



**INSTITUTO TECNOLÓGICO
SUPERIOR JAPÓN**

GUÍA
METODOLÓGICA
DE
DESARROLLO DE SISTEMAS

COMPILADO POR:

MAGÍSTER CÉSAR VALDIVIEZO
DESARROLLO DE SOFTWARE 2019

AMOR AL CONOCIMIENTO



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

METODOLOGIAS DE DESARROLLO DE SISTEMAS

1ER SEMESTRE DE DESARROLLO DE SOFTWARE

Descripción

“...La obtención de un software de calidad implica la utilización de modelos o procedimientos estándares para su análisis, diseño, desarrollo y prueba que permitan uniformar la filosofía de trabajo ...”

Tlgo. Esp. César Stalin Valdiviezo Castro

stalinaldiviezo@gmail.com

Tabla de contenido

METODOLOGIA DE DESARROLLO DE SOFTWARE 1ER SEMESTRE.....	1
UNIDAD 1.....	1
INTRODUCCION	1
CONCEPTOS GENERALES	1
Términos y Definiciones	1
• Área de desarrollo.-.....	1
• Cliente.-	1
• Coordinador.-	2
• Desarrollador.-	2
• Desarrollador Líder.-	2
• Entregables.-	2
• Método.-.....	2
• Metodología.-	2
• Metodología de Desarrollo de Software.-.....	2
• Módulo Informático.-	2
• Proyecto.-	2
• Software.-	2
• Stakeholders.-.....	3
• Usuario.-	3
INGENIERÍA DE SOFTWARE.....	4
Método	4
Metodología	4
METODOLOGIAS DE DESARROLLO DE SOFTWARE.....	5
Selección de sistema de información para el desarrollo del software	5
Recopilación de la información del desarrollo de software.....	5
TECNICAS	5
Entrevista.....	5
Cuestionario.....	6
Encuesta.....	7
La observación	8
Diccionario de Datos.....	9



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Otros	11
Procesamiento de la información recopilada.....	12
Las etapas para el Procesamiento de Datos son las siguientes:	12
Entrada:	12
Proceso:	12
Salida:	12
CODIFICACIÓN	12
TABULACION	12
REGISTRO DE DATOS	13
Presentación de datos	13
Análisis de la problemática del sistema actual.....	14
Introducción al Diseño de Sistemas	14
Análisis de Sistemas.....	15
CONCEPTUALIZACIÓN DE SISTEMAS	16
PLANTEAMIENTO DEL PROBLEMA	16
DESCRIPCIÓN DEL PROBLEMA.....	16
PLANTEAMIENTO DEL PROBLEMA.	16
FORMULACIÓN DEL PROBLEMA.....	16
SISTEMATIZACIÓN DEL PROBLEMA.	17
UNIDAD 2.....	18
APLICACIÓN DE LA METODOLOGIA DEL DESARROLLO DE SOFTWARE	18
METODOLOGIAS TRADICIONALES	18
Metodología en cascada: Framework lineal.	18
Principios Básicos Del Modelo De Cascada	19
Método de Prototipos	20
Principios Básicos del método de prototipos.....	21
Modelo Incremental o Iterativo y Creciente	22
Fases del Modelo Incremental	22

Página | 1

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Principios básicos del Modelo Incremental.....	23
Modelo en Espiral.....	23
Principios básicos del modelo en Espiral.....	25
RAD: Desarrollo Rápido de Aplicaciones (Rapid Application Development)	26
Principios básicos del Modelo RAD	26
METODOLOGÍAS ÁGILES.....	28
Manifiesto Ágil.....	28
• Al Individuo y las Iteraciones del Equipo.	28
PRINCIPALES METODOLOGÍAS ÁGILES	30
METODOLOGÍA SCRUM.....	30
Funcionamiento de los Procesos Scrum.....	31
Equipos que Componen los Procesos Scrum	32
METODOLOGÍA KANBAN	33
Garantía de Calidad.	34
METODOLOGÍA XP	36
Equipo de Trabajo dentro de una Metodología XP.....	40
CLASIFICACION DE LAS METODOLOGIAS DE DESARROLLO DE SOFTWARE CRONOLOGICAMENTE.....	41
LENGUAJE UNIFICADO DE MODELADO – UML.....	41
VERSIONES DE UML.....	42
TIPOS DE DIAGRAMAS EN UML	43
DIAGRAMAS DE COMPORTAMIENTO.....	43
DIAGRAMAS DE CASOS DE USO.....	44
DIAGRAMAS DE CLASES.....	51
DIAGRAMAS DE ESTADO Y ACTIVIDADES	61
DIAGRAMAS DE DESPLIEGUE.....	64
DIAGRAMAS DE BASE DE DATOS DE SOFTWARE	68
DIAGRAMAS DE INTERFACES DEL SOFTWARE.....	73

Página | 2

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

DIAGRAMAS DE MENU DEL SOFTWARE.....	77
DISEÑO DE INTERFAZ ACCESO DEL SOFTWARE.....	80
PRUEBA DEMOSTRATIVA DEL SOFTWARE.....	82
PROCESO DE RECUPERACION.....	84
HERRAMIENTAS O PROGRAMAS PARA TRABAJAR CON UML.....	85
Bibliografía.....	86

Página | 3



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

METODOLOGIA DE DESARROLLO DE SOFTWARE 1ER SEMESTRE

UNIDAD 1

Página | 1

INTRODUCCION

La ingeniería del software posee técnicas y herramientas que han madurado mucho actualmente se invierten grandes cantidades de recursos para potenciar la industria del software y convertirla en uno de los sectores estratégicos de crecimiento.

La obtención de un software de calidad implica la utilización de modelos o procedimientos estándares para su análisis, diseño, desarrollo y prueba que permitan uniformar la filosofía de trabajo, para lograr una mayor confiabilidad mantenimiento y facilidad de prueba. A su vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software. Se puede considerar que un software es de calidad si cumple los siguientes objetivos: concordancia con los requerimientos, desarrollo coherente y desarrollo de requerimientos implícitos en el proyecto.

En el campo del desarrollo de software, existen dos grupos de metodologías, las denominadas tradicionales (formales) y las ágiles.

Las primeras son un tanto rígidas, exigen una documentación exhaustiva y se centran en cumplir con el plan del proyecto definido totalmente en la fase inicial del desarrollo del mismo; mientras que la segunda enfatiza el esfuerzo en la capacidad de respuesta a los cambios, las habilidades del equipo y mantener una buena relación con el usuario.

Ambas propuestas tienen sus propias ventajas y desventajas; de cualquier manera, las metodologías de desarrollo nos dicen el ¿Qué hacer? más no el ¿Cómo hacer?, esto significa que la metodología que elijamos, debe ser adaptada al contexto del proyecto, teniendo en cuenta los recursos técnicos y humanos; tiempo de desarrollo y tipo de sistema.

CONCEPTOS GENERALES

Términos y Definiciones

- *Área de desarrollo.*- Se refiere a una unidad operativa en la cual se desarrollan productos de software. Por ejemplo: Coordinación de Transferencia Tecnológica Interna, Coordinación de Transferencia Tecnológica Externa y Coordinación de Tecnología Web son áreas de desarrollo.
- *Cliente.*-Es la persona o empresa receptora de un bien, servicio, producto o idea, a cambio de dinero u otro artículo de valor

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

- *Coordinador.*-Es el responsable de gestionar el proyecto. Para mayor información revise la sección Roles.
- *Desarrollador.*-Es un especialista en informática que es capaz de concebir y elaborar sistemas informáticos; así como de implementarlos y ponerlos a punto, utilizando el lenguaje de programación definido. Para mayor información revise la sección Roles.
- *Desarrollador Líder.*-Tiene como función realizar el seguimiento y el control del cronograma de actividades. Para mayor información revise la sección Roles.
- *Entregables.*- Son los productos intermedios que generan las fases. Los entregables permiten evaluar la marcha del proyecto mediante comprobaciones de su adecuación o no a los requisitos funcionales y de condiciones de realización previamente establecidos.
- *Método.*- Un método se compone de diversos aspectos que nos permitirán conseguir una meta o lograr un objetivo. Se define más claramente como un conjunto de herramientas, las cuales, utilizadas mediante las técnicas correctas, permiten la ejecución de procesos que nos llevarán a cumplir los objetivos que buscamos.
- *Metodología.*- Hace referencia al conjunto de procedimientos racionales utilizados para alcanzar el objetivo o la gama de objetivos que rige una investigación científica, una exposición doctrinal o tareas que requieran habilidades, conocimientos o cuidados específicos. Con frecuencia puede definirse la metodología como el estudio o elección de un método pertinente o adecuadamente aplicable a determinado objeto.
- *Metodología de Desarrollo de Software.*- Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. En un proyecto de desarrollo de software la metodología ayuda a definir: Quién debe hacer Qué Cuándo y Cómo debe hacerlo. La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende actividades a seguir para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado.
- *Módulo Informático.*- Es una porción de software que realiza tareas específicas dentro de un sistema. Por ejemplo: módulo de matrícula.
- *Proyecto.*-Según la definición que nos proporciona PMI en su guía PMBOOK, un proyecto se podría definir como “un servicio temporal que se lleva a cabo para crear un producto, servicio o resultado único” Podemos decir entonces que un proyecto tiene un inicio y un fin, este fin se tiene que alcanzar dentro de un tiempo fijado.
- *Software.*- IEEE Std. 610 define el software como “programas, procedimientos, documentación y datos asociados, relacionados con la operación de un sistema informático” El software se puede definir como el conjunto de tres componentes: Programas (instrucciones): este componente proporciona la funcionalidad deseada y el

Página | 2

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

rendimiento cuando se ejecute. Datos: este componente incluye los datos necesarios para manejar y probar los programas y las estructuras requeridas para mantener y manipular estos datos. Documentos: este componente describe la operación y uso del programa.

Página | 3

Componentes del Software:

Es importante contar con una definición exhaustiva del software ya que de otra manera se podrían olvidar algunos componentes. Una percepción común es que el software sólo consiste en programas. Sin embargo, los programas no son los únicos componentes del software.

Programas

Los programas son conjuntos de instrucciones que proporcionan la funcionalidad deseada cuando son ejecutadas por el ordenador. Están escritos usando lenguajes específicos que los ordenadores pueden leer y ejecutar, tales como lenguaje ensamblador, Basic, FORTRAN, COBOL, C... Los programas también pueden ser generados usando generadores de programas.

Datos

Los programas proporcionan la funcionalidad requerida manipulando datos. Usan datos para ejercer el control apropiado en lo que hacen. El mantenimiento y las pruebas de los programas también necesitan datos. El diseño del programa asume la disponibilidad de las estructuras de datos tales como bases de datos y archivos que contienen datos.

Documentos

Además de los programas y los datos, los usuarios necesitan también una explicación de cómo usar el programa. Documentos como manuales de usuario y de operación son necesarios para permitir a los usuarios operar con el sistema. Los documentos también son requeridos por las personas encargadas de mantener el software para entender el interior del software y modificarlo, en el caso en que sea necesario.

- *Stakeholders.*- Es una palabra del inglés que, en el ámbito empresarial, significa 'interesado' o 'parte interesada', y que se refiere a todas aquellas personas u organizaciones afectadas por las actividades y las decisiones de una empresa. En toda organización, además de sus propietarios, participan diversos actores claves y grupos sociales que están constituidos por las personas o entes que, de una manera y otra, tienen interés en el desempeño de una empresa porque están relacionadas directa o indirectamente, con ella.
- *Usuario.*- Es un usuario quien tiene un conjunto de permisos y de recursos (o dispositivos) a los cuales se tiene acceso. Es decir, un usuario puede ser tanto una persona como una máquina, un programa, etc. Para efecto del presente documento se utilizará "Usuario" para referirse a la persona con la que se establece comunicación para tratar temas relacionados a los requerimientos.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

INGENIERÍA DE SOFTWARE

El desarrollo de software, es uno de los sectores tecnológicos más competitivos y no es algo nuevo, ya que durante muchos años lo ha sido, sin embargo ha tenido una evolución constante en lo que se refiere a las metodologías o bien, las formas en las cuales se realiza la planeación para el diseño del software, básicamente con el objetivo de mejorar, optimizar procesos y ofrecer una mejor calidad.

Página | 4

Todas las empresas desarrolladoras de proyectos o servicios software asumen que, la implantación de una metodología es necesaria si se quieren gestionar adecuadamente los proyectos o servicios software. “...La metodología no debe adaptarse a la organización, sino que es la organización la que debe adaptarse a la metodología...”.

Sin embargo, antes de hablar acerca de metodologías y todo este tema tan amplio, analicemos a detalle brevemente **¿Qué es un método?** y para que lo acompañemos también veamos **¿qué es una metodología?** Seguramente los términos te suenan familiar, sin embargo el saber que significan de forma correcta es indispensable.

Método

Un Método se compone de diversos aspectos que nos permitirán conseguir una meta o lograr un objetivo. Se define más claramente como un conjunto de herramientas, las cuales utilizadas mediante las técnicas correctas, permiten la ejecución de procesos que nos llevarán a cumplir los objetivos que buscamos. En pocas palabras y aunque esto lo puedes encontrar como tal en internet, es un conjunto de herramientas, técnicas y procesos que facilitan la obtención de un objetivo.

Metodología

Se define como el enfoque de un problema de manera total, organizada, sistemática y disciplinada. Esta definición muestra una distinción entre “metodología” y “técnica”. La técnica se considera como un componente de la metodología, como el medio o procedimiento que se usa para realizar la metodología misma.

Una metodología es un conjunto de procedimientos, técnicas, herramientas y un **soporte documental** que ayuda a los desarrolladores a realizar un nuevo software. 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.

Finalmente dependerá de la metodología utilizada los productos del proyecto, por esta razón es necesario conocer a fondo cada una de ellas y poder diferenciar entre una y otra, para de este modo saber elegir la correcta en el momento de desarrollar un nuevo software, de otra manera el producto no será el mejor e incluso puede ser inútil.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

En el desarrollo de software, una metodología hace cierto énfasis al entorno en el cuál se plantea y estructura el desarrollo de un sistema. Como se mencionó al principio, existen una gran cantidad de metodologías de la programación que se han utilizado desde los tiempos atrás y que con el paso del tiempo han ido evolucionando. Esto se debe principalmente a que no todos los sistemas de la información, son compatibles con todas las metodologías, pues el ciclo de vida del software puede ser variable. Por esta razón, es importante que dependiendo del tipo de software que se vaya a desarrollar, se identifique la metodología para el diseño de software idónea.

Página | 5

METODOLOGIAS DE DESARROLLO DE SOFTWARE

Selección de sistema de información para el desarrollo del software

Una Metodología de desarrollo de software, consiste principalmente en hacer uso de diversas herramientas, técnicas, métodos y modelos para el desarrollo. Regularmente este tipo de metodología, tienen la necesidad de venir documentadas, para que los programadores que estarán dentro de la planeación del proyecto, comprendan perfectamente la metodología y en algunos casos el ciclo de vida del software que se pretende seguir.

Aunque actualmente existen mucha variedad en metodologías de programación. La realidad es que todas están basadas en ciertos enfoques generalistas que se crearon hace muchos años, algunos tipos de metodologías de desarrollo de software que se utilizaron e inventaron al principio de nuestra era tecnológica y son las que veremos a continuación.

Recopilación de la información del desarrollo de software

La recolección de datos se refiere al uso de una gran diversidad de técnicas y herramientas que pueden ser utilizadas por el analista para desarrollar los sistemas de información, los cuales pueden ser la entrevistas, la encuesta, el cuestionario, la observación, el diagrama de flujo y el diccionario de datos.

Todos estos instrumentos se aplicarán en un momento en particular, con la finalidad de buscar información que será útil a una investigación en común.

TECNICAS

Entrevista

Las entrevistas se utilizan para recabar información en forma verbal, a través de preguntas que propone el analista. Quienes responden pueden ser gerentes o empleados, los cuales son usuarios actuales del sistema existente, usuarios potenciales del sistema propuesto o aquellos que proporcionarán datos o serán afectados por la aplicación propuesta. El analista puede entrevistar al personal en forma individual o en grupos algunos analistas prefieren este método a las otras técnicas que se estudiarán

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

más adelante. Sin embargo, las entrevistas no siempre son la mejor fuente de datos de aplicación.

Cuestionario

Los cuestionarios proporcionan una alternativa muy útil para la entrevista; si embargo, existen ciertas características que pueden ser apropiada en algunas situaciones e inapropiadas en otra. Al igual que la entrevistas, deben diseñarse cuidadosamente para una máxima efectividad.

Página | 6

Cuestionario Abierto:

Al igual que las entrevistas, los cuestionarios pueden ser abiertos y se aplican cuando se quieren conocer los sentimientos, opiniones y experiencias generales; también son útiles al explorar el problema básico, por ejemplo, un analista que utiliza cuestionarios para estudiar los métodos de verificación de crédito, es un medio.

El formato abierto proporciona una amplia oportunidad para quienes respondan escriba las razones de sus ideas. Algunas personas, sin embargo, encuentran más fácil escoger una de un conjunto de respuestas preparadas que pensar por sí mismas.

Cuestionario laboral realizado para evaluar a un candidato a un puesto de trabajo.

1. Describe tu experiencia laboral previa relacionada con el puesto que estás solicitando.
2. ¿Cómo te describirías a ti mismo?
3. ¿Cuáles son tus mayores aptitudes y herramientas laborales?
4. ¿Qué defectos consideras que tienes?
5. ¿Qué estrategias implementarías para desempeñar tu labor en caso de ser contratado para el puesto solicitado?
6. ¿Cómo describirías las actividades principales del puesto solicitado?
7. ¿En qué consideras que se vería beneficiada la empresa al contratarte para el puesto de trabajo solicitado?
8. ¿Por qué te interesa laborar con nosotros?
9. ¿Qué opinas de nuestra empresa?

En el cuestionario abierto, la persona encuestada desarrolla su respuesta, de la que el encuestador toma nota. En este caso, la encuesta de cuestionario se parece a una entrevista individual de tipo direccional.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

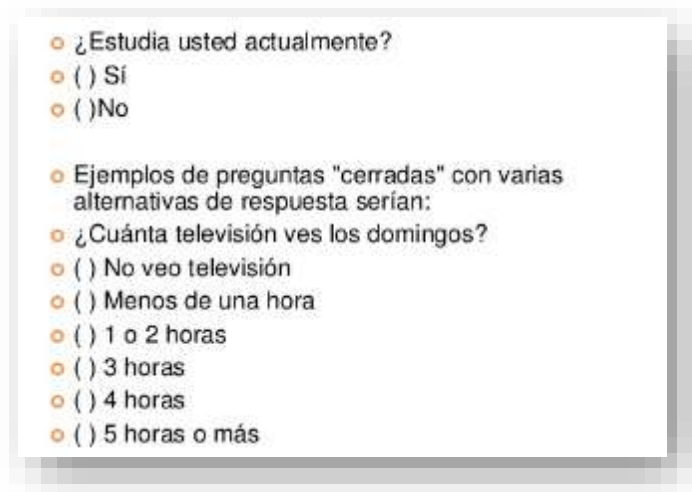
Cuestionario Cerrado:

El cuestionario cerrado limita las respuestas posibles del interrogado.

Por medio de un cuidadoso estilo en la pregunta, el analista puede controlar el marco de referencia. Este formato es el método para obtener información sobre los hechos.

También fuerza a los individuos para que tomen una posición y forma su opinión sobre los aspectos importantes.

Página | 7



Las preguntas son aquellas interpelaciones que un emisor le hace a uno o varios receptores con el objeto de obtener alguna información como respuesta. Las preguntas cerradas son aquellas que ya han configurado las opciones que tendrá el encargado de responder, que solo deberá optar entre una de ellas.

Encuesta

La encuesta es un instrumento de investigación para obtener información representativa de un grupo de personas. Se trata de aplicar un cuestionario a determinado número de individuos, con el objeto de obtener un resultado.

El requisito es que debe aplicarse a un número representativo.

Forma parte de una investigación; es sólo un instrumento.

Por ejemplo, para conocer si los jóvenes piensan votar en las próximas elecciones, puedes realizar una encuesta entre 10 o 20 chicos de las principales universidades del Ecuador.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

¿Qué prendas te resulta mas difícil conseguir?

<input type="checkbox"/> Pantalones	<input type="checkbox"/> Traje Sastre
<input type="checkbox"/> Faldas	<input type="checkbox"/> Blusas
<input type="checkbox"/> Shorts	<input type="checkbox"/> Playeras
<input type="checkbox"/> Pants	<input type="checkbox"/> Sacos
<input type="checkbox"/> Otro (explique)	

¿Qué marcas de ropa conoces que tengan tallas grandes?

<input type="checkbox"/> Forever 21
<input type="checkbox"/> Rainbow
<input type="checkbox"/> H&M
<input type="checkbox"/> Ninguna
<input type="checkbox"/> Otro (explique)

¿Qué aspectos crees que deben tener las tallas grandes? Marca todas las que creas necesarias.

<input type="checkbox"/> Precio accesible
<input type="checkbox"/> Diseños atractivos
<input type="checkbox"/> Comodidad
<input type="checkbox"/> Moda actual
<input type="checkbox"/> Facilidad de conseguir
<input type="checkbox"/> Otro (explique)

¿Crees que nos falta algo respecto a este tema?

Gracias por tu valioso tiempo, esperamos poder servirte nuevamente.

Fecha _____

Página | 8

La observación

Otra técnica útil para el analista en su progreso de investigación, consiste en observar a las personas cuando efectúan su trabajo.

Como técnica de investigación, la observación tiene amplia aceptación científica.

Los sociólogos, sicólogos e ingenieros industriales utilizan extensamente ésta técnica con el fin de estudiar a las personas en sus actividades de grupo y como miembros de la organización.



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

El propósito de la organización es múltiple: permite al analista determinar que se está haciendo, como se está haciendo, quien lo hace, cuando se lleva a cabo, cuanto tiempo toma, dónde se hace y porque se hace.

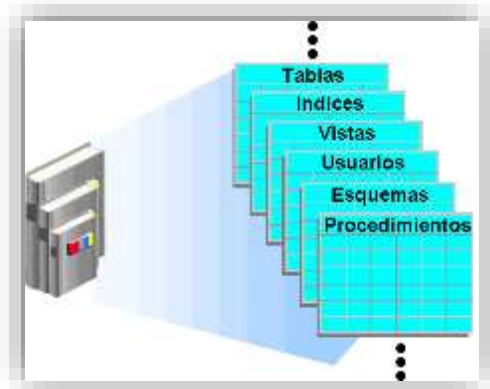
Diccionario de Datos

Los diccionarios de datos son el segundo componente del análisis de flujo de datos, el diccionario de datos proporciona información adicional sobre el sistema.

El diccionario de datos se almacena detalles y descripciones de los elementos.

El desarrollo de datos se desarrolla durante el flujo de datos y ayuda al analista involucrado en la determinación de requerimientos del sistema.

Para comprender mejor el significado de diccionario de datos puede considerarse su contenido como “Datos acerca de los datos”; es decir descripciones de todos los demás objetos (Archivos, programas, informes, sinónimos) existentes en un sistema.





INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Nombre de Archivo: APEmpleado.		Fecha de Creación: 23/07/2007	
Descripción: Archivo Principal de Empleados, contendrá información de cada uno de ellos.			
Campo	Tipo	Tamaño	Descripción
cveEmpleado	Númerico	12	Clave de Empleado
cNombre	Carácter	50	Nombre del empleado
nSueldo	Númerico	10:2	Sueldo del empleado con dos decimales para fraccionarios
cveDpto	Númerico	5	Clave del departamento a donde pertenece el empleado.
cveArea	Númerico	5	Clave del área donde se encuentra el empleado.
cdespuesto	Carácter	75	Descripción del puesto que ocupa.
Relaciones: CatDepto con el campo cveDpto. CatArea con el campo cveArea.		Campos Clave: cveEmpleado, cveDpto, cveArea	

Página | 10

Descripción de los datos en el diccionario

Cada entrada en el diccionario de datos consiste en un conjunto de detalles que describe los datos utilizados o producidos en el sistema.

Nombre de los datos

Para distinguir un dato de otro, los analistas le asignan nombres significativos que se utilizan para tener una referencia de cada elemento

Ejemplo:

La fecha de factura es más significativa si se llama FECHA FACTURA

Que si se le conoce como ABCXXX.

