Universidad de las Ciencias Informáticas
Facultad 10

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Título: Análisis y diseño del Habilitador Metodológico para la gestión de la metodología a utilizar en el desarrollo de Software.

Autor

Joannis Bacallao Guzmán

Yanet Salazar Gutiérrez

Tutor

MSc. Maidely Calderón Montero

Ing. Allan Pierra Fuentes

Caracas, Venezuela.

Junio de 2008

AGRADECIMIENTOS

A mis padres

A mi nene (Abel), porque más que mi pareja has sido mi padre, mi familia, mi guía, mi apoyo, mi amor.

A mis hermanas y mi bella sobrina.

A mi nueva familia que adoro y que me adora.

A mis amigos,

Yanet.

A mi familia por el amor y el apoyo que siempre me han dado.

A chily, por formar parte de lo que soy y por aguantar mis malcriadeces, te amo.

Gracias a todo que ha formado parte del transcurso de mi vida y mi desarrollo profesional.

Joannis.

A nuestros tutores y a todos nuestros amigos.

RESUMEN

Con esta investigación se pretende realizar el análisis y diseño de un Habilitador Metodológico que facilite la gestión de las metodologías de desarrollo de software. Para ello se realiza un estudio de los procesos de desarrollo de software así como de las metodologías más utilizadas, el cual tiene como propósito analizar los elementos que componen las diferentes metodologías para brindar una conceptualización de los mismos. Se analizan dos herramientas que ofrecen soluciones análogas a la propuesta en esta tesis.

Se exponen los requisitos funcionales y no funcionales del sistema, se especifican los Casos de Uso y se elaboran los diagramas correspondientes a estos. Además, se construyen los modelos de análisis y los diagramas de clases de diseño Web para cada Caso de Uso, asociados a estos se construyen los diagramas de secuencia. En el documento se presenta además, el modelado de la Base de Datos correspondiente al sistema.

Palabras claves: Metodologías de desarrollo, procesos de desarrollo de software, Habilitador Metodológico.

INDICE

Introducción 1

Capítulo 1: Fundamentación Teórica 5

INTRODUCCIÓN 5

PROCESOS DE DESARROLLO DE SOFTWARE. 5

Modelo Lineal Secuencial (MLS) 8

Modelo de Construcción del prototipo. 10

El modelo de Desarrollo Rápido de Aplicaciones (DRA) 11

Modelo de desarrollo evolutivo. 13

Modelo incremental 13

Modelo en espiral 13

Modelo espiral WINWIN (Victoria-Victoria). 14

Modelo de desarrollo concurrente. 15

Desarrollo basado en componentes. 15

Modelo de métodos formales. 16

Técnicas de cuarta generación. 16

METODOLOGÍAS DE DESARROLLO. 17

Metodologías tradicionales 17

Metodologías ágiles 18

Caracterización de las Metodologías de desarrollo 19

Proceso Unificado de Desarrollo (RUP) 19

Programación Extrema (Extreme Programming XP). 21

SCRUM 23

Dynamic Systems Development Method (DSDM). 25

Adaptive Software Development (ASD). 27

Feature Driven Development (FDD) 28
Lean Development (LD). 30
Crystal Methodologies 31
HERRAMIENTAS ANALIZADAS 32
Eclipse Process Framework Composer (EPF Composer) 32
Rational Method Composer de IBM (RMC): 33
CONCLUSIONES 35
Capítulo 2: Características del Sistema 37
INTRODUCCIÓN 37
DESCRIPCIÓN DEL PROBLEMA 37
MODELO DEL DOMINIO 38
Conceptos del Modelo del Dominio 38
Descripción del modelo del dominio 41
Reglas del negocio 41
ESPECIFICACIÓN DE LOS REQUERIMIENTOS 42
Requerimientos funcionales 42
Requisitos no Funcionales 45
MODELO DEL SISTEMA 48
Actores 48
Diagrama de Casos de Uso del Sistema 49
Descripción de los Casos de Uso del Sistema 50
CONCLUSIONES 99
Capítulo 3: Análisis y diseño del sistema 100
DIAGRAMAS DE CLASES DE ANÁLISIS 100
DIAGRAMAS DE CLASES DE DISEÑO 104

Descripción de las clases de Diseño 105
Diagramas de Secuencia 106
Diagrama Entidad Relación 110
Conclusiones 116
Recomendaciones 117
Referencias bibliográficas 118
Bibliografía 119
Anexo 1. Diagramas de clases de diseño 121
Anexo 2. Diagramas de Secuencia 130
Glosario de términos 150

Introducción

En el año 2000 surge en la República Bolivariana de Venezuela el Centro Nacional de Tecnologías de Información (CNTI), institución adscrita al recién creado Ministerio del Poder Popular para las Telecomunicaciones y la Informática encargada, prioritariamente, de acelerar el proceso de migración de la administración pública a Software Libre (SL en lo adelante), estandarizar la plataforma tecnológica del Estado, articular los esfuerzos que, en materia de capacitación tecnológica, adelantan diferentes organismos gubernamentales, fortalecer la interoperabilidad de los sistemas y consolidar una Industria Nacional de Software.

Como parte de los proyectos que patrocina el CNTI, en el año 2006 se crea la Red Nacional de Integración y Desarrollo de Software Libre (RINDE), es una plataforma de desarrollo integrada para la producción y mantenimiento de aplicaciones informáticas, y con el fin de estimular la participación de comunidades, grupos de usuarios, organismos de la Administración Pública Nacional, instituciones, universidades, sector productivo y demás personas interesadas en el desarrollo y consolidación de la industria del SL en la República Bolivariana de Venezuela.

RINDE esta compuesta por una herramienta de trabajo colaborativo llamada FINDE donde se genera un espacio de trabajo para todos los proyectos a nivel de país y se brindan una serie de servicios como: Administración de Versiones de Ficheros, Administración de documentos, Administración de Tareas y otros. Además de un portal con el objetivo de informar los proyectos que sostiene CNTI, las acciones que se realizan en apoyo a la migración de SL y la vinculación con todas las comunidades de SL en Venezuela, convirtiéndose así en un portal informativo. Después se integra una Wiki para tener un sitio donde publicar la documentación de ayuda y analizar los documentos legales. Esto constituyó la primera versión de RINDE.

Como una segunda versión de RINDE era preciso integrar a este una plataforma que brindara un estándar para el proceso de desarrollo de software y que pudiese ser empleada y adaptada según los requerimientos de cualquier comunidad u organización para el desarrollo de sistemas, en vista a esta necesidad, fue creado el proyecto de SL MeRinde por la Universidad de las Fuerzas Armadas (UNEFA). La idea que estaba concebida era presentar una combinación y adaptación de metodologías ampliamente utilizadas para el desarrollo de software y así brindar un apoyo a las

comunidades de desarrollo de SL en sus proyectos, suministrando las herramientas necesarias para que estos cumplieran con el proceso de desarrollo y la documentación de sus sistemas. MeRinde no materializó la idea expuesta anteriormente, lo que se obtuvo fue un Sitio Web, el cual facilita las prácticas del Proceso Unificado de Rational (RUP según sus siglas en inglés), como metodología de desarrollo para apoyar el proceso de software de los diversos entes que integran la red nacional.

El no contar con una herramienta que permita a los ingenieros de software consultar los aspectos fundamentales referentes a las metodologías de desarrollo y además, configurar estas de acuerdo a las necesidades particulares de cada proyecto, trae aparejado que estos tengan que dedicar parte de su tiempo a hacer búsquedas por la red, mientras que si contaran con una documentación centralizada y organizada sería mucho más factible además, de esta forma se podría crear un estándar para la utilización de las diferentes metodologías, ya sean tradicionales o ágiles, de manera que fueran empleadas en los proyectos teniendo en cuenta los mismos elementos.

Es por ello que el CNTI requiere la creación de un Habilitador Metodológico como herramienta accesible vía Web, que permitiera la consulta y configuración de las diferentes metodologías por parte de la comunidad de SL, de acuerdo a las características de cada proyecto de software y de esta forma avalar que los proyectos cumplen con los estándares establecidos para su desarrollo.

Para darle solución a la situación problémica descrita anteriormente, se plantea como **problema científico**:

¿Cómo facilitar el proceso de adecuación de las metodologías de desarrollo que se utilizan en el desarrollo de software?

Siendo el **objeto de estudio** los modelos y metodologías de desarrollo de software.

La investigación tiene como **campo de acción,** asistentes para la configuración de las metodologías de desarrollo de software.

Para la realización de este trabajo se han formulado diversas **preguntas científicas**, las cuales servirán de guía a lo largo de la investigación:

1. ¿Cuáles son las metodologías de desarrollo de software más utilizadas en la actualidad?

2. ¿Cuáles son los elementos fundamentales de los procesos y metodologías de desarrollo de software?
3. ¿Como diseñar un Habilitador Metodológico que permita la configuración de los elementos de las metodologías de desarrollo de software?

Para darle solución al problema científico planteado anteriormente se trazó el siguiente **objetivo general**:

Realizar el análisis y diseño del sistema "Habilitador Metodológico" para la gestión de la metodología a utilizar en el desarrollo de un Software.

Para darle cumplimiento a las preguntas científicas planteadas anteriormente, se exponen las siguientes **tareas de investigación**:

1. Realizar un estudio del estado del arte sobre los procesos y metodologías para desarrollo de software.
2. Estudiar y analizar las alternativas existentes.
3. Realizar un estudio de la arquitectura de RINDE.
4. Analizar los frameworks existentes para el desarrollo Web.

Los **resultados esperados** para esta investigación son:

1. Relación de los principales elementos o componentes de los modelos de procesos genéricos y de cada una de las metodologías de desarrollo del software más significativas.
2. Análisis y diseño de la herramienta Habilitador Metodológico.

Métodos científicos de investigación

Para el desarrollo de la investigación se utilizarán métodos teóricos y empíricos.

Métodos teóricos:

Análisis y Síntesis: Durante la investigación se realizó un estudio general de los procesos de desarrollo de software y las metodologías para conocer los principales elementos que los

componen, estos elementos son las fases, disciplinas, actividades, roles, artefactos y otros. Los resultados obtenidos unidos al conocimiento empírico nos permitieron llegar a conclusiones maduras sobre el tema.

Análisis histórico-lógico: A través de este método se ha realizado un análisis del surgimiento y evolución de los procesos y metodologías de desarrollo de software.

Métodos empíricos:

Entrevista: Se realizaron tres entrevistas dos de ellas relacionadas con el Proyecto RINDE, específicamente para obtener información sobre la composición de RINDE y su arquitectura, permitiendo recopilar información cualitativa de la investigación, la otra entrevista se realizó con el objetivo de obtener información referente a MeRinde, donde se obtuvo información sobre su surgimiento, la idea que estaba concebida que se desarrollara y lo que resultó realmente.

El presente trabajo está conformado por 3 capítulos.

En el capítulo 1 se hace un análisis del estado del arte y se exponen las ideas acerca de los procesos y metodologías de desarrollo de software más utilizadas. Además, se hace un análisis de herramientas que ofrezcan soluciones análogas a la propuesta realizada.

En el capítulo 2 se describe el sistema, se presentan los requisitos funcionales y no funcionales del software, una descripción de los Casos de Uso y los diagramas más significativos desde el punto de vista de la arquitectura.

En el capítulo 3 se modelan los diagramas de análisis y diseño del Habilitador Metodológico, además se modela la Base de Datos del sistema.

Capítulo 1: Fundamentación Teórica

Introducción

En el presente capítulo se exponen los elementos que caracterizan los procesos de desarrollo de software más significativos y las diferentes metodologías existentes, algunos de estos elementos son: las fases, disciplinas, artefactos que se generan, roles que desempeñan, funcionalidades dentro del ciclo de vida del software, principios y buenas prácticas. Además, se realiza un estudio de herramientas que brindan un entorno para la gestión de metodologías y que por ende son análogas a la solución propuesta, de estas se reflejan sus características fundamentales en cuanto a las funcionalidades que brindan y la arquitectura, también se analiza los aspectos referentes a las prestaciones necesarias para su utilización y en cuanto a las licencias que las regulan.

Procesos de desarrollo de software.

Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software (Jacobson, 1999). Una visión gráfica de lo dicho anteriormente se muestra en la Figura 1.

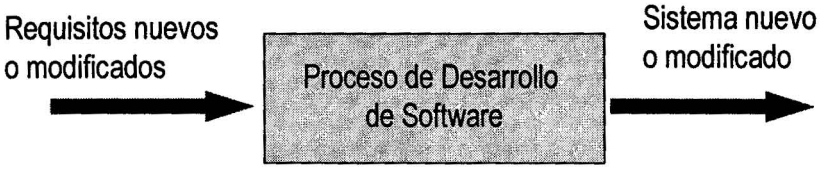


Figura 1: Proceso de desarrollo de software (Jacobson, 1999).

El proceso de desarrollo de software no es único. No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo.

A pesar de la variedad de propuestas de proceso de software, existe un conjunto de actividades fundamentales que se encuentran presentes en todos ellos:

1. Especificación de software: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.

2. Diseño e Implementación: Se diseña y construye el software de acuerdo a la especificación.

3. Validación: El software debe validarse, para asegurar que cumpla con lo que quiere el cliente.

4. Evolución: El software debe evolucionar, para adaptarse a las necesidades del cliente.

Además de estas actividades fundamentales, Pressman menciona un conjunto de actividades protectoras, que se aplican a lo largo de todo el proceso del software. Ellas se señalan a continuación:

- Seguimiento y control de proyecto de software.
- Revisiones técnicas formales.
- Garantía de calidad del software.
- Gestión de configuración del software.
- Preparación y producción de documentos.
- Gestión de reutilización.
- Mediciones.
- Gestión de riesgos.

Otra perspectiva utilizada para determinar los elementos del proceso de desarrollo de software es establecer las relaciones entre elementos que permitan responder **Quién** debe hacer **Qué**, **Cuándo** y **Cómo** debe hacerlo.

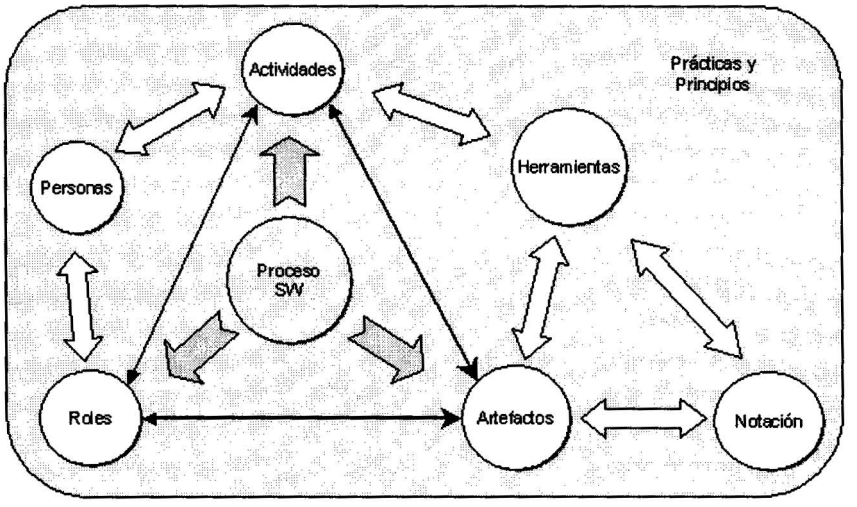


Figura 2: Relación entre elementos del proceso del software[1]

En la figura anterior, Figura 2, se muestran los elementos de un proceso de desarrollo de software y sus relaciones. Así las interrogantes se responden de la siguiente forma:

- **Quién**: Las Personas participantes en el proyecto de desarrollo desempeñando uno o más Roles específicos.
- **Qué**: Un Artefacto es producido por un Rol en una de sus Actividades. Los Artefactos se especifican utilizando Notaciones específicas. Las Herramientas apoyan la elaboración de Artefactos soportando ciertas Notaciones.
- **Cómo** y **Cuándo**: Las Actividades son una serie de pasos que lleva a cabo un Rol durante el proceso de desarrollo. El avance del proyecto está controlado mediante hitos que establecen un determinado estado de terminación de ciertos Artefactos.

