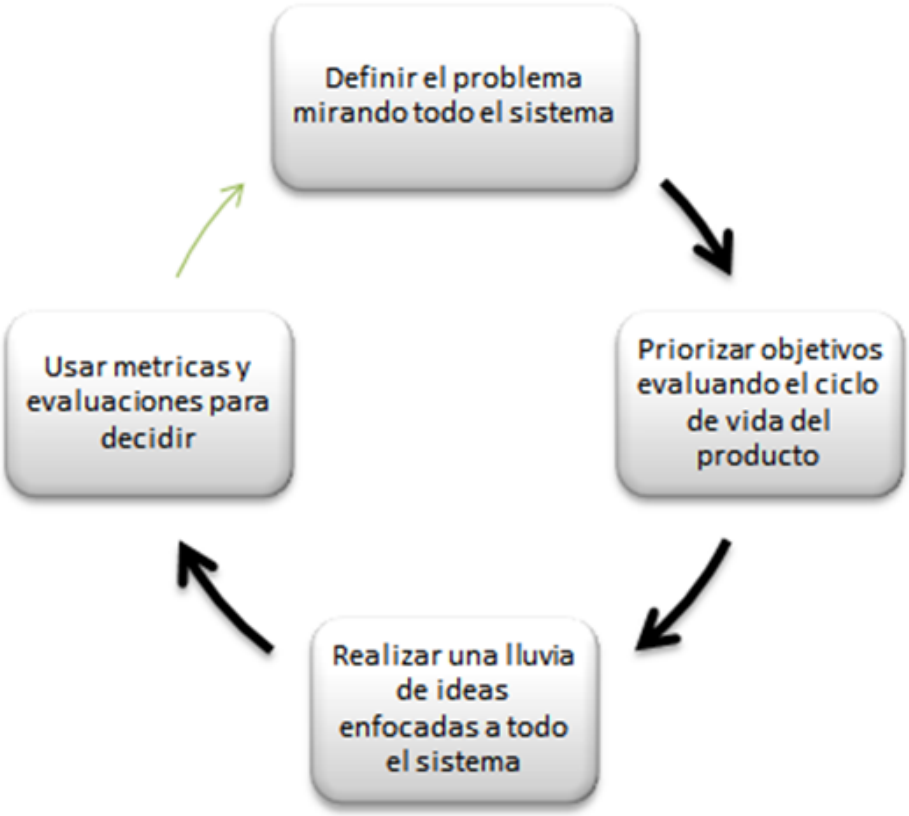
Unidad IV: Modelo de Diseño

4.1. Estrategias de diseño

El diseño se define como la búsqueda de una solución en cualquier campo, sin embargo las soluciones no llegan de una manera simple, muchas veces realizamos soluciones complejas a problemas sencillos o en otras ocasiones una gran solución conlleva a otro problema.

La cuestión está en cómo abordamos esos retos de diseño. Estudios demuestran que nos enfocamos en resolver los problemas de manera individual alejándonos cada vez más del sistema completo en el que estamos trabajando, “es como diseñar una ventana sin el edificio”, recuerden todo tiene un impacto y en un sistema todo está relacionado.

Así que por otro lado que tal si vemos todo el sistema y así planeamos mejor nuestro diseño, haciendo que las soluciones de una parte no perjudiquen a la otra, o mejor que se complementen entre si. Esto nos ayudara a ver que lugar social, ambiental y técnico nuestro producto hace parte. Se recomienda que tengamos en cuenta: Metas ambiciosas que resuelvan varios problemas, colaboración a través de diferentes disciplinas, establecer parámetros base, definir la vida útil, iniciar desde cero los diseños, usar datos medibles y no asumir reglas entre otros. En las siguientes imágenes podemos ver un sencillo pero útil flujo de trabajo para iniciar nuestros diseños o la mejora de ellos.



4.2. Diseño de objetos

Para los sistemas es posible definir un diseño en pirámide con las siguientes cuatro capas:

Subsistema. Contiene una representación de cada uno de los subsistemas que le permiten al software conseguir los requisitos definidos por el cliente e implementar la infraestructura técnica que los soporta.

Clases y Objetos. Contiene las jerarquías de clases que permiten crear el sistema utilizando generalizaciones y especializaciones mejor definidas incrementalmente. También contiene representaciones de diseño para cada objeto.

Mensajes. Contiene los detalles que permiten a cada objeto comunicarse con sus colaboradores. Establece las interfaces externas e internas para el sistema.

