Proceso de desarrollo de software

Marcelo Vendan **www.softdownlaod.com.ar**

1. Abstract

En el siguiente trabajo se pretende desarrollar un tema bastante especial en el proceso de desarrollo de software, el cual es la base para que todo proyecto –independientemente de cual sea su porte- se realice de forma correcta y entendible.
El trabajo consta de tres grandes partes las cuales son: 1) Los fundamentos principales de una especificación de requerimientos; 2) Las metodologías que se aplican hoy en día para su construcción y 3) Una tercera parte en donde se pretende conocer como se lleva a cabo dicho proceso en una empresa de software de Uruguay.

2. Fundamentos del Análisis de Requerimientos

Definición: Es el conjunto de técnicas y procedimientos que nos permiten conocer los elementos necesarios para definir un proyecto de software.
Es la etapa más crucial del desarrollo de un proyecto de software.
La IEEE los divide en funcionales y no funcionales:
Funcionales: Condición o capacidad de un sistema requerida por el usuario para resolver un problema o alcanzar un objetivo.
No Funcionales: Condición o capacidad que debe poseer un sistema par satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto.
Para realizar bien el desarrollo de software es esencial realizar una especificación completa de los requerimientos de los mismos. Independientemente de lo bien diseñado o codificado que esté, un programa pobremente especificado decepcionará al usuario y hará fracasar el desarrollo.
La tarea de análisis de los requerimientos es un proceso de descubrimiento y refinamiento, El ámbito del programa, establecido inicialmente durante la ingeniería del sistema, es refinado en detalle. Se analizan y asignan a los distintos elementos de los programas las soluciones alternativas.
Tanto el que desarrolla el software como el cliente tienen un papel activo en la especificación de requerimientos. El cliente intenta reformular su concepto, algo nebuloso, de la función y comportamiento de los programas en detalles concretos, El que desarrolla el software actúa como interrogador, consultor y el que resuelve los problemas.
El análisis y especificación de requerimientos puede parecer una tarea relativamente sencilla, pero las apariencias engañan. Puesto que el contenido de comunicación es muy alto, abundan los cambios por mala interpretación o falta de información. El dilema con el que se enfrenta un ingeniero de software puede ser comprendido repitiendo la esentencia de un cliente anónimo: “Sé que crees que comprendes lo que piensas que he dicho, pero no estoy seguro de que lo que creíste oír sea lo que yo quise decir”.

3. Análisis de Requerimientos

El análisis de requerimientos es la tarea que plantea la asignación de software a nivel de sistema y el diseño de programas (Figura 1). El análisis de requerimientos facilita al ingeniero de sistemas especificar la función y comportamiento de los programas, indicar la interfaz con otros elementos del sistema y establecer las ligaduras de diseño que debe cumplir el programa. El análisis de requerimientos permite al ingeniero refinar la asignación de software y representar el dominio de la información que será tratada por el programa. El análisis de requerimientos de al diseñador la representación de la información y las funciones que pueden ser traducidas en datos, arquitectura y diseño procedimental. Finalmente, la especificación de requerimientos suministra al técnico y al cliente, los medios para valorar la calidad de los programas, una vez que se haya construido.

4. Tareas del Análisis

El análisis de requerimientos puede dividirse en cuatro áreas:
1.- Reconocimiento del problema
2.- Evaluación y síntesis
3.- Especificación
4.- Revisión.
Inicialmente, el analista estudia la especificación del sistema (si existe) y el plan de proyecto. Es importante comprender el contexto del sistema y revisar el ámbito de los programas que se usó para

generar las estimaciones de la planificación. A continuación, debe establecerse la comunicación necesaria para el análisis, de forma que se asegure el reconocimiento del problema.
Las formas de comunicación requeridas para el análisis se ilustran en la Figura 2. El analista debe establecer contacto con el equipo técnico y de gestión del usuario/cliente y con la empresa que vaya a desarrollar el software. El gestor del programa puede servir como coordinador para facilitar el establecimiento de los caminos de comunicación. El objetivo del analista es reconocer los elementos básicos del programa tal como lo percibe el usuario/cliente.