Beneficios de un buen proceso de desarrollo de software.

El enfoque que se le debe dar a un proceso de desarrollo, es incrementar la calidad del software que se quiere desarrollar a través de un mayor control sobre el proceso que se realiza. Para obtener el

[1] http://www.dsic.upv.es/asignaturas/facultad/lsi/doc/IntroduccionProcesoSW.doc

producto final con el menor número de fallos según las necesidades del cliente, es fundamental entregar el producto en el tiempo y costo estimados.

Es muy importante saber seleccionar un proceso para el desarrollo del sistema ya que dependiendo del que se elija se pueden obtener beneficios como:

- Aumentar la velocidad de desarrollo.
- Predecir tiempos y costos.
- Mejorar la calidad, el control y el seguimiento del sistema.
- Minimizar gastos y riesgos.
- Mejorar las relaciones con los usuarios.
- Definir roles y perfiles.

Definición de un Modelo de Proceso de desarrollo de software.

Un proceso de desarrollo de software es una vista de las actividades que ocurren durante el desarrollo de software, intenta determinar el orden de las etapas involucradas y los criterios de transición asociadas entre estas etapas (Pressman, 1997). Dicho proceso describe las fases principales de desarrollo de software, ayuda a administrar el progreso del desarrollo y provee un espacio de trabajo para la definición de un detallado proceso de desarrollo de software.

Modelo Lineal Secuencial (MLS)

El Modelo Lineal Secuencial es también denominado como ciclo de vida clásico o modelo de cascada. Consiste en la ejecución secuencial de una serie de fases que se suceden, lo que da nombre al modelo. Cada fase genera documentación para la siguiente, esta documentación debe ser aprobada. Para que una fase comience, la anterior tiene que haber terminado y requiere que se disponga de unos requisitos completos y precisos al principio del desarrollo.

El MLS propone las siguientes fases (Pressman, 1997):

Análisis de los requerimientos del software: es la fase en la cual se capturan los requisitos que debe cumplir el software. En esta etapa es fundamental la presencia del cliente que documenta y revisa dichos requisitos.

Diseño: es una etapa dirigida hacia la estructura de datos, la arquitectura del software, las representaciones de la interfaz y el detalle procedimental (algoritmo). En forma general se hace un esbozo de lo solicitado y se documenta haciéndose parte del software.

Generación del código: es la etapa en la cual se traduce el diseño para que sea comprensible por la máquina. Depende de lo detallado en el diseño.

Pruebas: esta etapa se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en la detección de errores.

Mantenimiento: debido a que el programa puede tener errores, puede no ser del completo agrado del cliente o puede necesitar, eventualmente acoplarse a su entorno; se realizan algunos cambios, no se rehace el sistema.

El MLS es el paradigma de desarrollo de software más antiguo que existe, sin embargo presenta errores reales que serán mencionados a continuación (Pressman, 1997):

- Los proyectos raramente siguen el paradigma secuencial que propone el proyecto.
- A menudo es difícil que el cliente exponga exactamente todos los requisitos además, de que si el desarrollo del software es duradero puede que pase algún tiempo antes de la implementación en el cual los intereses del cliente puedan variar.
- El cliente recibe el primer producto al finalizar el ciclo de desarrollo cuando ya se han empleado todos los recursos disponibles.
- Los responsables del desarrollo de software siempre se retrasan innecesariamente.
- Los errores de análisis y diseño son costosos de eliminar, y se propagan a las fases siguientes.

Aunque presenta los inconvenientes mencionados anteriormente, al emplear este modelo podemos crear un marco de trabajo claro para todos los implicados en el proyecto, definiendo las etapas y las actividades a realizarse en cada una de ellas.

Para la aplicación adecuada de este modelo el proyecto debe disponer de unos requisitos completos y consistentes al inicio del desarrollo, además debe aplicarse en proyectos pequeños, en los que el período de congelación de los requisitos es corto, o un proyecto con unos requisitos bastante estables.

Modelo de Construcción del prototipo.

Un cliente, en ocasiones, define los objetivos generales para la construcción de un software, pero no identifica los requisitos de entrada o salida. Por lo que un paradigma de construcción de prototipo puede ofrecer el mejor enfoque.

El modelo de construcción de prototipos (Fig. 3) comienza con la recolección de los requisitos del software que se llevará a cabo. El desarrollador y el cliente encuentran y definen diez objetivos globales para el software, identifican los requisitos conocidos y las tareas del esquema donde es obligatoria más definición. Entonces aparece un diseño rápido. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente (por ejemplo: enfoques de entrada y formatos de salida). El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

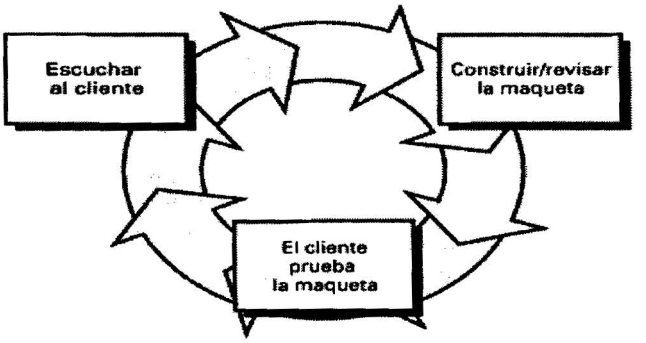


Figura 3. Paradigma de construcción de prototipos (Pressman, 1997).

El modelo de Desarrollo Rápido de Aplicaciones (DRA)

Es un modelo de proceso de desarrollo lineal secuencial que enfatiza un ciclo de desarrollo muy corto, el DRA (RAD por sus siglas en inglés) es una adaptación del modelo lineal secuencial pero a una alta velocidad, en el que se logra un desarrollo rápido utilizando una construcción basada en componentes.

Si los requisitos son bien comprendidos desde el inicio y se limita el ámbito de desarrollo del proyecto, el RAD permite al equipo de desarrollo crear un sistema completamente funcional dentro de períodos cortos, aproximadamente de 60 a 90 días.

El enfoque RAD propone las siguientes fases (Pressman, 1997):

Modelado de gestión: El flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión?, ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la procesa?

Modelado de datos: El flujo de información definido como parte de la fase de modelado de gestión se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen características (atributos) de cada uno de los objetos y las relaciones entre estos objetos.

Modelado de proceso: Los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos.

Pruebas y entrega: Como el modelo RAD enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfaces a fondo.

Una aplicación es candidata de un enfoque RAD si puede modularse de forma que puedan completarse cada una de sus funciones principales en menos de tres meses, cada una de estas funciones pueden ser afrontadas por un equipo RAD separado y luego ser integradas en un solo conjunto.

El modelo RAD al igual que otros enfoques presenta una serie de desventajas:

- Para proyectos grandes aunque por escalas, el RAD requiere recursos humanos suficientes como para crear el número correcto de equipos RAD.
- RAD requiere clientes y desarrolladores comprometido en las rápidas actividades necesarias para completar un sistema en un marco de tiempo abreviado, esto es necesario para que los proyectos RAD no fracasen.
- No todos los tipos de aplicaciones son apropiados para RAD. Si un sistema no se puede modularizar correctamente, la composición de los componentes para RAD será problemático. Si está en juego el alto rendimiento, y se va a conseguir el rendimiento convirtiendo interfaces en componentes de sistemas, el enfoque RAD puede que no funcione.
- RAD no es adecuado cuando los riesgos técnicos son altos, esto ocurre cuando una nueva aplicación hace uso de tecnologías nuevas, o cuando el software nuevo requiere de un alto grado de interoperatividad con programas de computadora ya existentes.

Modelo de desarrollo evolutivo.

El software, al igual que todos los sistemas complejos, evoluciona con el tiempo. Es por ello que los ingenieros del software necesitan un proceso de desarrollo que se ha diseñado explícitamente para un producto que evolucione con el tiempo, tal es el caso del modelo de desarrollo evolutivo. La idea detrás de este modelo es una implantación del sistema inicial, refinarla en N versiones hasta que se desarrolle el sistema adecuado.

Existen cuatro tipos de modelos dentro del proceso de desarrollo evolutivo. Se caracterizan por ser iterativos y por la forma en que permiten a los ingenieros de software desarrollar versiones cada vez mas completas:

Modelo incremental

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

Cuando se utiliza este tipo de modelo, el primer incremento a menudo es un producto esencial. Es decir, se afrontan requisitos básicos pero muchas funciones suplementarias (algunas conocidas, otras no) quedan sin extraer (Pressman, 1997).

Modelo en espiral

Conjuga la iteratividad del modelo de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial.

Este modelo se divide en un número de actividades, también llamadas regiones de tareas (Pressman, 1997):

- Comunicación con el cliente: Son las tareas que requieren establecer comunicación entre el desarrollador y el cliente.
- Planificación: Esta tarea es para definir recursos, el tiempo y cualquier otra información relacionada con el proyecto.
- Análisis de riesgos: Se utiliza para evaluar riesgos técnicos y de gestión.
- Ingeniería: Esta tarea es requerida para construir una o más representaciones de la aplicación.
- Construcción y acción: Es utilizada para construir, probar, instalar y proporcionar soporte al usuario.
- Evaluación del cliente: Se obtiene la reacción del cliente según la evaluación de las representaciones del software durante la etapa de ingeniería.

En este proceso el equipo de ingeniería de software gira alrededor de una espiral, comenzando siempre por el centro. El primer circuito de esta espiral puede producir el desarrollo de una especificación de productos y en los siguientes pasos versiones más sofisticadas del software que se está desarrollando.

Modelo espiral WINWIN (Victoria-Victoria).

La idea esencial de este modelo es que el cliente gane obteniendo el producto o sistema que satisface la mayor parte de sus necesidades y el desarrollador gane trabajando para conseguir presupuestos y lograr una fecha de entrega realista.

Define un conjunto de actividades de negociación al principio de cada paso alrededor de la espiral (Pressman, 1997):

- Identificación de sistemas o subsistemas claves de los directivos.
- Determinación de la condiciones de victoria de los directivos.
- Negociación de las condiciones de victoria de los directivos para reunirlas en un conjunto de condiciones victoria-victoria para todos los afectados incluyendo el equipo de proyecto de software.

Introduce además tres hitos en el proceso llamados puntos de fijación (Pressman, 1997):

- Objetivos del ciclo de vida (OCV), define un conjunto de objetivos para cada actividad principal de ingeniería de software.
- Arquitectura del ciclo de vida (ACV) establece los objetivos que se deben conocer mientras se define la arquitectura del software y el sistema.
- Capacidad operativa inicial (COI): represente un conjunto de objetivos asociados a la preparación del software para la instalación/distribución.

Modelo de desarrollo concurrente.

Está dirigido según las necesidades del usuario, las decisiones de la gestión y los resultados de las revisiones. Se puede representar en forma de esquemas como una serie de actividades técnicas importantes, tareas y otros estados asociados a ellas. Por ejemplo, la actividad de ingeniería definida para el modelo en espiral se lleva a cabo invocando las siguientes tareas: Modelado o construcción de prototipos y/o análisis, especificación de requisitos y diseño.

Este modelo define una serie de acontecimientos que provocan transiciones de estado a estado para cada una de las actividades de software que se realizan.

Desarrollo basado en componentes.

El modelo de desarrollo basado en componentes incorpora muchas características del modelo en espiral. Es evolutivo y exige además, ser iterativo para la creación del software. Sin embargo, este modelo configura aplicaciones desde componentes preparados de software.

La ingeniería en este caso comienza con la identificación de clases candidatas, esto se lleva a cabo examinando los datos que se van a manejar por parte de la aplicación y el algoritmo que se va a aplicar para conseguir el tratamiento. Los datos y los algoritmos correspondientes se empaquetan en una clase.

Las clases creadas en los proyectos de ingeniería de software anteriores, se almacenan en una biblioteca de clases o diccionario de datos. Una vez identificadas las clases candidatas, la biblioteca de datos se examina para ver si estas clases ya existen, en caso de que así fuera se extraen de la biblioteca y se vuelven a utilizar. Si una clase no pertenece a la biblioteca se aplican los métodos orientados a objetos (Pressman, 1997).

El modelo de desarrollo basado en componentes conduce a la reutilización, lo cual proporciona grandes beneficios a los ingenieros de software.

Modelo de métodos formales.

Los métodos formales permiten que un ingeniero de software realice una serie de actividades que conducen a la especificación matemática del software de computadora.

Cuando se utilizan métodos formales durante el desarrollo de un software, proporcionan un mecanismo para eliminar muchos de los problemas que son difíciles de superar con paradigmas de la ingeniería de software. La ambigüedad y lo incompleto, se descubren y se corrigen más fácilmente mediante la aplicación del análisis matemático (Pressman, 1997).

Cuando se utilizan métodos formales durante el diseño, sirven como base para la verificación de programas y permiten que el ingeniero del software descubra y corrija errores que no se pudieron detectar de otra manera.

Técnicas de cuarta generación.

El término de Técnicas de Cuarta Generación (T4G) abarca muchas herramientas que facilitan a los ingenieros de software la especificación de algunas características del software a alto nivel, dichas herramientas generan el código automáticamente basándose en las especificaciones de los desarrolladores.

El enfoque T4G para la ingeniería del software se orienta hacia la posibilidad de especificar el software usando formas de lenguaje especializado o notaciones gráficas que describa el problema que hay que resolver en términos que el cliente entienda (Pressman, 1997).

T4G al igual que otros enfoques comienza con la captura de requisitos, idealmente el cliente describe los requisitos que después son traducidos directamente a un prototipo operativo, esto en la práctica es imposible. El cliente puede que no este seguro de lo que necesita o puede no saber describir los requisitos en términos en que puede aceptar una herramienta T4G. Es por eso que la comunicación entre el cliente y los desarrolladores se hace imprescindible.

El modelo T4G presenta ventajas e inconvenientes, como ventaja podemos decir que el uso de T4G es un enfoque viable para muchas de las áreas de aplicación, junto con las diferentes herramientas CASE y los generadores de código ofrecen una solución fiable a muchos problemas del software además, se reduce el tiempo de desarrollo del software y se obtiene una mejora significativa en la productividad. Por otro lado, entre sus inconvenientes, según sus detractores, se encuentran que no es más fácil de usar que los lenguajes de programación, el código fuente producido por las herramientas utilizadas es ineficiente y el mantenimiento de grandes sistemas desarrollados con T4G es cuestionable.

Metodologías de desarrollo.

En los años 60 aparecen como solución a la problemática de que en los inicios los programadores desarrollaban software de una manera caótica, un grupo de metodologías que tenían como objetivos establecer un proceso de desarrollo de software disciplinado donde el trabajo fuese más eficiente y donde el equipo estuviera mejor preparado para los posibles cambios que provenían de diversas fuentes, tanto de los clientes, de los desarrolladores, de la tecnología o de los requerimientos.

