UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

PROCESO GENERAL DE DESARROLLO – ADMINISTRACIÓN DE BASES DE DATOS: PROPUESTA DE MODELO DE DESARROLLO DE BASES DE DATOS PARA PROCESOS DE DESARROLLO DE SOFTWARE

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO INFORMÁTICO

AUTOR: ALAIN OSORIO RODRÍGUEZ

TUTOR: ING. RÚDEL CÁRDENAS DÍAZ

Caracas, Junio 2008

DECLARACIÓN DE AUTORÍA

Declaramos que soy el único autor de este trabajo y autorizo a la dirección de la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____

Alain Osorio Rodríguez	Ingeniero Rúdel Cárdenas Díaz
Autor	Tutor

AGRADECIMIENTOS

Mencionar a todos los que, de una forma u otra, han contribuido en la realización de la presente investigación sería, de cierta forma, injusto. Este agradecimiento va a dedicado a todas esas personas que dieron su aporte de forma desinteresada con sus consejos, recomendaciones, opiniones y apoyo en la realización del presente trabajo.

Agradecer también a todos mis compañeros de trabajo los cuales han sido una parte activa en los resultados de este trabajo.

DEDICATORIA

Dedico este trabajo a mis dos más grandes amores: mí querida hija Claudia y mi señora Massiel, las cuales siempre han estado conmigo en las buenas y en las malas, son mi razón de vivir e inspiración para seguir trabajando por ellas.

RESUMEN

Con esta investigación se pretende desarrollar un tema en el cual el hombre ha venido trabajando de forma tal que forma parte indispensable de muchos procesos[1] en términos de eficiencia, eficacia y rendimiento. El tema se refiere a la guía y organización estructural de procesos que el hombre lleva a cabo para realizar determinada actividad, la cual requiere de la participación activa de un grupo de personas y diferentes tareas que se integran con el fin de obtener un resultado, el cual debe responder a una serie de características y cualidades determinadas por las personas que se beneficiarán con el producto final. Es por eso que todo proceso llevado a cabo por el hombre debe tener una base organizativa – estructural – metodológica con el fin de que al culminar el producto, el mismo cumpla con todos los requerimientos necesarios.

El elemento principal de problemáticas en diferentes procesos es que no poseen una organización clara, bien definida o fundamentada, que guíe todos sus elementos e interacciones entre los mismos.

El Proceso de Desarrollo de Software[2], no por ser novedoso, deja de tener una base que guíe el mismo con el fin de encaminarlo durante todo su ciclo de vida y las diferentes fases de desarrollo. De ahí que existan diferentes metodologías de desarrollo de software[3], las cuales se desarrollan a la par de las exigencias de las nuevas tecnologías y de la industria de software, y es por ello que están en constante evolución.

El objetivo fundamental de la investigación es desarrollar un modelo organizativo – estructural que guíe el proceso de desarrollo – administración de bases de datos, en el marco del Proceso de Desarrollo de Software, con el fin ganar en eficiencia, claridad del proceso y evitar elementos que puedan atentar contra la calidad y funcionalidad requerida por el producto final; así como en atrasos en cuanto a plazos con los

[1] **Proceso**: Un proceso define "*quién"* esta haciendo "*qué"*, *"cuando"* y "*cómo"* alcanzar un determinado objetivo. En la Ingeniería de Software consiste en construir un producto software o mejorar uno existente. (Jacobson, y otros, 2000 pág. XVI).

[2] **Proceso de Desarrollo de Software**: definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto. (Jacobson, y otros, 2000 pág. 13).

[3] **Metodología de Desarrollo de Software**: Un conjunto de actividades que tienen un orden lógico, en las cuales se utilizan herramientas, técnicas y que garantizan que al concluir las actividades tengamos el software que realmente queremos. (Universidad de las Ciencias Informáticas).

clientes, al mismo tiempo que garantizar un adecuado control de los elementos que se manejan respecto al desarrollo – administración de bases de datos.

Este modelo podrá ser aplicado tanto a proyectos de software en producción como nuevos proyectos de software, los cuales requieran como estructura de persistencia física una Base de Datos, y configurado según las características del proceso.

TABLA DE CONTENIDOS

TABLA DE FIGURAS

INTRODUCCIÓN

En la actualidad no se concibe un Proceso de Desarrollo de Software sin una base organizativa – estructural – metodológica que lo guíe, y por el constante desarrollo de las nuevas tecnologías, estas bases no son rígidas y están en una constante interacción con el proceso, siempre sujetas a modificaciones en su beneficio y desarrollo. De ahí que existan diversas metodologías que se adaptan, configuran y personalizan tanto para el Proceso de Desarrollo de Software en general como para distintos procesos que son de vital importancia durante todo el ciclo de vida del desarrollo de software.

El término bases de datos, y su estructura física, no es nuevo y el mismo ha ido evolucionando como lo han hecho las herramientas, métodos y sistemas informáticos en general. Ejemplo de esto es la variedad de Sistemas Gestores de Bases de Datos[4] (SGBD) existentes en la actualidad. Y por consiguiente existe más de un método, procedimiento o teoría de *"cómo hacer cada cosa"* en cuanto a este término se refiere.

Las bases de datos existen para soportar una o más aplicaciones, y el desarrollo de las mismas no puede ser visto como un proceso independiente dentro del Proceso de Desarrollo de Software. El desarrollo de base de datos es desarrollo de software, a diferencia de lo que muchos desarrolladores puedan ver como proceso independiente que se integra mediante llamadas a procedimientos almacenados, vistas, consultas. Muchas veces este inconveniente parte de la diferencia de teorías, procedimientos, lenguajes, equipos de desarrollo y de herramientas utilizadas durante el Proceso de Desarrollo de Software, por solo citar algunos ejemplos. Pudiéramos mencionar un Proyecto de Software el cual utiliza como SGBD MySql[5] y como IDE[6] de implementación para la aplicación de interacción con el usuario Microsoft® Visual Studio®.Net[7]. Ambos difieren tanto en herramientas de implementación como en el modo de desarrollo y

[4] **Sistema Gestor de Bases de Datos**: conjunto de programas y/o procesos que permiten crear y mantener una Base de Datos, asegurando su integridad, confidencialidad y seguridad.

[5] **MySql**: Base de Datos de código abierto.

[6] **IDE**: (Integrated Development Environment) es un programa compuesto por un conjunto de herramientas para un programador.

[7] **Microsoft® Visual Studio®.Net**: conjunto de aplicaciones para la creación tanto de aplicaciones de escritorio como de aplicaciones Web de empresa para trabajo en equipo creado por Microsoft.

equipo de trabajo. Sin embargo se pudiera ver más integración si utilizáramos Microsoft® SQL Server®[8], ya que el mismo se puede integrar mediante la misma herramienta de desarrollo y pudiéramos ver un proceso de desarrollo un poco más íntegro.

Independientemente de lo explicado con anterioridad, el tema de bases de datos dentro del equipo de desarrollo se trata diferente, dadas las características propias que tiene un desarrollo de base de datos. Esta diferencia radica en el sentido de la codificación de las funcionalidades, técnicas a tener en cuenta, roles, responsabilidades, procesos que se manejan y el de administrar en sí una Base de Datos, actividad implícita dentro del desarrollo de bases de datos la cual es toda una materia de respeto, sobre todo si se trata de un SGBD de prestigio a nivel mundial por toda la funcionalidad que este puede ofrecer.

De ahí que una Metodología de Desarrollo de Software no plantee todos los componentes necesarios para guiar el desarrollo en cuanto a bases de datos, y se necesiten elementos, técnicas, procedimientos y buenas prácticas, las cuales existen y están bien fundamentadas, ya sean especificas de cada gestor o generales del proceso, para complementar el mismo de forma eficiente, las cuales aunque sean diferentes tienen similitudes si se saben definir y enmarcar dentro del ciclo de desarrollo, sin importar que tipo de Metodología de Desarrollo de Software utilicemos y/o sistemas gestores de bases de datos y/o herramientas de desarrollo.

El desarrollo del software es una tarea complicada que depende en gran medida de la experiencia de las personas involucradas. Es muy común que al iniciar un Proceso de Desarrollo de Software, en específico en términos de Base de Datos, tomar experiencias de otros proyectos sobre este tema, o solicitar consultas a personas conocedoras para tener un modelo fiable a seguir. Del mismo modo que consultar bibliografía actualizada y concentrar conocimientos del tema para aplicar mejores prácticas de desarrollo – administración de bases de datos según el entorno y el proceso de desarrollo. En algunos casos se fracasa por la poca experiencia o mal encaminamiento del proceso, o se arrastran problemas de tipo estructural – organizativo por aplicar de forma incorrecta un método o no llevar correctamente los elementos de configuración, artefactos[9] o el proceso. Sobre todo si tratamos con negocios de cierta complejidad que requieran de una o más bases de datos de tamaño considerable, e integración de las

[8] **Microsoft® SQL Server®**: Sistema Gestor de Bases de Datos creado por Microsoft.

[9] **Artefacto**: producto tangible resultante del proceso de desarrollo de software.

mismas. Al mismo tiempo que dificulta agregar nuevas funcionalidades sin que el sistema sufra gastos considerables en todos los aspectos, además de la implicación que conlleva el *"cambiar lo que ya estaba hecho"* o *"modificar lo que ya funciona"* si no se tiene en cuenta una integración de todos los elementos mencionados anteriormente.

La comprensión del software es uno de los problemas más complicados en la tarea de mantenimiento y evolución, la cual pudiera tornarse caótica a la vez que arrastramos más errores.

Debido a la ausencia de un modelo genérico de desarrollo de bases de datos en procesos de desarrollo de software, cabe preguntarse cómo establecer un modelo para el desarrollo de bases de datos en sentido general, capaz de guiar y controlar el mismo de una forma eficiente durante todo el ciclo de vida del desarrollo de software, sin que sea una dificultad una Metodología de Desarrollo de Software, herramientas o Sistemas Gestores de Base de Datos.

Dada esta problemática se pretende diseñar un proceso general de desarrollo – administración para bases de datos, en el marco de procesos de desarrollo de software, que permita describir al mismo como un todo, apoyado en las distintas técnicas y procedimientos existentes de cada uno de los aspectos fundamentales, con el fin de ganar en eficiencia, claridad del proceso, reducción de tiempos, disminución del esfuerzo de mantenimiento, consistencia, fiabilidad, comunicación entre desarrolladores y evitar elementos que puedan atentar contra la calidad y funcionalidad requerida por el producto final, en este caso un producto de software, así como en atrasos en cuanto a plazos con los clientes, al mismo tiempo que garantizar un adecuado control de los elementos que se manejan respecto al desarrollo – administración de bases de datos, permitiendo desarrollar un proceso estándar y a la vez configurable según las características y peculiaridades de un Proyecto de Software determinado.

Se han trazado los siguientes objetivos específicos:

- Estructurar el desarrollo de bases de datos en actividades.
- Definir tareas generales para cada actividad.
- Definir roles y responsabilidades para cada actividad.
- Definir artefactos para cada una de las actividades.

Constituye el objeto de la investigación Procesos de Desarrollo de Software, Metodologías de Desarrollo de Software, modelos, marcos de trabajo y mejores prácticas de desarrollo de bases de datos.

Constituye el campo de acción el estudio de diferentes procedimientos, técnicas y buenas practicas para guiar y controlar el proceso de desarrollo – administración de bases de datos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capitulo se abordarán temas relacionados con la Ingeniería de Software y el Proceso de Desarrollo de Software en sentido general. Veremos ejemplos de Metodologías de Desarrollo de Software y su punto de contacto con el desarrollo de bases de datos, y específicamente como se lleva este proceso en sentido general, así como particularmente en la Universidad de las Ciencias Informáticas por parte del los equipos de desarrollo de Bases de Datos en los principales proyectos productivos.

> **Nota:** Esta fuera del alcance de la investigación el estudio a fondo de las Metodologías de Desarrollo de Software. Solo se abordaran algunas de ellas y aquellos elementos de interrelación con el desarrollo de bases de datos en procesos de software que requieran como estructura de persistencia física una o más bases de datos.

INGENIERÍA DE SOFTWARE

La Ingeniería de Software es una disciplina o área de la Informática o Ciencias de la Computación, que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo. Hoy día es cada vez más frecuente la consideración de la Ingeniería de Software como una nueva área de la ingeniería.

La Ingeniería de Software trata con áreas muy diversas de la Informática y de las Ciencias de la Computación, tales como construcción de compiladores, sistemas operativos o desarrollos en Intranet - Internet, abordando todas las fases del ciclo de vida del desarrollo de cualquier tipo de sistemas de información y aplicables a una infinidad de áreas tales como: negocios, investigación científica, medicina, producción, logística, banca, control de tráfico, meteorología, el mundo del derecho, la red de redes, Internet, redes Intranet y Extranet.

En término general, la Ingeniería de Software es una tecnología multicapa en la que, según Roger S. Pressman, se pueden identificar: los métodos (indican cómo construir técnicamente el software), el proceso (es el fundamento de la Ingeniería de Software, es la unión que mantiene juntas las capas de la tecnología) y las herramientas (soporte automático o semiautomático para el proceso y los métodos).

Figura 1: La Ingeniería de Software es una tecnología multicapa**.**

El trabajo que se asocia a la Ingeniería de Software se puede dividir en tres fases genéricas, con independencia del área de aplicación, tamaño o complejidad del proyecto.

Fase de definición: se centra sobre el *"qué"*. Es decir, durante la definición, el que desarrolla el software intenta identificar qué información ha de ser procesada, qué función y rendimiento se desea, qué comportamiento del sistema, qué interfaces van a ser establecidas, qué restricciones de diseño existen, y qué criterios de validación se necesitan para definir un sistema correcto.

Fase de desarrollo: se centra en el *"cómo".* Es decir, durante el desarrollo un Ingeniero de Software intenta definir como han de diseñarse las estructuras de datos, como ha de implementarse la función dentro de una arquitectura de software, cómo han de implementarse los detalles procedimentales, como han de caracterizarse interfaces, como ha de traducirse el diseño en un lenguaje de programación y cómo han de realizarse las pruebas.

Fase de mantenimiento: se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software y a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente. Durante la fase de mantenimiento se encuentran cuatro tipos de cambios:

1. **Corrección:** Incluso llevando a cabo las mejores actividades de garantía de calidad, es muy probable que el cliente descubra defectos en el software. El mantenimiento correctivo cambia el software para corregir los defectos.

2. **Adaptación:** Con el paso del tiempo, es probable que cambie el entorno original (CPU, el Sistema Operativo, las reglas de empresa, las características externas de productos.) para el que se desarrolló el software. El mantenimiento adaptativo produce modificación en el software para acomodarlo a los cambios de su entorno externo.

3. **Mejora:** Conforme se utilice el software, el cliente/usuario puede descubrir funciones adicionales que van a producir beneficios. El mantenimiento perfectivo lleva al software más allá de sus requisitos funcionales originales.

4. **Prevención**: El software de computadora se deteriora debido al cambio, y por esto el mantenimiento preventivo también llamado reingeniería de software, se debe conducir a permitir que el software sirva para las necesidades de los usuarios finales. En esencia, el mantenimiento preventivo hace cambios en programas de computadora a fin de que se puedan corregir, adaptar y mejorar más fácilmente.

PROCESOS DE DESARROLLO DE SOFTWARE

Cuando se trabaja para construir un producto o un sistema, es importante seguir una serie de pasos predecibles, un mapa de carreteras que le ayude a obtener el resultado oportuno de calidad. El mapa de carreteras a seguir es llamado Proceso de Software, el cual es de suma importancia ya que proporciona estabilidad, control y organización a una actividad que puede, si no se controla, volverse caótica.

Un Proceso de Software proporciona la estructura desde la que se puede establecer un detallado plan para el desarrollo de software. Un pequeño número de actividades estructurales se puede aplicar a todos los proyectos de software, sin tener en cuenta su tamaño o complejidad.