La evaluación del problema y la síntesis de la solución es la siguiente área principal de trabajo del análisis. El analista debe evaluar el flujo y estructura de la información, refinar en detalle todas las funciones del programa, establecer las características de la interfase del sistema y descubrir las ligaduras del diseño, Cada una de las tareas sirven para descubrir el problema de forma que pueda sintetizarse un enfoque o solución global.
Las tareas asociadas con el análisis y especificación existen para dar una representación del programa que pueda ser revisada y aprobada por el cliente. En un mundo ideal el cliente desarrolla una especificación de requerimientos del software completamente por sí mismo. Esto se presenta raramente en el mundo real. En el mejor de los casos, la especificación se desarrolla conjuntamente entre el cliente y el técnico.
Una vez que se hayan descrito las funcionalidades básicas, comportamiento, interfase e información, se especifican los criterios de validación para demostrar una comprensión de una correcta implementación de los programas. Estos criterios sirven como base para hacer una prueba durante el desarrollo de los programas. Para definir las características y atributos del software se escribe una especificación de requerimientos formal. Además, para los casos en los que se desarrolle un prototipo se realiza un manual de usuario preliminar.
Puede parecer innecesario realizar un manual de usuario en una etapa tan temprana del proceso de desarrollo, Pero de hecho, este borrador del manual de usuario fuerza al analista a tomar el punto de vista del usuario del software. El manual permite al usuario / cliente revisar el software desde una perspectiva de ingeniería humana y frecuentemente produce el comentario: “La idea es correcta pero esta no es la forma en que pensé que se podría hacer esto”. Es mejor descubrir tales comentarios lo mas tempranamente posible en el proceso.
Los documentos del análisis de requerimiento (especificación y manual de usuario) sirven como base para una revisión conducida por el cliente y el técnico. La revisión de los requerimientos casi siempre produce modificaciones en la función, comportamiento, representación de la información, ligaduras o criterios de validación. Además, se realiza una nueva apreciación del plan del proyecto de software para determinar si las primeras estimaciones siguen siendo validas después del conocimiento adicional obtenido durante el análisis.

5. Principios del Análisis

En la pasada década, se desarrollaron varios métodos de análisis y especificación del software. Los investigadores han identificado los problemas y sus causas y desarrollando reglas y procedimientos para resolverlos. Cada método de análisis tiene una única notación y punto de vista. Sin embargo, todos los métodos de análisis están relacionados por un conjunto de principios fundamentales:

El dominio de la información, así como el dominio funcional de un problema debe ser representado y comprendido.
El problema debe subdividirse de forma que se descubran los detalles de una manera progresiva (o jerárquica)
Deben desarrollarse las representaciones lógicas y físicas del sistema.

Aplicando estos principios, el analista enfoca el problema sistemáticamente. Se examina el dominio de la información de forma que pueda comprenderse su función mas completamente. La partición se aplica para reducir la complejidad. La visión lógica y física del software, es necesaria para acomodar las ligaduras lógicas impuestas por los requerimientos de procesamiento, y las ligaduras físicas impuestas por otros elementos del sistema.

6. El dominio de la Información

Todas las aplicaciones del software pueden colectivamente llamarse procesamiento de datos. Este término contiene la clave de lo que entendemos por requerimientos del software. El software se

construye para procesar datos; para transformar datos de una forma a otra; esto es, para aceptar entrada, manipularla de alguna forma y producir una salida. Este establecimiento fundamental de los objetivos es verdad tanto si construimos software por lotes para un sistema de nominas, como software empotrado en tiempo real para controlar el flujo de la gasolina de un motor de automóvil; el dominio de la información contiene tres visiones diferentes de los datos que se procesan por los programas de computadoras: 1) el flujo de información; 2) el contenido de la información y 3)la estructura de la información. Para comprender completamente el dominio de la información, deben considerarse cada una de estas tres partes.
El flujo de la información representa la manera en la que los datos cambian conforme pasan a través de un sistema. Refiriéndonos a la Figura 3, la entrada se transforma en datos intermedios y más adelante se transforma en la salida.