Metodologías tradicionales

Estas metodologías no ofrecían una solución a la medida para muchos software a desarrollar pues para su aplicación, era necesario contar con grandes presupuestos ya que los costes eran muy

elevados, además del rigor que exigían para su utilización en cuanto a planificación, documentación detallada como entrada y salida de cada fase y la existencia de procesos bien definidos. Requieren además sobre todas las cosas la existencia de procesos bien definidos, una estricta planificación antes del comienzo del proceso y además una exhaustiva documentación y contratos.

Metodologías ágiles

En 1990 surge un grupo de metodologías con características diferentes a las llamadas tradicionales o pesadas, como solución a los problemas que presentaban las anteriores tomando de estas lo positivo, llamadas metodologías ágiles. Este término surge por primera vez en una reunión realizada con la participación de 17 especialistas en materia de desarrollo de software en marzo de 1991 donde se redacta el Manifiesto Ágil[2] en el que quedan descritos los principios en los que se basan estas metodologías, entre ellas se destacan (Canós, Letelier, and Panadés 2004):

- Extreme Programing (XP)
- Adaptive Software Development (ASD)
- Feature Driven Development (FDD)
- SCRUM
- Dynamic Systems Development Method (DSDM)
- Lean Development (LD)
- Crystal Methodologies

Tomando como punto de partida el Manifiesto Ágil se enuncian los valores que resume.

- Valorar más a los individuos y su interacción que a los procesos y las herramientas.
- Valorar más el software que funciona que la documentación exhaustiva.
- Valorar más la colaboración con el cliente que la negociación contractual.
- Valorar más la respuesta al cambio que el seguimiento de un plan.

---

[2] www.agilalliance.com

Caracterización de las Metodologías de desarrollo

Después de esta breve explicación sobre el surgimiento de estos dos grandes grupos de metodologías, hagamos una pausa para entender mejor cada enfoque, para ello ofrecemos una definición de cada una de las metodologías mencionadas.

Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado de Desarrollo o Rational Unified Process (RUP) por sus siglas en inglés, divide en 4 fases el desarrollo del software: (Booch, Rumbaugh, and Jacobson 1999)

- Inicio: El objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración: En esta etapa se obtiene la arquitectura óptima del sistema.
- Construcción: En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- Transmisión: Durante el desarrollo de esta fase o etapa se obtiene el release[3] del proyecto.

Cada una de estas etapas es desarrollada por iteraciones. Durante la etapa de inicio se desarrolla una iteración, en la de elaboración se realizan dos, en la de construcción tres y durante la etapa de transición dos. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes

Esta metodología se sustenta en seis disciplinas ingenieriles y tres de soporte las cuales se relacionan a continuación:

Disciplinas Ingenieriles.

- Modelado de Negocio: Durante esta etapa se lleva a cabo el análisis de las necesidades del negocio y se definen las posibles actividades a automatizar.

[3] Lanzamiento o liberación de una versión de un programa de software

- Requerimientos: Se especifican los requisitos del sistema y los describen los procesos.
- Análisis y Diseño: Se define la arquitectura de software.
- Implementación: Se crea el software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Se comprueba que se cumple el comportamiento requerido por el usuario y que todo los solicitado esta presente.
- Despliegue: Se implanta el sistema solicitado, se crean las condiciones para que esta implantación sea correcta.

Disciplina de Soporte.

- Configuración y administración del cambio: Se guardan todas las versiones del proyecto.
- Administración del proyecto: Se administran los horarios y recursos.
- Ambiente: Se gestiona el ambiente de desarrollo.

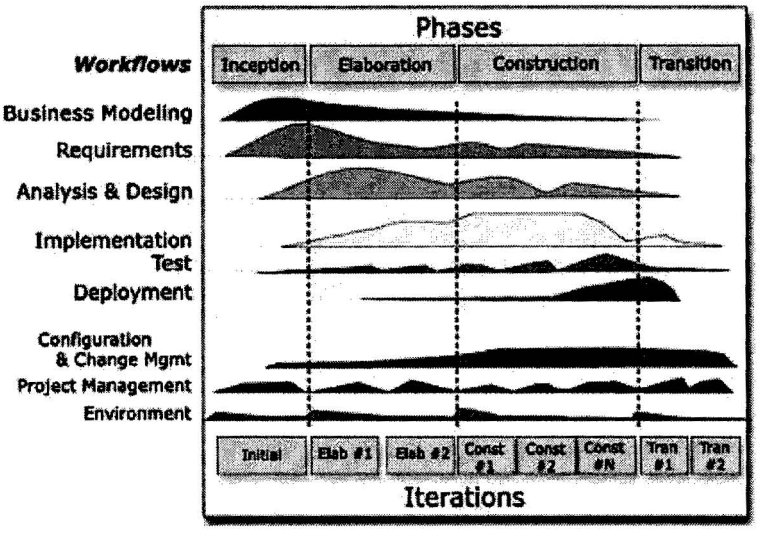


Figura 5: Fases e Iteraciones de la Metodología RUP (Reynoso, 2004)

Los elementos de RUP son:

- Actividades: Son los procesos que se llegan a determinar en cada iteración.

- Trabajadores: Las personas o entes involucrados en cada proceso.
- Artefactos: Puede ser un documento, un modelo, o un elemento de modelo.

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

Programación Extrema (Extreme Programming XP).

Fue desarrollada por Kent Beck en 1990, propone fases, roles, y buenas prácticas para guiar el proceso de desarrollo de software.

Las fases comprendidas para esta metodología son (Fernández Escribano 2002):

Planificación: Durante esta fase se mantienen en comunicación continua el cliente y los desarrolladores, en dicha comunicación ambas partes desarrollan roles importantes, el cliente define la prioridad de las funcionalidades a desarrollar, lo que debe estar incluido en cada versión y la fecha de entrega de las mismas. Por otra parte los desarrolladores son los responsables de establecer el tiempo necesario para implementar lo que quiere el cliente, de advertir de las consecuencias que traerían algunas decisiones y de establecer la política de trabajo en el equipo de desarrollo.

En esta fase se crean las historias de usuario, un plan de entrega, se controla la velocidad del proyecto, se divide el proyecto en iteraciones y al comienzo de cada iteración se traza un plan, además se rota al personal y cada día se convoca a una reunión de seguimiento.

Diseño: Para esta etapa se establecen algunas premisas (Marcelo Hernán, 2004):

- Simplicidad: Se aconseja que el diseño sea lo más sencillo posible, si se está en presencia de una implementación muy compleja, se debe dividir y así hacer más fácil el trabajo si se presenta cualquier modificación.
- Elegir una metáfora para el sistema: Esto nos permite mantener una coherencia de nombres para que todos comprendan cada objeto con que está relacionado.

- Usar tarjetas CRC: Permiten que el equipo completo contribuya en las tareas del diseño, representa un objeto. El nombre de la clase se escribe en la parte superior de la tarjeta y en la izquierda sus responsabilidades y las clases que se implican, también se pondrá el requerimiento correspondiente.
- No se añadirá funcionalidades en las primeras etapas: Esta premisa nos induce a no programar más de lo que está planificado en cada momento.
- Reutilizar cuando sea posible: Nos permite eliminar redundancia y por tanto código inútil, esto, dentro del ciclo de vida del software incrementa la calidad, implica mantener el código limpio, lo cual costará mucho trabajo al inicio pero resultará una actividad fundamental a la hora de realizar los futuros diseños.

Desarrollo: Esta etapa debe identificarse por las siguientes características (Marcelo Hernán 2004):

- El cliente debe estar disponible: Es necesario que el cliente siempre este disponible formando parte del equipo de desarrollo, este participará en las decisiones en cuanto a la prioridad de las historias de usuario, en el plan de entregas, en el diseño de las pruebas y en otros aspectos relacionados con el sistema a realizar.
- Se utilizará estándares de codificación: Facilita el entendimiento del código y la modificación por parte de cualquier miembro del equipo de desarrollo, además es una base para que se cumpla el principio de que el código debe ser común para todos.
- Se desarrollará la unidad de prueba primero: Ayuda al programador a tener una visión acerca de comportamiento del programa.
- Programación en parejas: Se formarán equipos de dos personas con conocimientos similares y así la calidad del software desarrollado será mayor y no se afectará el cronograma de entrega.
- Habrá una pareja encargada de la integración del código: Solo una pareja de programadores será responsable de realizar las pruebas para determinar si existen errores en la integración de los módulos que han sido desarrollados y no han sido probados.

- Integrar frecuentemente: Los desarrolladores actualizarán sus módulos con las últimas versiones realizadas, la integración propuesta solo se realizará después que todas las pruebas correspondientes tengan éxito.
- El código será común para todos: Esto permite que cualquier miembro del equipo de desarrollo contribuya al desarrollo de las funcionalidades del sistema y que pueda eliminar algún error en el código.
- Dejar las optimizaciones para el final: Solo al final se pensará en optimizar el código.
- Se trabajarán solamente 40 horas semanales: Propone que el equipo solo trabajará 40 horas semanales y así no absorber la motivación del equipo y evitar a la vez que haga un mal trabajo.

Prueba: El desarrollo está orientado por las pruebas. Los clientes ayudan a escribir las pruebas funcionales antes que se escriba el código. El propósito del código real no es cumplir un requerimiento, sino pasar las pruebas. Las pruebas y el código son escritas por el mismo programador. Hay dos clases de prueba: la prueba unitaria, que verifica una sola clase, o un pequeño conjunto de clases; la prueba de aceptación verifica todo el sistema, o una gran parte (Taber and Fowler 2001).

SCRUM

Esta metodología fue desarrollada en 1995 por Ken Schwaber y Mike Beedle, es una forma de gestionar proyectos de software. Está diseñada para proyectos con un rápido cambio en los requerimientos del sistema, presenta dos características fundamentales, una de ellas es que el desarrollo se realiza mediante iteraciones llamadas sprints que tienen aproximadamente 30 días y la otra es que se realizan reuniones a lo largo del proyecto y especialmente las que se realizan a diario durante 15 minutos.

Para la aplicación de SCRUM están definidas una serie de reglas básicas (Canós, Letelier, and Panadés 2004) :

- Una vez que se pasan las tareas más prioritarias del "Product Backlog" al "Sprint Backlog", estas no se pueden cambiar, esto quiere decir, que el trabajo de un mes queda fijado. Esta es la regla más importante de todas.
- Al final del mes se tiene que tener un ejecutable con las funcionalidades del "Sprint Backlog". A esto se le denomina "Sprint".
- Todo el mundo puede añadir funcionalidades al "Product Backlog", pero sólo una persona puede ordenarlo. A esta persona se le denomina "Product Owner". Es el responsable del producto final.
- Cada día se hace una reunión de menos de 15 minutos, en la que se reúne todo el equipo.
- Al final del mes, es decir, al final del Sprint, se presenta el producto y se toma del "Product Backlog" ordenado, las funcionalidades para cubrir en el siguiente mes.

En el proceso de desarrollo de SCRUM se identifican varios roles:

Scrum Master: Es un rol fundamentalmente administrativo, es el encargado de revisar las tareas diarias, evaluar los progresos realizados y comparar los mismos con los progresos estimados, aquellos que se previeron durante la planificación del Sprint. También es el encargado de definir con el cliente la persona que llevará el rol de "Product Owner" y, una vez definido éste, se encargará de generar y revisar el "Product Backlog". Durante los Sprints el Scrum Master es el encargado de eliminar todos los posibles inconvenientes que puedan surgir y que supongan un riesgo para alcanzar la meta fijada, por tanto deberá tomar decisiones importantes de cuyo acierto dependerá en gran medida el éxito o fracaso del Sprint.

Product Owner: Es responsable, junto con el Scrum Master, de generar lo que se conoce como el Product Backlog, el cual no es más que una lista evolutiva y casi siempre cambiante de requisitos del producto a desarrollar. El Product Owner debe estar siempre disponible para aclarar cualquier duda que pueda surgir durante el desarrollo, y debe encargarse de mantener la comunicación con el cliente de lo que se está desarrollando y proporcionar el feedback necesario para continuar con el desarrollo. La figura del Product Owner es fundamental en Scrum, ya que involucra al cliente en el propio proceso de desarrollo, lo cual evita en gran medida esas situaciones en las que el cliente dice aquella frase de " Pero nosotros no habíamos pedido esto".

Scrum Team: Consiste en el grupo de personas que llevarán a cabo el trabajo durante los Sprints. Generalmente suele estar compuesto por Ingenieros de Software y por Diseñadores Gráficos, los cuales se encargarán de realizar todas y cada una de las tareas de las que esté compuesto el Product Backlog, ni más ni menos. De ahí reside la importancia de que el Product Backlog esté correctamente detallado por parte del Product Owner y correctamente revisado por el Scrum Master. Un Product Backlog mal detallado, o con inconsistencias en su diseño lleva inevitablemente a un Sprint catastrófico.

Además se desarrollan diversos artefactos:

Product Backlog: Es una lista de funcionalidades de la aplicación ordenadas de mayor a menor importancia, en la cual no es necesario que se reflejen todas las funcionalidades en un inicio.

Sprint Backlog: Es una lista donde aparecen las tareas que serán realizadas en el mes próximo, son tomadas del product Backlog, se toman las primeras funcionalidades y se descomponen en tareas.

Incremento: Parte del producto desarrollada en un sprint, en condiciones de ser usada (pruebas, codificación limpia y documentada).

La dimensión del equipo total de Scrum no debería ser superior a diez ingenieros. Si hay más, lo más recomendable es formar varios equipos. No hay una técnica oficial para coordinar equipos múltiples, pero se han documentado experiencias de hasta 800 miembros, divididos en pequeños equipos Scrum, definiendo un equipo central que se encarga de la coordinación, las pruebas y la rotación de los miembros.

Dynamic Systems Development Method (DSDM).

Es una metodología desarrollada por el Consorcio DSDM en 1994, basándose en las bases del proceso de desarrollo RAD, el objetivo que se perseguía era lograr un modelo de desarrollo independiente de herramientas y que todos lo conocieran. DSDM proporciona un framework completo de controles para RAD y lineamientos para su uso (Marcelo Hernán 2004).

Para la utilización de DSDM se definen varios principios (Marcelo Hernán, 2004):

- Participación activa del usuario.
- El equipo toma decisiones.
- Frecuentes entregas del producto.
- Ajustarse a los objetivos del negocio.
- Desarrollo iterativo e incremental.
- Cambios reversibles.
- Especificar requerimientos globales.
- Pruebas integradas durante todo el ciclo de vida.
- La cooperación entre el equipo, usuarios y stakeholders es esencial.

DSDM define roles. Los más importantes son (Marcelo Hernán, 2004):

Coordinador técnico: Define la arquitectura del sistema y es responsable por la calidad técnica del proyecto, el control técnico y la configuración del sistema.

Usuario embajador: Proporciona al proyecto conocimiento de la comunidad de usuarios y disemina información sobre el progreso del sistema hacia otros usuarios.

Visionario: Es un usuario participante que tiene la percepción más exacta de los objetivos del sistema y el proyecto. Asegura que los requerimientos esenciales se cumplan y que el proyecto vaya en la dirección adecuada.

Patrocinador Ejecutivo: Es la persona de la organización que tiene autoridad y responsabilidad financiera, y es quien tiene la última palabra en las decisiones importantes.

Facilitador: Es responsable de administrar el progreso del taller, el motor de la preparación y la comunicación.

Escriba: Registra los requerimientos, acuerdos y decisiones alcanzadas en las reuniones, talleres y sesiones de prototipado.