Descripción de los Datos

Establece brevemente lo que representa el Dato en el sistema.

Deben evitarse términos especiales o argot, todas las palabras deben ser entendibles para el lector.

Longitud del campo

Será importante conocer la cantidad del espacio que necesite para cada Dato.

Valores de los Datos

En algunos procesos solo se permiten valores de datos específicos.

Registro de las descripciones de los Datos

Dadas que las descripciones se utilizan en forma repetitiva a través de una información y después, durante el diseño

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Otros

Diagramas de Flujo

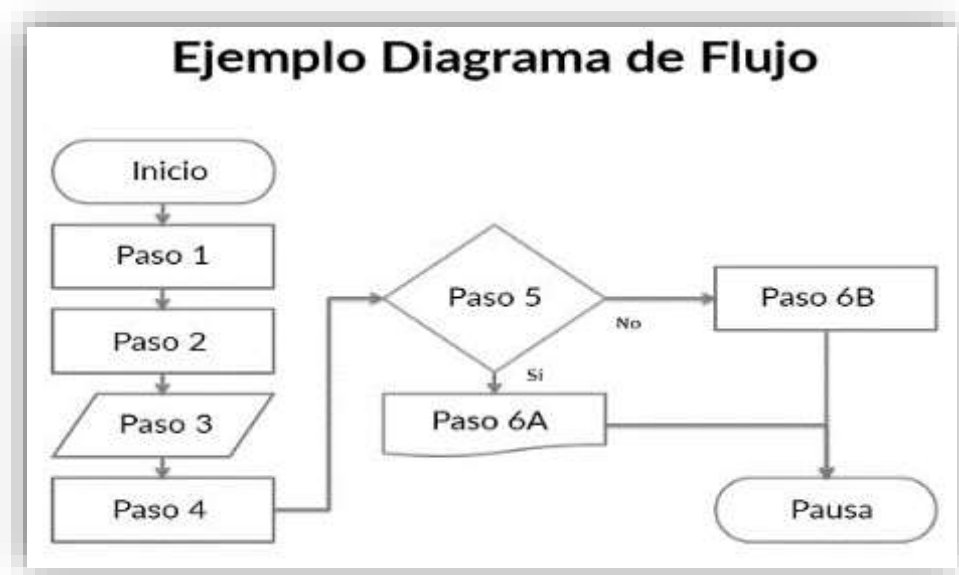
Es una representación pictórica de los pasos en proceso. Útil para determinar cómo funciona realmente el proceso para producir un resultado.

Página | 11

El resultado puede ser un producto, un servicio, información o una combinación de los tres. Al examinar cómo los diferentes pasos en un proceso se relacionan entre sí, se puede descubrir con frecuencia las fuentes de problemas potenciales.

Los diagramas de flujo se pueden aplicar a cualquier aspecto del proceso desde el flujo de materiales hasta los pasos para hacer la venta u ofrecer un producto.

Con frecuencia este nivel de detalle no es necesario, pero cuando se necesita, el equipo completo de trabajos más pequeños puede agregar niveles según sea necesario durante el proyecto.



¿Cuándo se utiliza un Diagrama de Flujo?

Cuando un equipo necesita ver cómo funciona realmente un proceso completo. Este esfuerzo con frecuencia revela problemas potenciales tales como cuellos de botella en el sistema, pasos innecesarios y círculos de duplicación de trabajo.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Procesamiento de la información recopilada

Es la técnica que consiste en la recolección de los datos primarios de entrada, que son evaluados y ordenados, para obtener información útil, que luego serán analizados por el usuario final, para que pueda tomar las decisiones o realizar las acciones que estime conveniente.

Página | 12

Las etapas para el Procesamiento de Datos son las siguientes:

Entrada:

Los datos deben ser obtenidos y llevados a un bloque central para ser procesados. Los datos en este caso, denominados de entrada, son clasificados para hacer que el proceso sea fácil y rápido.

Proceso:

Durante el proceso se ejecutarán las operaciones necesarias para convertir los datos en información significativa. Cuando la información esté completa se ejecutará la operación de salida, en la que se prepara un informe que servirá como base para tomar decisiones.

Salida:

En todo el procesamiento de datos se plantea como actividad adicional, la administración de los resultados de salida, que se puede definir como los procesos necesarios para que la información útil llegue al usuario.

La función de control asegura que los datos estén siendo procesados en forma correcta.

CODIFICACIÓN

La codificación consiste en asignar un código numérico a cada una de las alternativas de las preguntas del instrumento (cuestionario o guía) y de esta manera facilitar la tabulación y conteo de los datos.

Codificación de una pregunta cerrada: Para esta tipo de pregunta, dicho código se asigna en el momento que se diseña el instrumento.

Codificación de preguntas abiertas: las preguntas abiertas se codifican después de haber recopilado los datos, de manera de asignar los códigos a las respuestas dadas por los informantes.

TABULACION

La tabulación de los datos consiste en el recuento de las respuestas contenidas en los instrumentos, a través del conteo de los códigos numéricos de las alternativas de las preguntas cerradas y de los códigos asignados a las respuestas de las preguntas abiertas, con la finalidad de generar resultados que se muestran en cuadros (o tablas) y en gráficos. La tabulación puede ser tratada de forma manual o mecánica.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Tabulación manual: Es recomendada cuando el tamaño de la muestra es muy pequeño, sin embargo, la descripción del proceso de este tipo de tabulación, ayudaría a evitar posibles errores en la presentación de los resultados al procesar los datos a través de una tabulación electrónica.

Tabulación electrónica: cuando los datos por tabular sean un número relativamente grande es recomendable acudir al procesamiento electrónico de datos, a través de Excel o de paquetes de computación que facilitan la realización de cuadros o tablas estadísticas sencillas (de una variable) y cruzadas (dos o más variables).

Página | 13

REGISTRO DE DATOS

Un registro es un conjunto de campos que contienen los datos que pertenecen a una misma repetición de entidad. Se le asigna automáticamente un número consecutivo (número de registro) que en ocasiones es usado como índice, aunque lo normal y práctico es asignarle a cada registro un campo clave para su búsqueda.

En informática, o concretamente en el contexto de una base de datos relacional, un registro (también llamado fila o tupla) representa un objeto único de datos implícitamente estructurados en una tabla. En términos simples, una tabla de una base de datos puede imaginarse formada de filas y columnas o campos. Cada fila de una tabla representa un conjunto de datos relacionados, y todas las filas de la misma tabla tienen la misma estructura.

Presentación de datos

La presentación de datos estadísticos constituye en sus diferentes modalidades uno de los aspectos de más uso en la estadística descriptiva. A partir podemos visualizar a través de los diferentes medios escritos y televisivos de comunicación masiva la presentación de los datos estadísticos sobre el comportamiento de las principales variables económicas y sociales, nacionales e internacionales.

Presentación escrita: Esta forma de presentación de informaciones se usa cuando una serie de datos incluye pocos valores, por lo cual resulta más apropiada la palabra escrita como forma de escribir el comportamiento de los datos; mediante la forma escrita, se resalta la importancia de las informaciones principales.

Presentación tabular: Cuando los datos estadísticos se presentan a través de un conjunto de filas y de columnas que responden a un ordenamiento lógico; es de gran uso e importancia para el usuario ya que constituye la forma más exacta de presentar las informaciones. Una tabla consta de varias partes, las principales son las siguientes:

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Título: Es la parte más importante del cuadro y sirve para describir todo el contenido de este.

Encabezados: Son los diferentes subtítulos que se colocan en la parte superior de cada columna.

Columna matriz: Es la columna principal del cuadro.

Cuerpo: El cuerpo contiene todas las informaciones numéricas que aparecen en la tabla.

Fuente: La fuente de los datos contenidos en la tabla indica la procedencia de estos.

Notas al pie: Son usadas para hacer algunas aclaraciones sobre aspectos que aparecen en la tabla o cuadro y que no han sido explicados en otras partes.

Presentación gráfica: Proporciona al lector o usuario mayor rapidez en la comprensión de los datos, una gráfica es una expresión artística usada para representar un conjunto de datos. De acuerdo al tipo de variable que vamos a representar, las principales graficas son las siguientes:

- **Histograma:** Es un conjunto de barras o rectángulos unidos uno de otro, en razón de que lo utilizamos para representar variables continuas.
- **Polígono de frecuencias:** Esta grafica se usa para representar los puntos medios de clase en una distribución de frecuencias.
- **Gráfica de barras:** Es un conjunto de rectángulos o barras separadas una de la otra, en razón de que se usa para representar variables discretas; las barras deben ser de igual base o ancho y separadas a igual distancia. Pueden disponerse en forma vertical y horizontal.
- **Gráfica lineal:** Son usadas principalmente para representar datos clasificados por cantidad o tiempo; o sea, se usan para representar series de tiempo o cronológicas.
- **Gráfica de barra 100% y gráfica circular:** Se usan especialmente para representar las partes en que se divide una cantidad total.
- **La ojiva:** Esta grafica consiste en la representación de las frecuencias acumuladas de una distribución de frecuencias. Puede construirse de dos maneras diferentes; sobre la base "menor que" o sobre la base "o más". Puede determinar el valor de la mediana de la distribución.

Análisis de la problemática del sistema actual

Introducción al Diseño de Sistemas

“Diseño es el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso, o sistema, con los suficientes detalles como para permitir su realización física” El objetivo del diseñador es producir un modelo de una entidad que se construirá más

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

adelante. La actividad de Diseño se dedica a asignar porciones de la especificación estructurada (también conocida como modelo esencial) a procesadores adecuados (sean máquinas o humanos) y a labores apropiadas (o tareas, particiones, etc.)

Página | 15

El Diseño de sistemas se define el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un Sistema, con suficientes detalles como para permitir su interpretación y realización física.

La etapa del Diseño del Sistema encierra cuatro etapas:

El diseño de los datos

Trasforma el modelo de dominio de la información, creado durante el análisis, en las estructuras de datos necesarios para implementar el Software.

El Diseño Arquitectónico

Define la relación entre cada uno de los elementos estructurales del programa.

El Diseño de la Interfaz

Describe como se comunica el Software consigo mismo, con los sistemas que operan junto con él y con los operadores y usuarios que lo emplean.

El Diseño de procedimientos

Transforma elementos estructurales de la arquitectura del programa. La importancia del Diseño del Software se puede definir en una sola palabra Calidad, dentro del diseño es donde se fomenta la calidad del Proyecto. El Diseño es la única manera de materializar con precisión los requerimientos del cliente.

Análisis de Sistemas

Es un conjunto o disposición de procedimientos o programas relacionados de manera que juntos forman una sola unidad. Un conjunto de hechos, principios y reglas clasificadas y dispuestas de manera ordenada mostrando un plan lógico en la unión de las partes. Un método, plan o procedimientos de clasificación para hacer algo. También es un conjunto o arreglo de elementos para realizar un objetivo predefinido en el procesamiento de la información. Esto se lleva a cabo teniendo en cuenta ciertos principios:

- Debe presentarse y entenderse el dominio de la información de un problema.
- Defina las funciones que debe realizar el software.
- Represente el comportamiento del software a consecuencias de acontecimientos externos.
- Divida en forma jerárquica los modelos que representan la información, funciones y comportamiento.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

CONCEPTUALIZACIÓN DE SISTEMAS

Los sistemas agropecuarios son complejos y sólo a través de una metodología con ciertos pasos lógicos y ordenados es posible empezar a entenderlos. Para poder intercambiar ideas y apoyarnos mutuamente en el proceso de aprendizaje es necesario usar un vocabulario común. En otras palabras debemos contar con la habilidad de usar ciertos instrumentos para conceptualizar un determinado sistema.

Página | 16

Spedding (1975) sugiere nueve consideraciones que deben ser tomadas en cuenta para realizar la conceptualización de un sistema; estas son:

- El propósito
- El límite
- El contorno
- Los componentes
- Las interacciones
- Los recursos
- Los ingresos o insumos
- Los egresos o salidas
- Los subproductos

PLANTEAMIENTO DEL PROBLEMA

DESCRIPCIÓN DEL PROBLEMA

Un problema es todo aquello cuya solución se desconoce; ese desconocimiento puede ser para un grupo de personas o para la humanidad. Para la formulación correcta de un problema se debe tener en cuenta los siguientes aspectos: Aquello donde exista una situación actual que se desea mejorar, pero se desconoce la manera de lograrlo. Una situación actual indeseable, que se desea cambiar o modificar. Un problema debe expresarse en términos concretos y explícitos a través del planteamiento, formulación y sistematización.

PLANTEAMIENTO DEL PROBLEMA.

Es la descripción de la “Situación actual” que caracteriza al “objeto de conocimiento”. Aquí se describen los síntomas y las posibles causas, la identificación de las situaciones futuras si se sostiene dicha situación y la presentación de alternativas para superar la situación actual, las cuales permitan controlar el pronóstico planteado.

FORMULACIÓN DEL PROBLEMA.

Se plantea a través de una pregunta de investigación, la cual el investigador espera responderla y de esta forma resolver el problema planteado.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

SISTEMATIZACIÓN DEL PROBLEMA.

Se formula a través de subpreguntas que el investigador plantea sobre tópicos específicos relacionados y que se han observado en el planteamiento del problema. Pasos para realizar el planteamiento, formulación y sistematización del problema.

Página | 17

1. Antes de iniciar el planteamiento del problema usted debe tener:
 - a. **Un tema definido.** Materia o asunto que se investiga. Responde a la pregunta ¿Qué se va a estudiar?
 - b. **Área de investigación.** Se refiere al campo o disciplina científica o tecnológica que sirve de base a la investigación. Responde a la pregunta ¿Desde qué ángulo se realiza la investigación?
 - c. **Una bibliografía básica**
 - d. **Un profesor asesor**
 - e. **Fichas de lectura**
 - f. **Definido el ámbito espacial** (empresa, región, organización, sector económico, país etc.)
 - g. Apoyo de los miembros del ámbito espacial en el cual se desarrollará el trabajo.
 - h. Información preliminar acerca de los eventos que suceden en el ámbito espacial.
2. **Realizar el planteamiento del problema.** Guía Anteproyecto de Grado – Ingeniería de Sistemas - Este paso se inicia con un diagnóstico de la situación actual, preguntándose ¿Qué pasa con su objeto de investigación? El diagnóstico se fundamenta en:
 - a. La identificación de los hechos o situaciones que se observan al analizar el objeto de investigación. Estos son los síntomas del problema.
 - b. Con la lista de los síntomas del paso anterior identifique hechos o situaciones que producen los síntomas. Estos son las causas del problema.



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

UNIDAD 2

APLICACIÓN DE LA METODOLOGIA DEL DESARROLLO DE SOFTWARE

METODOLOGIAS TRADICIONALES

Como se mencionó anteriormente, regularmente, cada metodología de desarrollo de software, tiene un enfoque bien marcado, estos enfoques no son para nada nuevos y se siguen utilizando para la planeación y desarrollo de software aún en nuestros tiempos, así que vamos a ver cuáles son cada uno de ellos y aprenderemos como funcionan.

Página | 18

Metodología en cascada: Framework lineal.

El modelo de desarrollo de Software en cascada, es una metodología de la programación muy antigua. Si bien su creador nunca lo menciona como metodología en cascada, el funcionamiento y lineamiento de los procesos de la planeación, son exactamente iguales. Básicamente, el estilo del modelo en cascada, es que no podrás avanzar a la siguiente fase, si la anterior no se encuentra totalmente terminada, pues no tiene porque haber vuelta atrás. Vamos a ver cuáles son las fases de desarrollo de software del modelo en cascada, para que te puedas dar una idea.

1. **Análisis de Requisitos.** El primer nivel del modelo cascada, es el análisis de requisitos. Básicamente lo que se documenta aquí, son los objetivos de lo que el software debe hacer al terminar el desarrollo, sin entrar en detalles de la parte interna, los cuales se verán durante el proceso. Sin embargo es importante señalar que una vez avanzado el paso de análisis, no puede haber vuelta atrás, pues la metodología para el diseño de software con la cual se trabaja no lo permitirá.
2. **Diseño del Sistema.** Todo lo que conlleva el armado de un diseño para el sistema que vayas a utilizar, es lo que continua después del análisis de requisitos. Aquí se elaborará lo que es la estructura del sistema y se determinarán las especificaciones para cada una de las partes del sistema que se planea desarrollar. Siendo del mismo modo, una fase a la cuál ya no se podrá volver después de haber bajado al nivel 3.
3. **Diseño del Programa - Implementación.** En este punto, aún no ingresamos a lo que es la escritura de código, sin embargo ya se realizan los algoritmos que se van a utilizar en la programación. Si bien recuerdas, un algoritmo no necesariamente es código, simplemente tomas una hoja de papel y escribes el algoritmo que vas a utilizar. Esto es precisamente lo que se realiza en el paso número 3.
4. **Codificación.** Seguramente como programador, esta es la parte que estaba esperando, pues ahora es momento de empezar a escribir todo el código que será necesario para el desarrollo del software. Para este punto, la velocidad y el tiempo que se requiera, dependerá mucho del lenguaje de programación que se vaya a utilizar. Pues algunos lenguajes de programación permiten utilizar componente, bibliotecas e incluso algunas funciones para reutilizar código, las cuales podrán acelerar el proceso de codificación en gran manera.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

5. **Ejecución de Pruebas.** La codificación ha terminado y ahora es momento de verificar que nuestro sistema es realmente funcional, antes de que el cliente empiece a utilizarlo. Este es precisamente el objetivo de la fase 5 de pruebas. Aquí es recomendable que se intente mover lo más que se pueda su software, con el objetivo de dañarlo intencionalmente, de este modo, si supera las pruebas de daño realizadas, entonces estará listo para el usuario final.
6. **Verificación.** Después de haber realizado una gran cantidad de pruebas en la Fase 5, debemos migrar a la verificación. Esta fase consiste en la ejecución del Software por parte del usuario final. Si la fase cinco se realizó correcta y profundamente, el software no tendrá ningún tipo de problema y el usuario final quedará satisfecho con el resultado.
7. **Mantenimiento.** Seguramente se dará cuenta, de que las aplicaciones o el software actual, constantemente se está actualizando. Esto se debe principalmente a que se le da mantenimiento al software, se solucionan errores, se quitan algunos bugs, se añaden funcionalidades, todo después de que el usuario final ya lo ha probado y utilizado en su fase final. Esta es posiblemente una de las fases más tediosas del modelo de desarrollo de software, pues se debe estar atento a los comentarios de los usuarios, para ver qué cosas son las que no funcionan correctamente y a las cuales hay que darles mantenimiento.

Página | 19

Principios Básicos Del Modelo De Cascada

Como se puede ver, el proceso de desarrollo de software con el modelo de cascada es bastante complejo. Sin embargo, uno de sus principios es que cada una de las fases elaboradas, se encuentre documentada perfectamente, de este modo, si el desarrollo queda suspendido en alguna fase, cualquier usuario que quiera continuar con el proyecto lo podrá hacer leyendo la documentación.

Así también, es muy común encontrar metodologías para el desarrollo de software en cascada con fechas de objetivos, tiempos o presupuestos para determinadas fases. Aprovechando el hecho de que una vez que avanzaste de fase, es muy poco recomendable el volver atrás, aunque regularmente se tiene un cierto nivel de tolerancia, pero lo correcto en la utilización del modelo de cascada, es que no puedas ir atrás a realizar modificaciones de ningún tipo.



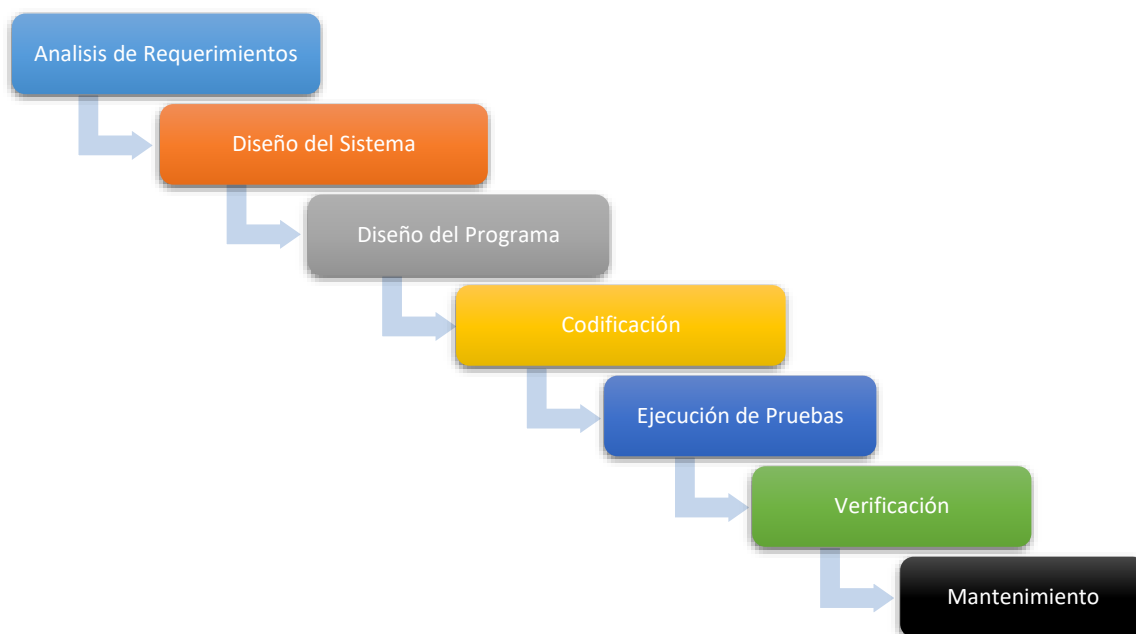
INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



Página | 20

Diseño Metodología en Cascada

Método de Prototipos

Esta metodología de la programación todavía sigue siendo la favorita de muchos. Consiste básicamente en que en base a los requerimientos y necesidades que tiene el cliente, se realiza de forma rápida un prototipo, este no vendrá completo ni mucho menos terminado, pero si permitirá contar con las bases necesarias para que cualquier programador pueda seguir trabajando en el hasta llegar al código final.

Por si no lo sabes aún, un prototipo es una versión no terminada del producto que se le entregará al cliente o usuario final. Esto nos genera cierta ventaja en el desarrollo de productos similares con funciones distintas, por ejemplo. Supongamos que vas a desarrollar un proyecto para 3 clientes distintos, ambos con una estructura idéntica, pero con funcionalidades muy distintas, entonces lo que hacemos es crear un prototipo base y entorno a él mostrarlo a nuestros clientes para que de ahí se empiecen a desarrollar las demás funciones.

Mejor vamos a ver cuáles son las etapas de desarrollo de software por las cuales tendrás que pasar, en caso de utilizar la metodología de prototipos.

1. **Planeación.** A diferencia de otras metodologías, la planeación debe ser muy rápida, en esta fase no puedes demorarte mucho, pues recuerda que solamente será un prototipo por el momento.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

2. **Modelado.** Nuevamente, una fase que deberá ser suficientemente rápida como para que nos quite nada de tiempo. Hacer el modelado será simple y te sigo recordando que solamente es un prototipo, al menos por ahora.
3. **Elaboración del Prototipo.** Ya que contamos con la planeación de lo que vamos a realizar y el modelado rápido, entonces es momento de elaborar el prototipo. Para esta instancia, ya no te diré que lo debes hacer rápido, puesto que te tomará el tiempo que tenga sea necesario elaborarlo, recuerda que este ya se muestra al cliente, así que ya es una fase importante.
4. **Desarrollo.** Posterior a contar con el prototipo elaborado y mostrado al cliente, es momento de comenzar el desarrollo. Este te tomará una gran cantidad de tiempo, dependiendo del tamaño del proyecto y el lenguaje de programación que se vaya a utilizar.
5. **Entrega y Retroalimentación.** Una de las cosas con las que cuenta el modelo de prototipos, es que una vez entregado el proyecto, debemos darle al cliente cierta retroalimentación sobre cómo utilizarlo y ciertamente es una fase que se encuentra dentro de las etapas de desarrollo de software esta metodología.
6. **Comunicación con el Cliente.** Es importante que una vez entregado el proyecto, tengamos cierta comunicación con el cliente, básicamente para que nos indique si el proyecto es correcto o si desea agregarle ciertas funciones, nuestra metodología lo permite. Si fuera en modo cascada, entonces sería algo realmente imposible de hacer.
7. **Entrega del Producto Final.** Por último, solamente quedará entregar el sistema elaborado mediante esta metodología. Aquí tendrás la ventaja de que el código es reutilizable, para que así con el prototipo ya puedas simplemente empezar de nuevo y con una buena base de código que te acelerará el proceso.