7. Partición

Normalmente los problemas son demasiado grandes y complejos para ser comprendidos como un todo. Por esta razón, tendemos a particionar (dividir) tales problemas en partes que puedan ser fácilmente comprendidas, y establecer interfases entre las partes, de forma que se realice la función global. Durante el análisis de requerimientos, el dominio funcional y el dominio de la información del software pueden ser particionados.
En esencia la partición descompone un problema en sus partes constituyentes. Conceptualmente, establecemos una representación jerárquica de la función o información y luego partimos el elemento superior mediante: 1) incrementando los detalles, moviéndonos verticalmente en la jerarquía, o 2) descomponiendo funcionalmente el problema, moviéndonos horizontalmente en la jerarquía.

8. Visiones Lógicas y Físicas

La visión lógica de los requerimientos del software presenta las funciones que han de realizarse y la información que ha de procesarse independientemente de los detalles de implementación.
La visión física de los requerimientos del software presenta una manifestación del mundo real de las funciones de procesamiento y las estructuras de información. En algunos casos se desarrolla una representación física como el primer paso del diseño del software. Sin embargo la mayoría de los sistemas basados en computador, se especifican de forma que se dictan ciertas recomendaciones físicas.

9. Construcción de Prototipos de Software

En análisis debe ser conducido independientemente del paradigma de ingeniería de software aplicado. Sin embargo, la forma que ese análisis tomara puede variar. En algunos casos es posible aplicar los principios de análisis fundamental y derivar a una especificación en papel del software desde el cual pueda desarrollarse un diseño. En otras situaciones, se va a una recolección de los requerimientos, se aplican los principios de análisis y se construye un modelo de software, llamado un prototipo, según las apreciaciones del cliente y del que lo desarrolla. Finalmente, hay circunstancias que requieren la construcción de un prototipo al comienzo del análisis, puesto que el modelo es el único mediante el que los requerimientos pueden ser derivados efectivamente.

10. Un escenario para la contruccion de prototipos

Todos los proyectos de ingeniería de software comienzan con una petición del cliente. La petición puede estar en la forma de una memoria que describe un problema, un informe que define un conjunto de objetivos comerciales o del producto, una petición de propuesta formal de una agencia o compañía exterior, o una especificación del sistema que ha asignado una función y comportamiento al software, como un elemento de un sistema mayor basado en computadora. Suponiendo que existe una petición para un programa de una de las formas dichas anteriormente, para construir un prototipo del software se aplican los siguientes pasos:

PASO 1. Evaluar la petición del software y determinar si el programa a desarrollar es un buen candidato para construir un prototipo.
Debido a que el cliente debe interaccionar con el prototipo en los últimos pasos, es esencial que: 1) el cliente participe en la evaluación y refinamiento del prototipo, y 2) el cliente sea capaz de tomar decisiones de requerimientos de una forma oportuna. Finalmente, la naturaleza del proyecto de desarrollo tendrá una fuerte influencia en la eficacia del prototipo.

PASO 2. Dado un proyecto candidato aceptable, el analista desarrolla una representación abreviada de los requerimientos.
Antes de que pueda comenzar la construcción de un prototipo, el analista debe representar los dominios funcionales y de información del programa y desarrollar un método razonable de partición. La aplicación de estos principios de análisis fundamentales, pueden realizarse mediante los métodos de análisis de requerimientos.

PASO 3. Después de que se haya revisado la representación de los requerimientos, se crea un conjunto de especificaciones de diseño abreviadas para el prototipo.
El diseño debe ocurrir antes de que comience la construcción del prototipo. Sin embargo, el diseño de un prototipo se enfoca normalmente hacia la arquitectura a nivel superior y a los aspectos de diseño de datos, en vez de hacia el diseño procedimental detallado.