A un nivel detallado, el proceso que adoptemos depende del software que estamos construyendo. Un proceso puede ser apropiado para crear software de un sistema de aviación, mientras que un proceso diferente por completo puede ser adecuado para la creación de un sitio web.

Desde el punto de vista de un Ingeniero de Software, los productos obtenidos son programas, documentos y datos que se producen como consecuencia de las actividades de la Ingeniería de Software definidas por el proceso.

Un proceso define *"quién"* está haciendo *"qué"*, *"cuándo"* y *"cómo"* para alcanzar un determinado objetivo. Un Proceso de Desarrollo de Software es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto.

Un proceso de software se puede caracterizar como se muestra en la siguiente figura.

Figura 2: El Proceso de Software.

Se establece un marco común del proceso definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos de software, con independencia de su tamaño o complejidad. Un número de conjuntos de tareas (cada uno es una colección de tareas de trabajo de ingeniería de software, hitos de proyectos, productos de trabajo, y puntos de garantía de calidad) que permiten que las actividades del marco de trabajo se adapten a las características del proyecto de software y a los requisitos del equipo del proyecto.

Finalmente, las actividades de protección (tales como garantía de calidad de software, gestión de configuración de software y medición) abarcan el modelo de procesos. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso.

Las fases genéricas que caracterizan el proceso de software, definición, desarrollo y soporte, son aplicables a todo software.

Existen diferentes modelos de procesos para la Ingeniería de Software. Cada uno representa un intento de ordenar una actividad inherentemente caótica.

MODELO LINEAL SECUENCIAL

Llamado algunas veces ciclo de vida básico o modelo en cascada, el modelo lineal secuencial sugiere un enfoque sistemático, secuencial, para el desarrollo del software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento. La siguiente figura muestra el modelo lineal secuencial para la Ingeniería de Software. Modelado según el ciclo de ingeniería convencional.

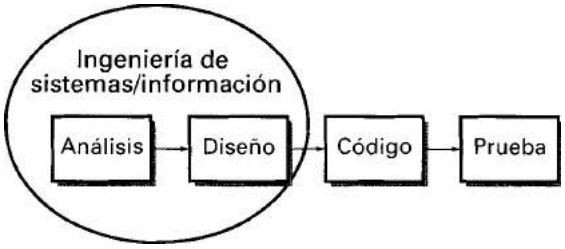

Figura 3: El modelo lineal secuencial.

MODELO DE CONSTRUCCIÓN DE PROTOTIPOS

El paradigma de construcción de prototipos comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un diseño rápido. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente (por ejemplo: enfoques de entrada y formatos de salida). El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos de software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

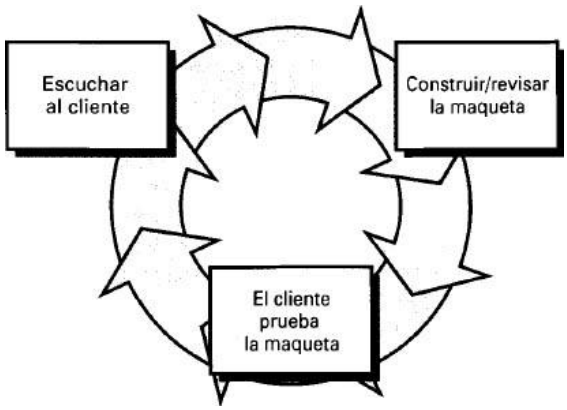

Figura 4: El paradigma de construcción de prototipos.

MODELO DRA

El Desarrollo Rápido de Aplicaciones (DRA) es un modelo de proceso del desarrollo de software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto. El modelo DRA es una adaptación a alta velocidad del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando una construcción basada en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite al equipo de desarrollo crear un sistema completamente funcional dentro de períodos cortos de tiempo, por ejemplo: de 60 a 90 días.

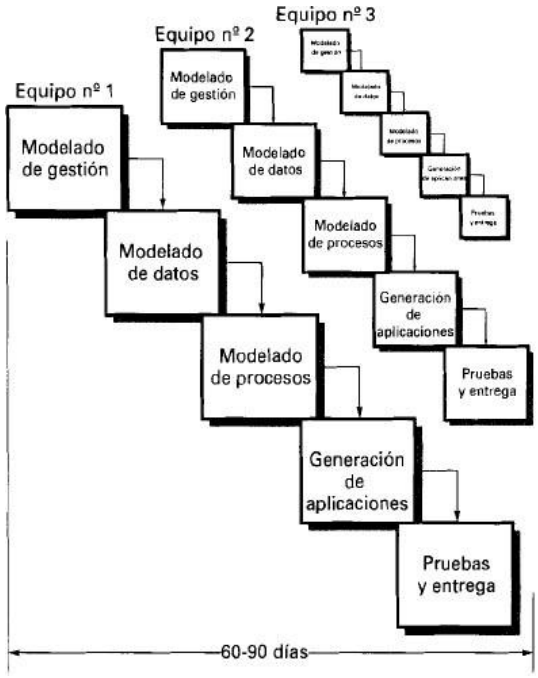

Figura 5: El modelo DRA.

MODELO INCREMENTAL

El modelo incremental combina elementos del modelo lineal secuencial (aplicados repetidamente) con la filosofía interactiva de construcción de prototipos. El modelo incremental aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software.

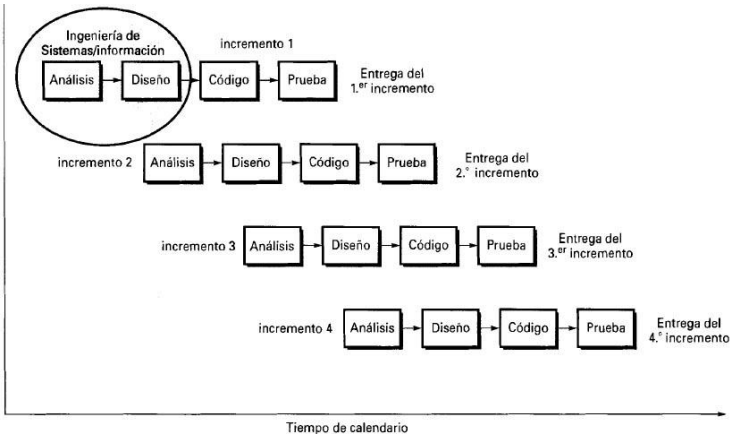

Figura 6: Modelo incremental.

MODELO ESPIRAL

El modelo en espiral es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software. En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.

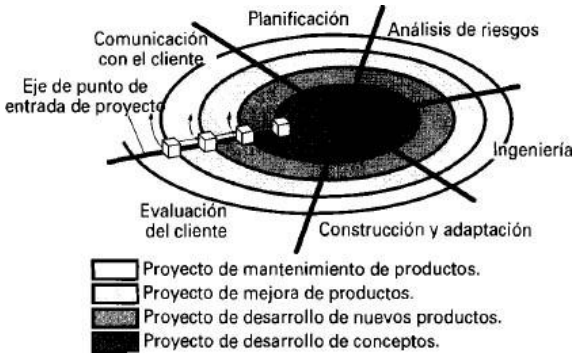

Figura 7: Un modelo espiral típico.

METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Una Metodología de Desarrollo de Software no es más que un conjunto de actividades que tienen un orden lógico, en las cuales se utilizan herramientas, técnicas y que garantizan que al concluir las actividades tengamos el software que realmente queremos. Una metodología es un proceso. No existe una Metodología de Software universal. Las características de cada proyecto (equipo de desarrollo, recursos.) exigen que el proceso sea configurable.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no llevamos una metodología de por medio, lo que obtenemos es clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos.

Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales.

RATIONAL UNIFIED PROCESS (RUP)

La metodología RUP, llamada así por sus siglas en inglés de Rational Unified Process, tiene como principal característica el estar dirigido por casos de uso[10], centrado en la arquitectura, iterativo e incremental. Esta metodología divide en cuatro fases el desarrollo del software:

1. **Inicio**: El Objetivo en esta etapa es determinar la visión del proyecto.
2. **Elaboración**: En esta etapa el objetivo es determinar la arquitectura óptima.
3. **Construcción**: En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
4. **Transición**: El objetivo es llegar a obtener el release[11] del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

El ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

Disciplina de Desarrollo

- **Ingeniería de Negocios**: Entendiendo las necesidades del negocio.
- **Requerimientos**: Trasladando las necesidades del negocio a un sistema automatizado.
- **Análisis y Diseño**: Trasladando los requerimientos dentro de la arquitectura de software.
- **Implementación**: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.

[10] **Casos de Uso**: Secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

[11] **Release**: Por su traducción se refiere a una versión o liberación del producto software.

- **Pruebas**: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado esta presente.

Disciplina de Soporte

- **Configuración y administración de cambio**: Guardando todas las versiones del proyecto.
- **Administrando el proyecto**: Administrando horarios y recursos.
- **Ambiente**: Administrando el ambiente de desarrollo.
- **Distribución**: Hacer todo lo necesario para la salida del proyecto.

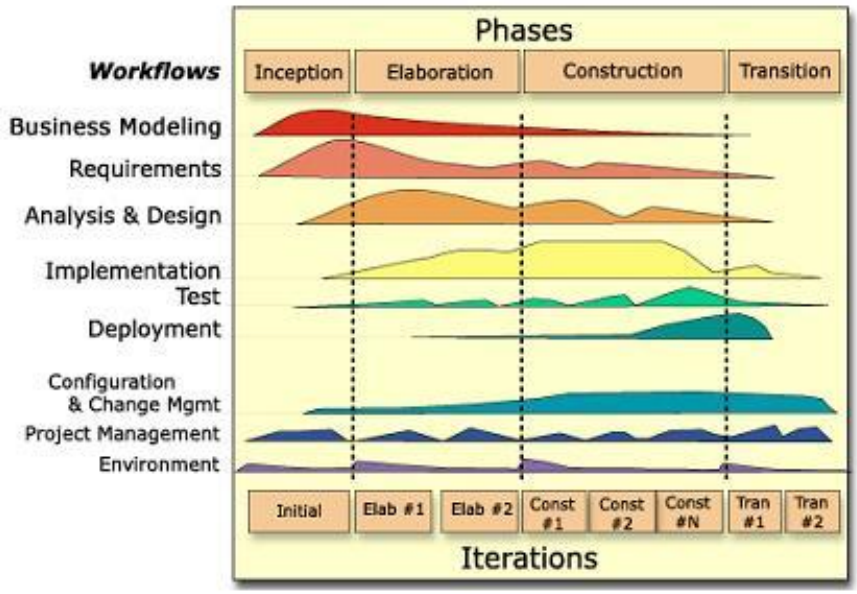

Figura 8: Fases e Iteraciones de RUP.

Los elementos del RUP son:

- **Actividades**: Son los procesos que se llegan a determinar en cada iteración.
- **Trabajadores**: Vienen hacer las personas o entes involucrados en cada proceso.
- **Artefactos**: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

EXTREME PROGRAMMING (XP)

Es una de las metodologías de desarrollo ágil de software más exitosas utilizadas en la actualidad para proyectos de corto plazo, de corto equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

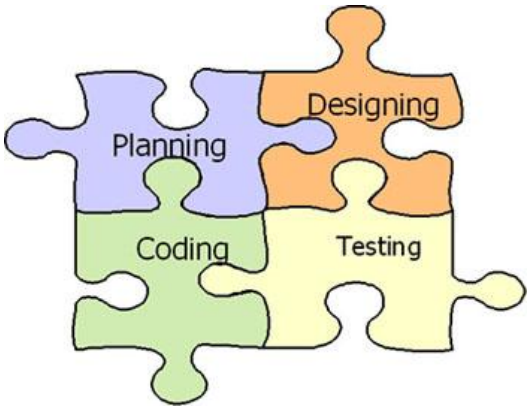

Figura 9: Metodología XP.

XP intenta minimizar el riesgo de fallo del proceso por medio de la disposición permanente de un representante competente del cliente a disposición del equipo de desarrollo. Este representante debería estar en condiciones de contestar rápida y correctamente cualquier pregunta del equipo de desarrollo de forma que no se retrase al tomar decisiones, de ahí lo de competente.

XP define UserStories como base del software a desarrollar. La misma las describe el cliente y describen escenarios sobre el funcionamiento del software, que no solo se limitan a la GUI[12] sino también pueden describir el modelo, dominio. A partir de las UserStories y de la arquitectura perseguida se crea un plan de releases[13] entre el equipo de desarrollo y el cliente.

Para cada release se discutirán los objetivos de la misma con el representante del cliente y se definirán las iteraciones (de pocas semanas de duración) necesarias para cumplir con los objetivos del release. El resultado de cada iteración es un programa que se transmite al cliente para que lo juzgue. En base a su opinión se definen las siguientes iteraciones del proyecto y si el cliente no esta contento se adaptará el plan de releases e iteraciones hasta que el cliente de su aprobación y el software este a su gusto.

La metodología se basa en:

- **Pruebas Unitarias:** se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.
- **Refabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

[12] **GUI**: Graphic User Interface. Interfaz de Grafica Usuario por su traducción.

[13] **Releases**: Ver Release 11.

XP propone:

- Empezar en pequeño y añadir funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

Derechos del Cliente

- Decidir que se implementa.
- Saber el estado real y el progreso del proyecto.
- Añadir, cambiar o quitar requerimientos en cualquier momento.
- Obtener lo máximo de cada semana de trabajo.
- Obtener un sistema funcionando cada 3 o 4 meses.

Derechos del Desarrollador

- Decidir como se implementan los procesos.
- Crear el sistema con la mejor calidad posible.
- Pedir al cliente en cualquier momento aclaraciones de los requerimientos.
- Estimar el esfuerzo para implementar el sistema.
- Cambiar los requerimientos en base a nuevos descubrimientos.

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

FEATURE DRIVEN DEVELOPMENT (FDD)

FDD es un proceso diseñado por Peter Coad, Erich Lefebvre y Jeff De Luca y se podría considerar a medio camino entre RUP y XP, más similar a XP.

FDD esta pensado para proyectos con tiempo de desarrollo relativamente cortos (menos de un año). Se basa en un proceso con iteraciones cortas (alrededor de 2 semanas) que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorizar.

Las iteraciones se deciden a base de features (de ahí el nombre del proceso) o funcionalidades, que son pequeñas partes del software con significado para el cliente.

Un proyecto que sigue FDD se divide en 5 fases:

1. Desarrollo de un modelo general.
2. Construcción de la lista de funcionalidades.
3. Plan de releases en base a las funcionalidades a implementar.
4. Diseñar en base a las funcionalidades.
5. Implementar en base a las funcionalidades.

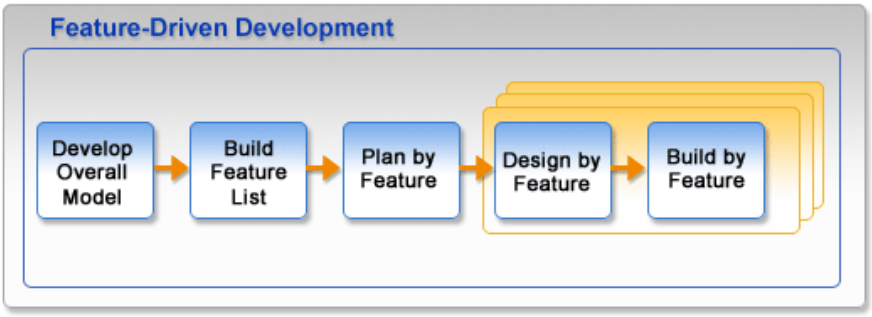

Figura 10: Vista general de FDD.

Las primeras tres fases ocupan gran parte del tiempo en las primeras iteraciones, siendo las dos últimas las que absorben la mayor parte del tiempo según va avanzando el proyecto, limitándose las primeras a un proceso de refinamiento.