El ciclo de vida de esta metodología se inicia con un estudio de la viabilidad del negocio, donde se calculan los costes, se comprueba que sea realmente viable y que DSDM sea el enfoque correcto, después le sigue el estudio de negocio donde se modela todo el proceso y existe una fuerte colaboración entre el equipo de desarrollo y el cliente. La siguiente etapa llega con una iteración del modelo funcional en el cual se hace un prototipo mínimo de lo que se va a desarrollar y se capturan los requisitos no funcionales, luego de realiza la iteración de diseño y construcción, aquí se refinan los requisitos del sistema y se desarrolla. Para terminar el ciclo de vida se realiza la implementación, llevándose a cabo la aprobación por el usuario y el entrenamiento de estos.

Para cada una de estas etapas se definen un conjunto de elementos o artefactos, lo cuales son:

Estudio de Viabilidad: Informe de viabilidad, Plan de desarrollo.

Estudio del Negocio: Definición del área de negocio, Definición de la Arquitectura del sistema, Prototipo del plan.

Iteración del Modelo Funcional: Modelo funcional, Prioridad de los requisitos, Revisión del prototipo funcional, Requisitos no funcionales, Análisis de los riesgos del desarrollo.

Iteración de Diseño y Construcción: Pruebas del sistema

Implementación: Manual de usuario, Reporte de la revisión del proyecto.

La rapidez de esta metodología se sustenta en la selección de las tareas o funcionalidades más prioritarias del negocio, esto se realiza a través del mecanismo llamado TimeBox, donde se registra una fecha de finalización y un conjunto de requerimientos a satisfacer indicando la prioridad de cada uno.

Adaptive Software Development (ASD).

Esta metodología fue diseñada por Jim Highsmith, surge con la intención de ofrecer una alternativa a la idea de que la optimización es la única solución para problemas de complejidad creciente, idea

planteada por CMM nivel 5. La particularidad de ASD radica en que no proporciona un proceso para el desarrollo sino que brinda los métodos para crear una cultura adaptativa para reconocer que el cambio y la incertidumbre son elementos del estado natural (Highsmith 1997).

Los objetivos de esta metodología son:

- Prestar soporte a una cultura adaptativa para que se espere cambio e incertidumbre y no se tenga una falsa expectativa de orden.
- Introducir marcos de referencia para orientar el proceso iterativo de gestión del cambio.
- Establecer la colaboración y la interacción de la gente en tres niveles: interpersonal, cultural y estructural.
- Agregar rigor y disciplina a una estrategia RAD, haciéndola escalable a la complejidad de los emprendimientos de la vida real.

El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

ASD se ocupa más de la calidad que de los procesos usados para producir un resultado. En los ciclos adaptativos de la fase de Colaboración, el planeamiento es parte del proceso iterativo, y las definiciones de los componentes se refinan continuamente. La base para los ciclos posteriores (el bucle de Aprendizaje) se obtiene a través de repetidas revisiones de calidad con presencia del cliente como experto.

Feature Driven Development (FDD)

FDD (por sus siglas en inglés) fue desarrollada por Jeff De Luca y Peter Coad alrededor de 1998. A igual que las otras metodologías ágiles, se enfoca en iteraciones cortas que entregan una funcionalidad tangible .

Esta metodología se centra en definir características que representan las funcionalidades que el sistema debe tener, las cuales tienen un alcance relativamente corto como para poder ser implementadas en dos semanas aproximadamente.

Una de las ventajas que posee FDD al utilizar las características del software está en formar un vocabulario común con vistas a obtener un diálogo fluido con los clientes, desarrollando de tal forma un modelo común del negocio.

El ciclo de vida propuesto por FDD contiene cinco procesos: (Marcelo Hernán, 2004).

Desarrollar un Modelo Global: Durante este proceso se intenta lograr un entendimiento general del negocio, se realiza una primera aproximación de las características del sistema, además de definir otras cuestiones como los requisitos no funcionales, para todo esto se definen las clases esenciales y las responsabilidades de las mismas; se plasman los objetivos del proyecto y como el mismo ayuda al negocio; un documento con los requerimientos no funcionales detectados; por último, el documento en el que aparecen las opciones de modelado surgidas durante esta actividad.

Construir una lista de características: Tiene como entrada las características iniciales que se detectaron en el proceso anterior para refinarlas, una vez refinadas, se agrupan para definir la jerarquía de las mismas y organizar el proceso de desarrollo, esta metodología propone una clasificación para las funcionalidades, estas son: A (debe tener), B (sería útil tener), C (agregar si es posible), o D (futuro).

Planificar por características: Este proceso toma como entrada la lista con las características priorizadas de la actividad anterior, durante esta se define el tiempo para el desarrollo de cada característica y los hitos, además se asignan responsabilidades. Participan el líder de proyecto, el líder de desarrollo y el programador jefe.

Diseñar por características: Este al igual que el de construir están relacionados con la parte productiva del proceso en general, es entonces donde los programadores definen los atributos y clases entre otros elementos para realizar las funcionalidades identificadas

Construir por características: Es aquí donde se comienza a desarrollar las clases definidas anteriormente, se construyen las pruebas unitarias y se revisa el código.

Lean Development (LD).

Esta metodología es creada por Bob Charette's, fue utilizada en numerosos proyectos de telecomunicaciones en Europa. En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades de mejoras para la productividad.

Su principal característica es introducir un mecanismo para implementar dichos cambios, es el método ágil menos divulgado entre los reconocidamente importantes.

Dado que LD es más una filosofía de administración que un proceso de desarrollo no hay mucho que decir del tamaño que debe poseer su equipo, la duración de las iteraciones que se realicen, los roles o sus etapas.

LD se inspira en doce valores centrados en estrategias de gestión:

- Satisfacer al cliente es la máxima prioridad.
- Proporcionar siempre el mejor valor por la inversión.
- El éxito depende de la activa participación del cliente.
- Cada proyecto LD es un esfuerzo de equipo.
- Todo se puede cambiar.
- Soluciones de dominio, no puntos.
- Completar, no construir.
- Una solución al 80% hoy, en vez de una al 100% mañana.
- La necesidad determina la tecnología.
- El crecimiento del producto es el incremento de sus prestaciones, no de su tamaño.
- Nunca empujes LD más allá de sus límites.

LD ha evolucionado pasando a ser Lean Software Development (LSD) y su figura representativa es Mary Poppendieck.

LSD ofrece principios que son aplicables en cualquier ambiente para mejorar el desarrollo del software. Su meta es desarrollar software en la mitad del tiempo que se estima con la mitad del presupuesto y con la menor cantidad de defectos posibles.

Los principios que presenta son (Reynoso, 2004):

- Eliminar lo sobrado – Referente a los diagramas y modelos que no agregan valor al producto.
- Minimizar inventario – Igualmente, suprimir artefactos tales como documentos de requerimiento y diseño.
- Maximizar el flujo – Utilizar desarrollo iterativo.
- Solicitar demanda – Soportar requerimientos flexibles.
- Otorgar poder a los trabajadores.
- Satisfacer los requerimientos del cliente – Trabajar junto a él, permitiéndole cambiar de ideas.
- Hacerlo bien la primera vez – Verificar temprano y refactorizar cuando sea preciso.
- Abolir la optimización local – Alcance de gestión flexible.
- Asociarse con quienes suministran – Evitar relaciones de adversidad.
- Crear una cultura de mejora continua.

LD y LSD han sido pensados como complemento de otros métodos como XP o Scrum, y no como una metodología excluyente a implementar en la empresa.

Crystal Methodologies

Se trata de un conjunto de metodologías para el desarrollo de software, las cuales han sido creadas por Alistair Cockburn. Se caracterizan por estar centradas en la comunicación entre las personas que componen el equipo de un proyecto .

La familia Crystal posee una codificación en colores para conocer el rango de la complejidad de una metodología, puesto que cuanto más oscuro sea un color, más peso tiene la metodología. Cuanto más crítico es un sistema, más rigor se requiere para desarrollarlo.

Dentro de las metodologías Crystal más conocidas se encuentran:

- La metodología Crystal Clear posee dentro de su equipo de desarrollo de 3 a 8 personas, se destaca por ser la más ágil de la serie (Marcelo Hernán, 2004)
- La Crystal Orange posee de 25 a 50 personas en su equipo de proyecto.

Crystal plantea la necesidad de tener un usuario real con el objetivo de definir correctamente los requisitos funcionales y no funcionales del software que se quiere realizar.

El desarrollo de software por parte de esta metodología se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar.

El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas.

Herramientas Analizadas

En este acápite se expone un estudio realizado sobre dos herramientas análogas a la propuesta.

Eclipse Process Framework Composer (EPF Composer)

Es una herramienta de código abierto, bajo la licencia Eclipse Public License - v 1.0 (EPL), utilizada por la Fundación de Eclipse para sus software (IBM, 2006).

EPF Composer es un método para el desarrollo de autorías de contenido y para la publicación de procesos. Le permite a una organización construir el proceso de desarrollo desde la base y personalizarlo a través de un framework existente, además integra las metodologías XP y SCRUM.

EPF Composer está desarrollado en Java, sobre componentes de código abierto y estándares. No incluye un ambiente de ejecución de Java, por sus siglas en inglés (JRE), evitando los asuntos de concesión de licencias. Requiere de operar la aplicación con Sun o IBM JRE versión 1.4.2 y alguna otra versión superior (IBM, 2006).

Esta herramienta está formada por diferentes componentes entre los cuales se encuentran:

UMA (Arquitectura de método unificada): Define cómo el contenido de los métodos y procesos de EPF están estructurados.

UMA agrupa dos categorías y varias subcategorías:

Contenido de los métodos: Describe los roles, las tareas que llevan a cabo, los artefactos producidos por estas tareas y una guía de soporte. Pueden ser categorizados en grupos lógicos con fines de visualización e indexados. Los elementos del contenido de los métodos son independientes de un ciclo de vida de desarrollo. En efecto, son rehusados a menudo en ciclos de vida de desarrollo múltiples.

Procesos: Describen el ciclo de vida de desarrollo. Definen secuencias de tareas a llevar a cabo por roles y artefactos realizados en tiempo extra. Los procesos son expresados típicamente como flujo de trabajo o estructuras de falla. La secuencia de las tareas dentro de la estructura de falla representa diferentes tipos de ciclo de vida de desarrollo, tales como: cascada, incremental, e iterativo generalmente.

EPF Composer está basado en Open Up, constituyendo una base para los procesos pequeños que tiendan a modificarse o ampliarse debido a requisitos particulares o a nuevas necesidades del cliente.

Rational Method Composer de IBM (RMC):

Es una plataforma de proceso de desarrollo de software flexible que ayuda a dar orientación personalizada a los equipos de proyecto. Es el nuevo release principal de RUP y representa un

avance en lo que se refiere a proporcionar una solución del proceso que vaya más allá del desarrollo de software (IBM 2007).

Las mejores prácticas del RMC, son un conjunto de procesos de ingeniería de software habilitados que dan guía para conducir las actividades de desarrollo del equipo.

Requisitos de hardware.

Intel Pentium III 800 MHz, o superior a este procesador.

Requiere 768 MB de RAM; aunque si posee más memoria mejora la capacidad de respuesta.

Espacio de disco.

Se necesita 750 MB para instalar Rational® Method Componer.

El directorio temporal requiere 1500 MB durante la instalación.

El display debe poseer una resolución de video de 1024 x 768 x 256 color, o superior.

Sistemas operativos.

Microsoft™ Windows™ 2000 Professional SP4

Microsoft Windows XP Professional (SP1® or SP2®)

Microsoft Windows 2003 Enterprise Edition SP1

Microsoft Windows 2003 Standard Edition SP1

Red Hat Enterprise Linux™ 4 WS (Update 4)

Red Hat Desktop 4 (Update 4)

SUSE Linux Enterprise Server 9 SP3

SUSE Linux Enterprise Server 10

Navegadores.

IE 6.0 SP™ 1

IE 5.5 SP 2

Firefox 1.5.0.6, o superior

Mozilla 1.7.12, o superior

Conclusiones

No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto ya que a estas las conforman diversos elementos y están regidas por principios diferentes. El estudio realizado arroja como resultado que la metodología RUP es la más completa de ellas, ofrece un detallado flujo de actividades las cuales están muy bien definidas, además de proponer la creación de diversos planes que ayudan con la organización de proceso de desarrollo de software y permite al equipo tener un control de las posibles debilidades que puedan presentar, además de que los obliga a planificar una estrategia de defensa ante la posible ocurrencia de estas, pero sucede que no en todos los proyectos RUP es aplicable, ya sea por el tamaño o por las características de su desarrollo, la misma esta diseñada para proyectos que utilicen la programación orientada a objetos y/o contemplen un gran desarrollo. En un escenario donde RUP no pueda ser aplicado podemos utilizar las diversas metodologías ágiles que permiten la modelación de los proceso de una forma sencilla y eficiente, cada una caracterizada por diferentes elementos y así configurar la metodología a utilizar para el desarrollo del software que se desee llevar a cabo.

La caracterización de estas metodologías permite concluir que los elementos que las componen coinciden en muchas de ellas, estos se resumen en: las fases o etapas, las disciplinas, las actividades que se realizan, los roles y los artefactos como un producto del trabajo que se construye, además en algunas de ellas se definen principios y buenas prácticas para el desarrollo de software.

En cuanto a las herramientas analizadas, se concluye que tienen grandes semejanzas marcadas fundamentalmente porque tanto el RMC y el EPF Composer ambas brindan la posibilidad de diseñar la metodología a utilizar en el software que se desea desarrollar, de igual forma tienen varios inconvenientes, como lo son el hecho de que una de ellas, RMC, es propietaria desarrollada por IBM, además que está basada íntegramente en RUP, por otra parte EPF Composer, está basada en RMC; ambas son aplicaciones de escritorio, no diseñadas con los mismos objetivos que se persiguen con la herramienta que se propone desarrollar en esta investigación.

Capítulo 2: Características del Sistema

Introducción

En este capítulo se presenta el modelo de dominio donde se describe y permite tener una mejor y completa comprensión del entorno en que se ubica el sistema a modelar. Además de un análisis de los elementos necesarios que permitirán la modelación del sistema, para ello se selecciona la metodología de desarrollo RUP, por el completo análisis que brinda acerca de cómo desarrollar un software con calidad, como lenguaje unificado de modelado se utiliza UML (por sus siglas en inglés), el cual permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, y como herramienta CASE el Visual Paradigm, herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Además, en el capítulo se relacionan los requisitos de software tanto los funcionales como los no funcionales, además se describen los procesos o Casos de Uso (CU en lo adelante) del sistema.