Página | 21

Principios Básicos del método de prototipos

Por supuesto, te habrás dado cuenta de que el modelo de prototipos puede llegar a ser un poco más tedioso, aunque todo dependerá del ámbito en que lo utilices. Sin embargo, uno de sus principios básicos que seguramente habrás notado, es que con el método de prototipos el proyecto se va dividiendo en partes cada vez más pequeñas, para evitar el peligro ante los riesgos frente a los que estamos expuestos.

Además, otros de sus principios básicos fundamentales, es que, con la metodología de prototipos, el cliente final se involucra mucho más en el proyecto que con otras metodologías, haciendo de esta forma que el producto final llegue rápidamente aunque con un poco más de presión en el proceso. La ventaja es que conforme se van haciendo prototipos pequeños, poco a poco se va llegando al producto final. Incluso en algún determinado momento podrás llegar a crear un prototipo que con solo ajustar ciertos detalles, se podría convertir en el producto que el usuario quiere.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Modelo Incremental o Iterativo y Creciente

El modelo Incremental, es una metodología de la programación muy utilizada hoy en día, pues su comodidad de desarrollo permite que te obtenga un producto final mucho más completo y exitoso. Se trata especialmente de la combinación de los modelos lineal e iterativo o bien, modelo de cascada y prototipos. Básicamente consiste en completar varias iteraciones de lo que es el modelo de cascada, pero sin completar ninguna, haciendo iteraciones lo que se hace es crear una evolución en el producto, permitiendo que se agreguen nuevas especificaciones, funcionalidades, opciones, funciones y lo que el usuario requiera después de cada iteración.

En pocas palabras, el Modelo Incremental repite el modelo de cascada una y otra vez, pero con pequeñas modificaciones o actualizaciones que se le puedan ir agregando al sistema. De este modo el usuario final se ve sumamente sumergido en el desarrollo y puedes proporcionarle un resultado óptimo.

Fases del Modelo Incremental

El modelo iterativo o incremental, cuenta con algunas fases de desarrollo de software que realmente no tienen mucha complejidad, vamos a verlas:

1. **Inicialización.** como en todo modelo de desarrollo, debe haber una inicialización, aquí se puede hablar de dar una idea, de tener algunos requisitos que se buscan en el proyecto y ciertas especificaciones que se pueden manejar. No es necesario que se haga una lista total de requerimientos pues recordemos que el proyecto se basa en iteraciones, así que el proyecto puede ir evolucionando poco a poco.
2. **Periodos de Iteración.** Durante el desarrollo del proyecto, es cuando damos inicio a las iteraciones. La primera iteración se realiza con las características iniciales, básicamente al final de la primera iteración, queda un pequeño prototipo de lo que será el proyecto, pero se puede volver a inicializar la iteración y realizar modificaciones en los procesos, como el análisis y las especificaciones o funciones que el usuario final requiere para su sistema. El número de iteraciones que se realicen son ilimitadas y dependerá tanto del desarrollador como del usuario final. Si nuestro objetivo es que el cliente o la persona que necesita el trabajo quede completamente satisfecha, entonces nos veremos en la necesidad de hacer la cantidad de iteraciones que se requieran al final del día.
3. **Lista de Control.** Es importante que conforme se vaya realizando cada iteración, se vaya llevando un control del mismo en una lista. Como si fuera un programa que recibe actualizaciones constantemente. Cada una de las actualizaciones o iteraciones deberá ser documentada y de ser posible, guardada en sus respectivas versiones, para que sea sencillo volver atrás, en caso de que una iteración no sea exitosa o el usuario ya no la requiera.



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Principios básicos del Modelo Incremental

Es importante definir los siguientes principios fundamentales, pues nos permiten saber con claridad por donde va la idea de la metodología.

La idea de un modelo incremental, es utilizar una serie de mini modelos de desarrollo de software en cascada, segmentando requerimientos y permitiendo que el usuario vaya de la mano con el proyecto durante la realización.

Básicamente las fases de cada iteración, son las mismas que se manejan en el modelo de cascada, aunque también se pueden agregar algunas, pero su objetivo es completar cada fase de la iteración, para que esta se vaya complementando poco a poco y no se genere un desarrollo tedioso y cansado que puede alargar la duración del proyecto en demasía.

Debes saber, que cada iteración te generará un prototipo cada vez más evolucionado, estos deberás guardarlos por si en determinado momento deseas volver atrás, pues a diferencia del modelo de cascada, podemos retroceder cuando se requiera y los prototipos se pueden volver a utilizar una y otra vez, pues el código fuente es reutilizable.

Página | 23



Diseño Método de Prototipos

Modelo en Espiral

El modelo en espiral, fue utilizado y diseñado por primera vez por Barry Boehm en 1986. Se trata nuevamente de una combinación entre el modelo lineal o de cascada y el



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

modelo iterativo o basado en prototipos, sin embargo, a este sistema lo que debemos añadirle es la gestión de riesgos, algo que en los modelos anteriores ni siquiera se menciona.

Este modelo, consiste en ciertas fases que se van realizando en modo de espiral, utilizando procesos de la misma forma en que se utilizan en el modelo de cascada, sin embargo, aquí estos no son obligatorios y no llevan precisamente el orden establecido. Básicamente se trata de un modelo evolutivo, que conforme avancen los ciclos, irá incrementando el nivel de código fuente desarrollado, un incremento en la gestión de riesgos y por supuesto un incremento en los tiempos de ejecución y planificación del sistema, esto es lo que tiene el modelo en espiral.

Para que tengas una idea más clara, el modelo en espiral es principalmente utilizado para el desarrollo de grandes proyectos como la creación de un sistema operativo. Sin embargo necesitas de ciertos requisitos, como el hecho de contar con personal completamente capacitado para las funciones que se requieran. Mejor veamos cuales son las fases o tareas dentro del modelo de espiral.

1. **Determinar Objetivo.** Es importante que siempre consideres una planeación inicial, esta solo se realizará una vez. Sin embargo, el proceso de determinar objetivos se hará constantemente durante cada iteración que se vaya realizando con el modelo espiral. Esto se debe a que poco a poco se irá incrementando más el tamaño del manual de usuario, los requisitos, las especificaciones e incluso las restricciones. Todo esto entra en lo que es la tarea de objetivos y con cada vuelta en el espiral entraremos a esta tarea, la cual como todas las demás, es fundamental.
2. **Análisis de Riesgo.** Una etapa donde incluso una lluvia de ideas podría ayudar, el análisis de riesgos. Aquí deberás tener en cuenta todo aquello que pueda dañar a tu proyecto, ya sea que se trate de ciertas amenazas o de posibles daños que se puedan ocasionar, teniendo además un Plan B por así decirlo, para que en caso de que ocurra algo inesperado, tener a la mano la solución para continuar con el proyecto. En esta fase del modelo espiral, podemos agregar lo que son la creación de prototipos, pues siempre es bueno tener un respaldo de nuestro código, se esta forma en caso de que algo malo suceda, volvemos a la versión anterior. Así que cada vez que vayamos a ingresar a la fase de pruebas e implementación, será necesario contar con un prototipo que nos respalde.
3. **Desarrollar, Validar y Probar.** Básicamente en esta fase, la forma en que se estará desarrollando el proyecto, dependerá del análisis de riesgos, pues siempre se va a ir desarrollando el proyecto enfocándose en los riesgos que podemos evitar en nuestro software, es decir, si la situación de riesgo más obvia se encuentra en la interfaz del usuario, entonces hay que trabajar con prototipos para este enfoque, creando evoluciones proporcionales, para ir evitando ese tipo de riesgos. Esto no significa que ignoremos el resto del proyecto o del desarrollo, sin embargo el modelo en espiral si

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

acomoda un poco más las prioridades al momento, independientemente de todo lo demás. Por lo que siempre en cada vuelta o iteración que se le dé al modelo de espiral, tu avance siempre dependerá del análisis de riesgo, hasta que este sea mínimo y el desarrollo pueda continuar de forma normal.

4. **Planificación.** Antes de proceder a realizar otra iteración o vuelta al espiral, debemos prestar atención a lo que sucedió en la vuelta anterior. Debemos analizar detalladamente si los riesgos encontrados ya tuvieron solución, lo cual debe ser lo ideal, puesto que ahora habría que analizar más especificaciones y más requisitos del sistema para continuar con el desarrollo. Básicamente la fase de planificación, nos servirá para determinar el avance de nuestro proyecto e indicar hacia dónde vamos a dirigirnos en la próxima iteración.

Página | 25

Principios básicos del modelo en Espiral

Está claro que el modelo en espiral, es sumamente distinto a los demás. Encontramos por fuera cuatro fases bien organizadas, las cuáles siempre deben llevar ese orden que se estipula desde el principio. Una determinación de objetivos, un análisis de riesgos, el desarrollo y las pruebas, junto con la planificación, la cual dependerá de los resultados de la iteración para saber cómo se actuará en la siguiente vuelta al espiral.

Básicamente, en el modelo de espiral, toda la atención está enfocada hacia el análisis de riesgos, pues el objetivo primario será reducir los riesgos que se vayan generando, de otra forma el sistema no llegará a ser seguro jamás.

Algo muy importante que se debe ver en el modelo de espiral, es que los interesados deben estar involucrados prácticamente en cada vuelta que se dé al espiral, para crear lo que son los requisitos antes de realizar una vuelta más y al final en la fase de planificación, se determinan los logros obtenidos, el avance y lo que se esperará de una siguiente vuelta.



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



Página | 26

Diseño Modelo en Espiral

RAD: Desarrollo Rápido de Aplicaciones (Rapid Application Development)

A diferencia de otras metodologías para el desarrollo de software, la metodología RAD o desarrollo rápido de aplicaciones, no cuenta con una serie de fases ordenadas por así decirlo. Aunque si está basada en lo que es el modelo de cascada y la creación de prototipos, sin embargo el proceso es muy independiente a contar con ciertas fases estipuladas como los modelos que hemos visto anteriormente. Así que vamos a ver los principios del modelo RAD.

Principios básicos del Modelo RAD

Del mismo modo que modelos anteriores, el Modelo RAD, está basado en el uso de las iteraciones y principalmente en el manejo de prototipos. Sin embargo, a diferencia del resto, la metodología RAD hace uso de las **Herramientas CASE**, las cuales permitirán acelerar el proceso considerablemente.

Del mismo modo que el modelo espiral y el de prototipos, RAD se subdivide en pequeñas secciones, las cuales se irán optimizando y de esta forma se irá avanzando mucho más rápido que con grandes segmentos que pueden volverse difíciles dentro de un proceso acelerado como lo esté este modelo.

Una de las ventajas del RAD, es que el enfoque y las prioridades van hacia la fase de desarrollo, a diferencia de modelos como el espiral, que se enfoca en que los riesgos al momento sean mucho menores. Acá con el RAD, se hace lo contrario, si hay riesgos reducimos los requerimientos para reducir los riesgos, no como en el espiral, que entre más riesgos, más requisitos aportamos para que se incremente. Acá la idea es reducir tiempos y no riesgos, o si, talvez también, pero la reducción de riesgos es totalmente inversa al modelo espiral.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

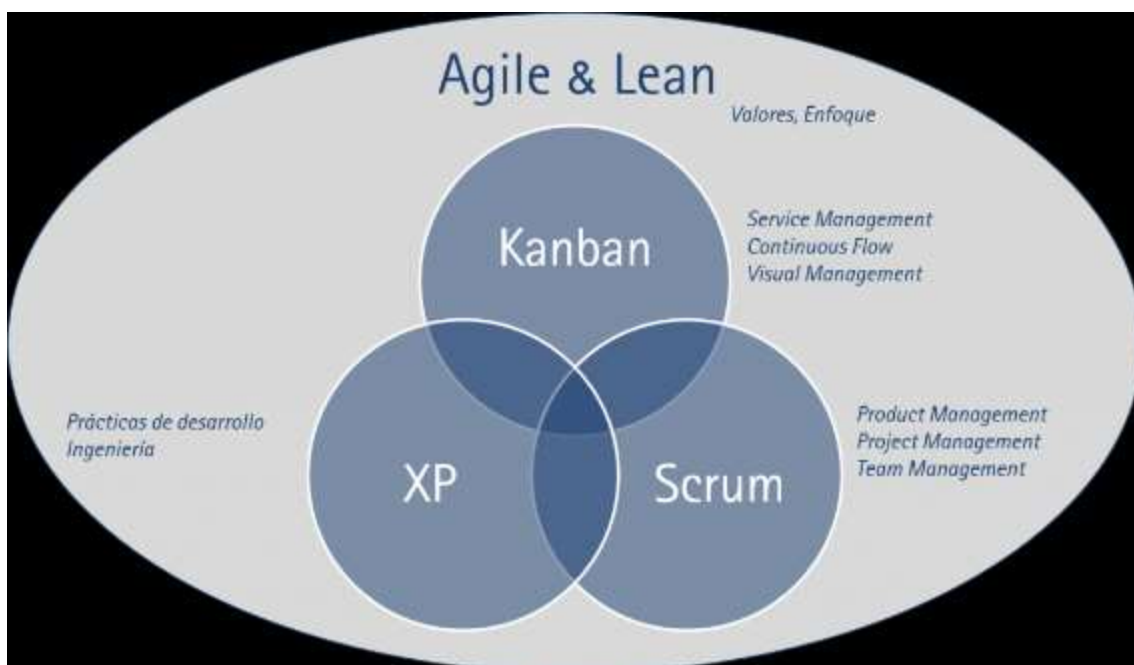
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Por supuesto, como en todos los modelos, vas a requerir de ciertos factores documentados, de preferencia todo lo que se pueda, para que en caso de que alguien más venga a continuar con este proyecto, tenga a la mano toda la información que necesita y con unas cuentas lecturas pueda empezar a desarrollar lo que se había quedado pendiente en un determinado momento.

Página | 27

Si bien hemos visto métodos muy diversos, información muy variada y ciclos de desarrollo de software con distintos enfoques. La realidad es que al final tu siempre decidirás como trabajar y con qué tipo de metodología te sentirías más a gusto. Obviamente depende de muchos factores, principalmente del tamaño del proyecto, pues modelos como el espiral, son especialmente para proyectos grandes y modelos como el RAD, son enfocados en proyectos pequeños, sin tanto requisito pero manejando siempre cierto nivel de calidad, el cual siempre debes considerar.

A continuación, analizaremos lo que son las metodologías ágiles de desarrollo de software más importantes, las cuáles a diferencia de las metodologías tradicionales, funcionan más como una combinación de estas para lograr un objetivo. Su finalidad siempre será el crear software de una forma más rápida de lo que se venía logrando con las metodologías de antaño. Así que vamos a ver un poco de información referente a lo que son las metodologías ágiles, como funcionan y que se necesita para implementarlas.



Esquema Metodología Ágiles

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

METODOLOGÍAS ÁGILES

Con el paso del tiempo, estaba claro que las metodologías tradicionales, simplemente no se iban a acoplar con las nuevas tecnologías, los nuevos lenguajes y sobre todo los programadores modernos. Es por eso que desde principios del Siglo, se han desarrollado lo que son las metodologías ágiles. Una metodología ágil, consiste principalmente en trabajar con menos documentación de la que, como vimos, las metodologías tradicionales utilizan en todo momento.

Página | 28

Existen una gran cantidad de metodologías ágiles de desarrollo de software y todas las vamos a ver a continuación. Sin embargo antes hay que comprender en que consiste detenidamente la metodología ágil, para lo cual contamos con el manifiesto ágil. Un documento en el cuál se resume la filosofía de este enfoque de desarrollo, así seguramente después de leer esos puntos, nos quedará aún mas clara la idea de hacia donde se pretende llegar y principalmente Cómo se pretende llegar a los objetivos.

Manifiesto Ágil

- *Al Individuo y las Iteraciones del Equipo.* Una de las cosas que sabemos muy bien, es que las personas son quienes consiguen los éxitos dentro de un proyecto de software. Sin embargo, también está claro que, si el equipo de trabajo falla, entonces todo se viene abajo y el enfoque que había de éxito se convierte en incertidumbre. A diferencia de si el equipo funciona perfectamente, se tienen más cerca los objetivos, aun cuando no exista una lista de procesos establecida como se hacía con las metodologías de antaño.

Con este primer valor del desarrollo ágil, se pretender hacer ver, que en realidad no importa que el equipo de trabajo no sean las personas más brillantes del sector. Con que sean personas que saben hacer bien el trabajo que se les asignará es más que suficiente. En este caso, las herramientas aunque son importantes para incrementar el rendimiento, también es cierto que si hay herramientas de más y que no sirven de nada en un principio, lo mejor es dejarlas de lado o estas nos quitarán valioso tiempo que no tendremos.

Básicamente el enfoque, es que no se intente crear un entorno de trabajo desde un principio, tratando de que los desarrolladores se adapten a ese entorno que nosotros deseamos, pues este tipo de proyectos suelen tardar mucho tiempo en desarrollarse, algunos incluso jamás terminan y se quedan estancados. El enfoque del desarrollo ágil nos dice, que es mejor formar primero un buen equipo de trabajo y posteriormente entre ellos vayan creando su propio entorno. Este proceso ayudará mucho más a la metodología ágil y por supuesto, la adaptación será un proceso fugaz.

- Software funcional en lugar de demasiada documentación. Crear documentación, es una de las actividades que, con las metodologías tradicionales, absorbían una gran

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

cantidad de tiempo. Si nos acercamos a analizar las metodologías de antaño, descubriremos que, en cada una de ellas, realizar la documentación era una parte vital. Sin embargo, en las metodologías ágiles, esto ha cambiado, pues existe una regla que dice de la siguiente forma: *“No producir documentación, al menos que sean sumamente necesarios al momento para tomar una decisión”*. Por lo que además se extiende hacia el enfoque de que la documentación debe ser corta y breve, nada sumamente extenso que requiera de una gran cantidad de tiempo de lectura.

Página | 29

Te preguntará y entonces, cuando un nuevo programador o desarrollador sea colocado en un puesto dentro del proyecto, como sabrá hacia donde ir y el enfoque que se está llevando a cabo. Para lo cual el manifiesto ágil nos dice que, existen dos elementos fundamentales para que un nuevo miembro del equipo se ponga al día. El primero es el código fuente, pues suponiendo que es una persona conocedora, sabrá hacia dónde va el curso del proyecto con tan solo leer el código. La segunda es que la interacción con el equipo de trabajo, será el complemento ideal para que se acople al proyecto. De este modo nos olvidamos de la extensa documentación para un software que al final del día debe estar totalmente funcional.

- Colaboración con el Cliente en lugar de hacer Contrato. Una de las cosas importantes dentro de las metodologías ágiles. Es que cambia el modo en que se trabajaba con el cliente anteriormente. Y es que en las metodologías de antaño, el trabajo consistía en tener una reunión previa con el cliente para analizar los requerimientos del sistema, aquí se analizaban las limitaciones del proyecto y se establecían los costos. Lo que generaba cierta barrera de bloqueo entre las posibilidades de desarrollar algo funcional para otras cosas o solo para el objetivo establecido en la reunión. Esto al final podía ser desastroso y dificultar la llegada al éxito por parte del proyecto.

Sin embargo, ahora en el manifiesto ágil, se propone que exista una interacción constante entre el cliente y el equipo de desarrolladores. La idea es que el cliente vaya viendo cómo va el sistema y analice nuevas funcionalidades o objetivos, ya que estos no tienen por qué plantearse desde el principio, si el desarrollo nos puede llevar a una infinidad de posibilidades. De esta forma el cliente al final queda totalmente satisfecho con el producto final y habremos llegado al éxito trabajando todos en equipo de forma simultánea.

- Posibilidad de Hacer cambios de planes a medio proyecto. Suena más o menos a lo que vimos en el punto anterior, pues básicamente la idea es evitar lo que es la planeación extensa y empezar a crear código que permita expansión. Recordemos que con las metodologías tradicionales, se acostumbraba a enlistar los requisitos del sistema y el desarrollo iba enfocado solamente a eso, lo cual ya no permitía que a medio desarrollo



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

hubiera cambios, pues era un código poco moldeable y si se requerían nuevas cosas, en algunas metodologías lo idea era volver a empezar.

La idea en las metodologías ágiles, es que durante el desarrollo del software, si el cliente tiene la idea de incrementar sus objetivos, especificaciones o requerimientos, lo pueda hacer sin ningún problema. Pues básicamente el sistema debe ser flexible para todo lo que pueda surgir en el proceso. De esta forma, no solamente llegaremos al éxito, si no que el cliente quedará totalmente satisfecho con el trabajo desarrollado, pues no tuvo que conformarse con lo primero que se le vino a la mente, si no que se actualizó con las ideas que en la marcha fueron surgiendo.

Página | 30

PRINCIPALES METODOLOGÍAS ÁGILES

Algo que debes saber antes de dar paso a la descripción de cada una de las metodologías de la programación ágiles, es que aunque entre sus creadores crearon lo que fue el manifiesto ágil, la realidad es que cada una de las metodologías cuenta con su propia personalidad y características únicas, que la diferencian de las demás. Por eso a continuación, veremos cada una de las metodologías ágiles más populares, para que tengas un conocimiento más sólido de lo que son y hacia dónde van cada una de ellas.

METODOLOGÍA SCRUM



Para que tener una idea rápida, para que un proyecto ingrese al marco de lo que es el modelo Scrum, debe contar con las siguientes características:

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Desarrollo Incremental.

Una metodología ágil sin desarrollo incremental, no puede ser considerada Scrum. Con incremental hago énfasis a olvidarnos de la planeación y de la ejecución de las líneas sin salirnos de lo pre establecido, pues con una metodología Scrum, el desarrollo se irá incrementando poco a poco, sin importar el orden en el cual se lleven a cabo los procesos.

Página | 31

Calidad de las personas.

Básicamente la calidad de un producto, no será analizada en base a la calidad de cada uno de los procesos llevados a cabo. Al contrario, la calidad dependerá de las personas, la auto organización y el conocimiento de los equipos de trabajo.

Adiós al Secuencial y Cascada.

Aquí en el modelo Scrum, hay algo a lo que se le denomina, solapamiento. Esto consiste en que no importa en que proceso te encuentres, si un proceso necesita ser trabajado, vuelves a el para realizar lo que tienes que hacer, a diferencia de las metodologías cascada o secuencial, donde no había vuelta atrás. Acá afortunadamente no hay ningún problema con eso y la ventaja es que se ahorran tiempos.

La comunicación es Fundamental.

Una de las cosas que se realizan, son los equipos de trabajo, sin embargo, acá la ventaja que tendrás es que podrás estar en constante comunicación con los otros equipos de trabajo, nadie está envuelto en su propia burbuja y toda la información que se maneje o lleve a cabo, será comunicada sin problema.

Funcionamiento de los Procesos Scrum

La metodología Scrum, es bastante amigable y fomenta lo que es el trabajo en equipo en todo momento, con la finalidad de conseguir los objetivos de una forma rápida. Veamos ahora cuales son los procesos con los cuales funciona la metodología, empezando por el Product Backlog, el cual nos permitirá llegar a los Sprints.

Product Backlog.

El Product Backlog no es más que una lista de las funcionalidades del producto a desarrollar. Este debe ser elaborado por el Product Owner, puesto que más adelante les explicaré. Sin embargo, no se trata de una lista cualquiera hecha con escritos y nada más. El Product Backlog debe estar ordenado de acuerdo a las prioridades del sistema de más a menos, con la idea de que las cosas con mayor prioridad sean las que se realicen antes de cualquier cosa. De forma concreta, digamos que el objetivo base del Product Owner, es que nos dé respuesta a la pregunta “¿Qué hay que hacer?”.

Sprint Backlog.

Una vez que ya contamos con el Product Backlog terminado, entonces aparecerá el primer Sprint Backlog. Pero ¿Qué es el Sprint Backlog? Consiste básicamente en

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

seleccionar algunos de los puntos escritos en el Product Backlog, los cuales procederán a ser realizados. Sin embargo en este punto el Sprint Backlog tiene como requisito marcar el tiempo en que se llevará a cabo el Sprint.

Sprint Planning Meeting.