El trabajo se realiza en grupo, aunque siempre habrá un responsable último, con mayor experiencia, que tendrá la última palabra en caso de no llegar a un acuerdo.

Las funcionalidades a implementar en una release se dividen entre los distintos subgrupos del equipo, y se proceden a implementarlas. Las clases escritas tienen propietario (es decir, solo quien las crea puede cambiarlas), es por ello que en el equipo que implementa una funcionalidad dad deberán estar todos los dueños de las clases implicadas, pudiendo encontrarse un programador en varios grupos, implementando distintas funcionalidades. Habrá también un programador jefe que hará las funciones de líder del grupo que implementa esa funcionalidad.

PUNTOS DE CONTACTO METODOLOGÍAS – BASES DE DATOS

Como vimos anteriormente una Metodología de Desarrollo de Software no plantea el *"cómo"* llevar el proceso de desarrollo – administración de Bases de Datos, y es por eso que en cada equipo de proyecto de cierta complejidad, exista un grupo especializado, o al menos un especialista, en este tema para llevar

todo el tema relacionado con la Base de Datos. Por otra parte, como también referíamos anteriormente, el tema de Bases de Datos no se ve aislado al Proceso de Desarrollo de Software. Y en este sentido las Metodología de Desarrollo de Software deben tener como mínimo un punto de inicio al desarrollo de Bases de Datos. Directa o indirectamente. Más o menos abarcador.

RUP + DESARROLLO DE BASES DE DATOS

RUP define varias disciplinas durante todo el ciclo de vida del desarrollo de software. Una de ellas es el **Análisis y Diseño**, que tiene como principales objetivos el transformar los requerimientos en un diseño de sistema, desarrollar una arquitectura sólida para el mismo y adaptar el diseño para que concuerde con el medio de implementación, diseñado para un buen rendimiento.

Y es precisamente en este flujo de trabajo que se ve el primer acercamiento al tema de Base de Datos. Este flujo de trabajo define una actividad nombrada **Diseño de la Base de Datos**, realizada por un trabajador definido como **Diseñador de Bases de Datos**. Los propósitos de esta actividad según RUP es asegurar que los datos persistentes sean almacenados consistente y eficientemente, y definir el comportamiento que debe ser implementado en la Base de Datos.

Los pasos a desarrollar son:

1. Desarrollar el Modelo Lógico de Datos (opcional).
2. Desarrollar el Diseño Físico de la Base de Datos.
    a. Definir dominios.
    b. Crear elementos iniciales de diseño físico de Bases de Datos.
    c. Definir tablas de referencia.
    d. Crear llaves primarias y restricciones.
    e. Definir datos e integridad referencial. reglas de ejecución.

f. De-normalizar para optimizar el rendimiento.

g. Optimizar el acceso a datos.

h. Definir características de almacenamiento.

i. Diseñar procedimientos almacenados para distribuir el comportamiento de clases a la base de datos.

3. Examinar resultados.

Pudiéramos decir hasta este punto que RUP define un grupo amplio de actividades para desarrollar respecto a bases de datos. Pero aun así, el desarrollo de bases de datos va un poco más allá de estas actividades que se definen como diseño, y de la forma en que se concibe en esta metodología. Especificar detalladamente las actividades que debemos realizar para desarrollar bases de datos es un tema no práctico ya que el mismo es todo un proceso iterativo – evolutivo que no solo depende de un buen diseño.

XP + DESARROLLO DE BASES DE DATOS

XP y el desarrollo de Bases de Datos es una parte de XP que no ha sido discutida a profundidad. La administración de bases de datos en conjunto con XP no es un tema frecuentemente tratado. XP se centra en la relación desarrollo – cliente, y por lo que es recomendado que XP sea usado solo con proyectos pequeños, se asume que los propios desarrolladores desempeñen las funcionalidades de Administradores de Bases de Datos.

Existe documentación respecto al uso de XP en proyectos grandes y las vías en que los principios básicos han sido adoptados por grandes equipos de desarrollo. Como también existe documentación respecto a opiniones y criterios de desarrolladores que se han visto afectados con el problema de XP y el desarrollo de bases de datos exponiendo criterios y fórmulas para mitigar un poco esta contradicción por llamarlo de alguna forma.

Es muy común que en proyectos de mediana o gran complejidad exista un equipo para llevar todo el proceso de desarrollo – administración de bases de datos, a diferencia de proyectos desarrollados con metodologías ágiles.

Uno de los principios de XP es la codificación en pareja, o sea en dúos. Según esta metodología todo el código a ser incluido en el sistema es creado por dúos trabajando juntos en una estación de trabajo. Cabe señalar que el desarrollo – administración de bases de datos va mucho más que la codificación de las funcionalidades requeridas, aunque bien se puede codificar en dúos sin mayores problemas en lo que concierne a esta tarea, siempre y cuando sean del mismo equipo de desarrollo de Bases de Datos.

En este sentido, la codificación en dúos es aplicable solo cuando se implementa en la Base de Datos (procedimientos, vistas, triggers.). La codificación en dúos puede no ser conveniente ya que tenemos dos diferentes grupos de habilidades para resolver un problema. Este tipo de programación en dúo necesita de un desarrollador que este familiarizado con la Base de Datos y un Administrador de Bases de Datos que este familiarizado con el lenguaje de programación de la aplicación. Lo fundamental es la comunicación entre los desarrolladores y los/el Administrador/es de Bases de Datos.

Por lo general no existe un equipo para desarrollar – administrar bases de datos dentro del equipo de desarrollo en XP, sino que los propios desarrolladores se valen de sus experiencias en el tema, la cual puede variar, al igual que los conocimientos sobre bases de datos. Esto trae, entre otras cosas, malas decisiones de diseño de bases de datos. La mayoría de las decisiones de diseño son llevadas por los propios desarrolladores (incluyendo el diseño de la Base de Datos) de ahí que muchas de estas no sean lo mejor que pudieran ser o la mejor disponible. Esto es debido al hecho de que no todos los desarrolladores tienes habilidades para el diseño de bases de datos. Y al mismo tiempo pasan por alto muchas cuestiones claves en el marco de la administración de bases de datos, que, más tarde o más temprano, darán al traste con ellas.

DESARROLLO DE BASES DE DATOS POR LA COMUNIDAD DE DESARROLLADORES

Los elementos a los cuales nos referiremos a continuación han sido ya referenciados de una u otra forma. En sentido general presentaremos una síntesis global de los mismos haciendo énfasis en el desarrollo de base de datos por parte de la comunidad de desarrolladores.

No existe tal cosa como una metodología o teoría escrita y probada de desarrollo de bases de datos que nos encamine en cómo llevar el proceso de una forma estándar para mejorar el desarrollo de bases de datos de principio a fin, como mismo no existe una metodología universal para el desarrollo de aplicaciones de software en sentido general dado las diversas variables que intervienen en un Proceso de Desarrollo de Software. Lo que muchos autores definen como metodología de desarrollo de bases de datos no es más que un proceso de desarrollo de software. Algunas cuestiones relacionadas con el desarrollo de bases de datos están vinculadas a los temas principales del desarrollo de software, implementación, despliegue y mantenimiento y actualmente no se conocen términos estándares para el desarrollo de bases de datos.

Uno de los problemas es que los desarrolladores no concuerdan en definiciones para bases de datos. Para algunos es solo almacenamiento. Constituido por tablas, consultas y vistas. Un *"cubo mudo"* para poner y obtener cosas. La administración de bases de datos es la principal tarea de ese cubo.

Para otros es un servidor de aplicaciones. Constituidos por procedimientos que realizan una determinada actividad, como ubicar los datos, pulirlos y entregarlos.

La mayoría concuerdan en que una base de datos puede hacer ambas cosas y se preguntan cual seria el mejor enfoque, y dónde las responsabilidades deberían estar: en el código de la aplicación o en la Base de Datos.

Desarrollo de bases de datos es sinónimo de desarrollo de software, y es por ese motivo que no siempre podemos ver la Base de Datos como un cubo mudo, y es por eso que existen procesos de desarrollo de software que requieren de ingeniería y reingeniería.

Un tema a considerar es que el manejo de bases de datos depende grandemente de las aplicaciones utilizadas y de su estructura. Una Base de Datos existe para soportar una o más aplicaciones de software,

y en este sentido todo profesional de la rama de la informática y del desarrollo de software concuerda en que no tiene sentido el desarrollar e implementar una Base de Datos aislada de aplicaciones de software.

Pudiéramos tomar una metodología de desarrollo y/o implementación y/o mantenimiento de software y aplicar un filtro conceptual: *"muéstrame todos los componentes de esta metodología que se relacionan con los componentes de Base de Datos de la aplicación de software"*.

El resultado puede ser la metodología que buscamos en el contexto de bases de datos. Pudiéramos realizar la misma operación en diferentes metodologías y comparar los contrastes y resultados. Vale destacar que para algunas metodologías esta vista conceptual puede ser nula, fundamentalmente en metodologías agiles.

El equipo de desarrollo comienza un desarrollo de aplicación de software, no un desarrollo de bases de datos. Como el equipo de desarrollo utilice la Base de Datos es una decisión arquitectural la cual el propio equipo tiene que determinar. El trabajo de base de datos es solo una parte de todo esfuerzo del desarrollo de aplicaciones, el cual no deja de ser importante y en ocasiones (por no decir siempre) es decisivo en el triunfo y logro del software.

Las bases de datos son altamente complejas en su propio derecho, y mientras que es bueno que desarrolladores de aplicación (para diferenciarlos del los desarrolladores comunes de Bases de Datos) deberían estar familiarizados con elementos de bases de datos, siempre es bueno tener a alguien con habilidades especializadas en el tema para realizar un *"buen trabajo"*.

Es recomendable de todas formas que cualquier desarrollador tenga conocimientos de administración de sistemas y bases de datos (y viceversa). Este tema tiene gran influencia en equipos pequeños de desarrollo ágil. Pero aun así no deja de ser importante en todo equipo de desarrollo por el aporte que todo el equipo pueda dar.

Un Administrador de Bases de Datos ve una Base de Datos con la necesidad de ser accedida por múltiples aplicaciones, las cuales pueden estar escritas en diferentes lenguajes soportando diferentes niveles de funcionalidad.

Un Administrador de Sistema ve una Base de Datos simplemente como otra aplicación.

Un programador ve la Base de Datos como un sistema de almacenamiento, y como un lugar donde las relaciones entre varias piezas de datos son almacenadas.

En cuanto a habilidades, definitivamente se necesita de un grupo de habilidades especializadas si nos enfrentamos con sistemas de gran rendimiento o con una lógica compleja de administración a nivel de bases de datos. También es importante tener alguien especializado en bases de datos en un proyecto. Al mismo tiempo que es de gran importancia trabajar en conjunto con los desarrolladores y comprender el negocio completo.

Para la mayoría de las bases de datos, casi siempre se necesita de alguien que sepa de elementos específicos como la optimización, algoritmos de unión, índices, parseo de sentencias SQL, bloqueo. Esto es algo que usualmente queda fuera de las habilidades de los desarrolladores de aplicaciones.

Especializarse puede ser de cierta forma una desventaja. Pero en realidad la especialización no es una dificultad. Debe tenerse una fuerte base general de conocimientos y luego por conveniencia se avanza en determinada área de desarrollo, como los es por ejemplo un ingeniero informático especializado en Administración de Datos. No por dedicarse a la Administración de Datos deja de ser un especialista en el área de la informática. No tener todas las habilidades no supone tampoco un impedimento, ya que pueden adquirirse con la experiencia. Un buen desarrollador tampoco debería tener muchos problemas con el desarrollo de bases de datos. El problema con los Administradores de Bases de Datos es que muchas veces después de especializados solo se dedican a este tema.

El desarrollo de aplicaciones a nivel mundial se divide en dos ramas principales: el desarrollo tradicional y robusto, el cual tiende a ser serio y no evolutivo en cuanto a enfoques sobre aspectos orientado a datos, y el desarrollo ágil el cual a dado buenos resultados y abre nuevos horizontes en el mundo del desarrollo de software y sus técnicas son puestas en practica en ambas formas de desarrollar con muy buenos resultados. De cierta forma todo Desarrollador de Base de Datos aplica consiente o inconscientemente técnicas agiles y evolutivas.

Esta nueva forma de desarrollar software, presenta nuevas y significantes demandas en el diseño de bases de datos. Una de las principales es la idea del diseño evolutivo. Es importante recalcar que estas técnicas tampoco resuelven el problema de la evolución de bases de datos, pero en cierta medida mejoran de gran forma el proceso.

En todo desarrollo de software, el equipo de desarrollo debe estar preparado para lidiar con cambios de una forma organizada para que el mismo no se convierta en un caos, sobre todo el de administrar un sistema de datos. Con el fin de hacer que esto funcione, se necesita una actitud diferente en el diseño. En lugar de pensar en el diseño como una fase, que es mayormente completado antes de empezar la construcción, debe verse el diseño como un proceso entrelazado con la construcción, prueba, e incluso la entrega.

Una de las grandes cuestiones es el como hacer que el diseño evolutivo de bases de datos funcione. La mayoría de las personas consideran que el diseño de bases de datos es algo que necesita absolutamente de planificación por adelantado. Cambiar el esquema de base de datos tarde en el desarrollo tiende a causar roturas y difusión de la aplicación de software.

A continuación describiremos técnicas para el desarrollo de bases de datos, no en el marco del desarrollo ágil, pero si partiendo del mismo, para de cierta forma generalizar estas buenas practicas.

Los Administradores de Bases de Datos deben colaborar estrechamente con los desarrolladores

Cada tarea en la que un desarrollador trabaje necesita de la ayuda y colaboración de un Administrador de Bases de Datos. Tanto los desarrolladores como los administradores de bases de datos tienen que considerar si una tarea de desarrollo va a conllevar a un cambio significante en el esquema de la Base de Datos. Si es así, el desarrollador tiene que consultar con el Administrador de Bases de Datos y decidir cómo hacer el cambio. El desarrollador sabe qué nueva funcionalidad se necesita, y el Administrador de Bases de Datos tiene una visión global de los datos en la aplicación. Para que esto suceda el Administrador de Bases de Datos ha de estar accesible y disponible en todo momento para no atrasar el desarrollo del software.

Todos los cambios son refactorizaciones de Base de Datos

La técnica de refactorizar es todo acerca de aplicar técnicas controlada y disciplinadamente a la hora de cambiar un código existente. Similarmente se han identificado numerosas refactorizaciones de bases de

datos que proveen un control y disciplina similar al cambio. Es importante comprender que no se agrega ninguna funcionalidad cuando se refactoriza. Cuando se refactoriza se mejora el código existente, cuando se agrega una funcionalidad se agrega un nuevo código. Se puede sin embargo refactorizar el código existente antes de agregar una nueva funcionalidad, así como refactorizar el nuevo código posteriormente. El punto es que refactorizar y agregar nuevas funcionalidades son dos tareas diferentes pero complementarias.

La refactorización de Base de Datos no es más que pequeños cambios al esquema de datos que mejora el diseño al mismo tiempo que preserva el comportamiento y semántica.

Una de las grandes diferencias en torno a la refactorización de Bases de Datos es que involucra tres diferentes cambios los cuales deben realizarse en conjunto (siempre y cuando se requiera):

- Cambiar el esquema de datos.
- Migrar los datos a la Base de Datos.
- Cambiar el código de acceso a la Base de Datos.

Muchas refactorizaciones de Bases de Datos, como agregar una columna, puede realizarse sin tener que actualizar todo el código que el sistema accede. Si el código utiliza el nuevo esquema sin estar consiente de él, la columna simplemente será inutilizable. Estos cambios son conocidos como *"cambios destructivos"*. Lo importante aquí es elegir el procedimiento más apropiado para el tipo de cambio que se esta realizando.