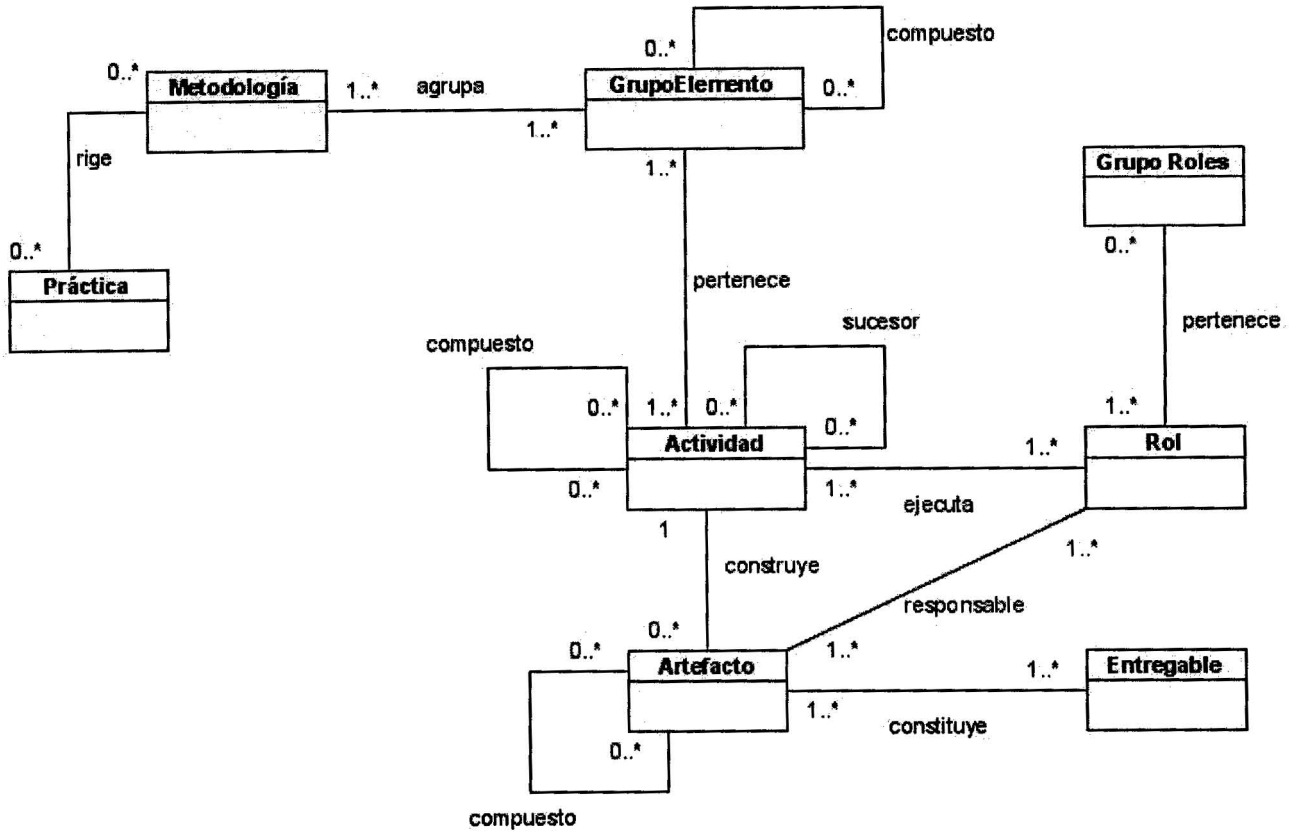
Descripción del problema

Actualmente el proceso de desarrollo de software para los ingenieros no es una tarea fácil, las metodologías de desarrollo son diversas, cada una brinda soluciones particulares basadas en sus propios principios, compuestas por diferente elementos, los cuales no pueden ser, en muchas ocasiones, adaptados a todo tipo de proyecto lo que hace que para algunos donde resulta muy factible determinada metodología para otros no sea una buena opción a seguir.

Para lograr que los ingenieros de software tengan una plataforma donde puedan consultar las metodologías de desarrollo y configurarla de acuerdo a las características de cada proyecto en particular, se ofrece la solución de un Habilitador Metodológico.

Modelo del dominio

Durante el análisis del entorno se identificaron varios conceptos que se relacionan entre sí, estos definen el dominio del sistema, que permite a los usuarios, clientes, desarrolladores e interesados, utilizar un vocabulario común lo que facilita la captura correcta de los requisitos de software y nos permite ofrecer una solución adecuada.



Modelo de Dominio

Conceptos del Modelo del Dominio

Metodología: Define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Es un proceso donde participan un grupo de roles desempeñando un grupo de actividades para crear un conjunto de

artefactos, es decir, una metodología está formada por un grupos de actividades, roles y artefactos interrelacionados, regidos por prácticas recomendadas y principios de ingeniería de software, ejemplo: Metodología del Proceso Unificado de Rational (RUP) como ejemplo más general de ellas.

Prácticas y Principios: Son afirmaciones, postulaciones, acotaciones, mandamientos o preceptos, los cuales, deben ser tomados en cuenta por las personas que desempeñan roles durante la ejecución de sus actividades, ejemplo: En Extreme Programming (XP) se debe programar en pares y hacer código simple.

Grupo elemento: Es una agrupación de elementos de acuerdo a un criterio dado, ejemplo: Fase (etapa), Disciplina, u otro nivel de organización según las siguientes consideraciones:

Fase: Es un período de tiempo entre dos objetivos importantes donde se desarrollan una serie de actividades.

Disciplina: Es una rama de la ciencia de Ingeniería de software que agrupa un conjunto de actividades, ejemplo: Modelamiento del negocio y Requerimientos es una disciplina de la Ingeniería de Software y entre sus actividades se encuentra: en Requerimiento, la definición de los requerimientos funcionales y no funcionales (Pressman, 1997).

Actividad: Una actividad describe una unidad de trabajo. Es una tarea que se ejecuta por un rol en específico, tiene un propósito claro, por lo general expresado en términos de la creación o actualización de algunos artefactos. Dentro de una tarea, cada rol que participa se traza un objetivo bien definido. No describe cuando se debe realizar el trabajo, sino que se describe todo el trabajo que se realiza en todo el ciclo de vida de desarrollo que contribuye a alcanzar el objetivo de la meta trazada.

Una actividad puede estar compuesta por otras actividades si éstas se visualizan en un nivel más bajo de abstracción, ejemplo: Realizar diagramas de clases del análisis constituye una actividad asociada al flujo de Análisis y Diseño de la Metodología RUP.

Artefacto: Son productos de trabajo bien definidos que son consumidos, producidos a modificados por las actividades. Los artefactos pueden estar compuestos por otros artefactos. Los roles usan los artefactos para ejecutar las tareas y para producir artefactos en la realización de las tareas, ejemplo: El artefacto Modelo de diseño puede estar compuesto por los artefactos Clases del diseño, Diseño de paquetes y Diseño de subsistemas.

Entregable: Es un artefacto que brinda una descripción y definición para empaquetar otros entregables, y pueden ser entregados a una parte externa o ser utilizados internamente por los miembros del equipo de desarrollo. El contenido y empaquetamiento de los entregables pueden necesitar ser modificados en cada uno de los proyectos sobre la base de esas recomendaciones. Los entregables se utilizan para representar a un producto de un proceso que tiene valor, material o de otro tipo, a un cliente u otras partes interesadas, ejemplo: El entregable Modelo del sistema puede estar compuesto por los artefactos Especificación de los Casos de Uso del sistema y el diagrama de estos.

Rol: Define un conjunto de habilidades, competencias y responsabilidades que deben poseer o desempeñar las personas que están ejecutando el rol. Los roles especifican quien realiza cada tarea y definen de quien es la responsabilidad de los artefactos. Son normalmente asignados a una persona, o un conjunto de los personas trabajando juntas como un equipo. Un miembro del equipo del proyecto, en general, cumple muchas funciones diferentes. Hay que tener en cuenta que los roles no necesariamente son personas ni equivalentes a los cargos de puestos de trabajo, en cambio, describen cómo los individuos deben comportarse en el proyecto y las responsabilidades que tienen. Los miembros de la organización juegan diferentes roles, ejemplo: Diseñador de Base de Datos, Programador.

Grupo de roles: Es el agrupamiento de lo roles en un nivel de organización, ejemplo: Grupo Analista, enmarca los roles que se encuentra involucrados en la obtención de los requerimientos de software, este grupo puede estar compuesto por el rol Arquitecto de negocio, que se encarga de toda la arquitectura del negocio, participa en todas las decisiones importantes relativas a la estructura y comportamiento del sistema; incluye identificar y documentar los aspectos significantes del negocio que se encuentran dentro del alcance del sistema. Además puede encontrarse dentro de este grupo

el rol Analista de sistema, su función es dirigir y coordinar la obtención de los requerimientos del sistema y determinar el alcance del mismo. Un rol puede encontrarse asociado a diversos grupos de roles, y un grupo de roles está constituido por varios, ejemplo: RUP propone que el rol Administrador de prueba se encuentra en los grupos de roles Administrador y Probador y que el rol Stakeholder se encuentra en los grupos Analista y General.

Descripción del modelo del dominio

Las metodologías de desarrollo de software están regidas por un conjunto de buenas prácticas que permiten un mejor desarrollo en la construcción del software, y dirigidas por un proceso de desarrollo específico, estas metodologías a su vez agrupan varios elementos, estos son las fases o etapas, disciplinas o flujos de trabajo, actividades, roles y artefactos, las actividades son ejecutadas por un rol, el cual es responsable de construir los artefactos asociados a la actividad a la que pertenece, los cuales constituyen entregables del proceso de desarrollo de software.

Reglas del negocio

1. El usuario solo podrá tener acceso a consultar las metodologías no podrá realizar ningún cambio sobre estas.
2. Para que la metodología pueda ser terminada tendrá que tener un mínimo de elementos configurados.
3. Se debe identificar el proceso de desarrollo que regirá la metodología.
4. Toda metodología debe tener asociado las prácticas y/o principios bajo los cuales se rige.
5. Toda metodología debe tener al menos una actividad definida y un rol que la realice dicha actividad.
6. Deben existir roles para crear asociaciones en grupos de roles.
7. Deben existir entregables para poder crear asociaciones de grupos de entregables.

Especificación de los requerimientos

Para determinar que debe hacer el sistema, se han identificado los siguientes requerimientos funcionales que representan las condiciones o capacidades que el software debe cumplir:

Requerimientos funcionales

R1. Permitir al usuario autenticarse en el sistema para iniciar sesión de usuario.

R2. Permitir al usuario finalizar sesión de usuario.

R3. El sistema debe permitir la creación de usuarios, para ello tendrá que indicar los siguientes datos:

Usuario, Clave o Contraseña, Nombre, Apellidos, País, E-mail, Dirección, Teléfono, Fax, además de una casilla para escribir un código de seguridad y evitar de esta manera los accesos no deseados.

R4. Brindar la posibilidad de eliminar usuarios.

R5. Permitir que los usuarios sean modificados.

R6. Brindar la posibilidad de cambiar la contraseña de un usuario determinado

R7. Gestionar la metodología de desarrollo de software a utilizar

R7.1. Permitir al usuario configurar una metodología, para ello tendrá que indicar los siguientes datos:

Nombre

Descripción

Prácticas que regirán la metodología

Proceso por el cuál se guiará la metodología

Proyecto

R7.2. Brindar la posibilidad de consultar cualquier metodología almacenada en el sistema.

R7.3. Permitir que el usuario pueda eliminar una metodología, siempre que esta haya sido configurada por él.

R7.4. Permitir al usuario seleccionar una metodología para ser modificada, para esto debe haber sido configurada por él

R7.5. Permitir que el usuario seleccione una metodología para exportarla.

R8. Gestionar las fases que conformarán la metodología.

R8.1. Permitir al usuario configurar fase de la metodología.

R8.2. Seleccionar una fase para ser modificada, con la condición de que esta haya sido configurada por él.

R8.3. Eliminar una determinada fase seleccionada por el usuario, validando que esta haya sido configurada por él

R9. Gestionar las disciplinas que conformarán la metodología.

R9.1. Permitir al usuario configurar una disciplina

R9.2. Posibilitar al usuario la modificación de una determinada disciplina, teniendo en cuenta que esta haya sido configurada por él.

R9.3. Eliminar una disciplina, validando que esta haya sido configurada por el usuario que solicita la acción.

R10. Gestionar las actividades que conformarán la metodología.

R10.1. Brindar la posibilidad al usuario de configurar una actividad

R10.2. Permitir al usuario modificar las actividades, siempre comprobando que esta haya sido configurada por él.

R10.3. Permitir eliminar una actividad configurada por el usuario que solicite la acción de eliminar.

R11. Gestionar los roles que conformarán la metodología.

R11.1. Posibilitar al usuario la configuración de los roles que intervendrán en el desarrollo del software.

R11.2. Facilitar al usuario modificar un rol determinado, teniendo como premisa que este haya sido configurado por él.

R11.3. Permitir al usuario seleccionar un rol para ser eliminado, teniendo en cuenta que este haya sido configurado por él

R12. Gestionar los artefactos que conformarán la metodología.

R12.1. Permitir la configuración de un artefacto.

R12.2. Permitir al usuario modificar un artefacto determinado, comprobando que este haya sido creado por el usuario que solicita la acción.

R12.3. Posibilitar la eliminación de un artefacto determinado, que seleccione el usuario, para ello el artefacto debe haber sido creado por él.

R13. Asociar una plantilla

R14. Asociar una imagen a los elementos que conforman una metodología

R15. Asociar tabla a los elementos que conforman una metodología

R16. Asociar icono a los elementos que conforman una metodología

R17. Asociar referencia o vínculo a los elementos que conforman una metodología

R18. Permitir que el usuario pueda asociar elementos de una metodología con otros existentes.

R19. Brindar la posibilidad al usuario de listar los elementos que están almacenados en el sistema.

R20. Validar metodología

Requisitos no Funcionales

3.1. Usabilidad

3.1.1. El sistema cuenta con una interfaz Web amigable para el usuario.

3.1.2. Al sistema es posible acceder desde la mayoría de los navegadores usados en el mundo (Internet Explorer, Mozilla Firefox).

3.1.3. Se utilizan los estándares de conexión IEEE 802.3u (Fast Ethernet) y IEEE 802.11 (Redes inalámbricas).

3.2. Rendimiento

3.2.1. El sistema utiliza las dependencias PostgreSQL-Server-8.1, Apache 2.2.3-4 y Squid 2.6.5-6.

3.2.2. El sistema debe entregar una respuesta a una petición de usuario en menos de 1 segundo.

3.2.3. El sistema debe ser capaz de soportar 100 usuarios concurrentes realizando peticiones.

3.3. Portabilidad

3.3.1. El sistema podrá ser portado por los usuarios en formato PDF y html.

3.4. Interoperatividad

3.4.1. El sistema se comunicará con otros sistemas a través de una capa de servicios Web para facilitar la comunicación.

3.5. Software

3.5.1. El sistema se desarrollará con lenguaje de programación PHP, JavaScript y Ajax.

3.5.2. Para el desarrollo Web se utiliza el Framework CodeIgniter.

3.5.3. Se utilizará un servidor con sistema operativo GNU/Linux Debian 4.0 (Etch).

3.5.4. En las terminales clientes se garantizará el sistema operativo GNU/Linux Debian 4.0 (Etch).

3.5.5. La comunicación de las terminales clientes con el servidor será a través del protocolo TCP/IP a una velocidad de 100 Mbps.

3.5.6. El sistema utilizará una Base de Datos implementada en PostgreSQL-server-8.1.

3.6. Hardware

3.6.1. Cada cliente donde se use el sistema debe tener las siguientes características mínimas: 256 MBytes de RAM, 60 GBytes de Disco Duro (HDD), P3-1.5 GHz.

3.6.2. Cada servidor donde se instale el sistema debe tener las siguientes características recomendadas: 2 GBytes de RAM, 120 GBytes de Disco Duro (HDD), P4-2.5 GHz para un máximo de 2000 usuarios.

3.7. Soporte

3.7.1. El servidor tendrá instalado un Servidor Apache con el módulo PHP5 activado.

3.7.2. El sistema recoge un sistema de *Logs* que se revisan periódicamente para comprobar su estado de funcionamiento.

3.7.3. Al ocurrir una falla grave, el sistema envía a través de correo electrónico un aviso a los administradores del mismo.

3.8. Seguridad

3.8.1. El sistema debe comunicarse usando un protocolo seguro (https) para los casos de autenticación y el resto del proceso bajo protocolo común (http).