Antes de iniciar un Sprint, el cual es la fase de desarrollo, se realiza lo que es un Sprint Planning Meeting. En este proceso del Scrum, es una reunión que se realiza para definir plazos y procesos a efectuarse para el proyecto establecido en el Product Backlog. Algo importante que debes saber, es que cada Sprint, se compone de diversos features, que no son otra cosa más que procesos o subprocesos que se deben realizar, puede ser la creación de un logo, la gestión de contenido, el diseño visual, etc. Todo dependerá del proceso que se desee llevar a cabo.

Página | 32

Daily Scrum o Stand-up Meeting.

Cuando un Sprint está en proceso, después de haber hecho la planeación del proyecto mediante plazos y procesos, entonces entramos a lo que son los Daily Scrum o Stand-up Meeting. Aquí básicamente lo que se hace son reuniones diarias mientras se está llevando a cabo un Sprint, para responder las siguientes preguntas: ¿Que hice ayer?, ¿Qué voy a hacer hoy, ¿Qué ayuda necesito? Aquí entra en función el **Scrum Master**, un puesto que igual más adelante les explicaré. Pero el será el encargado de determinar la solución de los problemas y cada complicación que suceda.

Sprint Review.

El Sprint Review, es básicamente una reseña de lo que fue el Sprint. Consiste específicamente en la revisión del Sprint terminado y para este punto ya tendría que haber algo que mostrarle al cliente, algo realmente visual o tangible para que se pueda analizar un cierto avance.

Sprint Retrospective.

Para concluir, el Sprint Retrospective, permite al equipo analizar los objetivos cumplidos, si se cometieron errores, visualizarlos y tratar de no cometerlos nuevamente más adelante. Básicamente también sirve este proceso para lo que son la implementación de mejoras.

Equipos que Componen los Procesos Scrum

Durante el punto anterior, te describí los procesos que se llevan a cabo en la Scrum Metodología, y en varios puntos mencioné ciertos equipos que son encargados de algunos aspectos importantes. Por eso a continuación veremos cuales son los equipos que conforman la metodología Scrum y con los cuales se trabajará arduamente, obvio, cada quien con sus respectivas responsabilidades.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Product Owner.

Si se trata de tener un líder de proyecto, entonces el Product Owner lo será. Básicamente son los ojos del cliente, será la persona encargada del proyecto y de vitorear que se lleve a cabo de tal forma que cumpla las expectativas de lo que se espera.

Scrum Master.

Ahora bien, para cada reunión realizada, siempre debe estar un líder, en este caso el Scrum Master será el líder de cada una de las reuniones y ayudará en los problemas que hayan surgido. Será básicamente como un “facilitador” el cual minimizará obstáculos, sin embargo no los omitirá. En realidad el Scrum Master debe ser una persona empapada de conocimientos sobre el lenguaje o lenguajes bajo los cuales se llevará a cabo el proyecto, de lo contrario no tendría como ayudar a solucionar problemas.

Scrum Team.

Básicamente es el núcleo de la metodología Scrum, pues es el equipo de desarrollo, encargado de lo que es la codificación del software y de cumplir los objetivos o metas propuestas por el Product Owner.

Cliente.

Aunque no lo creas, el cliente también forma parte del equipo, hablamos de eso hace un rato, cuando comenté que no es como en las metodologías tradicionales donde al cliente se le pedían requerimientos y se le daba un costo total. En la metodología Scrum, el cliente tiene la capacidad para influir en el proceso, debido a que siempre estará empapado de él, ya sea que proponga nuevas ideas o bien haciendo algún tipo de comentario.

METODOLOGÍA KANBAN

Siguiendo con las metodologías ágiles, nos encontramos con Kanban. Se trata de una metodología Japonesa, la cual consiste en ir etiquetando con tarjetas cada uno de los procesos que se deben llevar a cabo, también se le ha denominado como “Un sistema de producción de alta efectividad y productividad”. De hecho, empresas como la marca de autos Toyota, fueron una de las primeras en implementarla para acelerar los procesos de producción.

La palabra Kanban, en japonés significa “tarjetas visuales” y es precisamente lo que se maneja en ella. Algunos la trabajan con lo que son tarjetas virtuales o bien simulando que se pueden ver las tarjetas, sin embargo una forma correcta de hacerlo es con tarjetas físicas, que el equipo pueda ver y sentir para tener mayor efectividad.

Una de las principales ventajas de Kanban, es que además de ser una metodología Ágil, también es muy fácil de usar e implementar, sobretodo porque el equipo de trabajo se unirá y empezarán

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

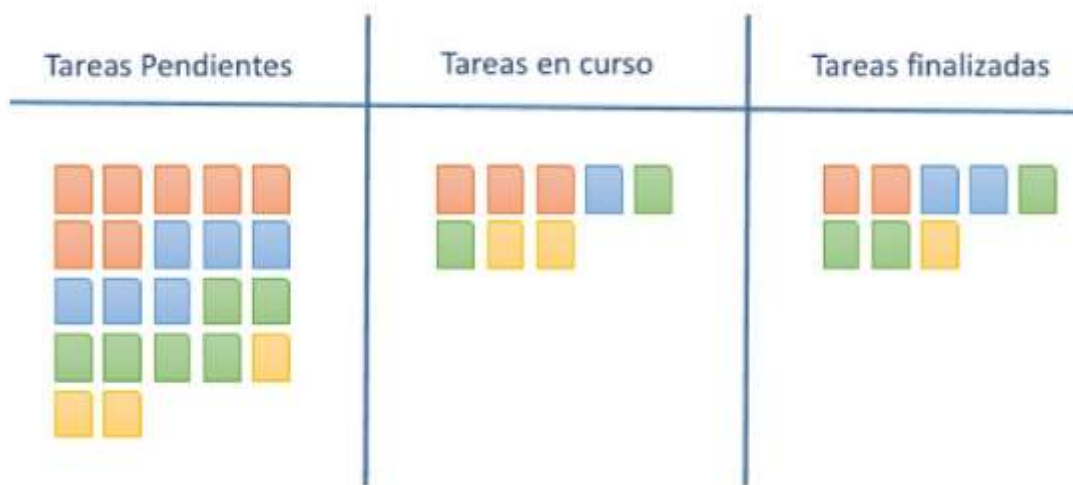
REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

a trabajar a la par en diferentes aspectos del desarrollo. Veamos ahora, cuales son los principios básicos de la metodología Kanban.



Página | 34

Ejemplo Metodología Kanban

Principios Básicos de Kanban

Garantía de Calidad.

Algo por lo que destaca Kanban, es que el ser una metodología ágil, no es sinónimo de trabajar a las carreras o de hacer todo de golpe. Kanban promueve la calidad antes que la velocidad, es decir, un producto bien hecho desde la primera vez que se elaboro es más rápido, que un producto mal hecho al cual se le tienen que volver a meter las manos para arreglarlo. Entendiendo esto, concluimos con que todo debe salir bien desde el inicio y no debe haber margen de error.

Desperdicios.

Basado en lo que es el principio YAGNI, la metodología Kanban trabaja de una forma en la cual, solamente se debe hacer lo necesario y requerido para que el sistema o el desarrollo quede bien. Evitando todo aquello que es considerado como extra, superficial o innecesario. De este modo no solamente se ofrece una mayor calidad en el producto, sino que además se optimizan tiempos y costos.

Mejora Continua.

Algo interesante de la metodología Kanban, es que no solamente se trata de un sistema diseñado para el proceso de desarrollo de Software, se puede implementar en el desarrollo de cualquier tipo de producto, tal y como lo hizo Toyota. Además es un

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

sistema que nos da la oportunidad de ir mejorando constantemente en los procesos, dependiendo claro de cual sea el objetivo o la meta final.

Es Flexible.

Aquí es donde volvemos a hacer comparaciones con las metodologías de antaño, donde la flexibilidad no existía, como si fueran metodologías del abuelo estricto, acá eso no existe. La flexibilidad es uno de los principios de Kanban y ¿qué obtenemos con ello? Gracias a que es flexible, podemos adelantarnos a un proceso que queramos hacer o que tenga cierto nivel de prioridad, no necesitamos seguir una línea de trabajo, lo cual hace de Kanban una metodología más dinámica y además permite resolver problemas que surjan de imprevisto, algo que con otras metodologías simplemente ni se considera.

Página | 35

Configurar una estrategia Kanban

Realmente una de las cosas que diferencian a Kanban del resto, es que acá vas a necesitar un tablero de verdad. No es nada extraño ni loco, con el avance de los días de trabajo notarás que fue una magnífica idea comprar ese tablero y los post-it para escribir objetivos. Así que vamos a ver los pasos para realizar bien la configuración de una metodología Kanban.

1. **Definir el Flujo de Trabajo.** Una vez que ya tienes el tablero que es requisito para esta metodología, es entonces cuando podrás empezar a seccionarlo dependiendo del número de tareas, fases o proyectos que tengas en puerta. Considera que el tablero debe estar a la vista de todo el equipo de trabajo, pues será necesario estar al pendiente de los procesos y de ir actualizándolos conforme se vaya avanzando en él. Ahora sí que como te comentaba, un tablero puede ser destinado para un solo proyecto o bien para muchos proyectos, todo dependerá de si deseas mantener el flujo de trabajo constante o medianamente pausado.
2. **Fases del Ciclo de Producción.** Es importante que te des cuenta, de cómo debes seccionar el tablero para ir marcando el flujo de producción. En este caso es realmente necesario que los procesos sean divididos en pequeños segmentos, para que se pueda agilizar y no se quede estancado en uno con demasiada duración. Es por eso que en los post-it que vayas colocando con cada proceso, deberá ser colocado el número de horas requeridas para completarlo, al finalizar las horas se determinará por qué no se ha terminado o bien ya se habrá avanzado a otra fase.
3. **Stop Starting, start finishing.** La filosofía de Kanban, trabaja de esta forma, no te voy a envolver en términos que te puedan confundir pues la idea es obvia: “No se empieza una nueva tarea, hasta terminar la otra”. Esto se debe principalmente a que la idea es tener un alto porcentaje de tareas completadas y no como ciertos equipos de desarrollo que tienen una gran cantidad de proyectos en puerta y muchas tareas por hacer, la mayoría empezadas, pero ninguna terminada. Esta es la situación principal que se trata de evitar con Kanban y es que aunque parezca muy obvia, la realidad es que muchos la pasan de largo, aun cuando es fundamental.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

4. **Tener un Control.** Algo con lo que trabaja Kanban, es con el control del flujo de trabajo. Si bien la idea es que los trabajadores tengan actividad realmente constante y no se detengan aun cuando terminen sus tareas. Kanban permite llevar un control de todo gracias a las notas que se van colocando. De esta forma además permite que se ejecuten varios proyectos se forma simultánea, una parte del equipo pudo haber terminado sus tareas del proyecto anterior y avanzar al siguiente, mientras los demás siguen todavía trabajando en ello. La idea es no provocar interrupciones a cada momento, además de que si necesitas un control, puedes ir almacenando las tarjetas y listo.

Kanban Board

Edumatics



Diagrama Metodología Kanban

METODOLOGÍA XP

Si hablamos de metodologías de la programación sin mencionar a la Metodología XP, es como no hablar de nada en absoluto. Esta metodología es posiblemente la mas destacada de las metodologías ágiles y esto se debe a su gran capacidad de adaptación ante cualquier tipo de imprevisto que surja. Pues la idea no es mantener ciertos requisitos desde que se está elaborando el proyecto, sino que durante el proceso, estos vayan cambiando o vayan evolucionando gradualmente sin complicaciones. Básicamente los creadores de esta

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

metodología XP, consideran que es mejor adaptarte en el proceso a los requisitos que vayan apareciendo, que iniciar con requisitos y desarrollar un proyecto en base a eso.

Si queremos ver la metodología con otra perspectiva, se podría decir que la metodología XP o metodología de programación extrema, como también se le conoce. Es la combinación de las demás metodologías, solamente que se van utilizando de acuerdo a como sea necesario, por eso es considerada como la más destacada de las metodologías ágiles. Así que es momento de entrar en detalles y vamos a ver cuáles son los valores que conforman a la metodología de programación XP.

Página | 37

Valores de la Metodología XP

Como toda metodología, la programación extrema cuenta con algunos valores que son fundamentales para que se lleve a cabo como debe ser. En algunas otras metodologías estos puntos los conocíamos como principios básicos, es realmente lo mismo solo que acá los mencionan como valores, veamos cuales son.

Comunicación.

Del mismo modo que otras metodologías como la Scrum, el cliente tiene una gran intervención, pero obviamente la comunicación dependerá de más factores. Por ejemplo. El código fuente de los programadores debe transmitir esa comunicación a todo el equipo, por eso las variables amigables. De igual forma, se deben documentar las cosas más relevantes, independientemente de que sean comentadas en el código, pero es importante tener un documento extra para explicaciones extensas, de lo contrario el código se verá infestado de escrito. En cuando a la comunicación entre personas, los programadores se comunican constantemente ya que trabajan en parejas, la comunicación que se tiene con el cliente debe ser constante, pues recordemos que incluso el forma parte del equipo de trabajo y es responsabilidad del cliente, comunicarnos algunas actualizaciones que requiera en el proceso, nuevas ideas, soluciones a problemas o sencillamente algún problema que el vea. Todo debe ser comunicado, esta parte es realmente fundamental para el desarrollo de un producto exitoso.

Simplicidad.

El primero de los valores de la metodología de programación XP, es la simplicidad. Ya te haz de imaginar en que consiste, puesto que la idea es que el desarrollo sea veloz, por lo cual todas las cuestiones de diseño se simplifican al máximo, lo mismo sucede con las líneas de código, si se pueden simplificar, se hacen, además de que regularmente el mismo código es donde va la documentación comentada, de esta forma nos evitamos el estar haciendo documentación extra. Por esta razón, además es importante que en el ciclo de desarrollo de software mediante la metodología XP, las variables, métodos y clases, tengan nombres amigables y relacionados, de este modo no solamente se ayuda

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

el equipo de trabajo, el cual de por sí ya se debe conformar de dos personas por equipo, sino que además, cuando una persona nueva ingrese al proyecto, será muy rápida su adaptación.

Retroalimentación.

El hecho de que el cliente se encuentre involucrado en el proyecto, ayudará inicialmente con la retroalimentación. Pues conforme pasan los días y se va analizando el código por pequeñas etapas, el cliente puede ir corrigiendo, agregando, quitando o excluyendo algunas cosas, esa es la ventaja de la programación por periodos cortos de tiempo, es decir, imagina que llevas varios meses desarrollando el proyecto y cuando vas con el cliente, el proyecto no le gusta y desea hacer tantos cambios que te llevará una eternidad. Precisamente eso es lo que se trata de evitar con la metodología XP, que se tiren varios meses de trabajo a la basura y mejor se vaya optimizando en pequeños lapsos de tiempo. Otra de las formas de retroalimentación con la cual contamos, es nuestro código fuente. Pues gracias a que podemos realizar diversas pruebas unitarias, podremos ver la salud de nuestro código, una gran ventaja, pues si hay problemas o errores, siempre estaremos a tiempo de realizar modificaciones y soluciones. A diferencia de un proyecto sumamente extenso, donde seguramente la cantidad de errores será tan impresionante que volver a empezar podrá ser una opción.

Página | 38

Valentía.

Hay elementos donde el coraje o la valentía de los programadores serán fundamental. Por ejemplo el dar solución a los problemas frente a los cuales se enfrente. El pasar por la eliminación de código fuente en el programa desarrollado, aun cuando todo ese código haya tomado una gran cantidad de tiempo en hacerse o que tal el hecho de diseñar para hoy y no para mañana. Muchos lo hacen con esa idea en ocasiones, pero con un poco de valentía y coraje que forman parte de los valores de la metodología, seguramente el éxito llegará de manera anticipada.

Respeto.

Originalmente, este quinto valor no se encontraba en la metodología XP, sin embargo es importante mencionarlo pues hoy en día ya lo conforma. El respeto es importante para que haya una buena comunión entre los programadores del equipo. Nunca hay que denigrar a nadie ni agregar u ofender, pues un autoestima alta en el equipo garantizará un trabajo mucho más eficiente. Por esta razón, cosas como el código fuente, las modificaciones, los fallos obtenidos, los problemas o la solución de problemas, son procesos que se deben respetar para que el ambiente de trabajo también sea óptimo.



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Características que componen la metodología XP

Ya vimos los valores o principios básicos, dejémoslo en valores pues así es como lo denomina la metodología, sin embargo ahora vamos a ver sus características, de esta forma te podrás dar una mejor idea de cómo funciona una metodología XP.

Tipo de Desarrollo Iterativo e incremental.

Como hemos visto en lo que llevamos hablando de la metodología XP, el método está basado en lo que son las mejoras continuas, a base de iteraciones y por supuesto un desarrollo incremental al estilo espiral.

Pruebas Unitarias.

Una de las características además son las pruebas unitarias. Se utiliza software de codificación eso sí, dependiendo del lenguaje que estemos usando es la herramienta que nos corresponde, pero de este modo se analiza el código y solucionan errores, antes de validarlo y darlo por bueno.

Trabajo en Equipo.

Más específico todavía, es el trabajo en parejas, el objetivo es que el enfoque en parejas sea mayor, las distracciones son menores y el aprendizaje del uno con el otro permite que el avance del proyecto sea mucho más eficiente que cuando una persona es la encargada.

Alguien del equipo trabaja con el cliente.

Es fundamental que el cliente intervenga en el desarrollo, pero obviamente el no estará en la sala de desarrollo, se debe asignar a una persona que sea le encargada de tener las reuniones con el cliente de forma constante. El será quien comunique al equipo los cambios o el seguimiento del proyecto.

Corrección de Errores.

Algo importante, el hecho de que la metodología XP sea realmente rápida para el desarrollo, no significa que se pasen por alto los errores, de hecho primero se le tiene que dar corrección a los errores antes de seguir avanzando en el proyecto.

Reestructuración del Código.

La idea es clara una refactorización del código siempre se debe realizar. Con esto lo que haremos es simplificar el código pero no las funciones. Pues regularmente cuando desarrollamos, agregamos algunas cosas que pueden ser innecesarias y que no afectan en el funcionamiento del sistema, estas son precisamente las que hay que re facturar.



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

El Código es de todos.

Realmente aunque se trabaje en equipos, al final todos tendrán la posibilidad de ver el código, proponer cambios o incluso hacerlos. La idea es que si uno no detecta un error, otro lo podrá hacer, por eso el código fuente es compartido entre todos.

Código simple es la clave.

Algo importante con la metodología extrema, es que la simplicidad siempre llevará la ventaja. Principalmente porque cuando se requiera hacer un cambio, si el código fuente es muy complejo, posiblemente lleve muchas horas realizar los cambios e incluso una alternativa sería ya no hacer ningún cambio para no perder tiempo. Esta es precisamente la razón por la cual el código simple, es fundamental en la metodología.

Básicamente, lo que es la simplicidad y la comunicación van de la mano. Puesto que a mayor simplicidad, la comunicación necesaria será menor. Haciendo que la eficiencia se incremente y la pérdida de tiempo en comunicación sea menor. Por eso es importante seguir al pie de la letra estas dos ideales de la metodología.

Equipo de Trabajo dentro de una Metodología XP

Ya para concluir con esta metodología, recordemos que es una de las mejores que podrían existir y que realmente vale la pena implementar. Veamos cuales son los roles que componen el equipo de trabajo en un proyecto que se elaborará mediante la metodología XP, para que tengas una idea de la formación que se debe efectuar.

Programador.

Realmente no creo que sea necesario que te diga lo que hace el programador. Bueno, es el encargado del código del sistema y además de hacer las pruebas unitarias que se solicitan.

Tester.

Básicamente es el encargado de las pruebas del desarrollo. Lo que se vaya implementando, el teste lo prueba y le dice al cliente o mejor dicho, le comunica al cliente las pruebas funcionales, para posteriormente comunicarle al equipo los resultados.

Tracker.

El seguimiento será lo suyo. Será el encargado de realizar las comparaciones entre los tiempos estimados antes de empezar un desarrollo y los tiempos reales que se obtuvieron. Tratando siempre de mantener al tanto al equipo para que traten de mejorar los tiempos.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Entrenador.

Este elemento es realmente importante, puesto que es el responsable del proyecto básicamente y precisamente hace las funciones de un entrenador. Se encarga de guiar al equipo por el camino que deben seguir.

Consultor.

Regularmente el consultor no formaba parte del equipo, bueno de hecho no lo integra. El consultor sigue siendo un externo, pero que cuenta con conocimientos específicos y que será capaz de ayudar en la solución de problemas.

Gestor.

Posiblemente el líder más alto. Si se trata de unir a los clientes con los programadores, el gestor es el intermedio, es decir. Es el encargado de vincular e interrelacionar al cliente con los programadores.

Página | 41

CLASIFICACION DE LAS METODOLOGIAS DE DESARROLLO DE SOFTWARE CRONOLOGICAMENTE

AÑO	METODO
1969	PROGRAMACION ESTRUCTURADA
1975	PROGRAMACION ESTRUCTURADA JACKSON
1980	STRUCTURED SYSTEMS ANALYSIS AND DESIGN METHODOLOGY (SSADM)
1980	STRUCTURED ANALYSIS AND DESIGN TECHNIQUE (SADT)
1981	INGENIERIA DE LA INFORMACION (IE/IEM)
1991	RAPID APPLICATION DEVELOPMENT (RAD)
90's	PROGRAMACIÓN ORIENTADA A OBJETOS (OOP) A LO LARGO DE LA DÉCADA DE LOS 90'S
90's	VIRTUAL FINITE STATE MACHINE (VFSM) DESDE 1990S
1995	DYNAMIC SYSTEMS DEVELOPMENT METHOD DESARROLLADO EN UK
90's	SCRUM FINALES DE LOS 90
1999	RATIONAL UNIFIED PROCESS (RUP)
1999	EXTREME PROGRAMMING(XP)
2002	ENTERPRISE UNIFIED PROCESS (EUP) EXTENSIONES RUP
2004	CONSTRUCTIONIST DESIGN METHODOLOGY (CDM)
2005	Agile Unified Process (AUP)
2018	PROGRAMACIÓN AL TIRO

LENGUAJE UNIFICADO DE MODELADO – UML

UML O Lenguaje Unificado de Modelado es un estándar para la representación de procesos o esquemas de software (programas informáticos).

UML son las siglas de “Unified Modeling Language” o “Lenguaje Unificado de Modelado”. Se trata de un estándar que se ha adoptado a nivel internacional por numerosos organismos y

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software (programas informáticos).



Página | 42

El término “lenguaje” ha generado bastante confusión respecto a lo que es **UML**. En realidad, el término lenguaje quizás no es el más apropiado, ya que no es un lenguaje propiamente dicho, sino una serie de normas y estándares gráficos respecto a cómo se deben representar los esquemas relativos al software. Mucha gente piensa por confusión que UML es un lenguaje de programación y esta idea es errónea: UML no es un lenguaje de programación. Como decimos, UML son una serie de normas y estándares que dicen cómo se debe representar algo.

UML es una herramienta propia de personas que tienen conocimientos relativamente avanzados de programación y es frecuentemente usada por analistas funcionales (aquellos que definen qué debe hacer un programa sin entrar a escribir el código) y analistas-programadores (aquellos que dado un problema, lo estudian y escriben el código informático para resolverlo en un lenguaje como Java, C#, Python o cualquier otro). Por tanto si estás dando tus primeros pasos en programación, te recomendaríamos que te olvides de UML hasta que tengas unos conocimientos mínimos como uso de condicionales, bucles, y conocimiento de la programación orientada a objetos. Esto es solo una recomendación, en realidad prácticamente cualquier persona puede usar UML, incluso podría usarse para realizar esquemas o documentación de procesos que no tengan que ver con la informática.

VERSIONES DE UML

Los antecedentes de UML se sitúan en la década de los 90 con distintos estándares para modelado de software, no obstante, podemos hablar de dos grandes versiones:

- **UML 1.X** (comprende UML 1.1, 1.2, 1.3, 1.4, 1.5): desde finales de los 90 se empezó a trabajar con el estándar UML. En los años sucesivos fueron apareciendo nuevas versiones que introducían mejoras o ampliaban a las anteriores.
- **UML 2.X** (comprende UML 2.1 hasta UML 2.5, 2.6, etc.): en torno a 2005 se difundió una nueva versión de UML a la que podemos denominar UML 2.X. Comprende varias revisiones.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Hay que tener en cuenta que UML es un conjunto muy amplio de normas. Prácticamente nadie las conoce todas. Según la empresa o universidad, institución o centro de trabajo se usan determinados programas para crear diagramas y se conocen ciertas partes de UML, pero no el conjunto de UML.