Automatizar las refactorizaciones

En el mundo de la codificación muestra herramientas para diferentes lenguajes para automatizar muchas refactorizaciones con las cuales pudiéramos tratar. Esas automatizaciones son validas para bases de datos; o al menos en áreas de cambios de esquema y migración de datos.

Como resultado, cada refactorización de bases de datos es automatizada en la forma de sentencias DDL[14] y DML[15]. Estos cambios nunca son aplicados de forma manual, sino que son aplicados ejecutando pequeños scripts de sentencias SQL[16] para realizar los cambios.

También es de gran utilidad tener un registro histórico de los cambios significantes en el esquema de la Base de Datos para una posterior ayuda de toma de decisiones en casos de temas de optimización y mantenimiento.

Esta habilidad de secuencia de cambios de manera automatizada es una herramienta esencial tanto para el proceso de integración continua en el desarrollo como para la migración de bases de datos de producción a una nueva versión. De la misma forma que se realizan cambios, se pueden aplicar estas mismas técnicas de refactorización para revertir los mismos.

Separar claramente todo el código de acceso a la Base de Datos

Importante también es tener una capa de acceso a la Base de Datos lo más clara posible en dependencia de las características del sistema así como de los requerimientos funcionales y no funcionales, para determinar dónde y cómo esta siendo accedida la Base de Datos. Para ello es bueno apoyarse en los diferentes patrones de diseño existentes, sin perder de vista claro esta, las especificaciones arquitectónicas del sistema. Tener una capa de acceso a la Base de Datos lo más clara posible, tiene una serie de numerosos beneficios. Minimiza las áreas del sistema donde los desarrolladores necesitan conocimiento de sentencias SQL para manipular la Base de Datos, lo cual le hace la vida más fácil a desarrolladores los cuales por lo general no están familiarizados con este tipo de sentencias. Para el Administrador de Bases de Datos provee una sección clara del código el cual puede revisar y ver cómo la Base de Datos esta siendo accedida. Esto ayuda a preparar índices, optimizaciones, e incluso ver como se puede reformular determinadas sentencias SQL para un mejor rendimiento. En sentido general, esto

[14] **DDL**: Data Definition Language. Lenguaje de Definición de Datos, por su traducción.

[15] **DML**: Data Manipulation Language. Lenguaje de Manipulación de Datos, por su traducción.

[16] **SQL**: Structured Query Language. Lenguaje Estructurado de Consultas, por su traducción.

permite a los Administradores de Bases de Datos tener un mejor entendimiento de cómo es utilizada la Base de Datos.

Como cualquier conjunto de prácticas, estas deben variar dependiendo de las circunstancias específicas del desarrollo de software. Si se determina automatizar el mayor conjunto de tareas, se puede manejar mucho más trabajo con menos personas involucradas.

DESARROLLO DE BASES DE DATOS EN LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

El desarrollo de bases de datos en la Universidad de las Ciencias Informáticas no presenta cambios significativos en la forma de desarrollar y organizar el trabajo respecto a bases de datos. Según la encuesta del Anexo 1: Encuesta sobre el proceso de desarrollo – administración de bases de datos en la Universidad de las Ciencias Informáticas, el desarrollo de bases de datos parte de la personalización de todos los elementos organizativos – estructurales según la experiencia del responsable de la Base de Datos de un proyecto de software determinado. Los roles se definen según el grado de división lógica que queramos darle a las actividades de desarrollo y mas que definir artefactos desde un inicio, se trabaja en artefactos generales como por ejemplo el Modelo de Datos, el acceso a datos.

Se desarrolla a medida de los requerimientos y enfocado en determinada metodología, y es usual la existencia de un equipo especializado en temas de bases de datos que desarrolle en este sentido a la par de los desarrolladores de la aplicación.

La nomenclatura utilizada es diferente entre proyectos de software y por lo general existe coherencia en la convención de nombres utilizada por cada proyecto de software en específico.

CAPÍTULO 2: PROCESO GENERAL DE DESARROLLO – ADMINISTRACIÓN DE BASES DE DATOS: PROPUESTA DE MODELO DE DESARROLLO DE BASES DE DATOS PARA PROCESOS DE DESARROLLO DE SOFTWARE

En este capitulo se detalla el modelo propuesto para guiar el desarrollo de bases de datos en procesos de desarrollo de software, así como elementos indispensables relacionados con el desarrollo – administración de bases de datos como el Modelo de Datos, las configuraciones del Sistema Gestor de Base de Datos, la implementación de funcionalidades y el Acceso a Datos.

ELEMENTOS INDISPENSABLES A TENER EN CUENTA EN CUANTO AL DESARROLLO DE BASES DE DATOS

MODELO DE DATOS

El Modelo de Datos constituye el principal elemento de diseño de bases de datos, y es comparable con los planos de una edificación, ya que es lo primero que se realiza y se toma como guía a la hora de ejecutar la obra constructiva.

Un Modelo de Datos no es más que la representación de un fenómeno de la realidad objetiva a través de los objetos, sus propiedades y las relaciones que se establecen entre ellos.

Un Modelo de Datos consiste en:

1. Objetos (entidades que existen y que se manipulan).
2. Atributos (características básicas de estos objetos).
3. Relaciones (forma en que se enlazan los distintos objetos entre si).

La metodología para el diseño de bases de datos, consta de los siguientes pasos:

- Determinación de entidades y atributos.
- Normalización de entidades.
- Determinación de relaciones.
- Obtención del modelo lógico global de los datos.
- Diseño físico de la Base de Datos.

Dentro del Modelo de Datos existen dos niveles de diseño:

- Lógico.
- Físico.

El Diseño Lógico parte del Esquema Conceptual y da como resultado un Esquema Lógico. El Diseño Conceptual parte de las especificaciones de requisitos de usuario y su resultado es el Esquema Conceptual de la Base de Datos. El objetivo del diseño conceptual es describir el contenido de información de la Base de Datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información.

En el Diseño Lógico deben describirse todos los elementos que forman la Base de Datos y debe condicionar al Diseño Físico.

El Diseño Físico depende fundamentalmente del Sistema Gestor de Base de Datos que se utilice y se expresa mediante su Lenguaje de Definición de Datos.

El Diseño Lógico y Físico se encuentran generalmente ligados a la arquitectura de la Base de Datos y especifican muchos detalles de implementación.

El Diseño Lógico corresponde con la fase de elaboración de los planos arquitectónicos, y la implementación física de la Base de Datos es la casa ya construida. El Diseño Lógico describe el tamaño, la forma y los sistemas necesarios para la Base de Datos: contiene las necesidades en cuanto a información y modo de operación del negocio. Después se construye la implementación física del Diseño Lógico de la Base de Datos mediante el Sistema Gestor de Base de Datos. Una Base de Datos bien diseñada contendrá información correcta, almacenará los datos más eficientemente y será más fácil de gestionar y de mantener.

Roger Pressman plantea en su libro, *"Ingeniería de Software, un enfoque práctico"*, que es recomendable desarrollar el Modelo de Datos de forma iterativa para así ir refinando los objetos, atributos y las relaciones que conectan los objetos.

Cabe señalar también que el Modelo de Datos forma parte importante y es punto de partida de la documentación de la estructura física de la Base de Datos.

NOMENCLATURA

Los nombres son la primera y más importante línea de documentación para una aplicación de software, y mucho más para las bases de datos. Tratar en detalle como sería la mejor forma de nombrar las cosas es un tema un poco engorroso y no práctico, el tema importante radica en la necesidad de *"coherencia"* a la hora de determinar una nomenclatura para nuestros objetos de desarrollo. Los nombres que se elijan no son sólo para permitir identificar el propósito de un objeto, sino de permitir que todos los futuros programadores, usuarios, comprendan de una forma rápida y fácil cómo una parte componente de la Base de Datos fue concebida a utilizarse; en el caso de una tabla, por ejemplo, que información debe almacenar, y en el caso de un procedimiento almacenado, a que funcionalidad da respuesta. También es importante tener una regla de nomenclatura estándar en el equipo de desarrollo para tener una consistencia al respecto, y para que los nuevos integrantes o sustitutos en el equipo mantengan la misma línea de nomenclatura.

Buenas Prácticas

Objetos de Base de Datos

- Limite el nombre a 30 caracteres. Más corto es mejor.
- Evite números.
- Trate de utilizar guiones bajos lo menos posible. Utilice la notación Pascal.
- Utilice una letra como primer caracter del nombre. No comience con guiones bajos.
- Evite abreviaciones. Puede malinterpretarse el nombre.
- Evite acrónimos. Algunos acrónimos tienen más de un significado.
- Evite espacios en nombres incluso si el sistema lo permite.
- Utilice prefijos y sufijos para enriquecer y describir la funcionalidad de los elementos estructurales y objetos de la Base de Datos.
- Para el Modelo Físico de la Base de Datos utilice solo nombres en mayúsculas.

Tablas

Cuando asigne nombres a las tablas de la Base de Datos, tenga en consideración otros pasos en el proceso de desarrollo. Tenga en cuenta que tendrá que utilizar el nombre de las tablas varias veces como parte de otros objetos, por ejemplo, procedimientos almacenados, triggers, o vistas suelen contener referencias al nombre de la tabla. Debe tenerse un nombre tan simple como corto posible.

Usados correctamente, los prefijos y sufijos en la nomenclatura de tablas pueden ayudar a organizar las mismas dentro de grupos relacionados o distinguirlas de otras tablas no relacionadas.

Para todas las partes del nombre de la tabla, incluyendo los prefijos y sufijos, utilice el estilo Pascal. Por ejemplo: *"NTipoPersonaMer"*. El estilo Pascal reduce la necesidad de guiones bajos para separar visualmente las palabras.

Los guiones bajos no deben usarse en nombres de tablas. Este caracter tiene lugar en otros nombres de objetos, pero no en nombre de tablas.

No utilice números en los nombres de tablas. Esto apunta hacia un mal diseño de datos.

No utilice acrónimos.

Evite el uso de abreviaciones en la medida de lo posible.

Nombre las tablas provenientes de una relación de muchos a muchos concatenando los nombres provenientes de las tablas con relación de uno a muchos. Por ejemplo la tabla *"Paciente"* relacionada de muchos a muchos con *"Consulta"* daría como resultado una nueva tabla relacionada de uno a muchos con las tablas mencionadas la cual deberíamos nombrar *"PacienteConsulta"*. Suele ser valido el uso de abreviaciones debido a la longitud que puedan tener algunos nombres.

Columnas

Cuando se nombren columnas, tenga presente que las mismas son miembros de la tabla, por tanto no hay necesidad de mencionar el nombre de la tabla en el nombre de la columna. El campo llave primaria es la única excepción a esta regla donde incluir el nombre esta justificado para tener un campo más descriptivo. Por ejemplo *"IdPersona"* en ves de *"Id"* como nombre del campo llave de una tabla llamada *"Persona"*. Al igual que el nombre de las tablas, evite el uso de abreviaturas, acrónimos o caracteres especiales. Todas las columnas deben escribirse en estilo Pascal.

Las llaves foráneas deben tener el mismo nombre que aparece en la tabla a la que hace referencia donde son llave primaria, y la relación correspondiente entre tablas, debe nombrarse con los nombres de ambas tablas.

Procedimientos almacenados

A diferencia de muchos otros objetos de Base de Datos, los procedimientos almacenados no están lógicamente ligados a ninguna tabla o columna. Los procedimientos almacenados realizan operaciones del tipo CRUD (create, read, update, delete) o la combinación de estas para describir operaciones transaccionales de negocio. Lo más aconsejable es nombrarlos según el tipo de funcionalidad a la que responden utilizando un verbo para describir el tipo de operación. El uso de prefijos y sufijos suele ser una buena práctica para agruparlos y/o diferenciarlos.

Triggers

Los triggers tienen muchas cosas en común con los procedimientos almacenados. Sin embargo son diferentes a estos en dos sentidos importantes. En primer lugar, no existen triggers por su propia cuenta. Ellos dependen de una tabla. Por lo tanto, es conveniente incluir el nombre de la tabla en el nombre del trigger. En segundo lugar, sólo se pueden ejecutar cuando un insertar, actualizar, o eliminar sucede en los registros de una tabla. Así que también tiene sentido incluir el tipo de acción que va a causar que el trigger se ejecute.

Para distinguir los triggers de otros objetos de la Base de Datos, es recomendable agregar *"Trg"* como prefijo o sufijo. Por ejemplo: *"TrgProductosInst".* Basta con que incluya el nombre de la tabla, la operación como sufijo y el prefijo *"Trg"*, para que a un desarrollador que trabaje en la Base de Datos le sea obvio con que tipo de objeto este trabajando y la funcionalidad que realiza. Si un trigger maneja más de una operación, insertar y eliminar por ejemplo, entonces incluya ambas operaciones en el nombre en forma de abreviaturas. Por ejemplo: *"TrgProductosInstUpd"*.

Estas son solo algunas consideraciones en cuanto a como nombrar los elementos de bases de datos. Queda como responsabilidad del equipo de desarrollo definir una nomenclatura de que sea más factible para el desarrollo.

Elija un nomenclatura y manténgala consistente.

CONFIGURACIONES SISTEMA OPERATIVO – SISTEMA GESTOR DE BASES DE DATOS

Las configuraciones tanto del Sistema Operativo como del Sistema Gestor de Base de Datos son imprescindibles para un correcto funcionamiento y puesta en marcha de la Base de Datos. La Base de Datos se configura en la medida de las necesidades y del entorno de su funcionamiento, soportada por la arquitectura del sistema. El Sistema Operativo es el que sustenta toda la operatividad del Sistema Gestor de Base de Datos. Algunas configuraciones suelen ser tan sencillas como variables de entorno, y otras un poco más complejas como pudieran ser servicios asociados a la Base de Datos.

Los tipos de configuraciones tiene relación directa según los entornos de desarrollo de aplicaciones como son: desarrollo, prueba y el sistema ya desplegado o en transición de despliegue, y del tipo de Base de Datos que estemos desarrollando; aunque suelen ser idénticas en ocasiones. Lo más importante respecto a las configuraciones es tener bien documentado y justificado todo el sistema de configuraciones, ya que así podemos llevar a cabo las tareas de administración, mantenimiento e instalación de una forma correcta y optima, logrando también poder incorporar nuevos integrantes al equipo de desarrollo adaptándose de una forma más sencilla al proceso de configuración.

SCRIPTS

Los scripts de Base de Datos no son más que sentencias SQL, que ejecutan operaciones DDL y DML, organizadas en ficheros que se ejecutan de manera organizada para realizar una determinada operación en la Base de Datos, los cuales son de gran utilidad ya que automatizan la forma de realizar diferentes acciones en la Base de Datos que suelen ser repetitivas y mayormente costosas, al mismo tiempo que podemos guardar determinada información en ficheros más manejables debido al tamaño, como puede ser la información de una tabla *“Clientes”*.

Respecto a las operaciones nos referimos a todo tipo de acciones sobre la Base de Datos: creación de tablas, procedimientos almacenados, compilación de objetos, inserción de tulpas, modificaciones. El proceso más costoso pudiera ser el crear estos scripts. Hoy día existen herramientas para generar todo tipo de scripts para Base de Datos, pero aun así, para scripts más avanzados se requiere de un minucioso trabajo manual. Los scripts complementan las configuraciones del la Base de Datos, ya que, como se

refería anteriormente, automatizan el proceso de instalación, implementación, configuración y actualización de la arquitectura de datos. Muchas de las operaciones repetitivas de implementación, administración y mantenimiento de la Base de Datos suelen ejecutarse mediante scripts previamente preparados, y es importante en este sentido automatizar la mayor cantidad de operaciones, siempre y cuando sea necesario y pertinente.