PASO 4. El software del prototipo se crea, prueba y refina
Idealmente, los bloques de construcción de software preexisten se utilizan para crear el prototipo de una forma rápida. Desafortunadamente, tales bloques construidos raramente existen.
Incluso si la implementación de un prototipo que funcione es impracticable, es escenario de construcción de prototipos puede aun aplicarse. Para las aplicaciones interactivas con el hombre, es posible frecuentemente crear un prototipo en papel que describa la interacción hombre-maquina usando una serie de hojas de historia.

PASO 5. Una vez que el prototipo ha sido probado, se presenta al cliente, el cual “conduce la prueba” de la aplicación y sugiere modificaciones.
Este paso es el núcleo del método de construcción de prototipo. Es aquí donde el cliente puede examinar una representación implementada de los requerimientos del programa, sugerir modificaciones que harán al programa cumplir mejor las necesidades reales.

PASO 6. Los pasos 4 y 5 se repiten iterativamente hasta que todos los requerimientos estén formalizados o hasta que el prototipo haya evolucionado hacia un sistema de producción.
El paradigma de construcción del prototipo puede ser conducido con uno o dos objetivos en mente: 1) el propósito del prototipado es establecer un conjunto de requerimientos formales que pueden luego ser traducidos en la producción de programas mediante el uso de métodos y técnicas de ingeniería de programación, o 2) el propósito de la construcción del prototipo es suministrar un continuo que pueda conducir al desarrollo evolutivo de la producción del software. Ambos métodos tienen sus meritos y amos crean problemas.

11. Especificación

No hay duda de que la forma de especificar tiene mucho que ver con la calidad de la solución. Los ingenieros de software que se han esforzado en trabajar con especificaciones incompletas, inconsistentes o mal establecidas han experimentado la frustración y confusión que invariablemente se produce. Las consecuencias se padecen en la calidad, oportunidad y completitud del software resultante.
Hemos visto que los requerimientos de software pueden ser analizados de varias formas diferentes. Las técnicas de análisis pueden conducir a una especificación en papel que contenga las descripciones graficas y el lenguaje natural de los requerimientos del software. La construcción de prototipos conduce a una especificación ejecutable, esto es, el prototipo sirve como una representación de los requerimientos. Los lenguajes de especificación formal conducen a representaciones formales de los requerimientos que pueden ser verificados o analizados.

12. Principios de Especificación

La especificación, independientemente del modo en que se realice, puede ser vista como un proceso de representación. Los requerimientos se representan de forma que conduzcan finalmente a una correcta implementación del software.
Baltzer y Goldman proponen ocho principios para una buena especificación:

PRINCIPIO #1. Separar funcionalidad de implementación.
Primero, por definición, una especificación es una descripción de lo que se desea, en vez de cómo se realiza (implementa). Las especificaciones pueden adoptar dos formas muy diferentes. La primera forma

es la de funciones matemáticas: dado algún conjunto de entrada, producir un conjunto particular de salida. La forma general de tales especificaciones es encontrar [un/el/todos] resultado tal que P (entrada), donde P representa un predicado arbitrario. En tales especificaciones, el resultado a ser obtenido ha sido expresado enteramente en una forma sobre el que (en vez de cómo). En parte, esto es debido a que el resultado es una función matemática de la entrada (la operación tiene puntos de comienzo y parada bien definidos) y no esta afectado por el entorno que le rodea.

PRINCIPIO #2**.** Se necesita un lenguaje de especificación de sistemas orientado al proceso. Considerar una situación en la que el entorno sea dinámico y sus cambios afecten al comportamiento de alguna entidad que interactúe con dicho entorno. Su comportamiento no puede ser expresado como una función matemática de su entrada. En vez de ello, debe emplearse una descripción orientada al proceso, en la cual la especificación del que se consigue mediante la especificación de un modelo del comportamiento deseado en términos de respuestas funcionales, a distintos estímulos del entorno.

