Técnicas de especificación de requerimientos:

- *Estáticas:* se describe el sistema a través de las entidades u objetos, sus atributos y sus relaciones con otros. No describe como las relaciones cambian con el tiempo. Cuando el tiempo no es un factor mayor en la operación del sistema, es una descripción útil y adecuada. Ejemplos (referencia indirecta, relaciones de recurrencia, definición axiomática, expresiones regulares, abstracciones de datos, etc.).

Es la descripción del sistema con una referencia indirecta al problema y su solución. Se define "QUE" se hace, no "COMO".

Definición axiomática: se definen las propiedades básicas de un sistema a través de operadores y axiomas (debe ser un conjunto completo y consistente). Se generan teoremas a través del comportamiento del sistema y se demuestran. Ejemplos: (Sistemas expertos, definición de TADs, etc.)

Abstracciones de datos: para aquellos sistemas en los que los datos determinan las clases de acciones que se realizan (importan para que son). Se categorizan los datos y se agrupan los semejantes. El diccionario contiene los tipos de datos (clases) y los datos (objetos). Se organizan de tal manera de aprovecharlas características compartidas.

- *Dinámicas:* se considera a un sistema en función de los cambios que ocurren a lo largo del tiempo. Se considera que el sistema está en un estado particular hasta que un estímulo lo obliga a cambiar su estado. Ejemplos (tablas de decisión, diagramas de transición de estados, diagramas de persianas, redes de Petri, etc.)

Validación:

Es el proceso de certificar la corrección del modelo de requerimientos contra las intenciones del usuario. Trata de mostrar que los requerimientos definidos son los que estipula el sistema. Se describe el ambiente en el que debe operar el sistema. Es importante, porque los errores en los requerimientos pueden conducir a grandes costos si se descubren más tarde.

Comprenden verificaciones de validez (para todos los usuarios), de consistencia (sin contradicciones), de completitud (todos los requerimientos) de realismo (se pueden implementar) y verificabilidad (se puede diseñar conjunto de pruebas).

Algunas técnicas de validación pueden ser manuales o automatizadas, revisiones de requerimientos (formales o informales), construcción de prototipos y generación de casos de prueba.

TABLAS DE DECISION

Es una herramienta que permite presentar de forma concisa las *reglas lógicas* que hay que utilizar para decidir *acciones a ejecutar* en función de las *condiciones* y la lógica de decisión de un problema específico. Describe el sistema como un conjunto de posibles CONDICIONES satisfechas por el sistema en un momento dado. REGLAS para reaccionar ante los estímulos que ocurren cuando se reúnen determinados conjuntos de condiciones y ACCIONES a ser tomadas como resultado.

La tabla se construye con condiciones simples y acciones simples, las condiciones deben ser atómicas y solo toman valores verdadero o falso y hay 2N reglas donde N es el número de condiciones.

Las especificaciones pueden ser completas (aquellas que determinan acciones para todas las reglas posibles), redundantes (aquellas que marcan para reglas que determinan las mismas condiciones acciones iguales) y contradictorias (aquellas que especifican para reglas que determinan las mismas condiciones acciones distintas).

REQUERIMIENTOS

Verificamos y validamos los requerimientos:

Correctitud. Consistencia. Completitud. Verificabilidad. Comprensibilidad. Adaptabilidad. Trazabilidad.

Gestión de los requerimientos:

Cambios en los requerimientos:

Estos cambian porque al analizar el problema, no se hacen las preguntas correctas a las personas correctas. Porque los clientes y los usuarios son distintos. Porque cambio el problema que se estaba resolviendo. Porque los usuarios cambiaron su forma de pensar o sus percepciones. Porque cambio el ambiente de negocios.

Evolución:

Requerimientos duraderos: relativamente estables, se derivan de la actividad principal de la organización.