3.8.2. La información almacenada en el sistema está protegida de acceso no autorizado.

3.8.3. El sistema cuenta con un módulo de seguridad que permite a los usuarios acceder solamente a los lugares a los que se le ha dado acceso.

3.8.4. A los usuarios autorizados se le garantizará el acceso a la información sin que los mecanismos utilizados para lograr la seguridad retrasen la obtención de los datos deseados en un momento dado.

3.9. Requerimiento de ayuda y documentación

3.9.1. Contará con una Guía de Usuario para facilitar el trabajo con el sistema.

3.9.2. El sistema estará integrado con una ayuda en línea para cada una de las partes del mismo.

3.10. Interfaz

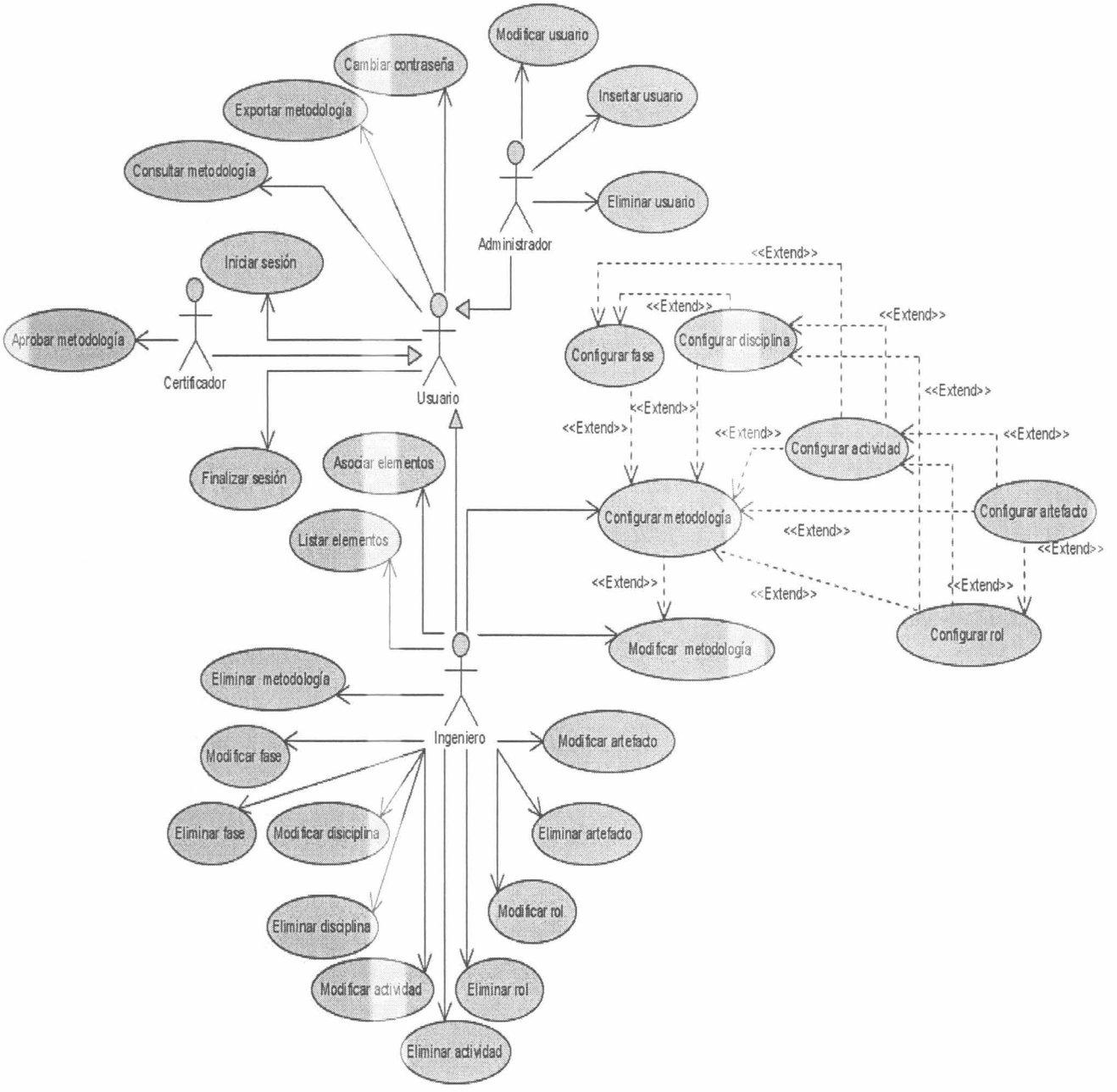
3.10.1. Se utilizará el protocolo de comunicación TCP/IP.

Modelo del Sistema

Actores

Actor del sistema	Descripción
Usuario	Representa a las personas que realizarán consultas en el Habilitador metodológico.
Ingeniero	Identifica a ingenieros o personal que tiene la responsabilidad de configurar la metodología de desarrollo que se utilizará en el proyecto al que pertenece.
Administrador	Persona con total acceso de administración en el sistema.
Certificador	Es la persona o grupo de personas con los conocimientos suficientes que es/son el/los responsable(s) de validar la metodología configurada por el Ingeniero.

Diagrama de Casos de Uso del Sistema



Descripción de los Casos de Uso del Sistema

Caso de Uso: Iniciar sesión

Caso de Uso (CU-1):	Iniciar sesión
Actores:	Usuario
Resumen:	El CU se inicia cuando el actor desea iniciar sesión, y finaliza cuando la sesión es iniciada con los privilegios correspondientes al actor. Para este proceso se utiliza SSO (Autenticación centralizada)
Referencia:	R1
CU asociados:	
Precondiciones:	Que el actor tenga una cuenta creada en el sistema.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Usuario selecciona la opción de Iniciar sesión.	1.1 El sistema muestra una interfaz donde el Usuario podrá introducir los datos para su autenticación (usuario y contraseña). Será a través de la Interfaz del SSO.
2. El Usuario introduce los datos para su autenticación.	2.1 El sistema verifica que los datos sean correctos. 2.2 El sistema inicia la sesión de usuario solicitada.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

	Si los datos no son correctos, el sistema le muestra un mensaje informándole al Usuario y vuelve al paso 2.
Poscondiciones:	Queda inicializada la sesión del Usuario.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Finalizar sesión

Caso de Uso (CU-2):	Finalizar sesión
Actores:	Usuario
Resumen:	El CU se inicia cuando el actor desea finalizar sesión, después de la confirmación del actor es finalizada y así es como termina el CU.
Referencia:	R2
CU asociados:	
Precondiciones:	El usuario debe estar autenticado.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Usuario decide finalizar sesión desde cualquier interfaz del sistema.	1.1 El sistema le muestra un mensaje para verificar si esta seguro de la acción.
2. El Usuario confirma la acción de finalizar sesión.	2.1 El sistema finaliza la sesión.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.1 Si el Usuario no confirma la acción de finalizar, no se finaliza la sesión.
Poscondiciones:	Queda finalizada la sesión del Usuario.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Insertar usuario

Caso de Uso (CU-3):	Insertar usuario
Actores:	Administrador
Resumen:	El CU se inicia cuando el actor desea insertar un nuevo usuario, después de la confirmación del actor es finalizada y así es como termina el CU. Este proceso ya está implementado para la plataforma RINDE con la cual se integrará el sistema.
Referencia:	R3.
CU asociados:	
Precondiciones:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El Administrador decide insertar un nuevo usuario en el sistema.	1.1 El sistema le muestra una interfaz con los datos que tendrá que especificar (usuario, clave o contraseña, nombre, apellidos, país, e-mail, dirección, teléfono, fax, además de una casilla para escribir un código de seguridad). Los datos que aparecen subrayados son obligatorios.
2. El Administrador introduce los datos.	2.1 El sistema valida los datos introducidos por el Administrador. 2.2 El sistema inserta el usuario.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si existe algún problema con los datos el sistema le envía un mensaje al Administrador y regresa al paso 2.
Poscondiciones:	El usuario es insertado en la Base de Datos del sistema.
Prioridad:	
Especificaciones Complementarias:	Este proceso de insertar usuario será a través del SSO (Autenticación centralizada) que está diseñado en la plataforma RINDE, con la cual este sistema va a ser integrado.

Caso de Uso: Eliminar usuario

Caso de Uso (CU-4):	Eliminar usuario

Actores:	Administrador
Resumen:	El Caso de Uso se inicia cuando el Administrador decide eliminar un usuario de la Base de Datos.
Referencia:	R4.
CU asociados:	
Precondiciones:	Que el actor se haya autenticado como Administrador del sistema.

Flujo Normal de Eventos	
Sección "nombre del escenario"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador solicita la opción de eliminar un usuario. 2. El Administrador entra el identificador del usuario. 3. El Administrador da clic en el botón eliminar.	1.1 El sistema muestra una interfaz para que entre el identificador del usuario. 2.1 El sistema busca el Usuario en la Base de Datos dado el parámetro. 3.1 El sistema elimina el usuario.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si el Usuario no es encontrado en la Base de Datos, se muestra un mensaje de advertencia y regresa al paso 2.

Poscondiciones:	El usuario finalmente es eliminado de la Base de Datos del sistema.
Prioridad:	

Especificaciones Complementarias:	Este proceso de eliminar usuario será a través del SSO (Autenticación centralizada) que está diseñado en la plataforma RINDE, con la cual este sistema va a ser integrado.

Caso de Uso: Modificar usuario

Caso de Uso (CU-5):	Modificar usuario	
Actores:	Administrador, Usuario	
Resumen:	El Caso de Uso se inicia cuando el actor decide cambiar algún dato, en el caso de un Usuario Común, podrá modificar solamente algunos datos de su perfil, en el caso del Administrador, tendrá permiso para realizar todas las modificaciones.	
Referencia:	R5.	
CU asociados:		
Precondiciones:		
Flujo Normal de Eventos		
Sección "nombre del escenario"		
Acción del Actor		**Respuesta del Sistema**

1. El Actor solicita la opción de modificar un Usuario.	1.1 Si el rol del que realiza la acción es Administrador, ir a la sección **"Actualizar usuario por un Administrador"**. 1.2 Si el rol que realiza la acción es de un Usuario Común, ir a la sección **"Actualizar usuario por un Usuario Común"**.
Sección "Modificar usuario por un Administrador"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
2. El Administrador introduce el identificador del usuario al cual desea modificar (puede modificar su propio perfil).	2.1 El sistema busca los datos del Usuario. 2.2 El sistema muestra el perfil relacionado con el Usuario a modificar.
3. El Administrador modifica los datos (excepto el usuario y la contraseña, puede modificar cualquier dato).	3.1 El sistema valida los datos entrados. 3.2 El sistema guarda los cambios realizados en la Base de Datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

	2.1 Si el sistema no encuentra el usuario solicitado por el Administrador, notifica al Administrador y brinda la posibilidad de volver a introducir el identificador del usuario.
	3.1 En caso de error notifica al Administrador y brinda la posibilidad de volver a introducir el identificador del usuario.
Sección "Modificar usuario por un Usuario Común"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
2. El Usuario solicita la acción de ver su perfil.	2.1 El sistema busca los datos del Usuario. 2.2 El sistema muestra el perfil relacionado con el Usuario.
3. El Usuario modifica los datos permitidos, estos son los relacionados con el Nombre, Apellidos, Correo Electrónico, dirección, teléfono.	3.1 El sistema valida los datos entrados. 3.2 El sistema guarda los cambios realizados en la Base de Datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 En caso de error notifica al Usuario y brinda la posibilidad de volver a introducir el identificador del usuario.

Poscondiciones:	Los datos de uno o varios usuarios finalmente son modificados y guardados en la Base de Datos del sistema.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Cambiar contraseña

Caso de Uso (CU-6):	Cambiar contraseña
Actores:	Administrador, Usuario
Resumen:	El Caso de Uso se inicia cuando el actor decide cambiar su contraseña.
Referencia:	R6
CU asociados:	
Precondiciones:	Que el actor se haya autenticado en el sistema.

Flujo Normal de Eventos	
Sección "nombre del escenario"	
Acción del Actor	**Respuesta del Sistema**
1. El actor solicita cambiar la contraseña.	1.1 Si el actor que está autenticado es un Administrador, ir a la sección **"Cambiar contraseña por Administrador".** 1.2. Si el actor que está autenticado es un Usuario, ir a la sección **"Cambiar contraseña por Usuario".**

Sección "Cambiar contraseña por Administrador"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador escoge la opción de cambiar la contraseña de un Usuario y no la opción de cambiar su propia contraseña.	1.1 El sistema le muestra un formulario para que indique el Usuario al que le desea cambiar la contraseña.
2. El Administrador entra el identificador del Usuario.	2.1 El sistema busca el Usuario en la Base de Datos. 2.2 El sistema le muestra el mismo formulario con los campos para la contraseña y la confirmación de esta.
3. El Administrador introduce y confirma la nueva contraseña.	3.1 El sistema valida la información. 3.2 El sistema actualiza la nueva contraseña del Usuario en la Base de Datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador escoge la opción de cambiar su propia contraseña.	1.1 El sistema le muestra un formulario para que entre la contraseña actual, la nueva y confirme esta última.
2. El Administrador indica los datos para realizar el cambio de la contraseña.	2.1 El sistema valida la información. 2.2 El sistema actualiza la nueva contraseña del Usuario en la Base de Datos.

	3.1 Si la contraseña entrada tiene errores, se envía un mensaje de advertencia y regresa al paso 3.
Sección "Cambiar contraseña por Usuario"	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema pide que inserte y confirme la nueva contraseña.
2. El Usuario inserta y confirma su nueva contraseña.	2.1 El sistema valida la información. 2.2 El sistema actualiza la contraseña del Usuario en la Base de Datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 El sistema envía un mensaje en caso de tener problema la contraseña entrada por el Usuario y retorna al paso 2.
Poscondiciones:	Los contraseña del actor finalmente es modificada y guardada en la Base de Datos del sistema.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Configurar Metodología

Caso de Uso (CU-7):	Configurar Metodología
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para configurar una metodología de desarrollo de software de acuerdo a las necesidades y/o exigencias de su proyecto, terminando cuando el actor la configura y esta queda salvada en la BD del sistema.
Referencia:	R7.1
CU asociados:	Configurar fase (extend); Configurar disciplina (extend); Configurar actividad (extend); Configurar rol (extend); Configurar artefacto (extend).
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de configurar metodología.	1.1 El sistema le muestra una interfaz donde el Ingeniero deberá especificar los datos referentes a la metodología (nombre, descripción de la misma, ya sea breve o ampliada, principios y/o prácticas, además debe dejar claro el proceso por el cuál se guiará su desarrollo).
2. El Ingeniero introduce los datos que corresponden a la metodología que desea configurar.	2.1 El sistema verifica que se haya introducido la información correspondiente. 2.2 El sistema le muestra una interfaz donde podrá configurar la metodología, permitiéndole

	indicar los elementos que la conformarán (fases y/o etapas, disciplinas, actividades, artefactos, roles), para ello el actor deberá ir a los Casos de Uso correspondientes: Caso de Uso extendido "Configurar fase y/o etapas". Caso de Uso extendido "Configurar disciplina". Caso de Uso extendido "Configurar actividad". Caso de Uso extendido **"Configurar rol".** Caso de Uso extendido "Configurar artefacto".
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

Poscondiciones:	El Ingeniero configura la metodología de desarrollo a utilizar en su proyecto.
Prioridad:	
Especificaciones Complementarias:	Esta acción terminará cuando el Ingeniero haya configurado los elementos de la metodología a utilizar, no necesariamente tendrá que configurarlos todos pero si se exigirá que la metodología esté conformada por un mínimo de elementos. Este CU tiene dos escenarios, uno es el que queda explicado en esta tabla y el otro es cuando el Ingeniero toma como base una metodología ya creada y la desea configurar/adaptar de acuerdo a las características de su proyecto de desarrollo de software, para el segundo caso el suceso de actividades será parecido al primer caso, solo que el actor contará con los

	datos de la metodología que será adaptada y solo tendrá que decidir que parte de esta tomar.