PRINCIPIO #3. Una especificación debe abarcar el sistema del cual el software es una componente. Un sistema esta compuesto de componentes que interactúan. Solo dentro del contexto del sistema completo y de la interacción entre sus partes puede ser definido el comportamiento de una componente especifica. En general, un sistema puede ser modelado como una colección de objetos pasivos y activos. Estos objetos están interrelacionados y dichas relaciones entre los objetos cambian con el tiempo. Estas relaciones dinámicas suministran los estímulos a los cuales los objetos activos, llamados agentes, responden. Las respuestas pueden causar posteriormente cambios y, por tanto, estímulos adicionales a los cuales los agentes deben responder.

PRINCIPIO #4. Una especificación debe abarcar el entorno en el que el sistema opera. Similarmente, el entorno en el que opera el sistema y con el que interactúa debe ser especificado. Afortunadamente, esto tan solo necesita reconocer que el propio entorno es un sistema compuesto de objetos que interactúan, pasivos y activos, de los cuales el sistema especificado es una agente, Los otros agentes, los cuales son por definición inalterables debido a que son parte del entorno, limitan el ámbito del diseño subsecuente y de la implementación. De hecho, la única diferencia entre el sistema y su entorno es que el esfuerzo de diseño e implementación subsecuente opera exclusivamente sobre la especificación del sistema. La especificación del entorno facilita que se especifique la interfaz del sistema de la misma forma que el propio sistema, en vez de introducir otro formalismo.

PRINCIPIO #5. Una especificación de sistema debe ser un modelo cognitivo.
La especificación de un sistema debe ser un modelo cognitivo, en vez de un modelo de diseño o implementación. Debe describir un sistema tal como es percibido por su comunidad de usuario. Los objetivos que manipula deben corresponderse con objetos reales de dicho dominio; los agentes deben modelar los individuos, organizaciones y equipo de ese dominio; y las acciones que ejecutan deben modelar lo que realmente ocurre en el dominio. Además, debe ser posible incorporar en la especificación las reglas o leyes que gobiernan los objetos del dominio. Algunas de estas leyes proscriben ciertos estados del sistema (tal como “dos objetos no pueden estar en el mismo lugar al mismo tiempo”), y por tanto limitan el comportamiento de los agentes o indican la necesidad de una posterior elaboración para prevenir que surjan estos estados.

PRINCIPIO #6. Una especificación debe ser operacional.
La especificación debe ser completa y lo bastante formal para que pueda usarse para determinar si una implementación propuesta satisface la especificación de pruebas elegidas arbitrariamente. Esto es, dado el resultado de una implementación sobre algún conjunto arbitrario de datos elegibles, debe ser posible usar la especificación para validar estos resultados. Esto implica que la especificación, aunque no sea una especificación completa del como, pueda actuar como un generador de posibles comportamientos, entre los que debe estar la implementación propuesta. Por tanto, en un sentido extenso, la especificación debe ser operacional.

PRINCIPIO #7. La especificación del sistema debe ser tolerante con la incompletitud y aumentable. Ninguna especificación puede ser siempre totalmente completa. El entorno en el que existe es demasiado complejo para ello. Una especificación es siempre un modelo, una abstracción, de alguna situación real (o imaginada). Por tanto, será incompleta. Además, al ser formulad existirán muchos niveles de detalle. La operacionalidad requerida anteriormente no necesita ser completa. Las herramientas de análisis empleadas para ayudar a los especificadores y para probar las

especificaciones, deben ser capaces de tratar con la incompletitud. Naturalmente esto debilita el análisis, el cual puede ser ejecutado ampliando el rango de comportamiento aceptables, los cuales satisfacen la especificación, pero tal degradación debe reflejar los restantes niveles de incertidumbre.

PRINCIPIO #8. Una especificación debe ser localizada y débilmente acoplada.
Los principios anteriores tratan con la especificación como una entidad estática. Esta surge de la dinámica de la especificación. Debe ser reconocido que aunque el principal propósito de una especificación sea servir como base para el diseño e implementación de algún sistema, no es un objeto estático precompuesto, sino un objeto dinámico que sufre considerables modificaciones. Tales modificaciones se presentan en tres actividades principales: formulación, cuando se está creando una especificación inicial; desarrollo, cuando la especificación se esta elaborando durante el proceso iterativo de diseño e implementación; y mantenimiento, cuando la especificación se cambia para reflejar un entorno modificado y/o requerimientos funcionales adicionales.