Requerimientos volátiles: cambian durante el desarrollo del sistema o después de que se puso en operación. Ejemplos: Req. Cambiantes (cambian porque se modifica el ambiente, el entorno). Req. Emergentes (surgen como ampliación). Req. Consecuentes (surgen por la introducción del sistema. Pueden cambiar los procesos de la

organización por desarrollar nuevas formas de trabajo). Req. De compatibilidad (cambian porque interactúan con otros sistemas que cambian).

Pasos a realizar:

1) Identificación de requerimientos. 2) Gestión del cambio (análisis del problema y especificación del cambio, análisis del cambio y cálculo del costo, implementación del cambio). 3) Políticas de rastreo (fuente, requerimientos, diseño). 4) Ayuda de herramientas CASE (almacenar requerimientos, gestionar el cambio, gestionar el rastreo).

MAQUINAS DE ESTADO FINITO

Describe al sistema como un conjunto de estados donde el sistema reacciona a ciertos eventos posibles (externos o internos) → f (Si, Cj) = Sk
Al estar en el estado Si, la ocurrencia de la condición Cj hace que el sistema cambie al estado Sk.

DTE (DIAGRAMA DE TRANSICION DE ESTADOS)

Es una máquina de estado finito, y estas describen al sistema como un conjunto de estados donde el sistema reacciona a ciertos eventos posibles (externos o internos).
Construcción de un DTE:

1) Identificar los estados. 2) Si hay un estado complejo se puedo explotar. 3) Desde el estado inicial, se identifican los cambios de estados con flechas. 4) Se analizan las condiciones y las acciones para pasar de un estado a otro. 5) Se verifica la consistencia, que se hayan definido, se puedan alcanzar y se pueda salir de todos los estados.

REDES DE PETRI

Fueron inventadas por Carl Petri en la universidad de Bonn, Alemania Occidental (1962). Utilizadas para especificar sistemas de tiempo real en las que son necesarios representar aspectos de concurrencia.
Los sistemas concurrentes se diseñan para permitir la ejecución simultanea de componentes de programación, llamadas tareas o procesos, en varios procesadores o intercalados en un solo procesador.
Las tareas concurrentes deben estar sincronizadas para permitir la comunicación entre ellas (pueden operar a distintas velocidades, deben prevenir la modificación de datos compartidos o condiciones de bloqueo). Pueden realizarse varias tareas en paralelo, pero son ejecutadas en un orden impredecible. Estas NO son secuenciales.
Los eventos se presentan como transiciones (T).
Los estados se presentan como lugares o sitios (P).
Los arcos indican, a través de una flecha, la relación entre sitios y transición y viceversa. A los sitios se les asignan tokens (fichas) que se representan mediante un número o puntos dentro del sitio y son ilimitados. Esta asignación de tokens a sitios constituye la marcación.
Luego de una marcación inicial se puede simular la ejecución de la red.
El conjunto de tokens asociado a cada estado sirve para manejar la coordinación de eventos y estados. Una vez que ocurre un evento, un token puede "viajar" de uno de los estados a otro.
Las reglas de disparo provocan que los tokens "viajen" de un lugar a otro cuando se cumplen las condiciones adecuadas.
La ejecución es controlada por el número y distribución de los tokens y se realiza disparando transiciones habilitadas.
Una transición está habilitada cuando cada lugar de entrada tiene al menos tantos tokens como arcos hacia la transición.
Disparar una transición habilitada implica remover tokens de los lugares de entrada y distribuir tokens en los lugares de salida (teniendo en cuenta la cantidad de arcos que llegan y la cantidad de arcos que salen de la transición).
La ocurrencia de los eventos (transiciones) depende del estado del sistema.
Una condición puede ser V (con token) o F (sin token)
La ocurrencia de un evento está sujeta a que se den ciertas condiciones (pre) y al ocurrir el evento causa que se hagan verdaderas las post-condiciones.
Las RP son asincrónicas y el orden en que ocurren los eventos es uno de los permitidos.

- La ejecución es NO DETERMINÍSTICA

Se acepta que el disparo de una transición es instantáneo.