IMPLEMENTACIÓN

La implementación no es más que la codificación de las funcionalidades requeridas por el sistema, en este caso en la Base de Datos. Nos referimos a la codificación de objetos de Base de Datos que responden a ciertas acciones a ejecutarse de forma manual o automática: procedimientos almacenados, triggers, jobs, funciones. Es muy importante en este punto mantener una nomenclatura consistente y seguir un estándar a la hora de codificar. Suele suceder que tenemos en un sistema funcionalidades escritas de varias formas, elemento el la cual debemos de evitar ya que esto incurre en una mala codificación y diseño, así como dificulta las tareas de documentación, optimización, administración.

La implementación de objetos en la Base de Datos responde a una necesidad de determinada funcionalidad en el sistema. El desarrollador de la aplicación solicita al desarrollador de Base de Datos una determinada funcionalidad y corresponde al equipo de Base de Datos el darle respuesta implementado uno o varios objetos como una operación transaccional.

Es importante tener identificado qué objetos de la Base de Datos responden a una o varias funcionalidades y a qué proceso de negocio dan respuesta, ya que así podemos tener una buena documentación de los procesos que se llevan a cabo en la Base de Datos y mejoramos las tareas de administración. Suele también suceder que tenemos un procedimiento almacenado el cual ya no es utilizado porque se reemplazó por otro y aun existe en la Base de Datos, lo cual no tiene gran repercusión pero el mismo no debería aun existir y al incurrir en este tema tendríamos código basura en la Base de Datos.

Importante también es tener bien documentado estos elementos y una buena comunicación entre todo el equipo de desarrollo para no repetir código, y al mismo tiempo propiciar la reutilización de código para

agilizar el desarrollo y dar robustez a la arquitectura del sistema. Por lo que es necesario realizar una buena gestión de la Base de Datos y para eso se necesita tener una organización de manera tal que existan personas responsables de las bases de datos además del desarrollador que es el primero que tiene que velar por esas cosas.

TUNING

La optimización de bases de datos describe un grupo de actividades con el fin de mejorar y homogeneizar el rendimiento. La optimización solapa temas como la optimización de consultas, configuración de ficheros de bases de datos, el Sistema Gestor de Base de Datos, el Sistema Operativo y el Hardware. Estos elementos pueden dividirse en dos grandes grupos: Optimización de Entrada/Salida (E/S) y Optimización del Sistema Gestor de Base de Datos. La optimización de E/S no es más que mejorar y balancear lecturas y escrituras a la Base de Datos. Las operaciones de E/S son generalmente las más costosas en el trabajo con bases de datos, y son típicamente el primer cuello de botella[17] encontrado. Los registros de transacciones y espacios temporales son grandes consumidores de operaciones E/S, y afectan el funcionamiento para todos los usuarios de la Base de Datos. Manejarlos apropiadamente es crucial. La optimización del Sistema Gestor de Base de Datos se refiere a la configuración de memoria y procesamiento de recursos del equipo que sustente el mismo. Esto se hace a través de la configuración del Sistema Gestor de Base de Datos, pero al mismo tiempo los recursos son compartidos con el sistema servidor, como por ejemplo la memoria física y virtual, porciento de CPU[18] utilizado entre otros.

El objetivo es maximizar el uso de los recursos del sistema para realizar un trabajo de la forma más eficiente y rápida posible. La mayoría de los sistemas se han diseñado para gestionar de manera eficiente el trabajo, pero es posible mejorar en gran medida el rendimiento mediante la configuración y la personalización de la Base de Datos y la optimización del Sistema Gestor de Base de Datos.

Un Administrador de Bases de datos es responsable por el rendimiento de una Base de Datos. Optimizar la misma para alcanzar un nivel de rendimiento deseado puede ser una tarea de enormes proporciones.

[17] **Cuello de Botella**: se refiere a elementos que atentan contra el rendimiento del sistema.

[18] **CPU**: Unidad Central de Procesamiento por su traducción.

En desarrollos de bases de datos de pequeña a mediana envergadura, una solo persona pudiera encargarse de realizar las tareas de optimización, mientras que para entornos de mayores niveles este trabajo suele dividirse entre diferentes administradores de bases de datos. Los problemas más comunes en sistemas en producción son referentes al rendimiento de las aplicaciones en general, y la mayoría son elementos que pudieron haberse previsto desde el inicio del desarrollo.

La optimización es un proceso iterativo. Eliminar el primer cuello de botella podría no conducir a una mejora inmediata, debido a que otro cuello de botella podría tener un determinado impacto en el rendimiento del sistema. Por esta razón este es un proceso iterativo. Diagnosticar con precisión el problema de rendimiento es el primer paso para garantizar que los cambios que se realicen en el sistema se traducirán en una mejora.

Antes de buscar en cualquier estadística de Base de Datos o Sistema Operativo, es fundamental obtener retroalimentación de los componentes más importantes del sistema: los usuarios del sistema y los clientes que pagan por la aplicación. Obtener retroalimentación de los usuarios facilita la determinación de los objetivos a optimizar, y este rendimiento podrá medirse en términos de objetivos de negocio reales en lugar de estadísticas del sistema.

ACCESO A DATOS

El acceso a datos se refiere a los programas informáticos y actividades relacionadas con el almacenamiento, recuperación, que actúan sobre los datos en una Base de Datos u otros repositorios.

Existen leguajes estándares, métodos y formatos que sirven como interfaces entre lenguajes específicos y la Base de Datos. Tales como SQL, ODBC, JDBC, ADO.NET, XML, XQuery, XPath, y Servicios Web.

Una Capa de Acceso a Datos es una capa de un programa de ordenador que proporciona un acceso simplificado a datos almacenados, como por ejemplo a una entidad de base de datos relacional. Esta Capa de Acceso a Datos se utiliza a su vez por otros módulos del programa para acceder y manipular los datos en el almacén de datos sin tener que hacer frente a las complejidades inherentes a este acceso. Por ejemplo, en lugar de utilizar comandos, como insertar, eliminar, actualizar para acceder a una tabla específica en una base de datos, se crearía una clase y unos procedimientos almacenados en la Base de

Datos. Los procedimientos se llamarán a partir de un método dentro de la clase, que devuelve un objeto que contiene los valores solicitados.

Esta capa sirve al mismo tiempo como puente entre la capa lógica de negocio y el proveedor de datos. De esta forma se tiene una separación tanto lógica como física del negocio, y de los datos, de manera tal que el negocio no tiene por que conocer la estructura de la Base de Datos, sino que la capa de acceso es la encargada de esto y que en caso de un cambio de estructura el negocio no es afectado.

La capa de datos:

- No guarda el estado.
- Sus parámetros deben de ser entidades de negocio (clases que si tiene el valor de cada propiedad, ejemplo: Cliente, Factura, Nomina.).
- Realiza operaciones elementales en el proveedor de los datos.
- No debe de existir ninguna validación ni operaciones complejas en esta capa solo las pertinentes para operar con el proveedor de los datos, como es el manejo de los nulos en los parámetros y en los datos que se reciben del proveedor.

Patrón de Acceso a Datos (DAO)

Data Access Object (DAO, Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de Datos o un archivo de datos.

El patrón DAO es una solución al problema del diferencial de impedancia entre un programa de aplicación orientado a objetos y una Base de Datos relacional, empleando únicamente la Interfaz de Programación (API) nativa del manejador de Base de Datos, o algún otro sustituto como el ODBC, DBI entre otros.

Utilizamos el patrón DAO para:

- Abstraer y encapsular los accesos.

- Gestionar las conexiones a la fuente de datos.
- Obtener los datos almacenados.

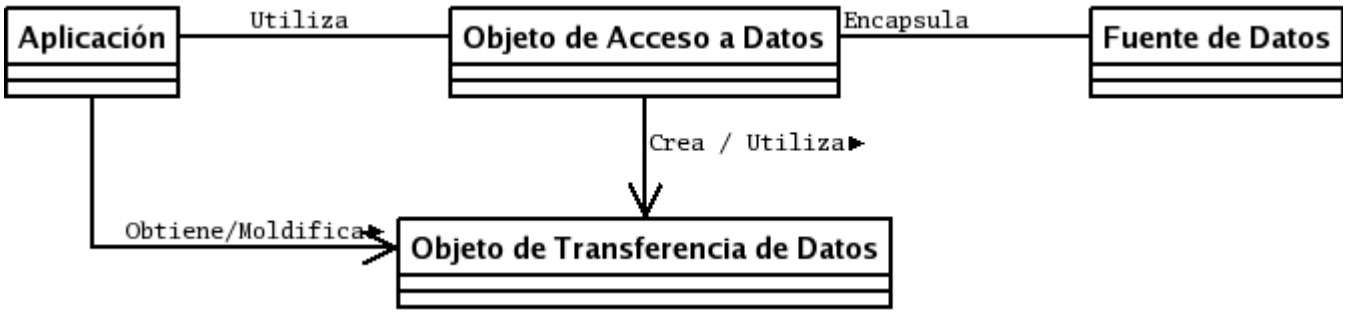

Figura 11: Patrón de Acceso a Datos.

Ventajas

- Cualquier objeto no requiere conocimiento directo del destino final de la información que se manipula.
- Se baja el nivel de acoplamiento entre clases, reduciendo la complejidad de realizar cambios.
- Se aíslan las conexiones a la fuente de datos en una capa fácilmente identificable y mantenible.

ROLES INVOLUCRADOS EN EL PROCESO

Los roles definen un grupo de responsabilidades en el trabajo con Bases de datos. Existen dos roles principales: el Administrador de Bases de datos el cual se encarga de elementos generales de administración de Bases de datos y el Desarrollador de Bases de Datos el cual se encarga de elementos más específicos de diseño e implementación; los cuales tienen tareas especificas y al mismo tiempo comparten determinadas actividades comunes como la toma de decisiones. En ocasiones solo existe un Administrador de Bases de Datos en un grupo de desarrollo de software, que realiza todas las tareas

relacionadas con la Base de Datos. La existencia de uno o varios roles en el desarrollo de bases de datos depende de la complejidad del software, metodología utilizada, y fundamentalmente del nivel de responsabilidad que se quiera especificar entre los miembros del equipo de desarrollo.

Administrador de Bases de Datos

Cada Base de Datos requiere de por lo menos un Administrador de Bases de Datos. La administración de bases de datos en ocasiones no es trabajo para una sola persona, sino de varios administradores que comparten las responsabilidades.

Entre las responsabilidades más importantes de un Administrador de Bases de Datos se encuentran:

- Instalación y actualización del servidor de Base de Datos.
- Asignación de almacenamiento del sistema y planificación de futuros requerimientos de almacenamiento para el sistema de bases de datos.
- Creación de estructuras de almacenamiento de Bases de Datos.
- Creación de objetos (tablas, vistas, índices).
- Modificación de la estructura de la Base de Datos, en caso necesario, a partir de la información dada por los desarrolladores de la aplicación.
- Creación de usuarios y mantener la seguridad del sistema.
- Controlar y monitorear el acceso de los usuarios a la Base de Datos.
- Monitorizar y optimizar el rendimiento de la Base de Datos.
- Planificar las copias de seguridad y la recuperación de la información.

Desarrollador de Base de Datos

Los Desarrolladores de Bases de Datos diseñan e implementan la Base de Datos. Sus responsabilidades incluyen las siguientes tareas:

- Diseñar y desarrollar la Base de Datos.
- Diseñar las estructuras de la Base de Datos.
- Estimar requerimientos de almacenamiento.
- Especificar modificaciones de la estructura de la Base de Datos.
- Trabajar en conjunto con el Administrador de Bases de Datos.
- Optimizar las operaciones de E/S referentes a la implementación y desarrollo.
- Establecer medidas de seguridad durante el desarrollo.

Los Desarrolladores de Bases de Datos pueden realizar algunas de las operaciones en colaboración con los Administradores de Bases de Datos.

Importante es el trabajo en conjunto de todo aquel involucrado en el diseño, desarrollo y administración de bases de datos de manera unificada, ya que las tareas no deben verse aisladas porque todas tributan a un buen desarrollo de Base de Datos y manejan aspectos comunes como por ejemplo la optimización.

HERRAMIENTAS DE AUTOMATIZACIÓN DEL PROCESO

Las herramientas constituyen el soporte automático o semiautomático para procesos y métodos en la Ingeniería de Software. Las herramientas son otros productos de software que nos facilitan y agilizan en gran medida la realización de diversas tareas en el desarrollo de software. De aquí su importancia.

Existen disimiles herramientas en el desarrollo de software para realizar diferentes tareas. Como también existen paquetes de herramientas para un grupo completo de actividades relacionadas. Las herramientas pueden clasificarse según el área de impacto como por ejemplo:

- Gestión de Software.
- Ingeniería de Software.
- Desarrollo de Software.
- Soporte.

Este tipo de herramientas se conoce como herramientas CASE (Computer Aided Software Engineering) las cuales son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Estas herramientas:

- Mejoran la productividad en el desarrollo y mantenimiento del software.
- Aumentan la calidad del software.
- Mejoran el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejoran la planificación de un proyecto.
- Automatizan el desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.
- Ayudan a la reutilización del software, portabilidad y estandarización de la documentación.

En el desarrollo de Base de Datos encontramos herramientas para modelar datos, generar código en la Base de Datos, generar código de acceso a datos, generar scripts de datos. Importante es utilizar al máximo estas herramientas y automatizar todo lo que pueda ser automatizado para aumentar el rendimiento y la productividad.

EL PROCESO

El proceso esta divido en cuatro actividades fundamentales las cuales incluyen tareas generales. Las actividades son: Modelado de Datos, Configuraciones, Diseño – Implementación y una cuarta actividad nombrada ADTP debido a la las tareas generales que se realizan en esta actividad (Acceso a Datos, Tuning, Prueba). Durante las actividades se generan una serie de artefactos básicos e indispensables y existen tres roles fundamentales que realizan diferentes tareas individuales y en común: el Arquitecto de Bases de Datos, el Administrador de Bases de Datos y el Desarrollador de Bases de Datos. El proceso transita según las fases genéricas[19] del ciclo de vida del desarrollo de software enfocado en mayor o en menor medida en determinada actividad según el momento del desarrollo del software. La primera fase es la fase de definición, la segunda fase es la fase de desarrollo junto a la fase de mantenimiento y la tercera fase es la fase de transición la cual se refiere a la etapa de transito del software al entorno real de ejecución, la cual implica pruebas finales de aceptación por parte de los clientes y usuarios del sistema. La siguiente imagen describe gráficamente como transitan las diferentes actividades del proceso, de manera aproximada, durante todo el ciclo de vida del desarrollo de software.

[19] **Fases Genéricas:** Ver Capítulo 1: Fundamentación Teórica, Ingeniería de Software.

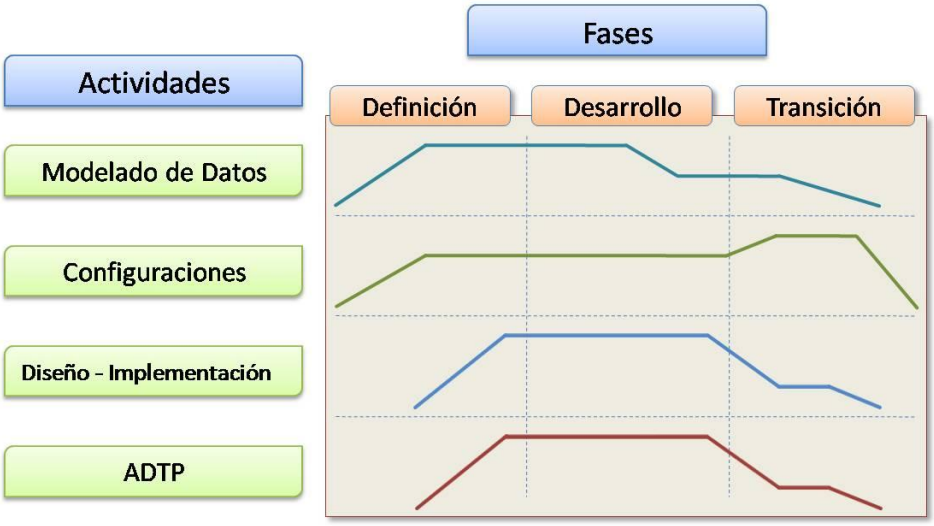

Figura 12: Descripción gráfica del Proceso General de Desarrollo – Administración de Bases de Datos.