TIPOS DE DIAGRAMAS EN UML

Página | 43

Usando UML se pueden construir numerosos tipos de diagramas. Vamos a citar los siguientes:

DIAGRAMAS DE COMPORTAMIENTO

Un diagrama de comportamiento se define formalmente como: Diagrama que expresa las secuencias de estados por los que pasa un objeto a lo largo de su vida en respuesta a eventos.

Hablando en un lenguaje más llano, se trata de diagramas que muestran diferentes estados de un proceso. Mediante estos diagramas se plasman de forma gráfica los procesos a programar.

Estos diagramas se usan para visualizar y especificar, a la vez que documentar, aspectos dinámicos de un sistema.

Cuando hablamos de desarrollo de software, estos aspectos dinámicos pueden ser mensajes que se generan, acciones de entrada de datos, eventos, etc.

Estos diagramas pueden ser muy simples o muy complejos en función del proceso que están representando. Hacen referencia a objetos, ya que se emplean mucho en la programación orientada a objetos. Un objeto puede ser cualquier entidad con un determinado estado y comportamiento.

Para aclarar de una forma visual el concepto, se puede ver la siguiente imagen donde se representa un proceso mediante un diagrama de comportamiento.



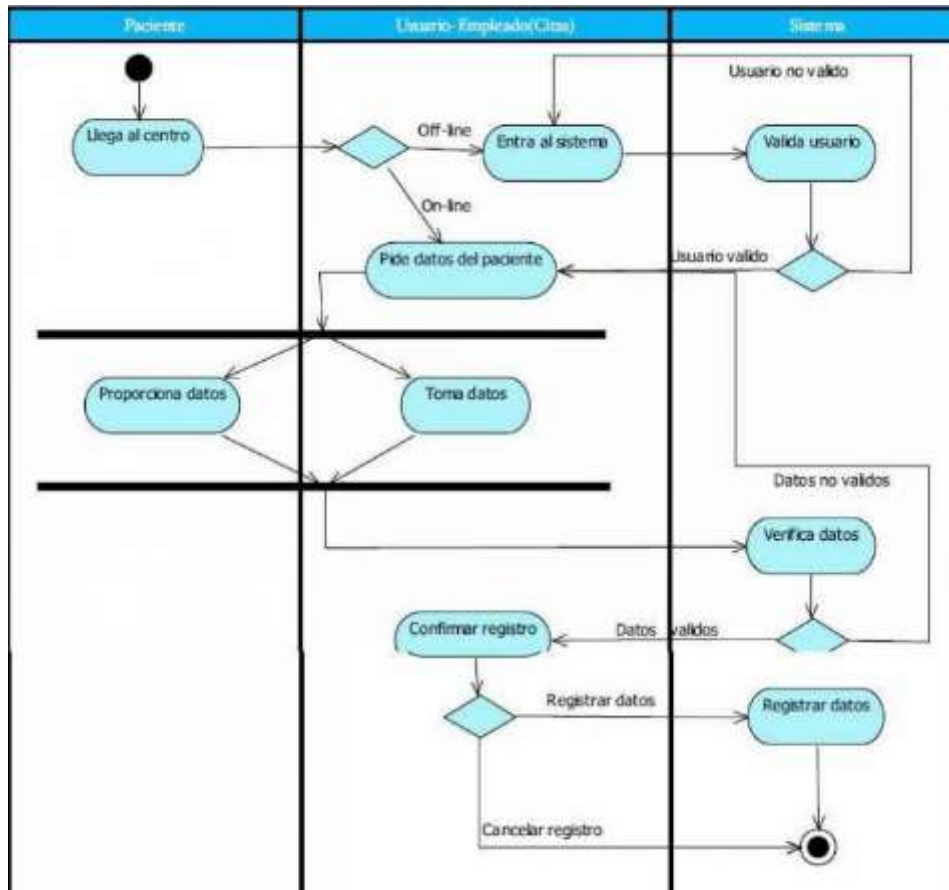
INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



Importancia de los diagramas de comportamiento en el desarrollo de software

- Se suele empezar definiendo el sistema a alto nivel para luego ir profundizando en diagramas más detallados.
- Estos diagramas proporcionan a los desarrolladores una visión clara y detallada de los procesos a implementar, con los diferentes detonadores de cambio de estado de cada objeto.
- No tiene por qué existir en un diseño todo este tipo de diagramas, pero si es necesario que existan para poder entender posteriormente toda la programación realizada.
- Además de ser una herramienta de diseño para el desarrollo, es una forma clara y concisa de documentar todos los procesos a desarrollar.

DIAGRAMAS DE CASOS DE USO

Representan a los actores y casos de uso (procesos principales) que intervienen en un desarrollo de software.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Este tipo de diagramas muestra las diferentes operaciones que puede realizar una aplicación o sistema, así como su relación con su entorno, que típicamente será el usuario u otras aplicaciones o sistemas.

Un ejemplo sería el definir los diferentes usos que puede tener un sistema que controle una máquina dispensadora. Esta máquina puede tener varios usos como: dispensar producto, dar cambio, informar de precio, etc.

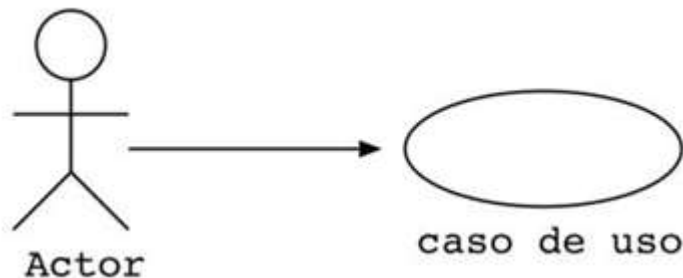
Página | 45

Elementos

Los elementos que pueden aparecer en un Diagrama de Casos de Uso son:

Actores

Un diagrama de caso de uso contiene los símbolos del actor y del caso de uso, junto con líneas conectoras. Los actores son similares a las entidades externas; existen fuera del sistema. El término actor se refiere a un rol específico de un usuario del sistema.



Por ejemplo:

Un actor puede ser un empleado, pero también puede ser un cliente en la tienda de la empresa. Incluso cuando es la misma persona en el mundo real, se representa como dos símbolos distintos en un diagrama de caso de uso, ya que la persona interactúa con el sistema en distintos roles.

Sistema

El rectángulo representa los límites del sistema que contiene los casos de uso. Los actores se ubican fuera de los límites del Sistema.



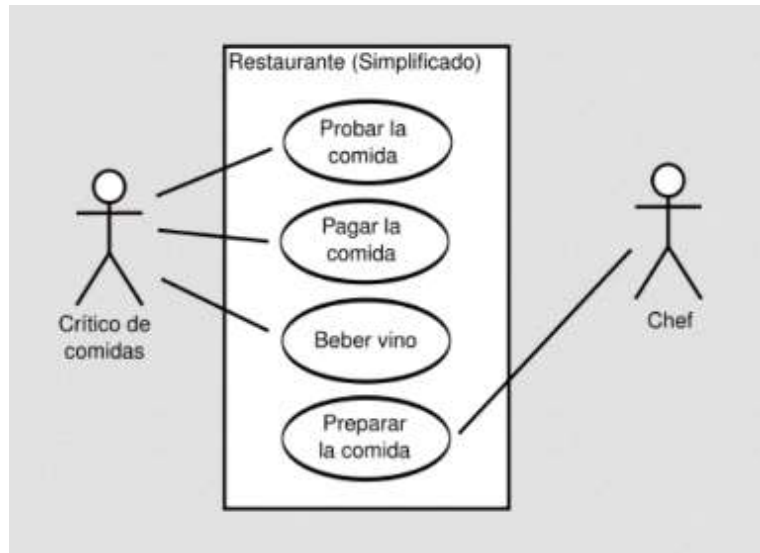
INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



Página | 46

Caso de uso

Se representan con óvalos. La etiqueta en el óvalo indica la función del sistema.



Relaciones

Las relaciones entre un actor y un caso de uso, se dibujan con una línea simple. Para relaciones entre casos de uso, se utilizan flechas etiquetadas “incluir” o “extender.” Una relación “incluir” indica que un caso de uso es necesitado por otro para poder cumplir una tarea. Una relación “extender” indica opciones alternativas para un cierto caso de uso.

Las relaciones conectan los casos de uso con los actores o los casos de uso entre sí.

Cuando conectan un actor con un caso de uso representa que ese actor interactúa de alguna manera con ese caso de uso y se representa con una línea continua con la identificación <<communicates>>.

Cuando conectan casos de uso entre sí se pueden diferenciar dos tipos de relaciones: <<include>> y <<extends>>. En español a veces se usa la nomenclatura <<usa>> y <<extiende>>:

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

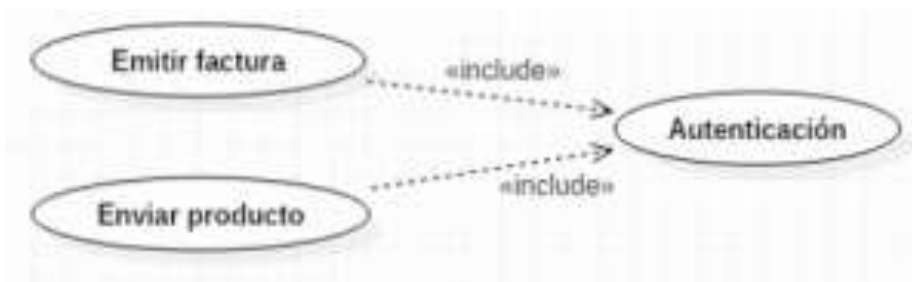
- **<<include>>**: Se utiliza para representar que un caso de uso utiliza siempre a otro caso de uso. Es decir, un caso de uso se ejecutará obligatoriamente (lo incluye, lo usa). Se representa con una flecha discontinua que va desde el caso de uso de origen al caso de uso que se incluye.



Página | 47

Relación include entre dos casos de uso

Un uso típico de este tipo de relaciones se produce cuando dos casos de uso **comparten una funcionalidad**. Esa funcionalidad es extraída de los dos y se crea un caso de uso nuevo que se relaciona con los anteriores con un include.



Ejemplo de uso de include

En este ejemplo, los casos de uso emitir factura y enviar producto ejecutarán ambos el caso de uso autenticación.

- **<<extend>>**: Este tipo de relaciones se utilizan cuando un caso de uso tiene un comportamiento opcional, reflejado en otro caso de uso. Es decir, un caso de uso puede ejecutar, normalmente dependiendo de alguna condición o flujo del programa, otro caso de uso. Se representa con una flecha discontinua que va desde el caso de uso opcional al original.



Relación extend entre dos casos de uso

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

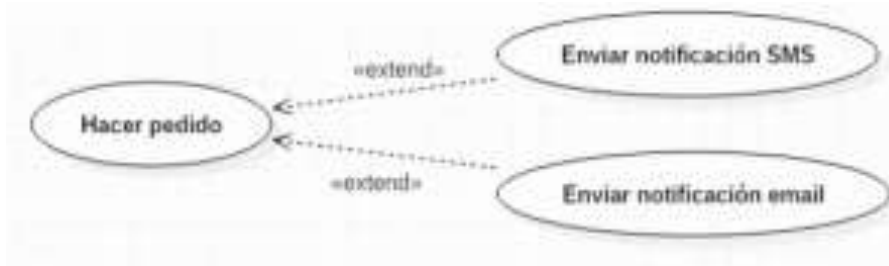
REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

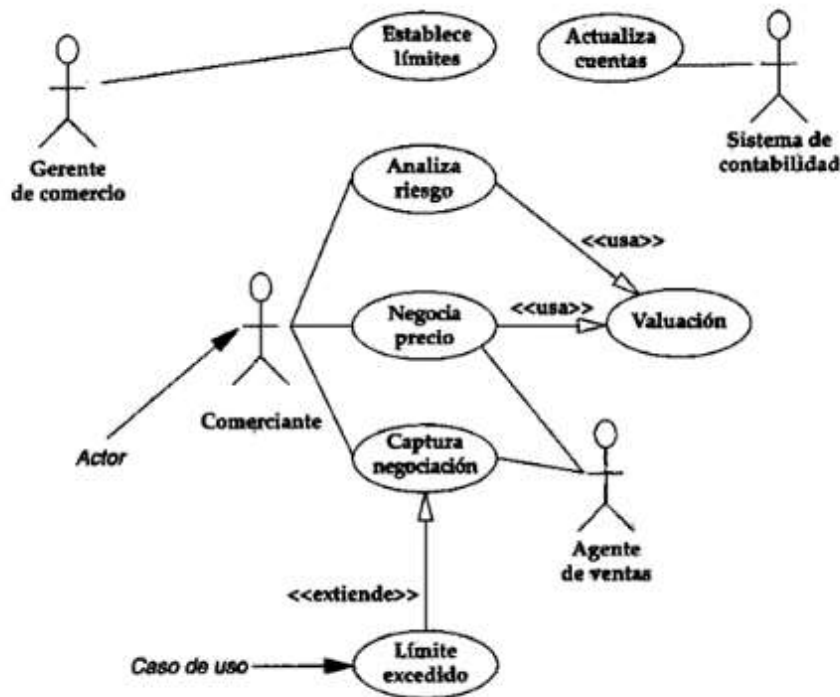
METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Un ejemplo de esta relación podría ser la siguiente:



Ejemplo de relaciones extend



Relaciones de los casos de uso

Las relaciones activas se conocen como relaciones de comportamiento y se utilizan principalmente en los diagramas de casos de uso. Hay cuatro tipos básicos de relaciones de comportamiento: comunica, incluye, extiende y generaliza.



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Relación	Símbolo	Significado
Comunica	—————	Para conectar un actor con un caso de uso se utiliza una línea sin puntas de flecha.
Incluye	<< Incluye >> ←-----	Un caso de uso contiene un comportamiento común para más de un caso de uso. La flecha apunta al caso de uso común.
Extiende	<< Extiende >> ----->	Un caso de uso distinto maneja las excepciones del caso de uso básico. La flecha apunta del caso de uso extendido al básico.
Generaliza	—————▷	Una "cosa" de UML es más general que otra "cosa". La flecha apunta a la "cosa" general.

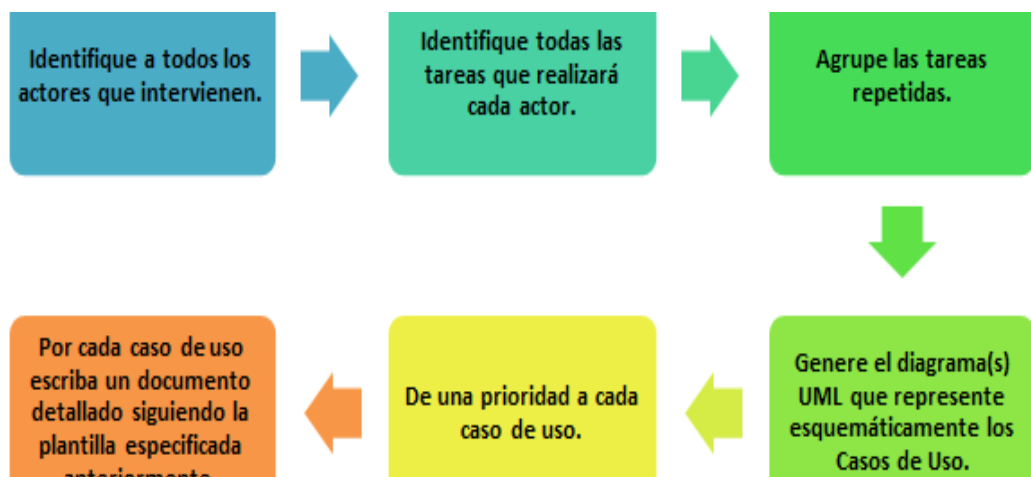
Página | 49

Documentación de los casos de uso

Existen dos formas principales de documentar un caso de uso:

- Un diagrama UML
- Un documento detallado

Documentar casos de usos no es una tarea fácil que se pueda dominar de un día para otro, requiere de tiempo, disciplina y experiencia, sin embargo podemos definir una serie de pasos identificables para escribir los casos de uso.



Formato de la documentación de caso de uso para los actores que participan:

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

ACTOR	PARTICIPA	TIPO	DESCRIPCION
Los actores son aquellos que relacionan directamente con el caso de uso, también se considera actor todo aquello que inicia un caso de uso (por ejemplo una tarea agendada) o responde a un caso de uso (un sistema externo de procesamiento en	Número de casos de uso en los que participe	Primario o Secundario	Breve descripción de la función de actor

a | 50

Documentación de un caso de uso:



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Nombre del caso de uso	Nombre del Caso de Uso	ID Único: ConfRG 003
Área	Planeación de la conferencia	
Actor(es)	Participante	
Interesados	Personal interesado en el caso de uso	
Nivel	Azul	
Descripción	Permite que el docente envíe la tarea a los estudiantes de la clase	
Evento Desencadenado	El docente utiliza el sistema para enviar la tarea a los estudiantes, para esto introduce en nombre del curso al cual se encuentran matriculados los estudiantes.	
Tipo de Desencadenador	Externo	
Pasos Realizados (Ruta Principal)		Información Para los Pasos
1. Se elige entre la lista de cursos asignados a cual enviar la tarea	ID de usuario, Nombre del Curso	
2. Se asigna una fecha máxima de entrega	Fecha	
3. Se ingresa la Descripción de la tarea	Descripción	
4. Se envía la información	Confirmación	
Precondiciones	El docente ya se registró y cuenta con ID de Usuario y contraseña	
Poscondiciones	La tarea se envió con éxito a los estudiantes	
Suposiciones	El docente tiene ID de Usuario y una contraseña	
Garantía de Éxito	La tarea se envió a los estudiantes y estos recibirán la notificación del plazo de entrega	
Garantía Mínima	El docente pudo ingresar al Sistema	
Requerimientos Cumplidos	Permitir el envío de tarea	
Cuestiones Pendientes	En caso de no realizarse con éxito el envío de la tarea se muestra un mensaje de error	
Prioridad	Alta	

Documentación de los Casos de Uso

DIAGRAMAS DE CLASES

Para UML una clase es una entidad, no una clase software. Un diagrama de clases UML puede ser un diagrama del dominio o representación de conceptos que intervienen en un problema, o también un diagrama de clases software. El sentido de un diagrama UML se lo da la persona que lo construye.

El **diagrama de clases** es uno de los diagramas incluidos en UML 2.5 clasificado dentro de los diagramas de estructura y, como tal, se utiliza para representar los elementos que componen un sistema de información desde un punto de vista estático.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Es importante destacar que, por esta misma razón, este diagrama no incluye la forma en la que se comportan a lo largo de la ejecución los distintos elementos, esa función puede ser representada a través de un diagrama de comportamiento, como por ejemplo un diagrama de secuencia o un diagrama de casos de uso.

El diagrama de clases es un **diagrama puramente orientado al modelo de programación orientado a objetos**, ya que define las clases que se utilizarán cuando se pase a la fase de construcción y la manera en que se relacionan las mismas. Se podría equiparar, salvando las distancias, al famoso diagrama de modelo Entidad-Relación (E/R), no recogido en UML, tiene una utilidad similar: la representación de datos y su interacción. Ambos diagramas muestran el modelo lógico de los datos de un sistema.

Página | 52

Elementos de un diagrama de clases

El diagrama UML de clases está formado por dos elementos: clases, relaciones e interfaces.

Clases

Las clases son el elemento principal del diagrama y representa, como su nombre indica, una clase dentro del paradigma de la orientación a objetos. Este tipo de elementos normalmente se utilizan para representar conceptos o entidades del “negocio”. Una clase define un grupo de objetos que comparten características, condiciones y significado. La manera más rápida para encontrar clases sobre un enunciado, sobre una idea de negocio o, en general, sobre un tema concreto es buscar los sustantivos que aparecen en el mismo. Por poner algún ejemplo, algunas clases podrían ser: Animal, Persona, Mensaje, Expediente... Es un concepto muy amplio y resulta fundamental identificar de forma efectiva estas clases, en caso de no hacerlo correctamente se obtendrán una serie de problemas en etapas posteriores, teniendo que volver a hacer el análisis y perdiendo parte o todo el trabajo que se ha hecho hasta ese momento.

Bajando de nivel una clase está compuesta por tres elementos:

- nombre de la clase,
- atributos,
- funciones

Estos elementos se incluyen en la representación (o no, dependiendo del nivel de análisis).

Para representar la clase con estos elementos se utiliza una caja que es dividida en tres zonas utilizando para ello líneas horizontales:



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Página | 53

- La primera de las zonas se utiliza para el nombre de la clase. En caso de que la clase sea abstracta se utilizará su nombre en cursiva.
- La segunda de las zonas se utiliza para escribir los atributos de la clase, uno por línea y utilizando el siguiente formato:

visibilidad nombre_atributo : tipo = valor-inicial { propiedades }

Aunque esta es la forma “oficial” de escribirlas, es común simplificando únicamente poniendo el nombre y el tipo o únicamente el nombre.

- La última de las zonas incluye cada una de las funciones que ofrece la clase. De forma parecida a los atributos, sigue el siguiente formato:

visibilidad nombre_funcion { parametros } : tipo-devuelto { propiedades }

De la misma manera que con los atributos, se suele simplificar indicando únicamente el nombre de la función y, en ocasiones, el tipo devuelto.

Nombre de la Clase
+ Atributos

+ Funciones()

Notación de una clase

Tanto los atributos como las funciones incluyen al principio de su descripción la visibilidad que tendrá. Esta visibilidad se identifica escribiendo un símbolo y podrá ser:

- **(+)** **Pública.** Representa que se puede acceder al atributo o función desde cualquier lugar de la aplicación.
- **(-)** **Privada.** Representa que se puede acceder al atributo o función únicamente desde la misma clase.
- **(#)** **Protegida.** Representa que el atributo o función puede ser accedida únicamente desde la misma clase o desde las clases que hereden de ella (clases derivadas).

Estos tres tipos de visibilidad son los más comunes. No obstante, pueden incluirse otros en base al lenguaje de programación que se esté usando (no es muy común). Por ejemplo: (/) Derivado o (~) Paquete.



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

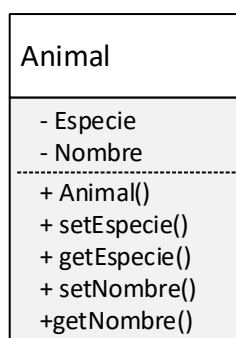
REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Un ejemplo de clase podría ser el siguiente:



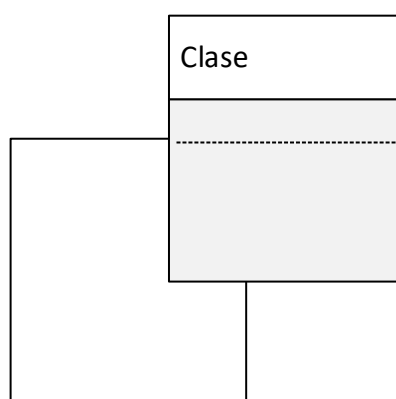
Página | 54

Ejemplo de una clase

En caso de que un atributo o función sea estático, se representa en el diagrama subrayando su nombre. Una característica estática se define como aquella que es compartida por cada clase y no instanciada para cada uno de los objetos de esa clase. Es un concepto muy común.