CASOS DE USO

Es el proceso de modelado de las "funcionalidades" del sistema en término de los eventos que interactúan e/ el usuario y el sistema. Tiene sus orígenes en el modelado orientado a objetos (Jacobson 1992) pero su eficiencia en modelado de

requerimientos hizo que se independice de la técnica de diseño utilizada, siendo aplicable a cualquier metodología de desarrollo. El uso de esta técnica facilita y alienta la participación de los usuarios.
Beneficios: Herramienta para capturar requerimientos funcionales. Descompone el alcance del sistema en piezas manejables. Medio de comunicación con los usuarios. Permite estimar el alcance del proyecto y el esfuerzo a realizar. Define una línea base para la definición de los planes de prueba. Define una línea base para toda la documentación del sistema. Proporciona una herramienta para el seguimiento de los requisitos.
Elementos:

- Diagrama de Casos de Uso (ilustra las interacciones entre el sistema y los actores).
- Escenarios (descripción de la interacción entre el actor y el sistema para realizar la funcionalidad).
- Caso de uso (representa un objetivo (funcionalidad) individual del sistema y describe la secuencia de actividades y de interacciones para alcanzarlo. Para que el CU sea considerado un requerimiento debe estar acompañado de su respectivo escenario).
- Actores (Un actor inicia una actividad (CU) en el sistema. Representa un papel desempeñado por un usuario que interactúa (rol). Puede ser una persona, sistema externo o dispositivo externo que dispare un evento (sensor, reloj).
- Relaciones:
    - *Asociaciones:* relación entre un actor y un CU en el que interactúan entre sí.
    - *Extensiones (extends):* un CU extiende una funcionalidad de otro CU. Un CU puede tener muchos CU extensiones. Los CU extensiones solo son iniciados por un CU.
    - *Uso o inclusión (uses):* reduce la redundancia entre dos o más CU al combinar los pasos comunes de los CU.
    - *Dependencia (depends):* relaciones entre CU que indica que un CU no puede realizarse hasta que se haya realizado otro CU.
    - *Herencia:* relación entre actores donde un actor hereda las funcionalidades de uno o varios actores.

Proceso de modelado:

- Identificar los actores.
- Identificar los CU para los requerimientos.
- Construir el diagrama.
- Realizar los escenarios.

Características importantes

- Un CU debe representar una funcionalidad concreta.
- La descripción de los pasos en los escenarios debe contener más de un paso, para representar la interacción entre los componentes.
- El uso de condicionales en el curso normal, es limitado a la invocación de excepciones, ya que este flujo representa la ejecución del caso de uso sin alteraciones.
- Las precondiciones no deben representarse en los cursos alternativos, ya que al ser una precondición no va a ocurrir.
- Los “uses” deben ser accedidos por lo menos desde dos CU.

HISTORIAS DE USUARIO

Una HU es una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario.
Son utilizadas en las metodologías de desarrollo agiles (ejemplo: XP, SCRUM) para la especificación de requisitos (acompañadas de las discusiones de los usuarios y las pruebas de validación).
Debe ser limitada. Son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Permiten responder más rápidamente los requisitos cambiantes. Al momento de implementar las historias, los desarrolladores deben tener la posibilidad de discutirlas con los clientes. Generalmente se espera que la estimación de tiempo de cada HU se situé entre unas 10 horas y un par de semanas.
Debe responder a tres preguntas:

¿Quién se beneficia?
¿Qué se quiere?
¿Cuál es el beneficio?