13. Metodos de Análisis de Requerimientos

Las metodologías de análisis de requerimientos combinan procedimientos sistemáticos con una notación única para analizar los dominios de información y funcional de un problema de software; suministra un conjunto de heurísticas para subdividir el problema y define una forma de representación para las visiones lógicas y físicas. En esencia, los métodos de análisis de requerimientos del software, facilitan al ingeniero de software aplicar principios de análisis fundamentales, dentro del contexto de un método bien definido.
La mayoría de los métodos de análisis de requerimientos son conducidos por la información. Estos es, el método suministra un mecanismo para representar el dominio de la información. Desde esta representación, se deriva la función y se desarrollan otras características de los programas.
El papel de los métodos de análisis de requerimientos, es asistir al analista en la construcción de “una descripción precisa e independiente” del elemento software de un sistema basado en computadora.

14. Metodologías de Análisis de Requerimientos

Las metodologías de análisis de requerimientos facilitan al analista la aplicación de los principios fundamentales del análisis de una manera sistemática.
Características Comunes
Aunque cada método introduce nueva notación y heurística de análisis, todos los métodos pueden ser evaluados en el contexto de las siguientes características comunes:

1. Mecanismos para el análisis del dominio de la información
2. Método de representación funcional
3. Definición de interfaces
4. Mecanismos para subdividir el problema
5. Soporte de la abstracción
6. Representación de visiones físicas y lógicas

Aunque el análisis del dominio de la información se conduce de forma diferente en cada metodología, pueden reconocerse algunas guías comunes. Todos los métodos se enfocan (directa o indirectamente) al flujo de datos y al contenido o estructura de datos. En la mayoría de los casos el flujo se caracteriza en el contexto de las transformaciones (funciones) que se aplican para cambiar la entrada en la salida. El contenido de los datos puede representarse explícitamente usando un mecanismo de diccionario o, implícitamente, enfocando primero la estructura jerárquica de los datos.
Las funciones se describen normalmente como transformaciones o procesos de la información. Cada función puede ser representada usando una notación especifica. Una descripción de la función puede desarrollarse usando el lenguaje natural, un leguaje procedimental con reglas sintácticas informales o un lenguaje de especificación forma.

15. Métodos de Análisis Orientados al Flujo de Datos

La información se transforma como un flujo a través de un sistema basado en computadora. El sistema acepta entrada de distintas formas; aplica un hardware, software y elementos humanos para transformar la entrada en salida; y produce una salida en distintas formas. La entrada puede ser una señal de control transmitida por un transductor, una serie de números escritos por un operador humano, un paquete de información transmitido por un enlace a red, o un voluminoso archivo de datos almacenado en memoria secundaria. La transformación puede comprender una sencilla comparación lógica, un complejo algoritmo numérico, o un método de inferencia basado en regla de un sistema experto. La salida puede

encender un sencillo led o producir un informe de 200 paginas. En efecto, un modelo de flujo de datos puede aplicarse a cualquier sistema basado en computadora independientemente del tamaño o complejidad.
Una técnica para representar el flujo de información a través del sistema basado en computadora se ilustra en la figura 4. La función global del sistema se representa como una transformación sencilla de la información, representada en la figura como una burbuja. Una o más entradas. Representadas como flechas con etiqueta, conducen la transformación para producir la información de salida. Puede observarse que el modelo puede aplicarse a todo el sistema o solo a un elemento de software. La clave es representar la información dada y producida por la transformación.

16. Diagramas de Flujos de Datos

Conforme con la información se mueve a través del software, se modifica mediante una serie de transformaciones. Un diagrama de flujos de datos (DFD), es una técnica grafica que describe el flujo de información y las transformaciones que se aplican a los datos, conforme se mueven de la entrada a la salida. La forma básica de un DFD se ilustra en la figura 5. El diagrama es similar en la forma a otros diagramas de flujo de actividades, y ha sido incorporado en técnicas de análisis y diseños propuesto por Yourdon y Constantine, DeMarco y Gane y Sarson. También se le conoce como un grafo de flujo de datos o un diagrama de burbujas.