Relaciones

Una relación identifica una dependencia. Esta dependencia puede ser entre dos o más clases (más común) o una clase hacia sí misma (menos común, pero existen), este último tipo de dependencia se denomina dependencia reflexiva. Las relaciones se representan con una línea que une las clases, esta línea variará dependiendo del tipo de relación



Relación reflexiva

Las relaciones en el diagrama de clases tienen varias propiedades, que dependiendo de la profundidad que se quiera dar al diagrama se representarán o no. Estas propiedades son las siguientes:

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

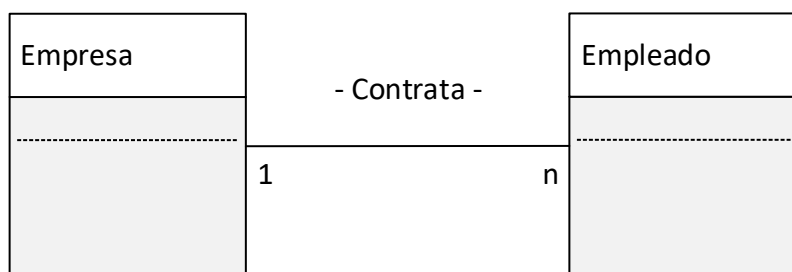
ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

- Multiplicidad. Es decir, el número de elementos de una clase que participan en una relación. Se puede indicar un número, un rango... Se utiliza n o $*$ para identificar un número cualquiera.
- Nombre de la asociación. En ocasiones se escriba una indicación de la asociación que ayuda a entender la relación que tienen dos clases. Suelen utilizarse verbos como, por ejemplo: “Una empresa contrata a n empleados”

Página | 55



Ejemplo de relación Empresa-Empleado

Tipos de relaciones

Un diagrama de clases incluye los siguientes tipos de relaciones:

- Asociación.
- Agregación.
- Composición.
- Dependencia.
- Herencia.

Asociación

Este tipo de relación es el más común y se utiliza para representar dependencia semántica. Se representa con una simple línea continua que une las clases que están incluidas en la asociación.

Un ejemplo de asociación podría ser: “Una mascota pertenece a una persona”.





INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Ejemplo de asociación

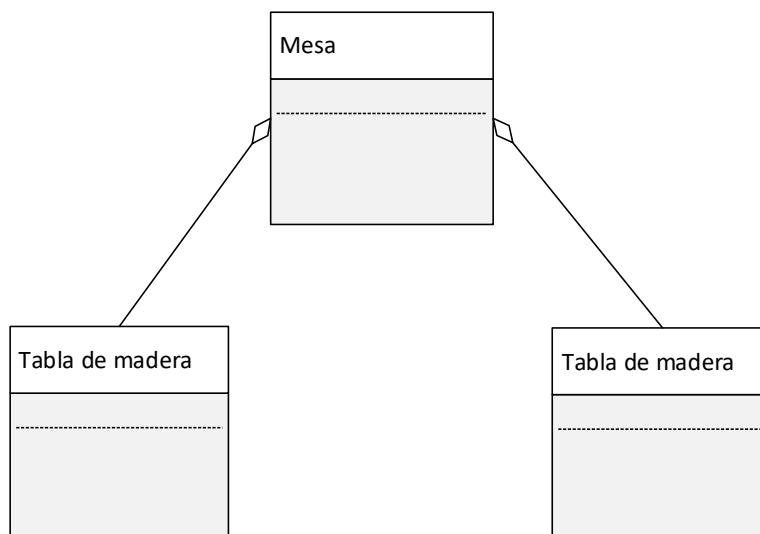
Agregación

Es una representación jerárquica que indica a un objeto y las partes que componen ese objeto. Es decir, representa relaciones en las que un objeto es parte de otro, pero aun así debe tener existencia en sí mismo.

Página | 56

Se representa con una línea que tiene un rombo en la parte de la clase que es una agregación de la otra clase (es decir, en la clase que contiene las otras).

Un ejemplo de esta relación podría ser: “Las mesas están formadas por tablas de madera y tornillos o, dicho de otra manera, los tornillos y las tablas forman parte de una mesa”. Como ves, el tornillo podría formar parte de más objetos, por lo que interesa especialmente su abstracción en otra clase.



Ejemplo de agregación

Composición

La composición es similar a la agregación, representa una relación jerárquica entre un objeto y las partes que lo componen, pero de una forma más fuerte. En este caso, los elementos que forman parte no tienen sentido de existencia cuando el primero no existe. Es decir, cuando el elemento que contiene los otros desaparece, deben desaparecer todos ya que no tienen sentido por sí mismos, sino que dependen del elemento que componen. Además, suelen tener los



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

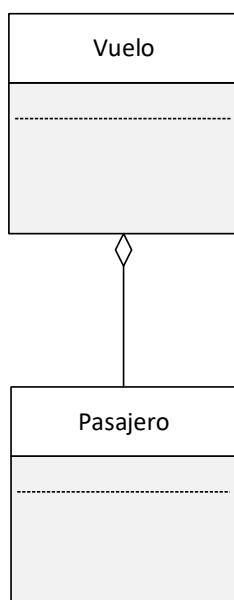
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

mismos tiempos de vida. Los componentes no se comparten entre varios elementos, esta es otra de las diferencias con la agregación.

Se representa con una línea continua con un rombo relleno en la clase que es compuesta.

Un ejemplo de esta relación sería: “Un vuelo de una compañía aérea está compuesto por pasajeros, que es lo mismo que decir que un pasajero está asignado a un vuelo”

Página | 57



Ejemplo de composición

Diferencia entre agregación y composición

La diferencia entre agregación y composición es semántica, por lo que a veces no está del todo definida. Ninguna de las dos tiene análogos en muchos lenguajes de programación (como por ejemplo Java).

Un “agregado” representa un todo que comprende varias partes; de esta manera, un Comité es un agregado de sus Miembros. Una reunión es un agregado de una agenda, una sala y los asistentes. En el momento de la implementación, esta relación no es de contención. (Una reunión no contiene una sala). Del mismo modo, las partes del agregado podrían estar haciendo otras cosas en otras partes del programa, por lo que podrían ser referenciadas por varios objetos que nada tienen que ver. En otras palabras, no existe una diferencia de nivel de implementación



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Página | 58

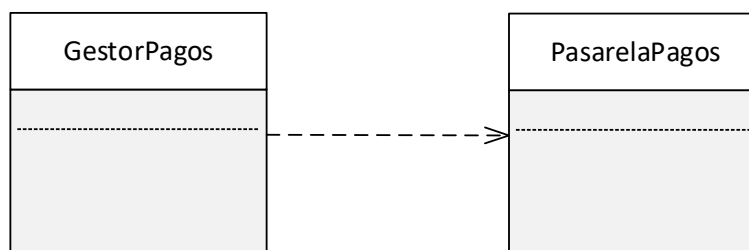
entre la agregación y una simple relación de “usos”. En ambos casos, un objeto tiene referencias a otros objetos. Aunque no existe una diferencia en la implementación, definitivamente vale la pena capturar la relación en el diagrama UML, tanto porque ayuda a comprender mejor el modelo de dominio, como porque puede haber problemas de implementación que pueden pasar desapercibidos. Podría permitir relaciones de acoplamiento más estrictas en una agregación de lo que haría con un simple “uso”, por ejemplo.

La composición, por otro lado, implica un acoplamiento aún más estricto que la agregación, y definitivamente implica la contención. El requisito básico es que, si una clase de objetos (llamado “contenedor”) se compone de otros objetos (llamados “elementos”), entonces los elementos aparecerán y también serán destruidos como un efecto secundario de crear o destruir el contenedor. Sería raro que un elemento no se declare como privado. Un ejemplo podría ser el nombre y la dirección del Cliente. Un cliente sin nombre o dirección no tiene valor. Por la misma razón, cuando se destruye al cliente, no tiene sentido mantener el nombre y la dirección. (Compare esta situación con la agregación, donde destruir al Comité no debe causar la destrucción de los miembros, ya que pueden ser miembros de otros Comités).

Dependencia

Se utiliza este tipo de relación para representar que una clase requiere de otra para ofrecer sus funcionalidades. Es muy sencilla y se representa con una flecha discontinua que va desde la clase que necesita la utilidad de la otra flecha hasta esta misma.

Un ejemplo de esta relación podría ser la siguiente:



Ejemplo de dependencia



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

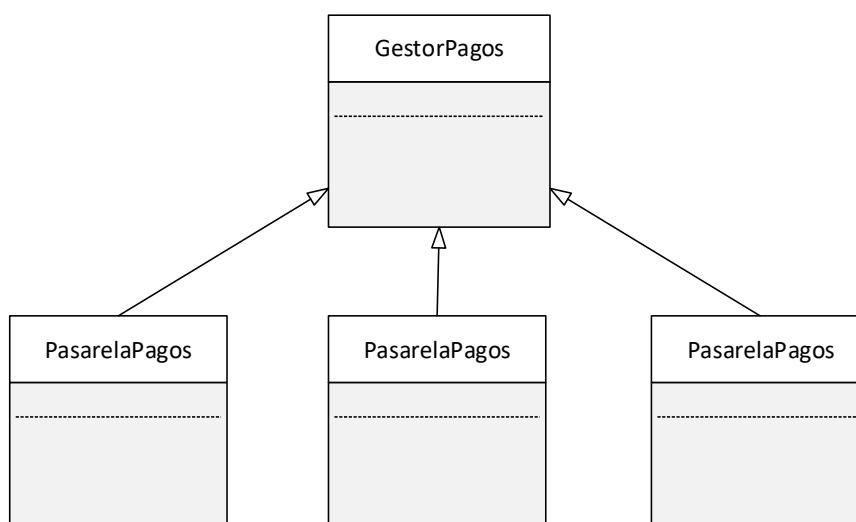
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Herencia

Otra relación muy común en el diagrama de clases es la herencia. Este tipo de relaciones permiten que una clase (clase hija o subclase) reciba los atributos y métodos de otra clase (clase padre o superclase). Estos atributos y métodos recibidos se suman a los que la clase tiene por sí misma. Se utiliza en relaciones “es un”.

Página | 59

Un ejemplo de esta relación podría ser la siguiente: Un pez, un perro y un gato son animales.



Ejemplo de Herencia

En este ejemplo, las tres clases (Pez, Perro, Gato) podrán utilizar la función respirar, ya que lo heredan de la clase animal, pero solamente la clase Pez podrá nadar, la clase Perro ladrar y la clase Gato maullar. La clase Animal podría plantearse ser definida abstracta, aunque no es necesario.

Interfaces

Una interfaz es una entidad que declara una serie de atributos, funciones y obligaciones. Es una especie de contrato donde toda instancia asociada a una interfaz debe de implementar los servicios que indica aquella interfaz.

Dado que únicamente son declaraciones no pueden ser instanciadas.

Las interfaces se asocian a clases. Una asociación entre una clase y una interfaz representa que esa clase cumple con el contrato que indica la



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

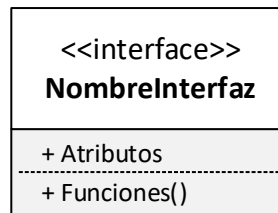
METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

interfaz, es decir, incluye aquellas funciones y atributos que indica la interfaz.

Su representación es similar a las clases, pero indicando arriba la palabra <<interface>>.

Página | 60



Notación de interfaz

Ejemplo de un diagrama de clases de una tienda
un ejemplo muy sencillo de una tienda:



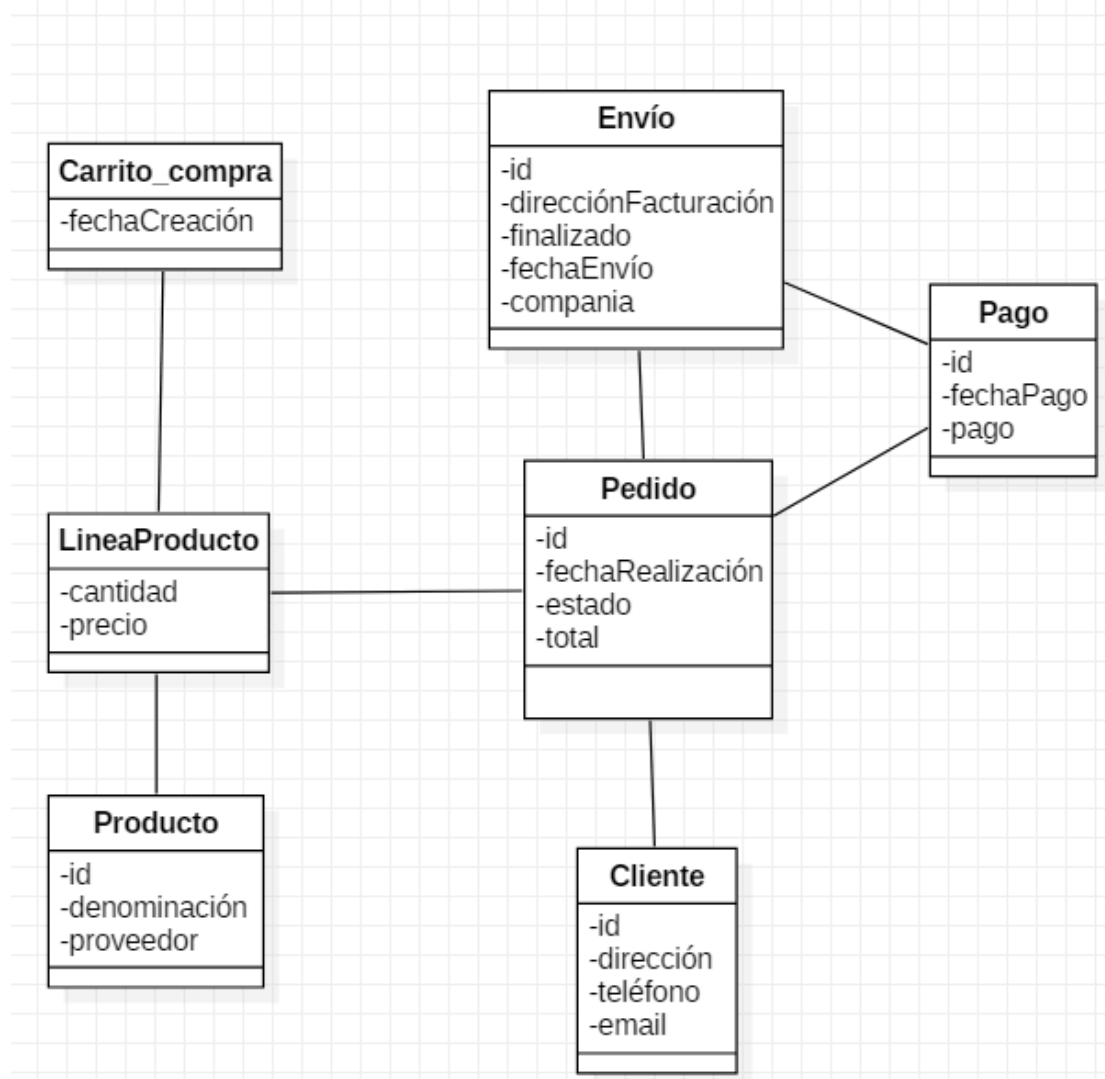
INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



Página | 61

DIAGRAMAS DE ESTADO Y ACTIVIDADES

Suelen usarse para representar cómo evoluciona un sistema (cómo va cambiando de estado) a medida que se producen determinados eventos.

Diagramas de estados

Los diagramas de estados nos muestran los diferentes estados por lo que puede pasar un objeto dentro de una aplicación, así como los cambios que le permiten pasar de un estado a otro.

Los diagramas de estado son útiles para describir el comportamiento de clases y sistemas que han sido concebidos haciendo uso de un modelo de estados.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

En un modelo de estados se identifican las situaciones en la que el comportamiento o capacidad de respuesta con cualitativamente diferentes, así como los eventos o condiciones bajo las que se pasa de una situación a otra (transiciones de estados).

Los diagramas de estados son intensivamente utilizados en:

- Sistemas de tiempo real y críticos.
- La descripción de sistemas reactivos.
- La descripción de sistemas basados en protocolos

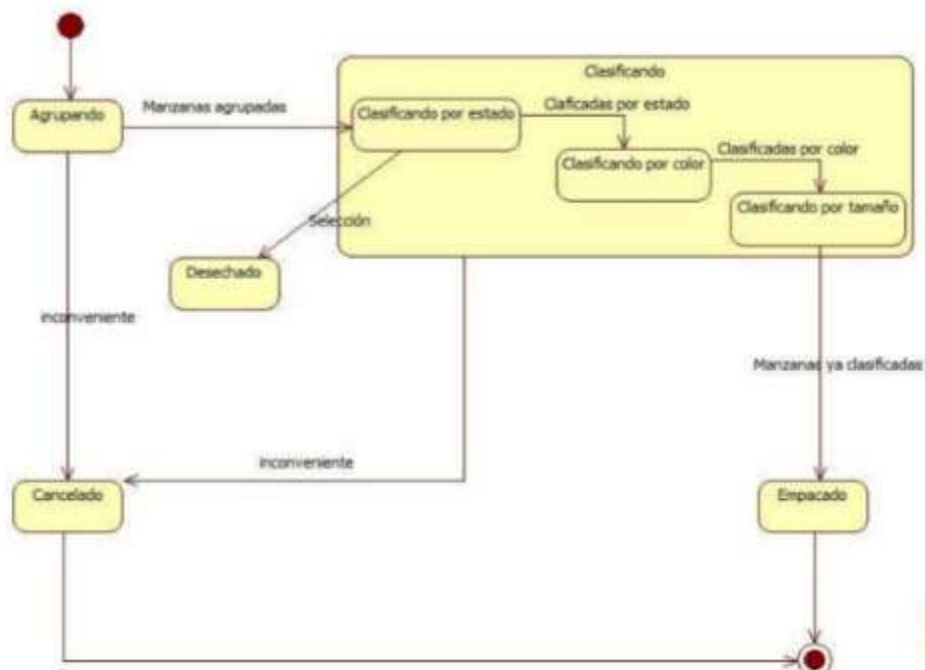
Página | 62

Un estado puede cualificarse:

- Estáticamente en función del valor que tienen sus atributos.
- Dinámicamente, esto es en función de la actividad que ejecuta.

Su comportamiento describe las acciones que se producen mientras que el sistema se encuentra en un estado:

- **entry/behavior** => Acción que se realiza cuando se llega a un estado.
- **do/behavior** => Actividad que se ejecuta mientras se está en un estado.
- **Exit/behavior** => Acciones que se ejecuta cuando se abandona un estado.
- **Transiciones internas** => Se formulan como trigger[guard]/behavior





INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Diagrama de Estados

Diagramas de actividades



Página | 63

Es un caso específico de los diagramas de estados donde los estados son estados de acción y la mayoría de transacciones se envían al finalizar la acción ejecutada en el estado anterior.

Este tipo de diagrama permiten el paralelismo de acciones mostrando las rutas de decisiones que existe en el proceso global.

Los diagramas de actividad permiten describir como un sistema implementa su funcionalidad.

Los diagramas de actividad modelan el comportamiento dinámico de un procedimiento, transacción o caso de uso haciendo énfasis en el proceso que se lleva a cabo.

Los diagramas de actividad es uno de los elementos de modelado que son mejor comprendidos por todos, ya que son herederos directos de los diagramas de flujo.

Los diagramas de actividad son más expresivos que los diagramas de flujo.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

También heredan características de:

- Los diagramas de estado.
- Los diagramas de flujo de datos.
- Las redes de Petri.

Página | 64

DIAGRAMAS DE DESPLIEGUE

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. Los estereotipos permiten precisar la naturaleza del equipo:

- Dispositivos
- Procesadores
- Memoria

Los nodos se interconectan mediante soportes bidireccionales que pueden a su vez estereotiparse. Esta vista permite determinar las consecuencias de la distribución y la asignación de recursos. Las instancias de los nodos pueden contener instancias de ejecución, como instancias de componentes y objetos. El modelo puede mostrar dependencias entre las instancias y sus interfaces, y también modelar la migración de entidades entre nodos u otros contenedores.

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos, procesos (caso particular de un objeto). En general un nodo será una unidad de computación de algún tipo, desde un sensor a un mainframe. Las instancias de componentes software pueden estar unidas por relaciones de dependencia, posiblemente a interfaces (ya que un componente puede tener más de una interfaz).

Componentes del diagrama de despliegue

Un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento. Pueden representarse instancias o tipos de nodos que se representa como un cubo 3D en los diagramas de implementación.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192

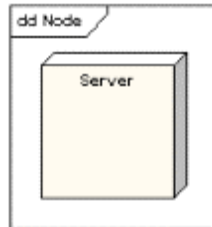


INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



Página | 65

Las instancias de componentes de software muestran unidades de software en tiempo de ejecución y generalmente ayudan a identificar sus dependencias y su localización en nodos. Pueden mostrar también qué interfaces implementan y qué objetos contienen. Su representación es un rectángulo atravesado por una elipse y dos rectángulos más pequeños.

Instancia de un nodo.

Una instancia de nodo se puede mostrar en un diagrama. Una instancia se puede distinguir desde un nodo por el hecho de que su nombre está subrayado y tiene dos puntos antes del tipo de nodo base. Una instancia puede o no tener un nombre antes de los dos puntos. El siguiente diagrama muestra una instancia nombrada de una computadora.

Estereotipo de Nodo.

Un número de estereotipos estándar se proveen para los nodos, nombrados «cdrom», «cdrom», «computer», «disk array», «pc», «pc client», «pc server», «secure», «server», «storage», «unix server», «user pc». Estos mostrarán un icono apropiado en la esquina derecha arriba del símbolo nodo.

Artefactos.

Un artefacto puede ser algo como un archivo, un programa, una biblioteca, o una base de datos construida o modificada en un proyecto. Estos artefactos implementan colecciones de componentes. Los nodos internos indican ambientes, un concepto más amplio que el hardware propiamente dicho, ya que un ambiente puede incluir al lenguaje de programación, a un sistema operativo, un ordenador o un clúster de terminales.

Estandares

1. **Executable:** especifica un artefacto que se puede ejecutar en un nodo.
2. **Library:** Biblioteca de objetos estática o dinámica.
3. **File:** artefacto que representa un documento que contiene código fuente o datos.
4. **Document:** artefacto que representa un documento.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Asociación

En el contexto del diagrama de despliegue, una asociación representa una ruta de comunicación entre los nodos. El siguiente diagrama muestra un diagrama de despliegue para una red, mostrando los protocolos de red como estereotipos y también mostrando multiplicidades en los extremos de la asociación.

Página | 66

Nodo como contenedor

Un nodo puede contener otros elementos, como componentes o artefactos. El siguiente diagrama muestra un diagrama de despliegue para una parte del sistema embebido y muestra un artefacto ejecutable como contenido por el nodo madre (motherboard).

Usos

Esta vista cubre principalmente la distribución, entrega e instalación de las partes que configuran un sistema físico. Los diagramas de despliegue se suelen utilizar para modelar: Sistemas empotrados: Un sistema empotrado es una colección de hardware con una gran cantidad de software que interactúa con el mundo físico. Los sistemas empotrados involucran software que controla dispositivo (motores, actuadores) que a su vez están controlados por estímulos externos como sensores.

- **Sistemas cliente-servidor:** Los sistemas cliente-servidor son un extremo del espectro de los sistemas distribuidos y requieren tomar decisiones sobre la conectividad de red de los clientes a los servidores y sobre la distribución física de los componentes software del sistema a través de nodos.
- **Sistemas completamente distribuidos:** En el otro extremo encontramos aquellos sistemas que son ampliamente o totalmente distribuidos y que normalmente incluyen varios niveles de servidores. Tales sistemas contienen a menudo varias versiones de componentes software, alguno de los cuales pueden incluso migrar de un nodo a otro. El diseño de tales sistemas requiere tomar decisiones que permitan un cambio continuo de la topología del sistema.

Dependencias

- **Diagrama de Componentes:** permiten modelar sistemas de software de cualquier tamaño y complejidad. Permite especificar un componente como unidad modular con interfaces bien definidas.
- **Diagrama de Paquetes:** más que un diagrama constituye una herramienta para mostrar los elementos que se integran en un sistema, aplicación o

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

módulos. Muestra como el sistema está dividido en agrupaciones lógicas mostrando las dependencias entre agrupaciones.