Esquema: **Como** (rol) **quiero** (algo) **para poder** (beneficio).
Ejemplo: Como usuario registrado quiero loguearme para poder empezar a utilizar la aplicación.
Características:

- *Independientes unas de otras:* de ser necesario, combinar las historias dependientes o buscar otra forma de dividir las historias de manera que resulten independientes.
- *Negociables:* la historia en sí misma no es lo suficiente explicita para considerarse un contrato, la discusión con los usuarios debe permitir esclarecer su alcance y este debe dejarse explicito bajo las formas de prueba de validación.
- *Valoradas por los clientes o usuarios:* los intereses de los clientes y de los usuarios no siempre coinciden, pero en todo caso, cada historia debe ser importante para alguno de ellos más que para el desarrollador.
- *Estimables:* un resultado de la discusión de una HU es la estimación del tiempo que tomara completarla. Esto permite estimar el tiempo total del proyecto.
- *Pequeñas:* las historias muy largas son difíciles de estimar e imponen restricciones sobre la planificación de un desarrollo iterativo. Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.
- *Verificables:* las HU cubren requerimientos funcionales, por lo que generalmente son verificables. Cuando sea posible, la verificación debe automatizarse, de manera que pueda ser verificable en cada entrega del proyecto.

Criterios de aceptación
Es el criterio por el cual se define si una historia de usuario fue desarrollada según la expectativa del Product Manager/Owner (como representante de los criterios del cliente) y si se puede dar como hecha.
Deben ser definidos durante la etapa inicial antes de la codificación, acompañan a la historia de usuario, porque la complementan y ayudan al equipo de desarrollo a entender mejor como se espera que el producto se comporte.
Son utilizados para expresar el resultado de las conversaciones del cliente con el desarrollador. El cliente debería ser quien las escriba más que el desarrollador.
Representan el inicia de la definición del cómo. No están diseñados para ser tan detallados como una especificación de diseño tradicional.
Si una HU tiene más de 4 criterios de aceptación, debe evaluarse subdividir la historia.
Puede añadirse un numero de escenario para identificar al criterio, asociado a la historia de usuario en cuestión.
Beneficios: - Al ser muy corta, esta representa requisitos del modelo de negocio que pueden implementarse rápidamente (días o semanas). – Necesitan poco mantenimiento. – Mantienen una relación cercana con el cliente. – Permite dividir los proyectos en pequeñas entregas. – Permite estimar fácilmente el esfuerzo de desarrollo. – Es ideal para proyectos con requisitos volátiles o no muy claros.
Limitaciones: - Sin pruebas de validación pueden quedar abiertas a distintas interpretaciones haciendo difícil utilizarlas como base para un contrato. – Se requiere un contacto permanente con el cliente durante el proyecto lo cual puede ser difícil o costoso. – Podría resultar difícil escalar a proyectos grandes. – Requiere desarrolladores muy competentes.

DIAGRAMA de FLUJO de DATOS (DFD)

Es una herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre si por “conductos” y almacenamiento de datos.
Representa la transformación de entradas a salidas y es también llamado diagrama de burbujas.
Es una herramienta comúnmente utilizada por sistemas operacionales en los cuales las funciones del sistema son de gran importancia y son más complejas que los datos que este maneja.

- Procesos: se representan con círculos o burbujas y representan las funciones individuales que ejecuta el sistema. Las funciones transforman entradas en salidas.
- Flujos: se representan con flechas continuas, la información que los procesos necesitan como entrada o producen como salida.
- Almacenamientos: se representan con líneas dobles los datos permanentes del sistema de operación. Al concretarse el diseño dará origen a las bases de datos y archivos.
- Entidades externas o terminadores: muestran productores o consumidores de información que residen fuera de los límites del sistema.

Diccionario de datos:
Listado organizado de todos los datos pertinentes al sistema:

- Definición sin ambigüedad de los datos y elementos del sistema.
- Permite revisar consistencia.

- Representa el contenido de la información.
- Define el significado de los flujos y los almacenes.
- Un dato debe contener: Tipo, nombre y descripción.
- Notación:
    - = está compuesto de
    - + y (secuencia)
    - () optativo
    - {} iteración
    - [] selección de alternativas
    - ” comentarios
    - @ campo clave de archivo
    - | separar opciones

Modelo Esencial:

Debe indicarse lo que el sistema debe hacer para satisfacer los requerimientos del usuario, con una mínima (en lo posible nula) explicación de cómo lo hace. Evitar el detalle de cualquier restricción o aspecto derivado de la implementación. Permite que sobreviva a cambios tecnológicos.