17. Diccionario de Datos

Un análisis del dominio de la información puede ser incompleto si solo se considera el flujo de datos. Cada flecha de un diagrama de flujo de datos representa uno o más elementos de información. Por tanto, el analista debe disponer de algún otro método para representar el contenido de cada flecha de un DFD.
Se ha propuesto el diccionario de datos como una gramática casi formal para describir el contenido de los elementos de información y ha sido definido da la siguiente forma:
El diccionario de datos contiene las definiciones de todos los datos mencionados en el DFD, en una especificación del proceso y en el propio diccionario de datos. Los datos compuestos (datos que pueden ser además divididos) se definen en términos de sus componentes; los datos elementales (datos que no pueden ser divididos) se definen en términos del significado de cada uno de los valores que puede asumir. Por tanto, el diccionario de datos esta compuesto de definiciones de flujo de datos, archivos [datos almacenados] y datos usados en los procesos [transformaciones]...

18. Descripciones Funcionales

Una vez que ha sido representado el dominio de la información (usando un DFD y un diccionario de datos), el analista describe cada función (transformación) representada, usando el lenguaje natural o alguna otra notación estilizada. Una de tales notaciones se llama ingles estructurado (también llamado lenguaje de diseño del programa o proceso(LDP)). El ingles estructurado incorpora construcciones procedimentales básicas –secuencia, selección y repetición- junto con frases del lenguaje natural, de forma que pueden desarrollarse descripciones procedimentales precisas de las funciones representadas dentro de un DFD.

19. Métodos Orientados a la Estructura de Datos

Hemos ya observado que el dominio de la información para un problema de software comprende el flujo de datos, el contenido de datos y la estructura de datos. Los métodos de análisis orientados a la estructura de datos representan los requerimientos del software enfocándose hacia la estructura de datos en vez de al flujo de datos. Aunque cada método orientado a la estructura de datos tiene un enfoque y notación distinta, todos tienen algunas características en común: 1) todos asisten al analista en la identificación de los objetos de información clave (también llamados entidades o ítems) y operaciones (también llamadas acciones o procesos); 2)todos suponen que la estructura de la información es jerárquica; 3)todos requiere que la estructura de datos se represente usando la secuencia, selección y repetición; y 4) todos dan una conjunto de pasos para transformar una estructura de datos jerárquica en una estructura de programa.
Como los métodos orientados al flujo de datos, los métodos de análisis orientados a la estructura de datos proporcionan la base para el diseño de software. Siempre puede extenderse un método de análisis para que abarque el diseño arquitectural y procedimental del software.

20. Desarrollo de Sistemas de Jackson

El desarrollo de sistema de Jackson (DSJ) se obtuvo a partir del trabajo de M.A. Jackson sobre el análisis del dominio de la información y sus relaciones con el diseño de programas y sistemas. En palabras de Jackson: “El que desarrolla el software comienza creando un modelo de la realidad a la que se refiere el sistema, la realidad que proporciona su materia objeto [del sistema]...”
Para construir un DSJ el analista aplica los siguientes pasos:

Paso de las acciones y entidades. Usando un método muy similar a la técnica de análisis orientada al objeto, en este paso se identifican las entidades (persona, objetos u organizaciones que necesita un sistema para producir o usar información) y acciones (los sucesos que ocurren en el mudo real que afectan a las entidades).

Paso de estructuración de las entidades. Las acciones que afectan a cada entidad son ordenadas en el tiempo y representadas mediante diagramas de Jackson (una notación similar a un árbol).

Paso de modelación inicial. Las entidades y acciones se representan como un modelo del proceso; se definen las conexiones entre el modelo y el mundo real.

Paso de las funciones. Se especifican las funciones que corresponden alas acciones definidas.

Paso de temporización del sistema. Se establecen y especifican las características de planificación del proceso.