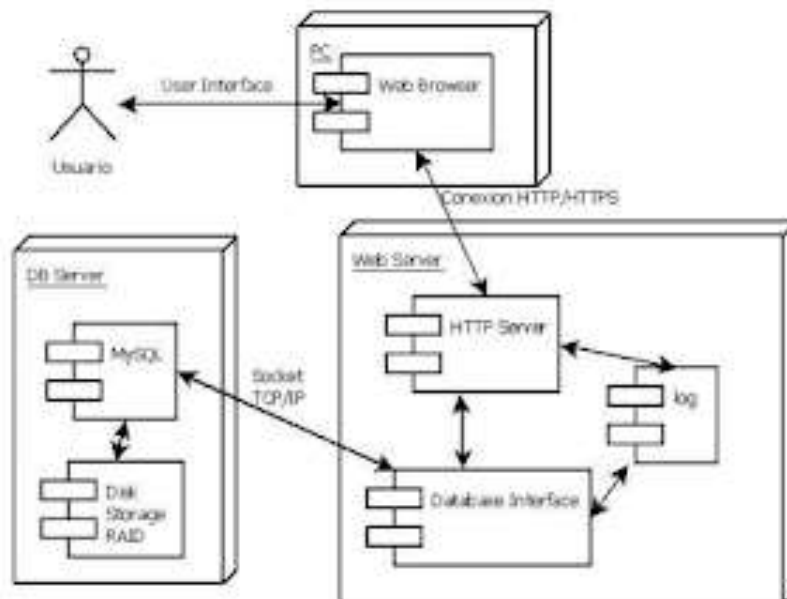
Ventajas

- Muestra un conjunto de nodos y sus relaciones.
- Se utilizan para describir la vista de despliegue estática de un sistema.
- Se relacionan con los diagramas de componentes, ya que un nodo normalmente incluye uno o más componentes.

Página | 67

Desventajas

- La posible falla en la modelación de un hardware.
- Tales sistemas contienen a menudo varias versiones de componentes software, alguno de los cuales pueden incluso migrar de un nodo a otro. El diseño de tales sistemas requiere tomar decisiones que permitan un cambio continuo de la topología del sistema.



Ejemplo de Diagrama de Despliegue







INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Representación	Nombre
	Nodo
	Componente
	Interface
	Dependencia

Página | 68

Componentes de un diagrama de despliegue

DIAGRAMAS DE BASE DE DATOS DE SOFTWARE

Una base de datos es una forma de almacenar información y sus relaciones dentro de tablas, para poder acceder a estos datos más adelante. Las bases de datos las encontramos típicamente detrás de plataformas web, que deben almacenar información que los usuarios suben y que quieren compartir o acceder a ella más tarde.

Uno de los pasos cruciales en la construcción de una aplicación que maneje una base de datos, es sin duda, el diseño de la base de datos.

Si las tablas no son definidas apropiadamente, podemos tener muchos dolores de cabeza al momento de ejecutar consultas a la base de datos para tratar de obtener algún tipo de información.

No importa si nuestra base de datos tiene sólo 20 registros, o algunos cuantos miles, es importante asegurarnos que nuestra base de datos está correctamente diseñada para que tenga eficiencia y que se pueda seguir utilizando por largo del tiempo.

Dependiendo de los requerimientos de la base de datos, el diseño puede ser algo complejo, pero con algunas reglas simples que tengamos en la cabeza será mucho más fácil crear una base de datos perfecta para nuestro siguiente proyecto.

El Diagrama de Clase presenta un mecanismo de implementación neutral para modelar los aspectos de almacenamiento de datos del sistema. Las clases persistentes, sus atributos, y sus relaciones pueden ser implementadas directamente en una base de datos orientada a objetos. Aun así, en el entorno de desarrollo actual, la base de datos relacional es el método más usado para el almacenamiento de datos. Es en el modelado de esta área donde UML se queda corto.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

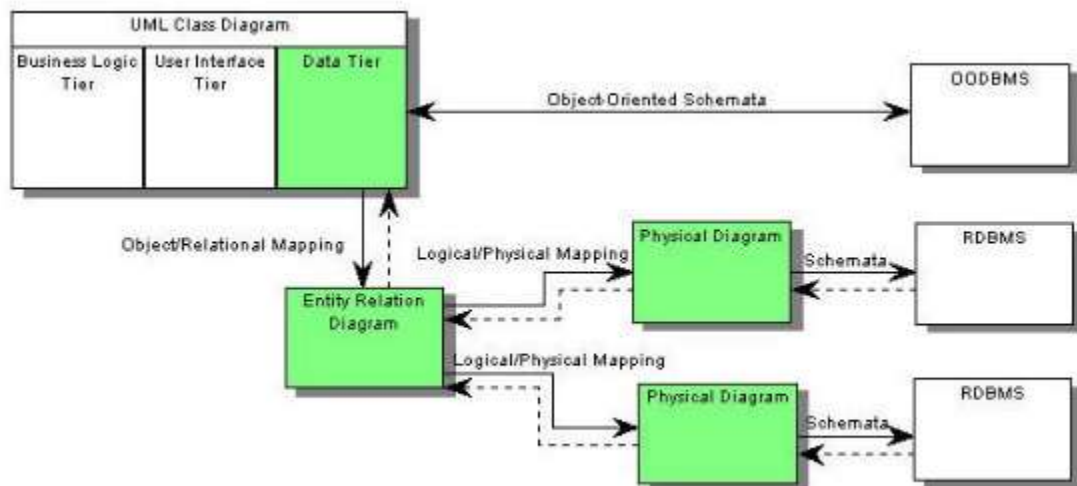
METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

El diagrama de clase de UML se puede usar para modelar algunos aspectos del diseño de bases de datos relacionales, pero no cubre toda la semántica involucrada en el modelado relacional, mayoritariamente la noción de atributos clave que relacionan entre sí las tablas unas con otras. Para capturar esta información, un Diagrama de Relación de Entidad (ER diagrama) se recomienda como extensión a UML.

Página | 69

El Diagrama de Clase se puede usar para modelar la estructura lógica de la base de datos, independientemente de si es orientada a objetos o relacional, con clases representando tablas, y atributos de clase representando columnas. Si una base de datos relacional es el método de implementación escogido, entonces el diagrama de clase puede ser referenciados a un diagrama de relación de entidad lógico. Las clases persistentes y sus atributos hacen referencia directamente a las entidades lógicas y a sus atributos; el modelador dispone de varias opciones sobre cómo inferir asociaciones en relaciones entre entidades. Las relaciones de herencia son referenciadas directamente a super-sub relaciones entre entidades en un diagrama de relación de entidad (ER diagrama).



Extensión de UML -- Diseño de Bases de Datos Relacionales con el Diagrama de Relación de Entidad

La idea es construir un modelo lógico que sea conforme a las reglas de normalización de datos; mientras que, el diagrama físico se usa para modelar propiedades específicas de cada fabricante para el RDBMS. Se crean varios diagramas físicos si hay varios RDBMSs siendo 'deployed'; cada diagrama físico representa uno de los RDBMS que son nuestro objetivo.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

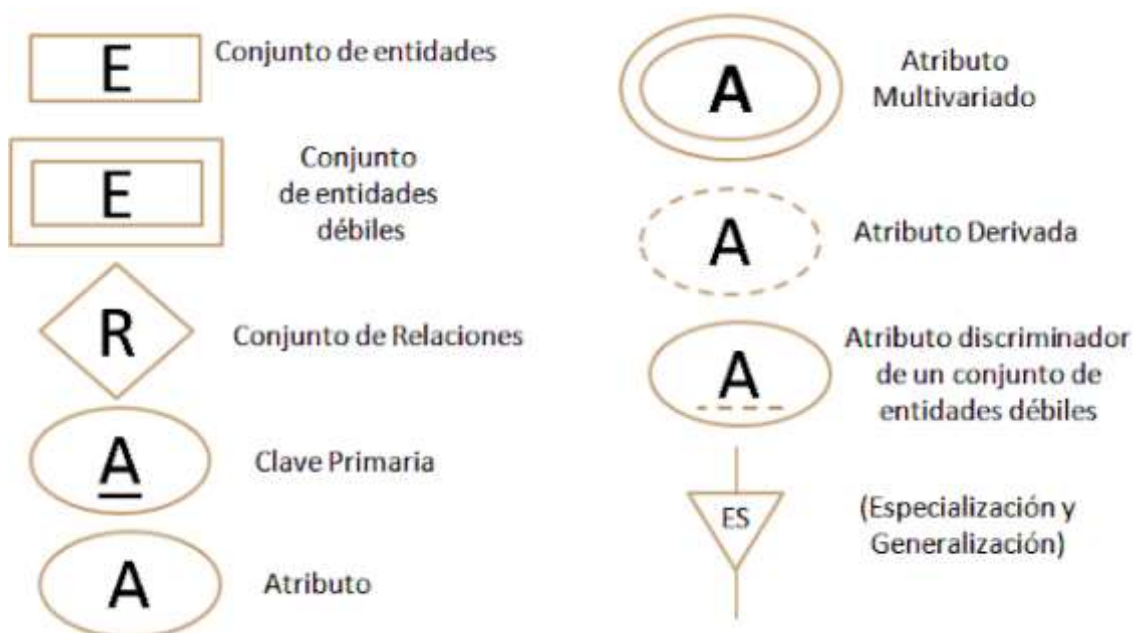
METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Componentes de los Diagramas de Base de Datos (Entidad Relación)

Es la representación gráfica del Modelo Entidad-Relación y permite ilustrar la estructura de la base de datos del negocio modelado está compuesto por los siguientes elementos.

- **Rectángulos:** representan conjuntos de entidades.
- **Elipses:** representan atributos.
- **Rombos:** representan relaciones.
- **Líneas:** unen atributos a conjuntos de entidades y conjuntos de entidades a conjuntos de relaciones.
- **Elipses dobles:** representan atributos multi valorados.
- **Elipses discontinuas:** que denotan atributos derivados.
- **Líneas dobles:** indican participación total de una entidad en un conjunto de relaciones.
- **Rectángulos dobles:** representan conjuntos de entidades débiles.

Página | 70



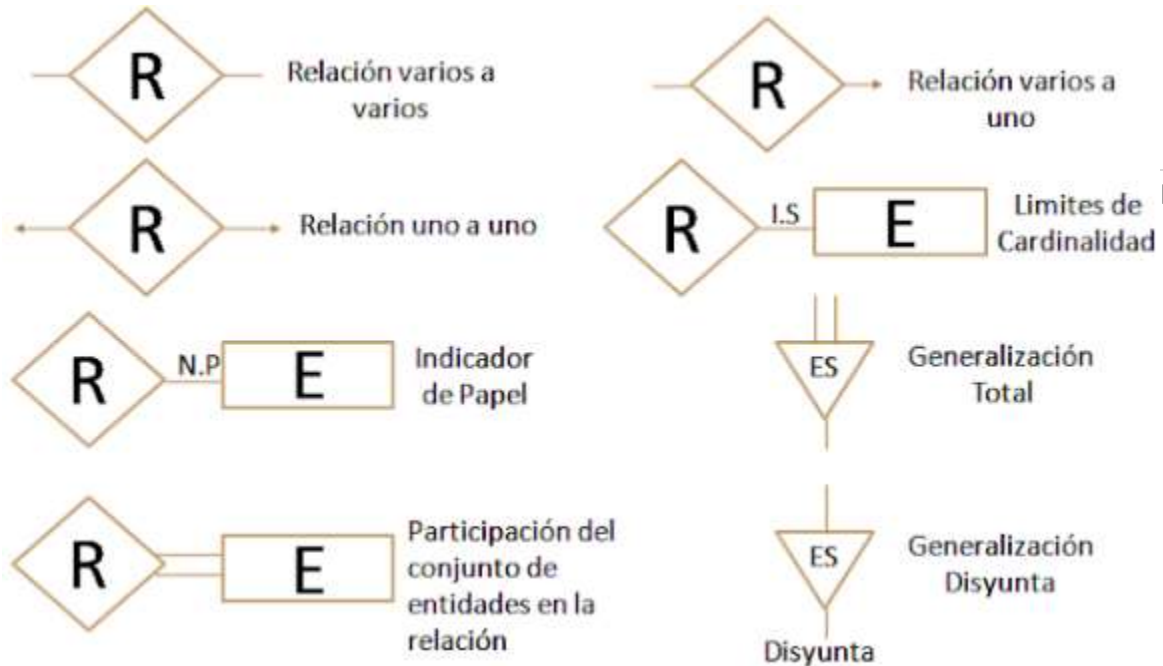


INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



Página | 71

Simbología de Diagrama de Base de Datos (Entidad Relación)

LA CARDINALIDAD

Es simplemente la forma en que se relacionan las entidades, o expresa cuántas entidades se relacionan con otras entidades. Hay varias maneras de mostrar las cardinalidades:

Poner etiquetas en las líneas que unen las relaciones con las entidades, consiste en un mínimo y máximo que contiene un cero (varios a varios) y lo usual es poner una "M".

Existen 4 tipos de relaciones que pueden establecerse entre entidades, las cuales establecen con cuántas ocurrencias de entidad de tipo B se puede relacionar una ocurrencia de entidad de tipo A:

- Relación uno a uno.
- Relación uno a varios (n).
- Relación varios (n) a uno.
- Relación varios a varios (n)- (n)

TIPOS DE RELACIONES

- Relaciones "uno a uno"
- Relaciones de "uno a varios"
- Relaciones de "varios con varios"

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



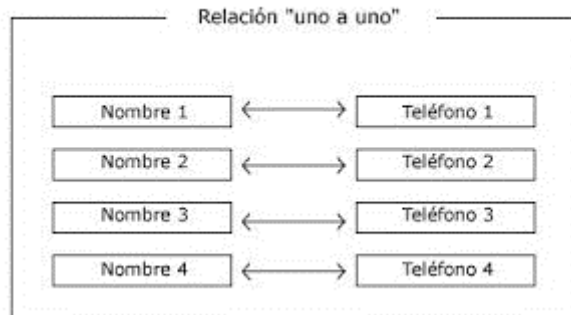
INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

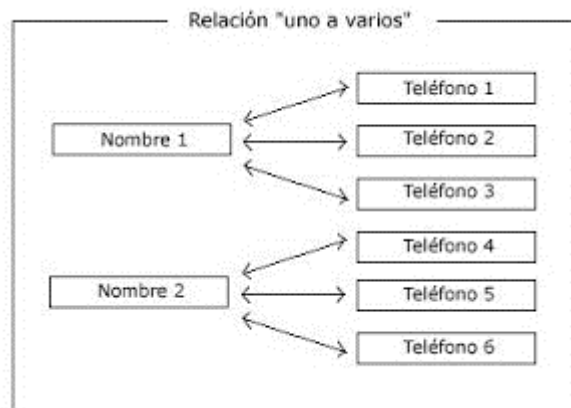
ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

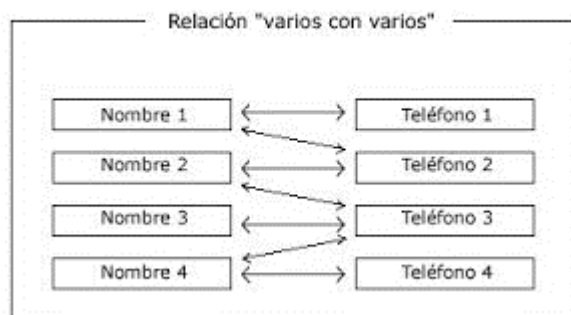
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



Ejemplo Relación uno a uno (1->1)



Ejemplo Relación uno a varios (1->n)



Ejemplo Relación varios a varios(n->n)



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N°. 17 – 082

ACUERDO N° 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

DIAGRAMAS DE INTERFACES DEL SOFTWARE

La interfaz gráfica de usuario, que muchas veces es referida como GUI por sus siglas en inglés (Graphical User Interface), es un tipo de interfaz de usuario que se caracteriza y diferencia por el hecho de utilizar un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

Página | 73

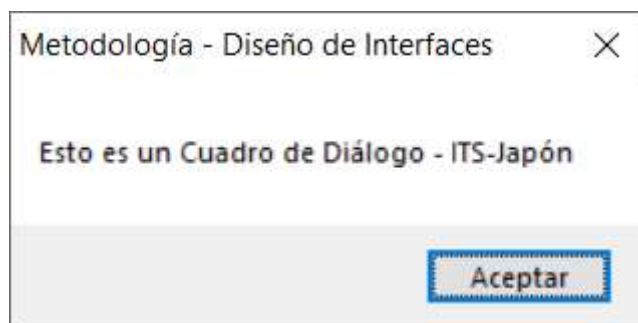
Actualmente todos o la gran mayoría de sistemas operativos cuentan con una interfaz gráfica de usuario. Anteriormente solamente se contaba con acceso a línea de comandos, desde donde se utilizaban instrucciones o comandos para realizar determinados procesos.

Ejemplos típicos de interfaz gráfica de usuario son el escritorio del sistema operativo Windows, el entorno X-Window de Linux, en entorno Aqua de Mac OS X, etc.

Elementos de una interfaz gráfica de usuario

No todas las interfaces gráficas de usuario tienen los mismos elementos, pero típicamente tienen componentes tales como:

- **Cuadros de diálogo:** es una pequeña ventana que se utiliza para abrir un diálogo con el usuario para intercambiar información.



Ejemplo Cuadro de Diálogo

- **Menús:** sirven para desplegar un conjunto de opciones disponibles en un programa o aplicación.



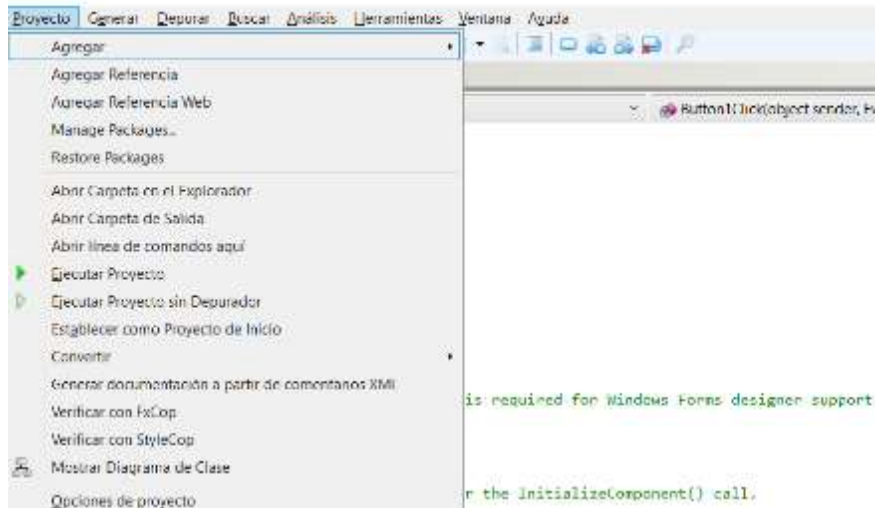
INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



Página | 74

Ejemplo Menús

- **Pestañas de propiedades:** dan acceso a un conjunto de propiedades o configuraciones ordenadas por secciones:



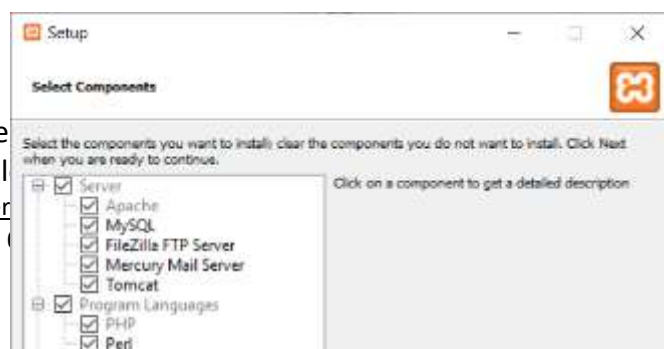
Ejemplo Pestañas

- **Barras de herramientas:** presentan opciones de fácil acceso, que aunque usualmente también se encuentran disponibles en las opciones de menú o mediante combinaciones de teclas, las barras de herramientas ofrecen las principales funcionalidades de formato y otras opciones, de forma fácil y accesible a un solo clic o a un par de clics de distancia:



Ejemplo Barra de herramientas

- **Asistentes:** son ventanas sucesivas que van guiando a un usuario en la realización de un proceso, como la instalación o desinstalación de un programa, por ejemplo.





INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

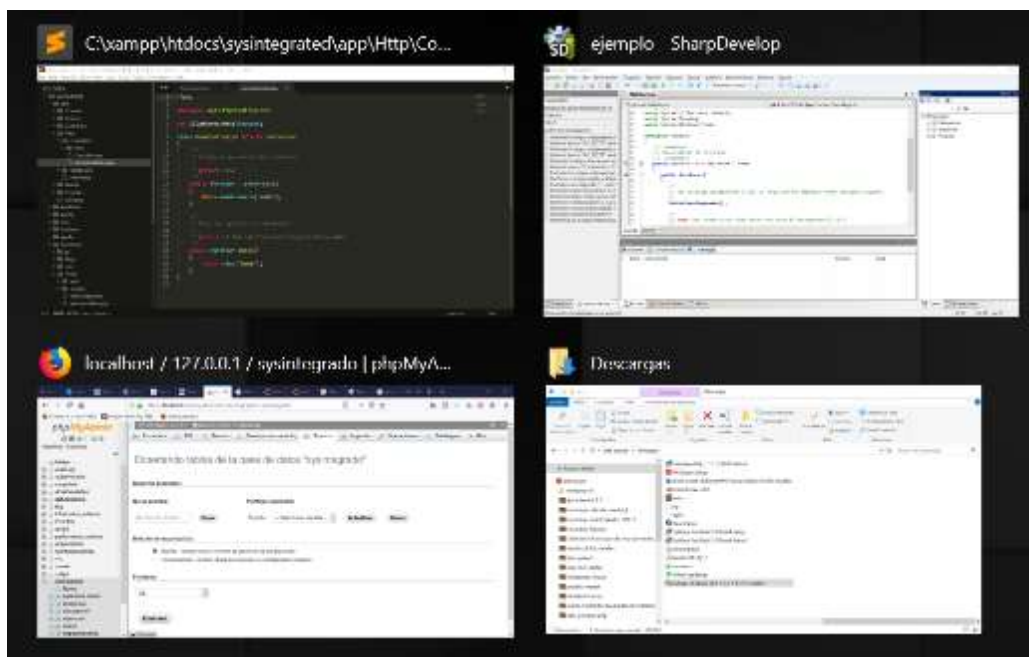
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



Página | 75

Ejemplo de Asistentes

- **Ventanas:** son áreas visuales, normalmente de forma rectangular, que contienen algún tipo de interfaz de usuario, mostrando la salida y permitiendo la entrada de datos para realizar determinados procesos. Se utilizan en las interfaces gráficas de usuario y pueden ser manipuladas con un puntero



CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21

Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

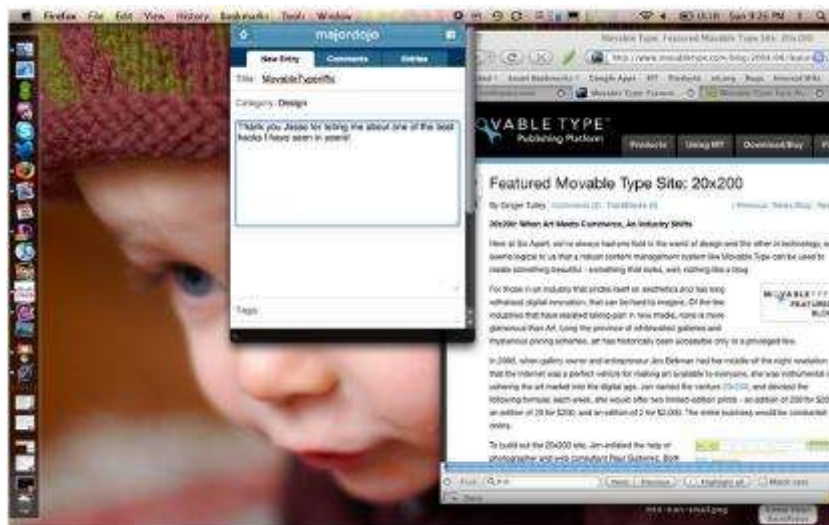
METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Ejemplo Ventanas

- **Entorno de escritorio:** es una solución completa de interfaz gráfica de usuario, que provee al usuario elementos tales como íconos, barras de herramientas, aplicaciones e integración entre aplicaciones con posibilidades de realizar acciones como arrastrar y soltar. En general todo esto proporciona un manejo amigable y fácil del sistema operativo.

Página | 76



Ejemplo Entorno de escritorio Mac OS X

Importancia de las interfaces gráficas de usuario en los sistemas

Si bien es cierto que muchos expertos están acostumbrados a trabajar desde una consola haciendo uso de comandos (por ejemplo, muchos usuarios avanzados de Linux), eso no es de ninguna manera argumento para restar valor o importancia a las interfaces gráficas de usuario. Se debe considerar que los sistemas casi siempre van enfocados a tener algún tipo de interacción con usuarios finales, y éstos aprecian y además esperan que se les proporcionen sistemas con interfaces gráficas de calidad, amenas o amigables y fáciles de usar.