- Componentes:

1- *Modelo ambiental:* define las interfaces entre el sistema y el ambiente donde el mismo se ejecuta.

1.1- Declaración de propósito: en forma sintética debe indicarse el objetivo del sistema, de que es responsable el sistema.

1.2- Diagrama de contexto: es un caso especial de DFD donde el sistema se representa en una sola burbuja vinculada con las entidades externas y los almacenamientos externos.

1.3- Lista de acontecimientos: se trata de un listado de eventos (“estímulos”) a los que el sistema debe responder

Tipos de acontecimientos:

Flujo (F): llega algún o algunos datos al sistema.

Temporal (T): comienzan con la llegada de un momento dado en el tiempo.

Control (C)

La construcción del modelo ambiental es lo primero y más importante en la construcción del modelo de requerimientos del usuario para el nuevo sistema.

Una vez concluido el modelo ambiental hay que chequearlo con los usuarios clave y con el grupo de análisis para que sea la base del modelo de comportamiento del sistema.

2- Modelo de comportamiento:

a. El modelo preliminar de comportamiento contiene:
    i. Un diagrama preliminar de flujos de datos del sistema (DFD)
    ii. Un diagrama preliminar de entidad-relación (ER)
    iii. Una primera versión del diccionario de datos (DD)
    iv. Un diagrama de transición de estados (DTE)
b. El desarrollo descendente del modelo preliminar propone partir directamente del diagrama de contexto y obtener una primera versión (nivel 0) del DFD. Problema: parálisis del análisis.
c. Construcción:
    i. Una burbuja o proceso por cada acontecimiento de la lista
    ii. La burbuja se nombra identificando la respuesta del sistema al acontecimiento
    iii. Se dibujan las entradas-salidas y los almacenamientos apropiados para que la burbuja “funcione”.
    iv. Se chequea el borrador de DFD obtenido con el diagrama de contexto y la lista de acontecimientos.

El modelo de comportamiento es la representación del comportamiento final que el sistema debe tener para manejar con éxito el ambiente, dentro de las especificaciones requeridas por el usuario.

Nivelación de un DFD:

- A partir del DFD preliminar se realizan nivelaciones:
    - Ascendentes: agrupa las burbujas con algún criterio.
        - Tiene una utilidad de presentación al usuario

        - El DFD preliminar tiene un proceso por cada acontecimiento
        - Utilizando el principio de “ocultamiento de la información” agrupa los procesos que acceden al mismo almacenamiento
    - Descendentes: descompone las burbujas funcionalmente.
        - Las burbujas que no tiene más explosiones son las burbujas primitivas

SISTEMAS DE TIEMPO REAL

Características: responden a un mundo real en un tiempo prefijado, deben ser fiables, reinicializables y recuperables a fallas.
Ejemplos: control de procesos, investigación médica, comunicaciones, etc.
En aplicaciones de tiempo real, el sistema debe controlar la información continua en el tiempo generada por algún proceso del mundo real.
La notación del flujo de datos convencional no hace distenciones entre datos discretos y datos continuos en el tiempo.
Una ampliación de la notación básica del análisis estructurado proporciona un mecanismo para representar el flujo de datos continuo en el tiempo.
Para representar el flujo continuo en el tiempo se usa la flecha de dos cabezas, mientras que el flujo de datos discreto se representa con una flecha de una sola cabeza.

DFC

Muchas aplicaciones de software son dependientes del tiempo y procesan más información orientada al control que a los datos, por ej.: control de naves, procesos de fabricación, etc.
Las primeras ampliaciones que se hacen a este método están efectuadas por Ward y Mellor, y posteriormente lo hacen Hatley y Pirbhai y GoldSmith.
Estas ampliaciones permiten reflejar el flujo de control y el procesamiento de control, así como el procesamiento y el flujo de datos.