Responsabilidades. Contiene las estructuras de datos y el diseño algorítmico para todos los atributos y operaciones de cada objeto.

Todo el programa está construido en base a diferentes componentes (Objetos), todo objeto del mundo real tiene 2 componentes: características y comportamiento.

Una clase es una plantilla genérica para un conjunto de objetos de similares características.

La herencia básicamente consiste en que una clase puede heredar sus variables y métodos a varias subclases.

Por ejemplo:

Los mensajes son invocaciones a los métodos de los objetos.

esta es una técnica de diseño, la cual se caracteriza por la determinación y delegación de responsabilidades.

Análisis orientado a objetos

El modelo del análisis orientado a objetos ilustra información, funcionamiento y comportamiento.

El diseño orientado a objetos transforma el modelo del análisis en un modelo de diseño que sirve como anteproyecto para la construcción de software.

Modelos del diseño

Estáticos. Estructura de subsistemas y/o clases y sus relaciones.

Dinámicos. Se describen las estructuras que muestran la interacción entre objetos. Ejemplos de UML: diagramas de secuencia, diagramas de estado.

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.

Tipos: de creación, estructurales, de comportamiento.

4.3. Diseño de sistema

Durante el diseño de objetos, se ha desarrollado un diseño detallado con base en la arquitectura especificada. En general, el diseño de sistemas incluye aspectos como los siguientes:

•Selección del lenguaje de programación a utilizarse

•Incorporación de bibliotecas (para interfaces gráficas, estructuras de datos, etc.)

•Incorporación de una base de datos

•Incorporación de archivos en sus diferentes formatos

•Consideraciones de procesamiento, como concurrencia, paralelismo, distribución y tiempo real.

Estos aspectos pueden variar entre un sistema y otro.

Además, pueden afectar de manera importante la arquitectura final del sistema.

Existen diferentes enfoques para la incorporación del ambiente de implementación a la arquitectura del sistema:

•Agregar clases abstractas o interfaces que luego se especializarán según el ambiente de implementación.

•Instanciar objetos especializados para la administración delambiente de implementación.

•Configurar múltiples versiones del sistemacorrespondientes a diferentes plataformas.

4.4. Revisión del diseño

El proceso de revisión se realiza en tres etapas en correspondencia con los pasos del proceso de diseño:

- Revisión del diseño preliminar.
- Revisión crítica del diseño.
- Revisión del diseño de programas.

Revisión del diseño preliminar:

Los clientes y usuarios se reúnen para validar el diseño conceptual.

Se asegura que todos los aspectos relativos a los requerimientos han sido apropiadamente contemplados en el diseño.

Se invita a participar a ciertas personas claves:

- Cliente (s), quien ayuda a definir los req. del sistema.
- Analista (s), quien colabora para definir los req. del sistema
- Usuario (s), potenciales del sistema.
- Diseñador (es) del sistema.

- Un moderador (solo coordina), un secretario (no se involucra).
- Otros desarrolladores (entregan perspectiva externa)

4.5. Diagramas de secuencias del Diseño

La fase de diseño (y los modelos UML resultantes) expande y detalla los modelos de análisis tomando en cuenta todas las implicaciones y restricciones técnicas. El propósito del diseño es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente y construir una arquitectura simple y fácilmente extensible. Las clases definidas en el análisis fueron detalladas, y se añadieron nuevas clases para manejar áreas técnicas como base de datos, interfaz del usuario, comunicación, dispositivos, etc.

Diagrama de secuencia

Los casos de uso deben ser realizados durante esta etapa. Para describir el comportamiento dinámico del sistema, cualquiera de los diagramas de interacción del UML puede ser utilizados. Debido a que Rational Rose no soporta los diagramas de actividad y ofrece soporte limitado para los diagramas de colaboración (en notación completa del UML) usaremos diagramas de secuencia.