Al término de la transición de la fase de definición a la fase de desarrollo contamos con una versión inicial de la arquitectura de la Base de Datos la cual se caracteriza por una estructura de datos principal y configuraciones iniciales que soportarán el futuro desarrollo, así como la implementación de las funcionalidades básicas esenciales. Al inicio de la fase de desarrollo tendremos en marcha los procesos fundamentales del desarrollo de la Base de Datos.

En el transito de la fase de desarrollo a la fase de transición, tenemos una versión avanzada – funcional – estable de la Base de Datos, la cual tiene como características una arquitectura de datos robusta casi definitiva (aproximadamente más de un 95% de la estructura definitiva), al mismo tiempo que las configuraciones y la implementación. Al inicio de la fase de transición contamos con un alto grado de terminación e integración de las funcionalidades entre el negocio del sistema y la estructura de la Base de Datos.

La fase de transición se caracteriza por las pruebas definitivas de aceptación por parte de los clientes y usuarios del sistema y la puesta en marcha del mismo en su entorno de ejecución real. Esta fase esta encaminada a completar la versión final del producto y todos sus artefactos.

Se define como artefacto general del proceso, en forma de plantilla[20], la nomenclatura de todos los objetos de la Base de Datos y sus partes.

Figura 13: Artefacto general del proceso.

LAS ACTIVIDADES

Las actividades se definen según tareas generales relacionadas y el rol responsable en dirigir, organizar y ejecutar una actividad determinada, siendo la interrelación entre las mismas el ciclo de vida y núcleo principal del modelo propuesto. Las actividades de Diseño – Implementación y ADTP suelen estar muy relacionadas por las tareas que definen y las mismas tienen dependencia directa de la actividad Modelado de Datos, y de cierta forma todas dependen de la actividad Configuraciones[21]. Las actividades no se desarrollan en paralelo, sino que dependiendo de la fase de desarrollo interactúan según determinado elemento de desarrollo que ejecutemos. En la fase de definición el mayor esfuerzo esta en desarrollar la primera estructura de la Base de Datos (Modelado de Datos) y a su vez las configuraciones iniciales (Configuraciones), lo cual tributa a los primeros pasos del desarrollo (Diseño – Implementación, ADTP). En la fase de desarrollo suele ocupar el mayor tiempo las tareas de implementación y acceso a datos

[20] **Ver anexos**: Plantilla General Nomenclatura de Base de Datos.

[21] **Ver Figura 25**: Relación entre actividades.

(Diseño – Implementación, ADTP), con determinados cambios en el modelo de datos (Modelado de Datos) y en la fase de transición suele ocupar la mayor importancia las configuraciones para la puesta en marcha del software a su entorno real de ejecución (Configuraciones).

MODELADO DE DATOS

La actividad tiene como tareas generales la estructuración del Modelo de Datos y su respectiva documentación, la cual tributa a la documentación general de la arquitectura de la Base de Datos.

Los objetivos de esta actividad son:

1. Estructurar iterativamente el Modelo de Datos.
2. Documentar la estructura de la Base de Datos a la par del desarrollo del Modelo de Datos.

Para dar cumplimiento al objetivo No. 1, se debe analizar el impacto que trae aparejado un cambio determinado y necesario dado los requerimientos de negocio del sistema, en la estructura total de la Base de Datos, y realizar los cambios pertinentes que garanticen la continua robustez del Modelo de Datos de forma evolutiva.

Para dar cumplimiento al objetivo No. 2, se debe ir refinando la documentación asociada a la estructura de la Base de Datos a medida que evoluciona el Modelo de Datos. No es imprescindible realizar paralelamente las tareas de documentación y cambios del modelo, pero si necesario antes del próximo cambio estructural del modelo considerado como una nueva iteración.

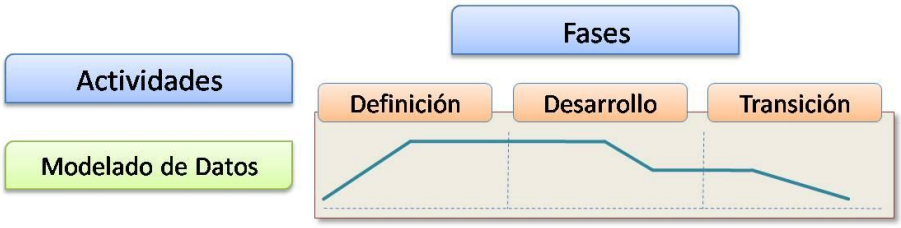

Figura 14: Descripción gráfica aproximada de la actividad Modelado de Datos.

Artefactos

Los artefactos que se generan en esta actividad son:

1. El Modelo de Datos, el cual se compone del Modelo Lógico y el Modelo Físico.
2. Documentación de estructura de la Base de Datos.

Figura 15: Artefactos de la actividad Modelado de Datos.

CONFIGURACIONES

La actividad tiene como tareas generales el establecimiento de las configuraciones de diferentes entornos de desarrollo, así como la estructuración de una estrategia de instalación, montaje y configuración de la arquitectura de la Base de Datos para diferentes entornos de desarrollo.

Los objetivos de esta actividad son:

1. Definir las configuraciones del Sistema Operativo – Sistema Gestor de Bases de Datos – Base de Datos para diferentes entornos de desarrollo.
2. Aplicar las correspondientes configuraciones en el momento que se requiera, mediante automatizaciones en caso de ser posible.

Para dar cumplimiento al objetivo No. 1, se debe tener por escrito y agrupado por entorno de desarrollo en forma de plantilla[22], todas las configuraciones del Sistema Operativo – Sistema Gestor de Bases de Datos – Base de Datos. Inicialmente podemos no tener todas las configuraciones que se requieran. A medida que cambien los requerimientos y las configuraciones, se debe actualizar dichos documentos.

Para dar cumplimiento al objetivo No. 2, se debe conformar una estructura, a partir de elementos de automatización, en la medida de lo posible, para ejecutar las tareas de instalación, montaje y configuración. En el caso de la instalación – configuración del Sistema Operativo son muy útiles los archivos por lotes MS-DOS[23] (*.bat) para automatizar este tipo de acciones y para la instalación – configuración de la Base de Datos es de gran utilidad definir una estructura de scripts, no siendo imprescindible la actualización continua de las sentencias DML y DDL de los mismos, sino en el momento que se requiera, mediante automatizaciones generar la misma. Un ejemplo de estructura y su orden de ejecución puede ser:

1. Roles.
2. Usuarios.
    a. Permisos.
3. Tablas nomencladoras.

[22] **Ver Anexo 2**: Plantilla General Nomenclatura de Base de Datos..

[23] **Archivo por lotes (*.bat)**: Archivo MS-DOS que ejecuta comandos del Sistema Operativo de forma automática y secuencial.

   a. Datos.

4. Tablas de datos.
5. Objetos.
   a. Procedimientos almacenados.
   b. Triggers.
6. Otros.
   a. Datos de tablas de datos que lo requieran.

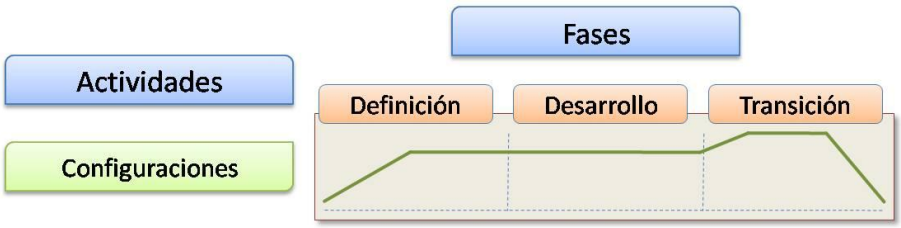

Figura 16: Descripción gráfica aproximada de la actividad Configuraciones.

Artefactos

Los artefactos que se generan en esta actividad son:

1. Documento de configuraciones del Sistema Operativo – Sistema Gestor de Bases de Datos – Base de Datos, agrupados por entorno de desarrollo.
2. Estructura de aplicación de las configuraciones.

Figura 17: Artefactos de la actividad Configuraciones.

DISEÑO – IMPLEMENTACIÓN

Esta actividad se centra en la implementación de todas las funcionalidades requeridas en la Base de Datos que correspondan a las necesidades del sistema. La misma se divide en dos tareas generales:

1. Diseño: se refiere a modelar gráficamente todas las funcionalidades que serán implementadas en la Base de Datos.
2. Implementación: se refiere a la codificación de las funcionalidades basándose en el diseño previamente realizado.

Se entiende como funcionalidad un objeto o grupo de estos los cuales correspondan a determinada funcionalidad del sistema.

Los objetivos de esta actividad son:

1. Diseñar gráficamente las funcionalidades que serán implementados en la Base de Datos.
2. Implementar las funcionalidades basándose en el diseño previamente realizado.

3. Actualizar la documentación de la estructura de la Base de Datos incorporando la documentación de los nuevos elementos de implementación.

Para dar cumplimiento al objetivo No. 1, debemos contar con una herramienta para el diseño visual de las funcionalidades de la Base de Datos. Diseñar gráficamente las funcionalidades nos facilita las labores de documentación, administración, mantenimiento, corrección de errores, toma de decisiones y ganamos en claridad en cuanto a la maquinaria del funcionamiento de la Base de Datos.

Para dar cumplimiento al objetivo No. 2 debemos seguir una traza respecto al diseño previamente realizado e implementar las funcionalidades según el diseño para lograr concordancia y una continuidad evolutiva en el proceso de implementación.

La reingeniería en esta actividad es un tema a considerar en determinadas ocasiones del ciclo de vida del desarrollo. Pudiéramos primero implementar y luego mediante ingeniería inversa realizar el diseño, sobre todo si queremos darle un enfoque ágil al proceso, siempre y cuando sea necesario y factible. Sí es importante mantener el diseño, ya que como se mencionaba anteriormente nos facilita muchas tareas relacionadas con la documentación, administración, mantenimiento, corrección de errores, toma de decisiones y claridad en el proceso.

Para dar cumplimiento al objetivo No. 3 debemos ir actualizando la documentación de la estructura de la Base de Datos incorporando la documentación de los nuevos elementos de implementación en el momento que se considere pertinente realizarlo, ya que de cierta forma cumpliendo el objetivo No. 2 tenemos parte de dicha documentación.

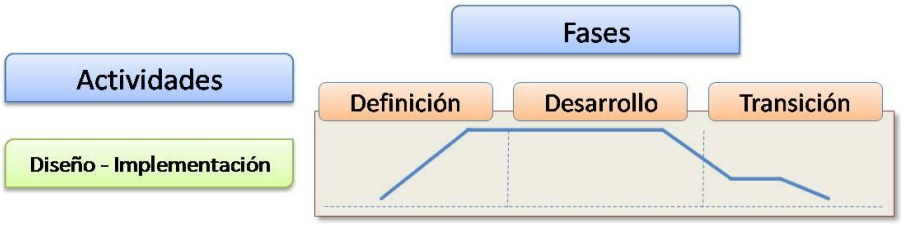

Figura 18: Descripción gráfica aproximada de la actividad Diseño – Implementación.

Artefactos

Los artefactos que se generan en esta actividad son:

1. Diseño grafico de las funcionalidades de la Base de Datos.
2. Codificación de las funcionalidades de la Base de Datos.

Los artefactos que se actualizan son:

1. Documentación de estructura de la Base de Datos.

Figura 19: Artefactos de la actividad Diseño – Implementación.

ADTP

El nombre de esta actividad viene dado por las tareas generales que se realizan:

1. Acceso a Datos.
2. Tuning.
3. Pruebas Generales.

La tarea general Acceso a Datos se refiere a la creación – modificación de la interfaz de comunicación entre la aplicación de negocio y la Base de Datos. La tarea general Tuning se refiere a las optimizaciones de E/S vinculadas con las funcionalidades implementadas en la Base de Datos. Por último la tarea general de Pruebas Generales se refiere a las pruebas en el marco de las tareas de Acceso a Datos y Tuning, y puede definirse como una tarea intermedia de comprobación y terminación.

Cabe destacar que en esta actividad, las tareas generales suelen estar muy relacionadas y solo en determinadas ocasiones se realizan aisladamente. Al mismo tiempo, esta actividad y la actividad de Diseño – Implementación, a medida que avanza el desarrollo de la Base de Datos, suelen estar muy relacionadas, pero es conveniente dividir lógicamente las tareas generales por el volumen de trabajo que suelen contener. La realización de esta actividad transita similarmente a la actividad de Diseño – Implementación en el ciclo del desarrollo de la Base de Datos.

Los objetivos de esta actividad son:

1. Estructurar el acceso a datos.
2. Optimizar las operaciones de E/S vinculadas a la codificación de las funcionalidades de la Base de Datos.
3. Probar las funcionalidades optimizadas hasta obtener un resultado eficiente en la medida de los requerimientos del sistema.
4. Garantizar un acceso a datos óptimo y seguro.

5. Actualizar la documentación de la estructura de la Base de Datos incorporando la documentación de los nuevos elementos de optimización.

Para dar cumplimiento al objetivo No. 1, debemos definir una estructura de acceso a datos lo más robusta para la arquitectura del sistema, que contemple elementos de seguridad, y a la vez flexible para los desarrolladores.

Para dar cumplimiento al objetivo No. 2, se debe identificar aquellos elementos candidatos a optimizar, trazar una estrategia de optimización y realizar los cambios pertinentes. Cabe señalar en este punto que las tareas de optimización de E/S no son propias únicamente de esta actividad, sino que desde el Modelo de Datos debemos definir elementos, como por ejemplo índices, y a la hora de implementar debemos hacerlo óptimamente. Las tareas de optimización en esta actividad suele ser el punto final de ajuste en este sentido.

El cumplimiento del objetivo No. 3 va implícito en el cumplimiento del objetivo No. 2.

Cumpliendo satisfactoriamente todos los objetivos previamente descritos, damos cumplimiento al objetivo No. 4.

Para dar cumplimiento al objetivo No. 5 debemos ir actualizando la documentación de la estructura de la Base de Datos incorporando la documentación de los nuevos elementos de optimización en el momento que se considere pertinente realizarlo.

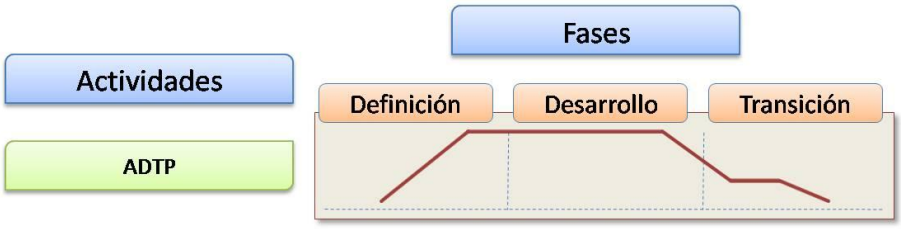

Figura 20: Descripción gráfica aproximada de la actividad ADTP.

Artefactos

Los artefactos que se generan en esta actividad son:

1. Acceso a Datos.

Los artefactos que se actualizan son:

1. Diseño grafico de las funcionalidades de la Base de Datos.
2. Codificación de las funcionalidades de la Base de Datos.
3. Documentación de estructura de la Base de Datos.