Caso de Uso: Modificar metodología

Caso de Uso(CU-8):	Modificar metodología
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para modificar una metodología de desarrollo de software de acuerdo a las necesidades y/o exigencias de su proyecto; finaliza el proceso cuando se salven los cambios realizados.
Referencia:	R7.4
CU asociados:	Configurar metodología (extend)
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de modificar metodología.	1.1 El sistema le brinda una interfaz para que el Ingeniero indique que metodología quiere modificar.
2. El Ingeniero indica la metodología que desea modificar.	2.1 El sistema verifica que la metodología seleccionada por el Ingeniero haya sido creada por él (dado el caso, no es obligatorio que el nombre sea modificado.) 2.2 El sistema le devuelve una interfaz donde

	aparecerán todos lo elementos de la metodología seleccionada.
3. El Ingeniero modifica los elementos de la metodología que seleccionó.	3.1 El sistema verifica que los datos introducidos por el Ingeniero estén correctos. 3.2 El sistema pide confirmación al Ingeniero para guardar los cambios.
4. El Ingeniero confirma la acción de guardar los cambios.	4.1 El sistema guarda los cambios en la Base de Datos del sistema.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si la metodología seleccionada no fue creada por el actor, ir al caso de uso **"Configurar metodología"**.
	3.1 Si los datos introducidos por el Ingeniero no son correctos, el sistema muestra un mensaje al actor para que este los corrija, regresar al punto 3 del flujo normal de eventos.
4. El Ingeniero no confirma la acción de modificar la metodología.	4.1 El sistema no almacena los cambios.
Poscondiciones:	Queda modificada la metodología que utilizará el Ingeniero en el proyecto.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Consultar metodología

Caso de Uso (CU-9):	Consultar metodología
Actores:	Usuario
Resumen:	El CU se inicia cuando el actor desea consultar algunas de las metodologías existentes en el sistema, finalizando cuando el sistema le muestra la metodología para que sea consultada.
Referencia:	R7.2
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Usuario selecciona la opción de consultar metodología.	1.1 El sistema brinda una interfaz donde aparecerá el nombre de las metodologías existentes.
2. El Usuario selecciona la metodología que desea consultar.	2.1 El sistema busca todos los elementos relacionados con la metodología seleccionada. 2.2 El sistema muestra una interfaz con la metodología que será consultada.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

Poscondiciones:	El Usuario consulta una metodología seleccionada.

Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Eliminar metodología

Caso de Uso(CU-10):	Eliminar metodología
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para eliminar una metodología de desarrollo de software confeccionada por el mismo; finalizando el CU cuando se realiza la acción.
Referencia:	R7.3
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de eliminar metodología.	1.1 El sistema le muestra una interfaz donde aparecerán las metodologías que él ha creado.
2. El Ingeniero selecciona la metodología que desea eliminar.	2.1 El sistema le pide confirmación al Ingeniero para eliminar la metodología.
3. El Ingeniero confirma la acción de eliminar la metodología seleccionada.	3.1 El sistema elimina la metodología de la BD.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si la acción de eliminar no es confirmada el sistema no realiza la acción.
Poscondiciones:	El Ingeniero elimina la metodología seleccionada.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Exportar Metodología

Caso de Uso(CU-11):	Exportar Metodología
Actores:	Usuario
Resumen:	El CU se inicia cuando el Usuario decide exportar una metodología de desarrollo de software, finalizando este obtenga la metodología exportada en el formato seleccionado.
Referencia:	R7.5
CU asociados:	
Precondiciones:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El Usuario selecciona la opción de exportar metodología.	1.1 El sistema le brinda una interfaz donde el Usuario deberá seleccionar el formato en el que desea que se exporte la (html, PDF)
2. El Usuario indica el tipo de formato.	2.1 El sistema exporta la metodología seleccionada por el Usuario, en el formato indicado.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones:	La metodología es exportada al formato indicado.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Configurar fase

Caso de Uso(CU-12):	Configurar fase
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor decide configurar una fase para la metodología de desarrollo a utilizar, finalizando cuando esta es configurada en el sistema.
Referencia:	R8.1
CU asociados:	Configurar metodología (extend), Configurar disciplina (extend), Configurar actividad (extend).

Precondiciones:	Que el actor haya configurado los elementos iniciales de la metodología a la que pertenecerá la fase a configurar.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de configurar fase.	1.1 El sistema le brinda una interfaz donde el Ingeniero deberá introducir los datos referentes a la fase que desea configurar (Nombre, Descripción) 1.2 El sistema brinda la posibilidad al Ingeniero de asociar a esta fase un icono o un enlace.
2 El Ingeniero introduce los datos requeridos por el sistema.	2.1 El sistema verifica que el Ingeniero haya introducido todos los datos requeridos.
3. El Ingeniero decide asociar un icono a la fase que está configurando.	3.1 El sistema le brinda la posibilidad de asociar el icono.
4. El Ingeniero decide asociar un enlace a la fase.	4.1 El sistema le brinda la posibilidad de indicar el enlace al cual hace referencia. 4.2 Si el Ingeniero decide indicar los elementos por los que estará compuesta dicha fase: a) Ir al CU "Configurar disciplina" b) Ir al CU "Configurar actividad"
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

	2.1 Si el Ingeniero no introdujo todos los datos requeridos, el sistema lo remite al paso 2 del flujo normal de eventos.
3. El Ingeniero decide no asociar icono a la fase que esta configurando.	3.1 El sistema no realiza ninguna acción.
4. El Ingeniero decide no asociar enlace a la fase que está configurando.	4.1 El sistema no realiza ninguna acción. 4.2 Si el Ingeniero no decide configurar los elementos de la fase el sistema solo guarda los datos especificados hasta el punto 4 del FN de eventos.
Poscondiciones:	
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Modificar fase

Caso de Uso(CU-13):	Modificar fase
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para modificar una fase de desarrollo de software de acuerdo a las necesidades y/o exigencias de su proyecto; finaliza el proceso cuando se salven los cambios realizados.
Referencia:	R8.2

CU asociados:	
Precondiciones:	El actor debe haber solicitado la acción desde el proceso de configurar metodología.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de modificar fase.	1.1 El sistema le brinda una interfaz para que el Ingeniero indique que fase quiere modificar.
2. El Ingeniero indica la fase que desea modificar.	2.1 El sistema verifica que la fase seleccionada por el Ingeniero haya sido creada por él. 2.2 El sistema le devuelve una interfaz donde aparecerán todos lo elementos configurables de la fase seleccionada (Nombre, Descripción).
3. El Ingeniero modifica los elementos de la fase que seleccionó.	3.1 El sistema verifica que los datos introducidos por el Ingeniero estén correctos. 3.2 El sistema pide confirmación al Ingeniero para guardar los cambios.
4. El Ingeniero confirma la acción de guardar los cambios.	4.1 El sistema guarda los cambios en la Base de Datos del sistema.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

	2.1 Si la fase seleccionada no fue creada por el Ingeniero, el sistema no le permite modificar la fase seleccionada y el sistema le muestra una interfaz advirtiendo al actor. Permitiéndole volver a seleccionar la fase.
	3.1 Si los datos introducidos por el Ingeniero no son correctos, el sistema muestra un mensaje al Ingeniero para que este los corrija, regresar al punto 3 del flujo normal de eventos.
4. El Ingeniero no confirma la acción de modificar la fase.	4.1 El sistema no almacena los cambios.

Poscondiciones:	El Ingeniero modifica la fase de desarrollo.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Eliminar fase

Caso de Uso(CU-14):	Eliminar fase
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para eliminar una fase confeccionada por el mismo; finalizando el CU cuando se realiza la acción.
Referencia:	R8.3
CU asociados:	

Precondiciones:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de eliminar fase.	1.1 El sistema le muestra una interfaz donde aparecerán las fases que él ha creado.
2. El Ingeniero selecciona la fase que desea eliminar.	2.1 El sistema le muestra una interfaz al Ingeniero donde aparecen los elementos con los que dicha fase tiene relación, para que el Ingeniero seleccione si desea que se eliminen sus relaciones (podrá decidir si elimina una, algunas o todas las relaciones).
3. El Ingeniero confirma la acción de eliminar la actividad seleccionada con las relaciones que decida eliminar.	3.1 El sistema realiza la acción de eliminar.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. El Ingeniero decide mantener algunos elementos que tenían relación con la actividad.	3.1 El sistema mantiene los elementos en la Base de Datos, de manera que estos puedan ser asociados en otro momento.
Poscondiciones:	El Ingeniero elimina la fase seleccionada.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Configurar disciplina

Caso de Uso(CU-15):	Configurar disciplina
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor decide configurar una disciplina para la metodología de desarrollo a utilizar, finalizando cuando esta es configurada en el sistema.
Referencia:	R9.1
CU asociados:	Configurar actividad (extend), Configurar rol (extend)
Precondiciones:	Que el actor haya configurado los elementos iniciales de la metodología a la que pertenecerá la disciplina a configurar.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de configurar disciplina.	1.1 El sistema le brinda una interfaz donde el Ingeniero deberá introducir los datos referentes a la disciplina que desea configurar (Nombre, Descripción) 1.2 El sistema brinda la posibilidad al Ingeniero de asociar a esta disciplina un icono o un enlace.
2. El Ingeniero introduce los datos requeridos por el sistema.	2.1 El sistema verifica que el Ingeniero haya introducido todos los datos requeridos.
3. El Ingeniero decide asociar un icono a la	3.1 El sistema le brinda la posibilidad de

disciplina que está configurando.	asociar el icono.
4. El Ingeniero decide asociar un enlace a la disciplina.	4.1 El sistema le brinda la posibilidad de indicar el enlace al cual hace referencia. 4.2 Si el Ingeniero decide indicar los elementos por los que estará compuesta dicha disciplina: a) Ir al CU "Configurar actividad" b) Ir al CU "Configurar rol"
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si el Ingeniero no introduce todos los datos requeridos, el sistema lo remite al paso 2 del flujo normal de eventos.
3. El Ingeniero decide no asociar icono a la disciplina que esta configurando.	3.1 El sistema no realiza ninguna acción.
4. El Ingeniero decide no asociar enlace a la disciplina que está configurando.	4.1 El sistema no realiza ninguna acción.
Poscondiciones:	
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Modificar disciplina

Caso de Uso(CU-16):	Modificar disciplina

Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para modificar una disciplina; finaliza el proceso cuando se salven los cambios realizados.
Referencia:	R9.2
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de modificar disciplina.	1.1 El sistema le brinda una interfaz donde aparecerán las metodologías que han sido creadas por él.
2. El Ingeniero indica la disciplina que desea modificar.	2.1 El sistema devuelve una interfaz donde aparecerán todos lo elementos configurables de la disciplina seleccionada (Nombre, Descripción).
3. El Ingeniero modifica los elementos de la disciplina que seleccionó.	3.1 El sistema verifica que los datos introducidos por el actor estén correctos. 3.2 El sistema pide confirmación al Ingeniero para guardar los cambios.
4. El Ingeniero confirma la acción de guardar los cambios.	4.1 El sistema guarda los cambios en la Base de Datos del sistema.
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
	3.1 Si los datos introducidos por el Ingeniero no son correctos, el sistema muestra un mensaje al actor para que este los corrija, regresar al punto 3 del flujo normal de eventos.
4. El Ingeniero no confirma la acción de modificar la disciplina.	4.1 El sistema no almacena los cambios.

Poscondiciones:	El Ingeniero configura la disciplina.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Eliminar disciplina

Caso de Uso(CU-17):	Eliminar disciplina
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para eliminar una disciplina confeccionada por el mismo; finalizando el CU cuando se realiza la acción.
Referencia:	R9.3
CU asociados:	
Precondiciones:	
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El Ingeniero selecciona la opción de eliminar disciplina.	1.1 El sistema le muestra una interfaz donde aparecerán las disciplinas que él ha creado.
2. El Ingeniero selecciona la disciplina que desea eliminar.	2.1 El sistema le muestra una interfaz al Ingeniero donde aparecen los elementos con los que dicha disciplina tiene relación, para que el Ingeniero seleccione si desea que se eliminen sus relaciones (podrá decidir si elimina una, algunas o todas las relaciones).
3. El Ingeniero confirma la acción de eliminar la actividad seleccionada con las relaciones que decida eliminar.	3.1 El sistema elimina la disciplina de la Base de Datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. El Ingeniero decide mantener algunos elementos que tenían relación con la disciplina.	3.1 El sistema mantiene los elementos en la Base de Datos, de manera que estos puedan ser asociados en otro momento.

Poscondiciones:	El Ingeniero elimina la disciplina seleccionada.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Configurar actividad

Caso de Uso(CU-18):	Configurar actividad
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor decide configurar una actividad para la metodología de desarrollo a utilizar, finalizando cuando esta es configurada en el sistema.
Referencia:	R10.1
CU asociados:	Configura rol (extend), Configurar artefacto (extend)
Precondiciones:	El actor podrá configurar la actividad desde la interfaz de Configurar metodología, pero además podrá hacerlo desde la interfaz de configurar disciplina, donde el actor podrá especificar las actividades por la que estará compuesta la disciplina.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de configurar actividad.	1.1 El sistema le brinda una interfaz donde el Ingeniero deberá introducir los datos referentes a la actividad que desea configurar (Nombre, Descripción). 1.2 El sistema brinda la posibilidad al Ingeniero de asociar a esta actividad un icono o un enlace.
2. El Ingeniero introduce los datos requeridos por el sistema.	2.1 El sistema verifica que el Ingeniero haya introducido todos los datos requeridos.
3. El Ingeniero decide asociar un icono a la	3.1 El sistema le brinda la posibilidad de

actividad que está configurando.	asociar el icono.
4. El Ingeniero decide asociar un enlace a la actividad.	4.1 El sistema le brinda la posibilidad de indicar el enlace al cual hace referencia. 4.2 Si el Ingeniero decide indicar los elementos por los que estará compuesta dicha actividad: a) Ir al CU "Configurar rol" b) Ir al CU "Configurar artefacto"
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si el Ingeniero no introduce todos los datos requeridos, el sistema lo remite al paso 2 del flujo normal de eventos.
3. El Ingeniero decide no asociar icono a la actividad que esta configurando.	3.1 El sistema no realiza ninguna acción.
4. El Ingeniero decide no asociar enlace a la actividad que está configurando.	4.1 El sistema no realiza ninguna acción.
Poscondiciones:	Queda configurada la actividad
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Modificar actividad

Caso de Uso(CU-19):	Modificar actividad

Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para modificar una actividad; finaliza el proceso cuando se salven los cambios realizados.
Referencia:	R10.2
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de modificar actividad.	1.1 El sistema le brinda una interfaz donde aparecerán las actividades creadas por él para que indique que actividad quiere modificar.
2. El Ingeniero indica la actividad que desea modificar.	2.3 El sistema devuelve una interfaz donde aparecerán todos lo elementos configurables de la actividad seleccionada (Nombre, Descripción).
3. El Ingeniero modifica los elementos de la actividad que seleccionó.	3.1 El sistema verifica que los datos introducidos por el actor estén correctos. 3.2 El sistema pide confirmación al Ingeniero para guardar los cambios.
4. El Ingeniero confirma la acción de guardar los cambios.	4.1 El sistema guarda los cambios en la Base de Datos del sistema.
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
	3.1 Si los datos introducidos por el Ingeniero no son correctos, el sistema muestra un mensaje al Ingeniero para que este los corrija, regresar al punto 3 del flujo normal de eventos.
4. El Ingeniero no confirma la acción de modificar la actividad.	4.1 El sistema no almacena los cambios.