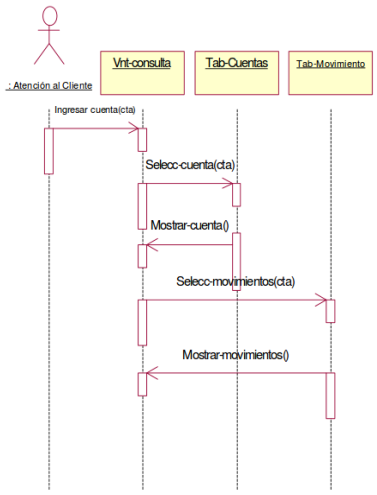
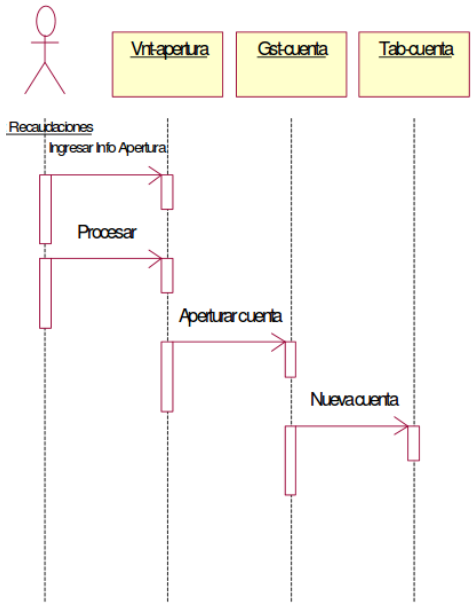
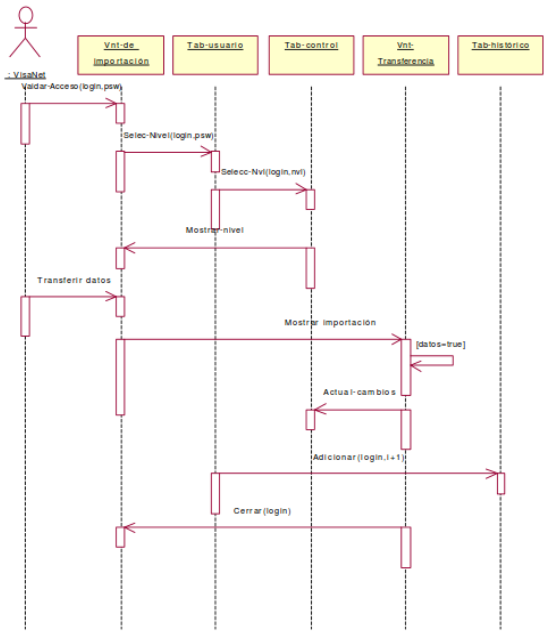


Diagrama de Secuencia Consultas y Reportes


Vnt-apertura
Gst-cuenta
Tab-cuenta
Recaudaciones
Ingresar Info Apertura
Procesar
Aperturar cuenta
Nueva cuenta


Vnt-de importacion
Tab-usuario
Tab-control
Vnt-Transferencia
Tab-historico
: VisaNet
Validar-Acceso(login,psw)
Selec-Nivel(login,psw)
Selecc-Nvl(login,nvl)
Mostrar-nivel
Transferir datos
Mostrar importación
[datos=true]
Actual-cambios
Adicionar(login,i+1)
Cerrar(login)

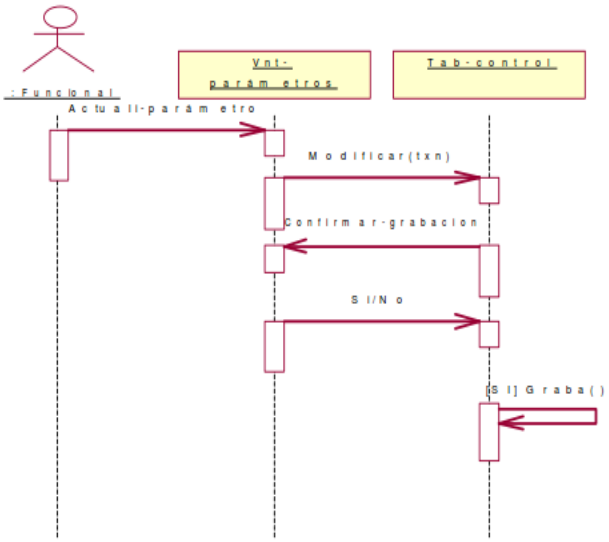


Diagrama de secuencia actualizar parámetros

4.6. Herramientas CASE para el diseño

La herramienta CASE (Computer-Aided Systems Engineering) cuyo significado en español es ingeniería de sistemas asistida por ordenador, es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo de sistemas y al igual que las herramientas CAD (Diseño Asistido por Computadora) o CAM (Manufactura Asistida por Computadora) su objetivo es acelerar el proceso para el que han sido diseñadas, en el caso de CASE para automatizar o apoyar una o mas fases del ciclo de vida del desarrollo de sistemas. La primera herramienta CASE como hoy la conocemos fue Excelerator en 1984, era para PC. Actualmente la oferta de herramientas CASE es muy amplia y tenemos por ejemplo el EASYCASE o WINPROJECT.