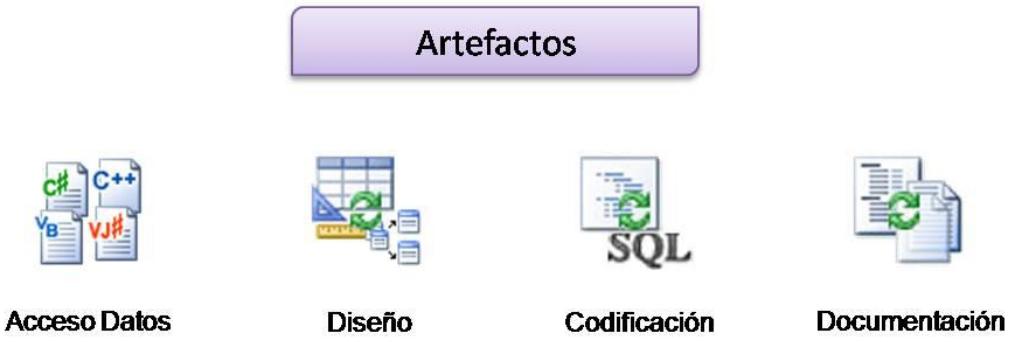

Figura 21: Artefactos de la actividad ADTP.

ROLES

Los roles definen responsabilidades dentro del proceso y los mismos pueden desempeñarse en una o más personas, como también una o más personas pueden desempeñar más de un rol de los definidos. Son responsables de una o más actividades y deben trabajar en conjunto en la toma de decisiones y no aisladamente.

ARQUITECTO DE BASE DE DATOS

El Arquitecto de Base de Datos es el encargado de organizar el proceso y que funcione de la mejor manera, es el máximo responsable por el buen funcionamiento de la Base de Datos, toma las principales decisiones y es el intermediario entre la aplicación de negocio y la aplicación de bases de datos.

El Arquitecto de Base de Datos forma parte de los arquitectos del sistema y pudiera definirse como el *"jefe"* de la Base de Datos. Es el encargado de estructurar y realizar las tareas generales definidas en la actividad de Modelado de Datos, así como sus artefactos. También es responsable del artefacto general del proceso Nomenclatura.

Figura 22: Rol Arquitecto de Base de Datos.

ADMINISTRADOR DE BASES DE DATOS

El Administrador de Base de Datos se define según la descripción para Administrador de Bases de Datos en la sección Roles involucrados en el proceso.

Es el encargado de estructurar y realizar las tareas generales de la actividad Configuraciones, así como sus artefactos.

Figura 23: Rol Administrador de Base de Datos.

DESARROLLADOR DE BASES DE DATOS

El Desarrollador de Base de Datos se define según la descripción para Desarrollador de Base de Datos en la sección Roles involucrados en el proceso.

Es el encargado de estructurar y realizar las tareas generales de la actividad Diseño – Implementación y ADTP, así como sus artefactos.

Figura 24: Rol Desarrollador de Base de Datos.

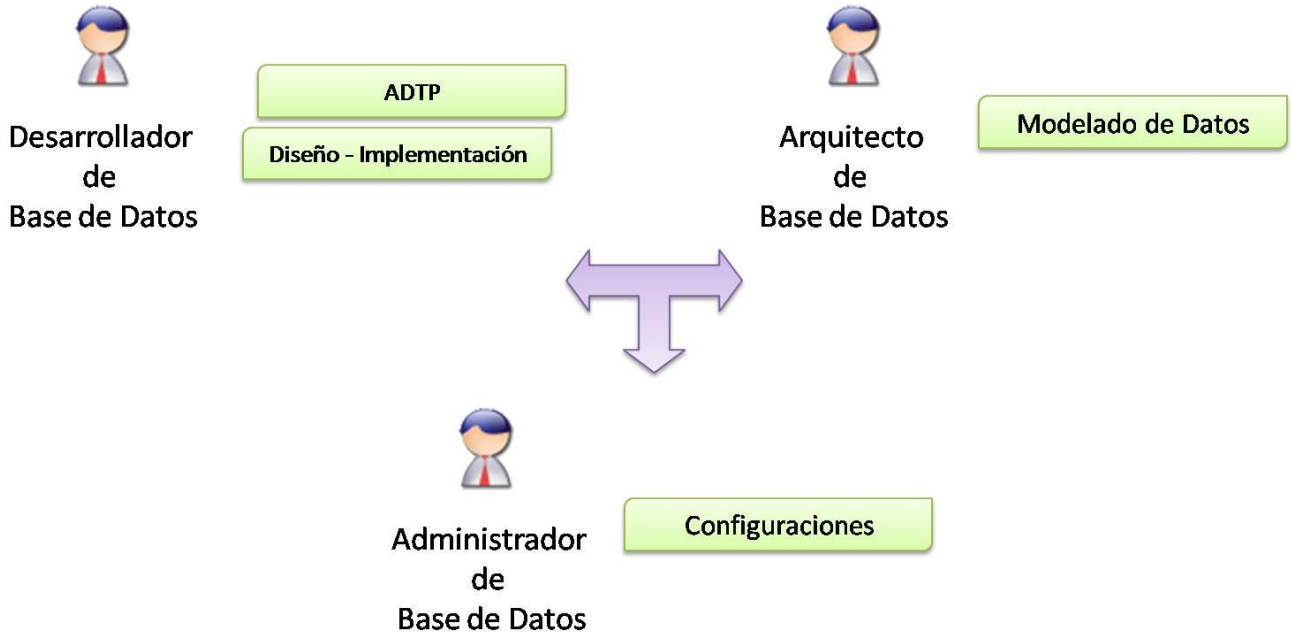

Figura 25: Relación entre actividades.

CONSIDERACIONES GENERALES DEL PROCESO

El proceso debe ajustarse al entorno de desarrollo, y el mismo puede considerarse como una división lógica por roles del desarrollo de bases de datos.

Detallar las tareas que se puedan definir en las actividades como parte del proceso es un tema no práctico por la variedad de estas que podemos llevar a cabo en determinado momento. Las tareas generales definidas son el núcleo de cada actividad, y de ellas se derivan cualquier otro tipo de tareas.

La concepción del proceso pretende minimizar la dependencia hombre – software que suele ocurrir en proyectos de software donde escasea la documentación y una sola persona maneja grandes responsabilidades y al mismo tiempo las soluciones respecto a la Base de Datos.

Cualquier integrante del equipo de desarrollo con conocimientos de bases de datos puede desempeñar cualquier rol de los planteados en el proceso y no es de estricto cumplimiento la existencia de un equipo aparte para llevar el desarrollo de la Base de Datos, si de al menos un especialista como se define al Arquitecto de Base de Datos.

CAPÍTULO 3: VALIDACIÓN DEL PROCESO

En el presente capitulo se aborda la validación del proceso propuesto. Validación necesaria para determinar la validez y necesidad del mismo, mediante criterios de expertos con el uso de técnicas propuestas por el método Delphi, el cual procede por medio de la interrogación a expertos con la ayuda de cuestionarios sucesivos, a fin de poner de manifiesto convergencias de opiniones y deducir eventuales consensos.

Los métodos de expertos utilizan como fuente de información un grupo de personas a las que se supone un conocimiento elevado de la materia que se va a tratar. Estos métodos se emplean cuando se da alguna de las siguientes condiciones:

1. No existen datos históricos con los que trabajar. Un caso típico de esta situación es la previsión de implantación de nuevas tecnologías.

2. El impacto de los factores externos tiene más influencia en la evolución que el de los internos. Así, la aparición de una legislación favorable y reguladora y el apoyo por parte de algunas empresas a determinadas tecnologías pueden provocar un gran desarrollo de éstas que de otra manera hubiese sido más lento.

3. Las consideraciones éticas o morales dominan sobre los económicas y tecnológicas en un proceso de evolutivo. En este caso, una tecnología puede ver dificultado su desarrollo si éste provoca un alto rechazo en la sociedad (un ejemplo lo tenemos en la tecnología genética, que ve dificultado su avance por los problemas morales que implica la posibilidad de manipulación del genotipo).

El método de expertos ideal sería aquel que extrajese los beneficios de la interacción directa y eliminase sus inconvenientes. Esta intenta ser la filosofía de la metodología Delphi.

EL MÉTODO DELPHI

El método Delphi pretende extraer y maximizar las ventajas que presentan los métodos basados en grupos de expertos y minimizar sus inconvenientes. Para ello se aprovecha la sinergia del debate en el grupo y se eliminan las interacciones sociales indeseables que existen dentro de todo grupo. De esta forma se espera obtener un consenso lo más fiable posible del grupo de expertos.

Este método presenta tres características fundamentales:

1. Anonimato: Durante un Delphi, ningún experto conoce la identidad de los otros que componen el grupo de debate. Esto tiene una serie de aspectos positivos, como son:

   - Impide la posibilidad de que un miembro del grupo sea influenciado por la reputación de otro de los miembros o por el peso que supone oponerse a la mayoría. La única influencia posible es la de la congruencia de los argumentos.

   - Permite que un miembro pueda cambiar sus opiniones sin que eso suponga una pérdida de imagen.

   - El experto puede defender sus argumentos con la tranquilidad que da saber que en caso de que sean erróneos, su equivocación no va a ser conocida por los otros expertos.

2. Iteración y realimentación controlada: La iteración se consigue al presentar varias veces el mismo cuestionario. Como, además, se van presentando los resultados obtenidos con los cuestionarios anteriores, se consigue que los expertos vayan conociendo los distintos puntos de vista y puedan ir modificando su opinión si los argumentos presentados les parecen más apropiados que los suyos.

3. Respuesta del grupo en forma estadística: La información que se presenta a los expertos no es sólo el punto de vista de la mayoría, sino que se presentan todas las opiniones indicando el grado de acuerdo que se ha obtenido.

De manera resumida los pasos que se llevarán a cabo para garantizar la calidad de los resultados, para lanzar y analizar la Delphi deberían ser los siguientes:

1. Fase 1: Formulación del problema
2. Fase 2: Elección de expertos.
3. Fase 3: Elaboración y lanzamiento de los cuestionarios (en paralelo con la fase 2).
4. Fase 4: Desarrollo práctico y explotación de resultados.

SELECCIÓN DEL GRUPO DE EXPERTOS

Se entiende por experto, tanto al individuo en sí como a un grupo de personas u organizaciones capaces de ofrecer valoraciones conclusivas de un problema en cuestión y hacer recomendaciones con un máximo de competencia.

Para la selección del grupo de expertos se realizaron las siguientes actividades:

1. Determinación de las áreas del conocimiento que deben dominar los expertos.

Partiendo del problema planteado en la introducción se determinó que los expertos a consultar debían dominar las siguientes áreas del conocimiento: Ingeniería de Software, procesos de desarrollo de software, y desarrollo de bases de datos.

2. Elaboración del listado de expertos candidatos.

Luego de determinar las áreas del conocimiento se elaboró un listado de expertos candidatos teniendo en cuenta su experiencia productiva o docente en las áreas identificadas.

Aunque no hay forma de determinar el número óptimo de expertos para participar en una encuesta Delphi, estudios realizados por investigadores de la Rand Corporation[24], señalan que si bien parece necesario un mínimo de siete expertos, habida cuenta que el error disminuye notablemente por cada experto añadido hasta llegar a los siete expertos, no es aconsejable recurrir a más de 30 expertos, pues la mejora en la previsión es muy pequeña y normalmente el incremento en coste y trabajo de investigación no compensa la mejora.

3. Obtención del consentimiento de los expertos para participar.

El siguiente paso fue la obtención del consentimiento de los expertos para participar en la validación.

4. Determinación del coeficiente de conocimiento de los expertos.

Las características de los expertos influyen decisivamente en la confiabilidad de los resultados obtenidos. Estas características son: calificación técnica, capacidad de emitir una decisión al respecto, conocimientos específicos sobre el tema a evaluar, disposición a participar, entre otros.

Para la selección de los expertos es útil emplear la valoración por competencias. Este método consiste en calcular el coeficiente de competencia (k) del experto a partir de la autovaloración del experto sobre su conocimiento o información sobre el tema (kc) y el coeficiente de argumentación o valoración (ka) mediante la siguiente ecuación:

$$k = (k_c + k_a)/2$$

El código de interpretación de los coeficientes de competencias es como sigue:

Si $0.8 < k < 1.0$ coeficiente de competencia alto.

[24] **Rand Corporation**: www.rand.org

Si $0.5 < k < 0.8$ coeficiente de competencia medio.

Si $k < 0.5$ coeficiente de competencia bajo.

Recomendándose incluir en el grupo a los de coeficiente de competencia alto y medio.

En el Anexo 4: Encuesta para determinar el coeficiente de experticidad., se muestra la encuesta realizada para determinar el coeficiente de experticidad de cada experto.

VALIDACIÓN DE LA PROPUESTA

Para la validación de la propuesta se tomaron cinco expertos los cuales dio como resultado de experticidad entre 0.5 y 1.0. A los mismos se le aplicó la encuesta presentada en el anexo: Anexo 5: Encuesta de validación del proceso, la cual tiene como objetivos determinar la necesidad de un proceso genérico de desarrollo de bases de datos en procesos de desarrollo de software, así como validar la correcta estructuración de las actividades definidas, los roles y la correcta selección de los artefactos que se generan. Todos estos elementos de forma general dentro del desarrollo de software.

La encuesta respondida por los diferentes expertos arrojó los siguientes resultados:

- Es importante tener un modelo genérico que guíe el desarrollo de bases de datos en procesos de desarrollo de software. De aquí la vigencia del tema propuesto para la investigación.
- El modelo propuesto es efectivo en su aplicación genérica sin importar metodologías, sistemas de bases de datos y herramientas. De aquí la efectividad de ser implementado por los equipos de desarrollo.
- Actividades propuestas y sus tareas generales son necesarias y generales para desarrollar bases de datos.

- La organización de las actividades es correcta.
- Los artefactos propuestos son necesarios y generales para desarrollar bases de datos. Determinados expertos plantearon determinados artefactos que pudieran ser genéricos e importantes los cuales pudieran incluirse en el modelo.
- Los roles definidos son necesarios y generales para desarrollar bases de datos. Determinados expertos plantearon determinada estructuración de roles partiendo de los propuestos en el modelo dependiendo de las características del software que estemos desarrollando, lo cual forma parte de un principio del proceso propuesto, que no es más que la estructuración del desarrollo de bases de datos basado en una estructura genérica de desarrollo.

CONCLUSIONES

La organización del equipo de desarrollo, en proyectos de software, es un punto clave en el éxito del mismo. Las metodologías de desarrollo de software no plantean estos elementos estructurales – organizativos respecto al desarrollo de bases de datos en proyectos de software y queda a responsabilidad del equipo de desarrollo, o de un especialista en el tema, organizar el desarrollo de la Base de Datos. Independientemente que una Metodología de Desarrollo de Software guíe el software durante todo su ciclo de vida, el desarrollo de bases de datos requiere una atención diferente por las propias características del trabajo con bases de datos.

Hoy en la universidad se carece de personal preparado para enfrentar un proceso de desarrollo de base bases de datos, las tecnologías a utilizar son muy diversas y complejas, las formas de organizar el equipo muy variable. Es por ello que un proceso genérico es un elemento muy importante, ya que establece las reglas de gestión de configuración dentro del equipo de desarrollo de base de datos, la planificación del cronograma y la definición de los roles dentro del mismo.

No siempre se cuenta con el personal con los conocimientos suficientes para lograr un desarrollo de bases de datos exitoso. Es de gran utilidad contar con un modelo que guíe al desarrollador de bases de datos durante todo el proceso, que indique los elementos, técnicas y procedimientos a seguir.

El modelo propuesto estructura el desarrollo de bases de datos de forma tal que puede ser aplicado en cualquier entorno de desarrollo sin que sea un impedimento determinada metodología, Sistema Gestor De Base De Datos o herramientas utilizadas, ya que organiza los elementos que deben estar presentes a la hora de desarrollo de bases de datos de forma general. Por otro lado permite la organización tanto de las actividades como de los recursos humanos de los cuales se necesita, permitiendo desarrollar más rápidamente y con calidad.