Paso de implementación. Se especifica el hardware y software como un diseño.

Los últimos tres pasos del DSJ están muy relacionados con el diseño de sistemas.

21. Requerimientos de las Bases de Datos

El análisis de requerimientos para una base de datos incorpora las mismas tareas que el análisis de requerimientos del software. Es necesario un contacto estrecho con el cliente; es esencial la identificación de las funciones e interfaces; se requiere la especificación del flujo, estructura y asociatividad de la información y debe desarrollarse un documento formal de los requerimientos.
Un tratamiento completo del análisis de las bases de datos va mas allá del ámbito de este paper.

22. Características de las bases de datos

El termino base de datos se ha convertido en uno de los muchos tópicos del campo de las computadoras. Aunque existen muchas definiciones elegantes, definiremos una base de datos como: una colección de información organizada de forma que facilita el acceso, análisis y creación de informes. Una base de datos contiene entidades de información que están relacionadas vía organización y asociación. La arquitectura lógica de una base de datos se define mediante un esquema que representa las definiciones de las relaciones entre las entidades de información. La arquitectura física de una base de datos depende de la configuración del hardware residente. Sin embargo, tanto el esquema (descripción lógica) como la organización (descripción física) deben adecuarse para satisfacer los requerimientos funcionales y de comportamiento para el acceso al análisis y creación de informes.

23. Ingeniería de Requerimientos en el URUGUAY

En esta sección trataremos este tema desde el punto de vista de una empresa que se dedica a la producción de software en el Uruguay.
Cabe destacar que es muy difícil el pasaje de un marco teórico a una realidad practica en su totalidad, de hecho creo que no debe haber ninguna empresa que así lo realice.
Para el desarrollo de este tema hemos mantenido una reunión con el encargado del área de análisis de dicha empresa, la cual por pedido de su gerencia no quiso ser revelada.
En dicha entrevista se aprovechó no solo para averiguar el modelo de ingeniería de requerimientos utilizado sino también todo el modelo de desarrollo en etapas que suelen utilizar, las herramientas de análisis, de diseño, de implementación y de testing.
El ingeniero Ledesma nos comentó que, en su mayoría, se utilizan los métodos de análisis orientados al flujo de datos, como ser los diagramas de flujo, pero no siempre. Mucho depende de cual sea el tamaño del proyecto a realizar.

Luego de una larga charla llegamos a la conclusión de que en su empresa todavía no se tomaba el proceso de análisis de requerimientos como un proceso formal en toda su extensión, si bien es parte esencial, todavía no se tiene una metodología de trabajo especifica, sino que se trabaja en forma ad-hoc.
El Ingeniero nos comentó que la aplicación del modelo es bastante intuitiva y que no se sigue el modelo al pie de la letra. A través de los años se ha ido desarrollando la experiencia y muchas veces la especificación de requerimientos se hace en base a este método (métodos de análisis orientados al flujo de datos) pero en forma intuitiva.
Anteriormente, cuando en la empresa se trabajaba pura y exclusivamente con el Visual Basic, cuando en ese entonces era un lenguaje estructurado, sin la capacidad de incorporación de objetos al desarrollo, se utilizaban otros métodos, los cuales ya no eran orientados al flujo de datos sino que eran orientados a la estructura de la información.
Con dicha entrevista, nos quedó totalmente claro que ninguna empresa de este ramo puede sobrevivir sin la elaboración total o parcial de un esquema de especificación de requerimientos.

24. Bibliografía

- Roger S. Pressman, “Ingeniería del Software: Un enfoque práctico”, Segunda edición, Editorial McGraw Hill, 1990
- Freeman, P., “Requirements Analysis and Specification”, Proc. Intl. Computer Technology Conf., ASME, San Francisco, August, 1980
- Balzer, R., and N. Goodman, “Principles of Good Software Specification”, Proc, on Specifications of Reliable Software, IEEE, 1979, pp.58-67.

Marcelo Bendahan
Softdownload Argentina
www.softdownload.com.ar
info@softdownload.com.ar