La importancia de las GUI es evidente al considerar que los sistemas interactúan con usuarios finales y aun cuando no lo hicieran, incluso usuarios avanzados o expertos no tendrían problema en tener al frente una interfaz gráfica de calidad y que permita realizar todas las operaciones que se requieran de manera rápida, eficiente, llamativa y amigable.

Además, cuando se trata de vender un producto (o un sistema), la vista es uno de los sentidos más poderosos para capturar la atención. Un ambiente gráfico de calidad puede atraer más usuarios o hacer que los productos o sistemas se vendan en mayor cantidad o a un mayor precio. Además de los efectos o aspectos visuales, las GUI incorporan usualmente sonidos y

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

movimientos, es decir efectos multimedia que provocan una excelente impresión en los usuarios y ofrecen ventajas y beneficios indiscutibles.

DIAGRAMAS DE MENU DEL SOFTWARE

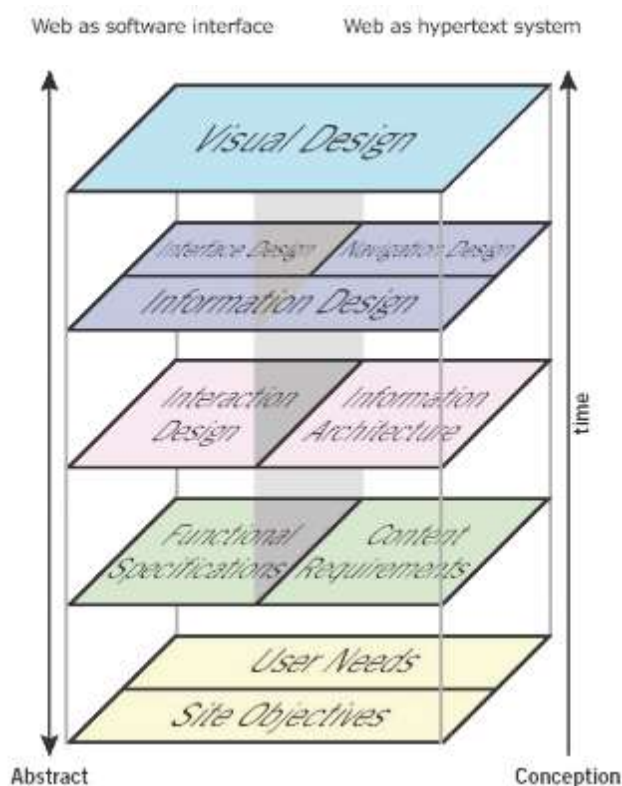
En informática, un menú es una serie de opciones que el usuario puede elegir para realizar determinadas tareas.

Página | 77

Los menús están contenidos en una barra de menú, la cual se puede decir que es una propiedad que tienen las ventanas para poseer menús, esto es porque la barra de menú en sí misma no es una ventana como lo puede ser un botón de comando o un cuadro de texto, pero tampoco es una barra de herramientas, sino un objeto contenedor de otros menús.

El diseño de navegación forma parte de la Arquitectura de la información, disciplina más amplia encargada de proponer principios prácticos para la estructuración, etiquetado y navegación de sistemas y/o sitios web. Ambas disciplinas se nutren, entre otras cosas, de resultados obtenidos por los estudios de Usabilidad.

Diagrama de los elementos de la experiencia de usuario según Garrett



En la práctica, es evidente que la Usabilidad y la Arquitectura de la Información se condicionan mutuamente. Cuando se desarrolla un sistema, los principios de la Arquitectura de la

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Información orientan la planificación del sitio, pero su eficacia real solamente puede ser medida mediante test's con usuarios reales siguiendo los principios de la Usabilidad. Cuando se trata de evaluar, a posterior, la calidad de un sitio, es imprescindible hacer estudios de usabilidad, pero, a pesar de todo, «el usuario no es un diseñador», así que también será necesario aplicar análisis de expertos en base, nuevamente, a los principios de la Arquitectura de la Información.

Página | 78

Volviendo al diseño de navegación (diagrama de Menús), éste se encarga de determinar y seleccionar los sistemas de navegación más adecuados a los contenidos, los objetivos y los usuarios propios de un determinado sistema:

- Un sistema (**diagrama de Menús**) de navegación es cada una de las piezas de la interfaz de usuario de una página Web encargada de permitir y favorecer la navegación hipertextual. Ejemplos de sistemas de navegación son menús, barras de navegación, migas de pan, mapas de navegación, índices, visitas guiadas.
- Existen diversas formas de clasificar y presentar los sistemas de navegación. Pero, a efectos de los aspectos estos presentan una clasificación articulada en
- torno a dos grandes tipos: sistemas de navegación integrados y sistemas de navegación suplementarios

Sistemas de navegación integrados (embedded navigation)

- **Globales:** proporcionan acceso a las principales secciones de un sistema.
- **Locales:** proporcionan acceso a los módulos vecinos de una misma sección del sistema.
- **Adicionales:** proporcionan atajos para el acceso a secciones seleccionadas y que no estarían disponibles de forma inmediata con los sistemas globales o locales.
- **Contextuales:** están integrados en el contenido, en el texto de las páginas para proporcionar acceso a información complementaria en el momento de la lectura.
- **De cortesía:** proporcionan acceso a ítems no usados de forma general pero que son proporcionados por comodidad, como por ejemplo formulario de sugerencias, información de contacto

Sistemas de navegación suplementarios (supplemental o remote navigation tools)

- **Mapas:** proporciona en una sola página una visión panorámica de la arquitectura de un sitio Web.
- **Índices:** listados temáticos, cronológicos o de nombres de temas con enlaces a las páginas relevantes.

Estructura de los menús

Los menús se organizan siguiendo el principio de los árboles, esto quiere decir que un menú puede tener menús hijos y menús padres. Inicialmente, al crear una barra de menú, el

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

identificador del objeto nuevo es un identificador de menús válido, además de que la barra de menú queda como la raíz, el nodo principal, o en otras palabras, un menú sin padres.

De ahí se pueden empezar a crear menús hijos como lo puede ser uno que se llame Archivo, otro Edición y otro Ver, que tendrán como padre al manipulador (handler) del menú raíz; estos menús pueden tener menús hijos, así, archivo, descendiendo de MenuBar puede engendrar Abrir, Guardar, Guardar como, Codificación, etc; y siguiendo estas mismas reglas, Codificación, descendiente de Archivo puede engendrar un submenú con los comandos ASCII, Unicode, UTF-8, entre otros.

La susodicha organización pertenece a una capa de abstracción que es tratada en el proceso de programación mediante APIs del sistema operativo.

Tipos de menús

Los tipos de menús más usuales son:

- **Normales.** Son los que tienen el predominio más alto en las aplicaciones.
- **Casillas de verificación.** Al hacer clic sobre ellos, se activa un indicador y su estado cambia a «marcado/desmarcado».
- **Botones de radio o Radio buttons.** Son grupos de botones donde sólo se puede tener activo uno de todos ellos y su indicador acostumbra ser una viñeta.
- **Submenús.** Son los menús que tienen menús hijos, es decir que no se puede hacer clic en él, en vez de eso hay que seleccionar uno de sus «hijos»; habitualmente traen consigo una flecha en la lateral derecha indicando la naturaleza del mismo.
- **Separadores.** Son menús sin nombre ni valor (pero sí un handler). Se muestran como líneas grises opacas entre la lista de comandos.

Características de los menús

Aunque los menús son personalizables, hay características que se pueden apreciar siempre que se ve un menú:

- **Icono.** En el lado izquierdo hay un espacio para almacenar ya sea un indicador del tipo de menú (viñeta para el radiobotón y paloma para la casilla de verificación) o un pequeño gráfico que haya sido implementado.
- **Mnemónico.** Una letra subrayada que indica qué carácter forma junto con la tecla Alt un atajo de teclado que habilita el menú.
- **Teclas de acceso rápido.** Como su mismo nombre lo dice, es una combinación de teclas que activa al menú una vez que ha sido presionada. Tienden a aparecer en el extremo derecho de cada comando de la lista.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

DISEÑO DE INTERFAZ ACCESO DEL SOFTWARE

La accesibilidad es el grado en el que todas las personas pueden utilizar un objeto, visitar un lugar o acceder a un servicio, independientemente de sus capacidades técnicas, cognitivas o físicas. La accesibilidad web se refiere a la capacidad de acceso a la Web y a sus contenidos por todas las personas independientemente de la discapacidad (física, intelectual o técnica) que presenten o de las que se deriven del contexto de uso (tecnológicas o ambientales). En este área se ofrece un conjunto de pautas y recursos que ofrecen información para desarrollar de interfaces web accesibles.

Página | 80

El diseño de interfaz de acceso o interfaz de usuario o ingeniería de la interfaz es el resultado de definir la forma, función, usabilidad, ergonomía, imagen de marca y otros aspectos que afectan a la apariencia externa de las interfaces de usuario en sistemas de todo tipo (computadoras de uso general, sistemas de control, dispositivos de comunicación móviles, software de sistemas, software de aplicaciones, sitios web, etc). El diseño de la interfaz de acceso del software es una disciplina asociada al diseño industrial y se enfoca en maximizar la usabilidad y la experiencia de usuario. El objetivo final del diseño de la interfaz de acceso del software es hacer que la interacción entre el usuario y el sistema del que es interfaz sea tan simple y eficiente como sea posible, en términos de cumplimiento de los objetivos del usuario. Sigue por ello una filosofía de diseño centrado en el usuario.

La incorporación de los contenidos de Accesibilidad Web en MADEJA se basará fundamentalmente en la adaptación de las pautas propuestas por el W3C (World Wide Web Consortium), iniciativa Web Accessibility Initiative (WAI), con objetivo de facilitar el acceso a las personas con discapacidad y a las que poseen dificultades de acceso debido a limitaciones tecnológicas. Para esto se desarrollarán pautas de accesibilidad y se ofrecerán herramientas para la evaluación y reparación de accesibilidad Web.

Las características dinámicas de un sistema se describen en términos de requisitos diálogo que aparecen definidos en los siete principios de diálogo del capítulo 10 del estándar ISO 9241 sobre ergonomía de la interacción persona-sistema.⁷ Este estándar establece una serie de conceptos y elementos básicos de ergonomía que suponen un punto de partida para facilitar el diálogo entre los sistemas y las personas que usan dichos sistemas, con definiciones de alto nivel, aplicaciones ilustrativas y ejemplos de los principios definidos. Los principios aplicables representan los aspectos dinámicos de la interfaz y pueden considerarse, de forma general, como la "sensación" que produce la interfaz.

Los siete principios son los siguientes:

- **Adecuación a la tarea:** el diálogo es adecuado a la tarea cuando asiste al usuario en la compleción eficaz y eficiente de la tarea.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

- **Carácter autodescriptivo:** el diálogo es autodescriptivo cuando cada paso del diálogo es inmediatamente comprensible ya sea mediante la información devuelta por el propio sistema o por una explicación a solicitud del usuario.
- **Conformidad con las expectativas del usuario:** el diálogo es conforme con las expectativas del usuario cuando es consistente y se ajusta a las características del usuario, tales como conocimiento de la tarea, educación, experiencia, y otros convenios comúnmente aceptados.
- **Adecuación al aprendizaje:** el diálogo es adecuado al aprendizaje cuando ofrece soporte y guía para que el usuario aprenda a utilizar el sistema.
- **Controlabilidad:** el diálogo es controlable cuando el usuario es capaz de iniciar y controlar la dirección y ritmo de la interacción hasta el punto en el que la tarea ha sido completada.
- **Tolerancia a errores:** el diálogo es tolerante a errores si, con independencia de que haya errores de la entrada, el resultado pretendido puede ser alcanzado sin acción necesaria por parte del usuario, o con una acción mínima.
- **Personalizable:** el diálogo es personalizable cuando la interfaz de software puede ser modificada para ajustarse a las necesidades de la tarea, preferencias individuales, y habilidades del usuario.

Página | 81

El concepto de usabilidad es definido en el estándar a partir de la eficacia, eficiencia de la interfaz, así como de la satisfacción del usuario:

- "La usabilidad es la medida con la que un producto se puede usar por usuarios determinados para conseguir objetivos específicos con eficacia, eficiencia y satisfacción en un contexto de uso concreto."
- La eficacia o efectividad mide la extensión (exactitud e integridad) con la que se alcanzan globalmente los objetivos de uso del sistema.
- La eficiencia mide los recursos que deben utilizarse para alcanzar los objetivos pretendidos.
- La satisfacción es un factor subjetivo que mide hasta donde el usuario encuentra globalmente al sistema aceptable.

La eficacia, la eficiencia y la satisfacción se pueden considerar como factores de calidad de la usabilidad. La evaluación de estos factores necesita de una descomposición adicional en subfactores y, finalmente, en métricas de usabilidad.

Los siete atributos de presentación son los siguientes:

- **Claridad:** el contenido de la información es presentado de forma rápida y precisa.
- **Discriminabilidad:** la información visualizada puede ser distinguida de forma precisa.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

- **Concisión:** los usuarios no son sobrecargados con información irrelevante.
- **Consistencia:** el diseño es único y conforme a las expectativas del usuario.
- **Detectabilidad:** la atención del usuario es dirigida hacia la información necesaria.
- **Legibilidad:** la información es fácil de leer.
- **Comprensibilidad:** el significado es claramente inteligible, no ambiguo, interpretable y reconocible.

Página | 82

PRUEBA DEMOSTRATIVA DEL SOFTWARE

Pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto, es una actividad más en el proceso de control de calidad.

Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo.

Características y fases de prueba de software

La prueba de software se puede definir como una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas (que cumpla con todos los requisitos del cliente). Seguidamente se realiza un proceso de evaluación en el que los resultados obtenidos se comparan con los resultados esperados para localizar fallos en el software. Estos fallos conducen a un proceso de depuración en el que es necesario identificar la falta asociada con cada fallo y corregir, pudiendo dar lugar a una nueva prueba. Como resultado final se puede obtener una determinada predicción de fiabilidad o un cierto nivel de confianza en el software probado

El objetivo de las pruebas no es asegurar la ausencia de defectos de un software, únicamente puede demostrar que existen defectos.

El objetivo es diseñar y hacer exhaustivamente pruebas que nos permitan encontrar el mayor número de errores haciéndolo con la menor cantidad de tiempo y esfuerzo.

Para encontrar el mayor número de fallos en el sistema será necesario que sean realizadas por un equipo ajeno al que desarrollo el software ya que si el ingeniero que creo el sistema tratara siempre de demostrar que su software funciona y las pruebas correctivas no tendrán mucho éxito

Tareas a realizar para probar tu software

- **Diseño de las pruebas** identificar las distintas técnicas de pruebas que se utilizaran para probar el software

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

- **Generación de los casos de prueba** los casos de prueba son los datos de entrada que se introducirán en el software para probarlo y retornarán resultados con un objetivo en particular
- **Definición de los procedimientos de prueba**, especifica como se va a llevar el proceso de prueba, quien lo va a realizar y cuando
- **Ejecución de prueba** después de haber aplicado los casos de prueba se van a comparar los datos retornados por el programa y se comparan con los resultados
- **Informe de prueba** con el resultado de los casos de prueba se identificaran cuales resultaron satisfactorios en caso de ser diferente a lo esperado, se identificaran los fallos

Página | 83

Técnicas y Herramientas de prueba

Las técnicas de evaluación dinámica o prueba proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas

Las Técnicas se agrupan en:

- **Técnicas de caja blanca o estructurales**, que se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa
- **Técnicas de caja negra o funcionales**, que realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa.

No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.

Pruebas de Caja Blanca o Estructurales

- A este tipo de técnicas se le conoce también como Técnicas de Caja Transparente o de Cristal.
- Este método se centra en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa. Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento.
- El objetivo de la técnica es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa.

Pruebas de Caja Negra o Funcionales

- También conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una “Caja Negra” cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. No

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE

DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

obstante, como el estudio de todas las posibles entradas y salidas de un programa sería impracticable se selecciona un conjunto de ellas sobre las que se realizan las pruebas.

- Para seleccionar el conjunto de entradas y salidas sobre las que trabajar, hay que tener en cuenta que en todo programa existe un conjunto de entradas que causan un comportamiento erróneo en nuestro sistema, y como consecuencia producen una serie de salidas que revelan la presencia de defectos. Entonces, dado que la prueba exhaustiva es imposible, el objetivo final es pues, encontrar una serie de datos de entrada cuya probabilidad de pertenecer al conjunto de entradas que causan dicho comportamiento erróneo sea lo más alto posible.
- Al igual que ocurría con las técnicas de Caja Blanca, para confeccionar los casos de prueba de Caja Negra existen distintos criterios. Algunos de ellos son:
 - Particiones de Equivalencia.
 - Análisis de Valores Límite.
 - Métodos Basados en Grafos.
 - Pruebas de Comparación.
 - Análisis Causa-Efecto.

Página | 84

Proceso de la Prueba de Software

La estrategia que se ha de seguir a la hora de evaluar dinámicamente un sistema software debe permitir comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Más concretamente, los pasos a seguir son:

1. **Pruebas unitarias.** Comienzan con la prueba de cada módulo.
2. **Pruebas de Integración.** A partir del esquema del diseño, los módulos probados se vuelven a probar combinados para probar sus interfaces.
3. **Prueba del Sistema.** El software ensamblado totalmente con cualquier componente hardware que requiere se prueba para comprobar que se cumplen los requisitos funcionales.
4. **Pruebas de Aceptación.** El cliente comprueba que el software funciona según sus expectativas.

PROCESO DE RECUPERACION

Para cada estrategia, el plan de recuperación evalúa y aplica un conjunto de directrices dirigidas al elemento clave en cuestión. Por ejemplo, si la estrategia a ejecutar va dirigida al producto, entonces el conjunto de directrices abarca acciones tales como: la estabilización de los requerimientos, la disminución del conjunto de prestaciones, la eliminación de las partes del producto caracterizadas por una baja calidad, la reducción del número de defectos, entre otras.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

La clave entre el éxito y el fracaso de un proyecto de software radica (en la gran mayoría de los casos), contar con un Departamento de Riesgos encargado de tomar las riendas de un proyecto de software.

La administración efectiva de un proyecto de software depende totalmente de la forma en la que se planea la totalidad del proyecto.

Página | 85

Dentro de las actividades que se deben tomar en cuenta al momento de querer retomar el rumbo correcto de un proyecto de software podemos mencionar:

- Cuanto personal se requiere para contemplar una actividad de recuperación
- Cuanto tiempo tomara realizar esta actividad
- Cuál será el costo total de realizar este tipo de actividades

HERRAMIENTAS O PROGRAMAS PARA TRABAJAR CON UML

Hay muchísimos programas que permiten trabajar con UML, aunque aprender a usarlos requiere tiempo.

Astah community: herramienta sencilla, adecuada para aprender. Se puede descargar una versión gratuita en <http://astah.net/editions/community>. Astah (antes conocido como Jude) también tiene una versión profesional.

Rational Rose: conjunto de herramientas IBM usado por muchas empresas.

Lucidchart: herramienta que permite crear muchos tipos de diagramas, entre ellos UML. Puede probarse visitando:

- <https://www.lucidchart.com/pages/es/ejemplos/diagrama-UML>

Microsoft Visio: herramienta de Microsoft que permite la creación de muchos tipos de diagramas, entre ellos diagramas UML.

Otros: Erwin, Oracle Designer, EasyCASE, Power Designer, etc. son herramientas que incorporan muchas utilidades, entre ellas UML.



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Bibliografía



- 10 de Marzo de 2014

<https://blog.es.logicalis.com/analytics/bid/370962/arquitectura-de-datos-an-lisis-y-dise-o>

-Anónimo- 4 de Mayo de 2012

<http://ingenieriadesistemas-shirley.blogspot.com/2012/05/tecnicas-de-recoleccion-de-datos.html>



- Pérez Hernández Marco Antonio – 29 de Mayo de 2013

<https://www.slideshare.net/MarcoSk81/fases-de-prueba-de-software>



- Yuri Angelica Guerrero Acuña – 30 de Septiembre de 2013

<https://www.monografias.com/trabajos98/de-metodologia-de-la-investigacion/de-metodologia-de-la-investigacion.shtml>

Energía y Consumo

<http://energiayconsumo16in.blogspot.com/2016/03/disenio-de-sistemas-conceptualizacion.html>

FIADATA . Universidad de San Martín de Porres, Lima-Perú - Ing. Juan José Montero Román

<https://www.usmp.edu.pe/publicaciones/boletin/fia/info86/articulos/problemasDesarrolloSoftware.html>



- Roberto MENENDEZ FERNANDEZ – 06 de Agosto de 2008

<https://www.xing.com/communities/posts/preguntas-cerradas-y-preguntas-abiertas-1004870945>



- Anónimo

https://www.ecured.cu/Dise%C3%B1o_de_Interfaces_de_Usuario

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL Nº. 17 – 082

ACUERDO Nº 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO



- Universitat Oberta de Catalunya

<http://multimedia.uoc.edu/blogs/dii/es/que-es-una-interficie/>

Componentes de los diagramas Entidad Relación – 08 de Marzo de 2016

Página | 87

<https://pacc27.wixsite.com/base-de-datos/single-post/2016/03/08/Componentes-de-los-diagramas-Entidad-Relacion>



- Manuel Cillero -

<https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/>



- Lucidchart

<https://www.lucidchart.com/pages/es/simbolos-de-diagramas-entidad-relacion>

-Anónimo-

<http://takeuchi-e.blogspot.com/p/simbologia-de-una-base-de-datos-entidad.html>



- ESPAM MFL - 11 de Julio de 2015

<http://ingsoftware-luiszambrano.blogspot.com/2015/07/diagrama-de-despliegue.html>



- Copyright 2016

https://www.tutorialspoint.com/es/software_engineering/software_user_interface_design.htm

Marco de Desarrollo de la Junta de Andalucía

<http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/interfaz-usuario/accesibilidad>

Santa Stracuzzi. Metodología de la Investigación Cuantitativa, Caracas, editorial Fedupel, 2012.

CAMPUS MATRIZ QUITO: Marieta de Veintimilla Pomasqui – Informes: (Luis Cordero OE-21 Edif. Valdivia y Juan León Mera)

EMAIL: sixtodrlawyer@gmail.com / itsj_japon@hotmail.com

Telf: 02 2356 368 / 2554192



INSTITUTO TECNOLÓGICO SUPERIOR JAPÓN

REGISTRO INSTITUCIONAL N.º. 17 – 082

ACUERDO N.º 175

METODOLOGIA DE DESARROLLO DE SOFTWARE
DOCENTE: TLGO. ESP. CESAR STALIN VALDIVIEZO CASTRO

Introducción a las Bases de Datos: THOMSON PARANINFO, S.A. 2005

Pressman, R. 2010. Ingeniería del Software un enfoque Práctico, 7ma. ed. México:Mc Graw

Martin Fowler. «Forms and controls». GUI architecture. thoughtworks publication. Consultado el 20 de mayo de 2018.

Ann Blandford. «Semi-structured qualitative studies». The Encyclopedia of Human-Computer Interaction, 2nd Ed. Interaction Design Foundation. Consultado el 20 de mayo de 2018.

INGENIERIA DE SOFTWARE. Dr. Pedro Mejia Alvarez. 2009. Obtención de Requerimientos PDF

DISEÑO DE CUESTIONARIOS PARA RECOLECCIÓN DE DATOS, Yadira Corral, Facultad de Ciencias Económicas y Sociales, Venezuela, 30 de Junio de 2010, PDF

Análisis de Sistemas de Información, LUIS ANTONIO DOMINGUEZ COUTIÑO, RED TERCER MILENIO, Derechos Reservados © 2012, por RED TERCER MILENIO S.C. PDF

SISTEMAS DE NAVEGACIÓN CON MENÚ DESPLEGABLES: COMPONENTES Y EDICIÓN EN LÍNEA, Cristófol Rovira y Lluís Codina, Rev. Esp. Doc. Cient., 29, 1, 2006, PDF

Página | 88



INSTITUTO TECNOLÓGICO
SUPERIOR JAPÓN

AMOR AL CONOCIMIENTO

POMASQUI-

c/Marieta Veintimilla E5-471 y Sta. Teresa 4ta transversal

Tlfs: 022356-368 - 0986915506

www.itsjapon.edu.ec