Los artefactos propuestos corresponden a la mayoría de los elementos de configuración y responden de cierta forma a la documentación de la arquitectura de la Base de Datos.

Las actividades, tareas generales, roles y responsabilidades definidas constituyen una división lógica de todo el esfuerzo en el desarrollo de bases de datos, estructurando genéricamente el mismo.

RECOMENDACIONES

- Buscar más fuentes de información respecto a la estructura organizativa de desarrollo de bases de datos en proyectos de software en producción, para definir más exactamente el modelo de desarrollo lo más general posible.
- Describir y fundamentar el modelo propuesto evolutivamente a través de la retroalimentación en la medida que los equipos de desarrollo ponen en marcha el desarrollo de bases de datos basados en el modelo propuesto.
- Describir mejor la relación entre las actividades propuestas.
- Especificar con más detalle las responsabilidades de los roles propuestos.
- Definir una nomenclatura exacta para otros elementos generales de Base de Datos no mencionados.
- Definir cómo documentar los diferentes elementos de Base de Datos.

BIBLIOGRAFÍA

Database Development . [En línea] http://www.databasedevelopment.co.uk/services.htm.

DevCampus Database Naming Conventions. [En línea] http://weblogs.asp.net/jamauss/articles/DatabaseNamingConventions.aspx#Tables.

About, Inc. *Database Development.* [En línea] http://databases.about.com/od/development/Database_Development.htm.

Agapea Factory S.A. DESARROLLO DE BASES DE DATOS: CASOS PRÁCTICOS DESDE EL ANÁLISIS A LA IMPLEMENTACIÓN. [En línea] http://www.agapea.com/Desarrollo-de-Bases-de-Datos-casos-practicos-desde-el-analisis-a-la-implementacion-n1234377i.htm.

Agile Alliance. *Agile Alliance.* [En línea] http://www.agilealliance.org/home.

Ambler, Scott W. Techniques for Successful Evolutionary/Agile Database Development. *Agile Data Home Page.* [En línea] http://www.agiledata.org/.

Argüello Venegas, José Ronald & Di Mare, Adolfo. 1981. Necesidad de un diseño lógico para una base de datos. *Revista COMPUTING.* 1981.

Astigarraga, Eneko. EL MÉTODO DELPHI. [En línea] http://www.codesyntax.com/prospectiva/Metodo_delphi.pdf.

Capítulo 8: Proceso de creación y Metodología de Desarrollo de Bases de Datos. [En línea] http://bibliologia.info/archivos/DisBasDatRela_8.pdf.

Database Development Services, Inc. FileMaker Consulting, Database Development Services, Inc. *Database Development Services.* [En línea] http://www.dbservices.com/.

Desarrollo +. *Programación en Capas Primera Parte (Capa de Acceso a Datos).* [En línea] http://jmhogua.blogspot.com/2007/01/programacin-en-capas-primera-parte-capa.html.

El método Delphi. [En línea] http://www.gtic.ssr.upm.es/encuestas/delphi.htm.

Fowler, Martin. Evolutionary Database Design. [En línea] http://martinfowler.com/articles/evodb.html.

Garcia, Rosa Maria Mato. 1999. Diseño de Bases de Datos. *Diseño de Bases de Datos.* Octubre de 1999.

Grupo Alarcos - Universidad de Castilla la Mancha. [En línea] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf.

Hassan, Ahmed M. y Elssamadisy, Amr. Extreme Programming and Database Administration: Problems, Solutions, and Issues.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : AddisonWesley, 2000. 84-7829-036-2.

Luis Mesa, Jose, y otros. Patron DAO Data Access Object Objeto de Acceso a Datos.

Marqués, Mercedes. 2001. Apuntes de Ficheros y Bases de Datos. 10 de Febrero de 2001.

Microsoft. *MSDN es Español.* [En línea] [Citado el: 21 de Noviembre de 2007.] http://msdn.microsoft.com/library/spa/default.asp?url=/library/spa/vsintro7/html/vsstartpage.asp.

—. MSF and databases. *Microsoft.* [En línea] http://www.microsoft.com/.

Molpeceres, Alberto. 2002. *Procesos de desarrollo: RUP, XP y FDD.* 2002.

MVI Solutions. Database Development. *MVI Solutions.* [En línea] http://www.mediavue.net/databases/database_development.html#custom.

MySQL. *MySQL.* [En línea] [Citado el: 21 de Noviembre de 2007.] http://www.mysql.com/.

One Step Education Network. One Step Education Network: Information Technology Solutions. [En línea] http://www.onestep.net.au/images/Database%20Development%20Process.pdf.

Oracle. 2006. *Oracle® Database Administrator's Guide 10g Release 1 (10.1).* 2006.

—. 2006. *Oracle® Database 2 Day + Performance Tuning Guide.* 2006.

Pressman, Roger S. *Ingenieria de Software. Un enfoque práctico.*

Rational Software Corporation. Rational Unified Process Help.

Red Gate Software. simple-talk. *Ten Common Database Design Mistakes.* [En línea] http://www.simple-talk.com/sql/database-administration/ten-common-database-design-mistakes/.

Sanchez, María A. Mendoza. Metodologías De Desarrollo De Software. [En línea] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

Sigmer Technologies Ltd. Sigmer Technologies. [En línea] http://www.sigmer.com/products_services/database_solutions_lifecyle.html.

SoftDevArticles. [En línea] http://www.softdevarticles.com/.

Southern Illinois University Edwardsville. Database Development Process. [En línea] http://www.siue.edu/~dbock/cmis450/2-databasedevelopment.htm.

Universidad de Bogotá Jorge Tadeo Lozano. Especialización en Desarrollo de Bases de Datos . [En línea] http://www.utadeo.edu.co/programas/postgrados/especializaciones/base_datos/index.php.

Universidad de las Ciencias Informáticas. *Inter-Nos.* [En línea] [Citado el: 20 de Noviembre de 2007.] http://inter-nos/Teleclases/Teleclases.asp?id_as=12.

—. Sitio de Ingenieria de Software I. [En línea] http://teleformacion.uci.cu/mod/resource/view.php?id=6655.

Universidad de Murcia. Metodologías de desarrollo de software. [En línea] http://www.um.es/docencia/barzana/IAGP/Iagp2.html#BM4.

Visconti, Marcello y Astudillo, Hernán. Fundamentos de Ingeniería de Software. [En línea] [Citado el: 11 de Diciembre de 2007.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/14-Persistencia.pdf.

Yahoo. [En línea] http://tech.groups.yahoo.com/group/agileDatabases/.

ANEXOS

ANEXO 1: ENCUESTA SOBRE EL PROCESO DE DESARROLLO – ADMINISTRACIÓN DE BASES DE DATOS EN LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

ENCUESTA
PROCESO DE DESARROLLO – ADMINISTRACIÓN DE BASES DE DATOS EN LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Nombre del proyecto:

Ejemplo: Registros & Notarias.

Nota: Los ejemplos mostrados en la encuesta son hipotéticos.

Proceso de desarrollo – metodología utilizada en el proyecto y que plantea la misma sobre el desarrollo de Bases de Datos:

Ejemplo: XP. La misma plantea que los integrantes del equipo de la Base de Datos trabajen en dúos.

Roles, responsabilidades y artefactos que manejan en el desarrollo – administración de Bases de Datos:

Ejemplo:

Rol	Responsabilidad	Artefactos
Administrador de Bases de Datos	1. Configuraciones. 2. Montaje de escenarios.	1. Documento de configuración por escenarios.
Diseñador de Bases de Datos	1. Diseñar la Base de Datos.	1. Modelo de Datos.
Desarrollador de Bases de Datos	1. Implementar las funcionalidades.	

Descripción – Estrategia del proceso de desarrollo – administración de la Base de Datos:

Nota: Ser lo más descriptivo posible.

Ejemplo: Somos un equipo de desarrolladores de Bases de Datos formado por 2 roles, los cuales realizamos nuestras actividades según los requerimientos. La estrategia es cada día terminar una tarea en específico y realizar una compilación semanal y a la vez actualizar la documentación.

Estrategia de diseño de la Base de Datos:

Ejemplo: A partir de los requerimientos existentes y los nuevos que se vallan presentando, vamos conformando el Modelo de Datos y la implementación de las funcionalidades que satisfagan el negocio.

Estrategia de configuración:

Ejemplo: El administrador de la Base de Datos se guía por el documento de configuración – administración de Postgre versión 5.7 del 2008 para la configuración – administración de instancias de Base de Datos.

Estrategia de implementación:

Ejemplo: A medida que los desarrolladores del negocio requieran cierta funcionalidad la implementamos y le damos una librería de acceso a datos.

Estrategia de optimización:

Ejemplo: Cada vez que las pruebas den como resultado búsquedas lentas tomamos decisiones respecto a la optimización de la misma y la implementamos.

Estrategia de nomenclatura:

> **Nota:** Ser lo más abarcador posible en caso de existir una nomenclatura común. En caso contrario especificar como se nombran los elementos relacionados con la Base de Datos.

Ejemplo:

1. Nombre de tablas: t<NombreTabla>
2. Nombre de las llaves primarias: pk<NombreLlavePrimaria>
3. Nombre de las relaciones de integridad: r1<NombreRelacion>
4. Procedimientos almacenados: sp<Nombre>

5. No llevamos una nomenclatura común.

Herramientas de automatización del proceso:

Ejemplo: Utilizamos un generador de capas de acceso a datos para agilizar este proceso.

Estrategia de documentación:

Ejemplo: Documentamos todos los procedimientos almacenados con una breve descripción para un mejor mantenimiento, administración y control de cambios.

Otros elementos organizativos que considere importantes y que definan el proceso de desarrollo – administración de la Base de Datos dentro de su proyecto:

ANEXO 2: PLANTILLA GENERAL NOMENCLATURA DE BASE DE DATOS.

PLANTILLA GENERAL NOMENCLATURA DE BASE DE DATOS

Este documento contempla la nomenclatura estándar a seguir en cuanto a objetos y elementos lógico – físicos de la estructura de la Base de Datos.

Elemento:

Nomenclatura:

Descripción:

Elemento:

Nomenclatura:

Descripción:

ANEXO 3: PLANTILLA GENERAL CONFIGURACIONES DE BASE DE DATOS.

PLANTILLA GENERAL CONFIGURACIONES DE BASE DE DATOS

Este documento contempla las configuraciones de la Base de Datos de <Centro de Datos><Oficina><Desarrollo> para garantizar un entorno de <Desarrollo><Prueba><Despliegue>.

Descripción General

Elemento:

Configuración:

Descripción:

ANEXO 4: ENCUESTA PARA DETERMINAR EL COEFICIENTE DE EXPERTICIDAD.

DETERMINACIÓN DEL COEFICIENTE DE INFORMACIÓN DEL EXPERTO

Marque con una X, en una escala del 1 al 10 el valor que se corresponde con el grado de conocimiento o información que usted considera que tiene respecto a procesos de desarrollo de software centrado en el tema de bases de datos, 1 indica que no tiene ningún conocimiento sobre el tema y 10 indica que tiene pleno conocimiento.

1	2	3	4	5	6	7	8	9	10

Señale con una X el nivel de influencia que ha tenido cada una de las fuentes de argumentación en su conocimiento sobre el desarrollo de bases de datos en procesos de desarrollo de software.

Fuentes De Argumentación	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted			
Su experiencia obtenida			
Trabajos de autores nacionales			
Trabajos de autores extranjeros			
Su propio conocimiento del estado del problema en el extranjero			
Su intuición			

ANEXO 5: ENCUESTA DE VALIDACIÓN DEL PROCESO

ENCUESTA DE VALIDACIÓN DEL PROCESO

1. ¿Considera usted que es importante tener un modelo genérico que guíe el desarrollo de bases de datos en procesos de desarrollo de software?

   Sí__ No__

   ¿Por qué?

2. ¿Considera usted que el modelo propuesto es efectivo en su aplicación genérica sin importar metodologías, sistemas de bases de datos y herramientas?

   Sí__ No__

   ¿Por qué?

3. Evalúe el modelo propuesto según los siguientes aspectos:

a) Las actividades propuestas y sus tareas generales son:

__Innecesarias.

__Necesarias pero no suficientes para desarrollar bases de datos.

__Necesarias y generales para desarrollar bases de datos.

Otras consideraciones al respecto:

b) La organización de las actividades:

__Correcta.

__Incorrecta.

Otras consideraciones al respecto:

c) Los artefactos propuestos son:

__ Innecesarios.

__ Necesarios pero no suficientes para desarrollar bases de datos.

__ Necesarios y generales para desarrollar bases de datos.

Otras consideraciones al respecto:

d) Los roles definidos son:

__Innecesarios.

__ Necesarios pero no suficientes para desarrollar bases de datos.

__ Necesarios y suficientes para desarrollar bases de datos.

Otras consideraciones al respecto:

4. Elabore un comentario general sobre el procedimiento que está siendo evaluado que aporte elementos a la mejora del mismo.

GLOSARIO DE TÉRMINOS

A

Artefacto: producto tangible resultante del proceso de desarrollo de software.

B

C

Casos de Uso: Secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

Cuello de Botella: se refiere a elementos que atentan contra el rendimiento del sistema.

CPU: Unidad Central de Procesamiento, por su traducción.

D

DDL: Data Definition Language. Lenguaje de Definición de Datos, por su traducción.

DML: Data Manipulation Language. Lenguaje de Manipulación de Datos, por su traducción.

E

Esquema de Persistencia: conjunto reutilizable de clases que presentan servicios a los objetos persistentes, siendo una Bases de Datos es su representación física.

F

Fases Genéricas: Ver Capítulo 1: Fundamentación Teórica, Ingeniería de Software.

G

GUI: Graphic User Interface. Interfaz de Grafica Usuario por su traducción.

H

I

IDE: (Integrated Development Environment) es un programa compuesto por un conjunto de herramientas para un programador.

J

K

L

M

Metodología de Desarrollo de Software: Un conjunto de actividades que tienen un orden lógico, en las cuales se utilizan herramientas, técnicas y que garantizan que al concluir las actividades tengamos el software que realmente queremos. (Universidad de las Ciencias Informáticas).

Microsoft® SQL Server®: Sistema Gestor de Bases de Datos creado por Microsoft.

Microsoft® Visual Studio®.Net: conjunto de aplicaciones para la creación tanto de aplicaciones de escritorio como de aplicaciones Web de empresa para trabajo en equipo creado por Microsoft.

Modelo de Datos: Modelo que describe como se representan los datos. Consiste en objetos, atributos y relaciones.

MySql: Base de Datos de código abierto.

N

Ñ

O

P

Proceso: Un proceso define "*quién"* esta haciendo "*qué"*, *"cuando"* y "*cómo"* alcanzar un determinado objetivo. En la Ingeniería de Software consiste en construir un producto software o mejorar uno existente. (Jacobson, y otros, 2000 pág. XVI)

Proceso de Desarrollo de Software: definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto. (Jacobson, y otros, 2000 pág. 13)

Q

R

Rand Corporation: www.rand.org.

Release: Por su traducción se refiere a una versión o liberación del producto software.

Releases: Ver Release.

RUP: Metodología de Desarrollo de Software que se caracteriza por estar dirigida por Casos de Uso, centrada en la arquitectura, iterativa e incremental.

S

Sistemas Gestores de Bases de Datos: conjunto de programas y/o procesos que permiten crear y mantener una Base de Datos, asegurando su integridad, confidencialidad y seguridad.

SQL: Structured Query Language. Lenguaje Estructurado de Consultas, por su traducción.

T

Tuning: (afinar, por su traducción al español) En términos de Bases de Datos se refiere a un alto grado de optimización de los procesos en sentido general.

U

V

W

X

XP: Metodología de Desarrollo de Software ágil la cual se caracteriza por ser concreta y de continua retroalimentación en ciclos cortos.

Y

Z