Poscondiciones:	El Ingeniero configura la actividad.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Eliminar actividad

Caso de Uso(CU-20):	Eliminar actividad
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para eliminar una actividad confeccionada por el mismo; finalizando el CU cuando se realiza la acción.
Referencia:	R10.3
CU asociados:	
Precondiciones:	
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El Ingeniero selecciona la opción de eliminar actividad.	1.1 El sistema le muestra una interfaz donde aparecerán las actividades que él ha creado.
2. El Ingeniero selecciona la actividad que desea eliminar.	2.1 El sistema le muestra una interfaz al Ingeniero donde aparecen los elementos con los que dicha actividad tiene relación, para que el actor seleccione si desea que se eliminen sus relaciones (podrá decidir si elimina una, algunas o todas las relaciones).
3. El Ingeniero confirma la acción de eliminar la actividad seleccionada con las relaciones que decida eliminar.	3.1 El sistema realiza la acción de eliminar.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. El Ingeniero decide mantener algunos elementos que tenían relación con la actividad.	3.1 El sistema mantiene los elementos en la Base de Datos, de manera que estos puedan ser asociados en otro momento.
Poscondiciones:	La actividad es eliminada.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Configurar rol

Caso de Uso(CU-21):	Configurar rol
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor decide configurar un rol para la metodología de desarrollo a utilizar, finalizando cuando esta es configurada en el sistema.
Referencia:	R11.1
CU asociados:	Configurar artefacto (extend)
Precondiciones:	El actor podrá configurar el rol desde la interfaz de Configurar metodología, pero además podrá hacerlo desde la interfaz de configurar actividad, donde podrá especificar el o los roles que serán responsables de la actividad que se esté configurando.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de configurar rol.	1.1 El sistema le brinda una interfaz donde el Ingeniero deberá introducir los datos referentes al rol que desea configurar (Nombre, Descripción, grupo al que pertenece). 1.2 El sistema brinda la posibilidad al Ingeniero de asociar a este rol un icono o un enlace.
2. El Ingeniero introduce los datos requeridos por el sistema.	2.1 El sistema verifica que el Ingeniero haya introducido todos los datos requeridos.
3. El Ingeniero decide asociar un icono al rol que	3.1 El sistema le brinda la posibilidad de

está configurando.	asociar el icono.
4. El Ingeniero decide asociar un enlace al rol.	4.1 El sistema le brinda la posibilidad de indicar el enlace al cual hace referencia.
5. El Ingeniero indica el enlace al cuál hace referencia y especifica el icono.	5.1 El sistema asocia el icono y mantiene el enlace indicado por el Ingeniero. 5.2 Si el Ingeniero decide indicar el artefacto o los artefactos por los que será responsable dicho rol: a) Ir al CU "Configurar artefacto"
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si el Ingeniero no introduce todos los datos requeridos, el sistema lo remite al paso 2 del flujo normal de eventos.
3. El Ingeniero decide no asociar icono al rol que está configurando.	3.1 El sistema no realiza ninguna acción.
4. El Ingeniero decide no asociar enlace al rol que está configurando.	4.1 El sistema no realiza ninguna acción. 4.2 Si el Ingeniero no desea especificar la relación del rol con sus responsabilidades, el CU finaliza en el punto 5.1 del FN de eventos.
Poscondiciones:	Queda configurado el rol
Prioridad:	
Especificaciones	

Complementarias:	

Caso de Uso: Modificar rol

Caso de Uso(CU-22):	Modificar rol
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para modificar un rol; finaliza el proceso cuando se salven los cambios realizados.
Referencia:	R11.2
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de modificar rol.	1.1 El sistema le brinda una interfaz con los roles que han sido creados por él para que indique cuál desea modificar.
2. El Ingeniero indica el rol que desea modificar.	2.1 El sistema devuelve una interfaz donde aparecerán todos lo elementos configurables del rol seleccionado (Nombre, Descripción).
3. El Ingeniero modifica los elementos del rol que seleccionó.	3.1 El sistema verifica que los datos introducidos por el actor estén correctos. 3.2 El sistema pide confirmación al actor para guardar los cambios.

4. El Ingeniero confirma la acción de guardar los cambios.	4.1 El sistema guarda los cambios en la BD del sistema.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si los datos introducidos por el actor no son correctos, el sistema muestra un mensaje al actor para que este los corrija, regresar al punto 3 del flujo normal de eventos.
4. El Ingeniero no confirma la acción de modificar el rol.	4.1 El sistema no almacena los cambios.
Poscondiciones:	El Ingeniero configura el rol.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Eliminar rol

Caso de Uso(CU-23):	Eliminar rol
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para eliminar un rol confeccionado por el mismo; finalizando el CU cuando se realiza la acción.
Referencia:	R11.3
CU asociados:	

Precondiciones:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de eliminar rol.	1.1 El sistema le muestra una interfaz donde aparecerán los roles que él ha creado.
2. El Ingeniero selecciona el rol que desea eliminar.	2.1 El sistema le muestra una interfaz al Ingeniero donde aparecen los elementos con los que dicho rol tiene relación, para que el Ingeniero seleccione si desea que se eliminen sus relaciones (podrá decidir si elimina una, algunas o todas las relaciones).
3. El Ingeniero confirma la acción de eliminar el rol seleccionado con las relaciones que decida eliminar.	3.1 El sistema realiza la acción de eliminar.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. El Ingeniero decide mantener algunos elementos que tenían relación con el rol.	3.1 El sistema mantiene los elementos en la Base de Datos, de manera que estos puedan ser asociados en otro momento.
Poscondiciones:	El rol es eliminado.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Configurar artefacto

Caso de Uso(CU-24):	Configurar artefacto
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor decide configurar un artefacto para la metodología de desarrollo a utilizar, finalizando cuando este es configurado en el sistema.
Referencia:	R12.1
CU asociados:	
Precondiciones:	El actor podrá configurar el artefacto desde la interfaz de Configurar metodología, pero además podrá hacerlo desde la interfaz de configurar rol, donde podrá especificar el o los artefactos que serán construidos por dicho rol.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de configurar artefacto.	1.1 El sistema le brinda una interfaz donde el Ingeniero deberá introducir los datos referentes al artefacto que desea configurar (Nombre, Descripción), además deberá especificar si dicho artefacto será un entregable. 1.2 El sistema brinda la posibilidad al Ingeniero de asociar a este artefacto un icono, un enlace o una plantilla (esta última en dependencia de lo establecido en la institución responsable del

	proyecto para el cual será configurada la metodología de desarrollo, a la cual pertenece este artefacto)
2. El Ingeniero introduce los datos requeridos por el sistema.	2.1 El sistema verifica que el Ingeniero haya introducido todos los datos requeridos.
3. El Ingeniero decide asociar un icono al rol que está configurando.	3.1 El sistema le brinda la posibilidad de asociar el icono.
4. El Ingeniero decide asociar un enlace al rol.	4.1 El sistema le brinda la posibilidad de indicar el enlace al cual hace referencia.
5. El Ingeniero decide asociar al artefacto una plantilla.	5.1 El sistema le brinda la posibilidad de buscar y asociar la plantilla al artefacto.
6. El Ingeniero indica el enlace al cuál hace referencia, especifica el icono y la plantilla que desea asociar. (Estas acciones no son excluyentes, ni es obligatorio realizar las tres.)	6.1 El sistema asocia el icono, construye el enlace indicado por el Ingeniero y/o asocia la plantilla.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si el Ingeniero no introduce todos los datos requeridos, el sistema lo remite al paso 2 del flujo normal de eventos.
3. El Ingeniero decide no asociar icono al artefacto que está configurando.	3.1 El sistema no asocia icono al artefacto.

4. El Ingeniero decide no asociar enlace al artefacto que está configurando.	4.1 El sistema no asocia enlace al artefacto.
4. El Ingeniero decide no asociar plantilla al artefacto que está configurando.	4.1 El sistema no asocia ninguna plantilla al artefacto.
Poscondiciones:	Queda configurada el rol
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Modificar artefacto

Caso de Uso(CU-25):	Modificar artefacto
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para modificar un artefacto; finaliza el proceso cuando se salven los cambios realizados.
Referencia:	R12.2
CU asociados:	
Precondiciones:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de modificar	1.1 El sistema le brinda una interfaz con los artefactos que han sido configurados por él

artefacto.	para que indique cuál desea modificar.
2. El Ingeniero indica el artefacto que desea modificar.	2.1 El sistema devuelve una interfaz donde aparecerán todos lo elementos configurables del artefacto seleccionado (Nombre, Descripción), además de poder modificar el hecho de si constituirá un artefacto o no.
3. El Ingeniero modifica los elementos del artefacto que seleccionó.	3.1 El sistema verifica que los datos introducidos por el Ingeniero estén correctos. 3.2 El sistema pide confirmación al Ingeniero para guardar los cambios.
4. El Ingeniero confirma la acción de guardar los cambios.	4.1 El sistema guarda los cambios en la Base de Datos del sistema.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si los datos introducidos por el Ingeniero no son correctos, el sistema muestra un mensaje al actor para que este los corrija, regresar al punto 3 del flujo normal de eventos.
4. El Ingeniero no confirma la acción de modificar artefacto.	4.1 El sistema no almacena los cambios.
Poscondiciones:	El Ingeniero configura el artefacto.
Prioridad:	

Especificaciones Complementarias:	

Caso de Uso: Eliminar artefacto

Caso de Uso(CU-26):	Eliminar artefacto
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor entra a la plataforma para eliminar un artefacto confeccionado por el mismo; finalizando el CU cuando se realiza la acción.
Referencia:	R12.3
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de eliminar artefacto.	1.1 El sistema le muestra una interfaz donde aparecerán los artefactos que él ha creado.
2. El Ingeniero selecciona el artefacto que desea eliminar.	2.1 El sistema le muestra una interfaz al Ingeniero donde aparecen los elementos con los que dicho artefacto tiene relación, para que el Ingeniero seleccione si desea que se eliminen sus relaciones (podrá decidir si elimina una, algunas o todas las relaciones).

3. El Ingeniero confirma la acción de eliminar el artefacto seleccionado con las relaciones que decida eliminar.	3.1 El sistema realiza la acción de eliminar.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. El Ingeniero decide mantener algunos elementos que tenían relación con el artefacto.	3.1 El sistema mantiene los elementos en la Base de Datos, de manera que estos puedan ser asociados en otro momento.
Poscondiciones:	El artefacto es eliminado.
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Listar elementos

Caso de Uso(CU-27):	Listar elementos
Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor desea ver un listado de los diferentes elementos que están configurados en el sistema, esta acción finaliza cuando el actor selecciona el elemento que desea listar y obtiene como respuesta del sistema dichos elementos.
Referencia:	R19
CU asociados:	Asociar elemento (extend)

Precondiciones:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona la opción de listar elementos.	1.1 El sistema le brinda una interfaz donde el Ingeniero podrá seleccionar el tipo de elemento que quiere que sea listado.
2. El Ingeniero selecciona el tipo de elemento.	2.1 El sistema muestra una interfaz con todos los elementos correspondientes al tipo seleccionado por el Ingeniero. 2.2 Si el Ingeniero desea asociar algún elemento a una metodología o a otro elemento entonces ir al CU **"Asociar elemento".**
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones:	El Ingeniero obtiene el listado de los elementos
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Asociar elementos

Caso de Uso(CU-28):	Asociar elementos

Actores:	Ingeniero
Resumen:	El CU se inicia cuando el actor decide asociar un elemento determinado a una metodología o a otro elemento que no pertenezca a su grupo, se finaliza cuando ya el elemento queda asociado según las necesidades del ingeniero.
Referencia:	R18
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Ingeniero selecciona el elemento que desea asociar.	1.1 El sistema muestra una interfaz donde aparecerán los elementos posibles a asociar agrupados en categorías, estas son: Fases, Disciplinas, Actividades, Roles y Artefactos.
2. El Ingeniero selecciona la categoría a la que pertenece el elemento que desea asociar.	2.1 El sistema muestra los elementos contenidos en la categoría seleccionada por el Ingeniero.
3. El Ingeniero selecciona de los elementos mostrados el que desea asociar.	3.1 El sistema le brinda al Ingeniero en la misma interfaz la posibilidad de seleccionar primeramente a que metodología desea asociar el elemento seleccionado.

4. El Ingeniero selecciona la metodología.	4.1 El sistema le muestra los elementos que componen dicha metodología, estos estarán agrupados en categorías también.
5. El Ingeniero selecciona la categoría a la cual desea asociar el elemento	5.1 El sistema le muestra los elementos categorizados con dicha especificación.
6. El Ingeniero escoge el elemento de la metodología al cual quiere realizar la asociación.	6.1 El sistema asocia los elementos como lo indica el Ingeniero.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
5. Si el Ingeniero no desea especificar la categoría a la cual pertenece el elemento a asociar.	5.1 El sistema deberá permitir asociarlo directamente a la metodología seleccionada.
Poscondiciones:	
Prioridad:	
Especificaciones Complementarias:	

Caso de Uso: Validar metodología

Caso de Uso(CU-29):	Validar metodología
Actores:	Certificador
Resumen:	El CU se inicia cuando el actor entra al sistema para realizar la acción de validar una metodología que haya sido configurada en el sistema, finaliza cuando ya la metodología queda validada y se le notifica al responsable

	de la misma.
Referencia:	R20
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Certificador selecciona la opción de validar metodología.	1.1 El sistema muestra una interfaz donde aparecerán las metodologías que aún no ha sido validadas.
2. El Certificador selecciona la metodología que va a validad, esta selección es por prioridad, la primera que este en la lista es la primera a la que se le realizará el proceso de validación.	2.1 El sistema muestra todo lo relacionado con la metodología que será validada.
3. El Certificador aprueba la metodología.	3.1 El sistema le brinda al Certificador la posibilidad de indicar si la metodología queda finalmente aprobada.
4. El Certificador indica que la metodología queda aprobada.	4.1 El sistema le envía un correo electrónico al responsable de la metodología indicándole que el nombre de esta para que sepa que fue aprobada.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. Si el Certificador no indica que la metodología	3.1 El sistema le envía un correo al

queda aprobada.	responsable de la metodología con los detalles por los que no fue aprobada, de tal forma podrá corregirla.
Poscondiciones:	
Prioridad:	
Especificaciones Complementarias:	

Conclusiones

Durante el desarrollo de este capítulo se realizó el modelo de dominio permitiendo una mejor comprensión del entorno y una descripción del problema que se desea resolver. Se expusieron además, los requisitos tanto funcionales como los no funcionales, la especificación de los Casos de Uso del sistema y el diagrama correspondiente.

Capítulo 3: Análisis y diseño del sistema

En este capítulo se desarrollarán diversas actividades referentes al flujo de trabajo de Análisis y Diseño fundamentalmente las que se enmarcan en la fase de Elaboración, estas actividades que se realizarán son: la realización de las clases de análisis, materializándose en el modelo de análisis, a través del cuál podrá comprenderse mejor los requisitos antes de tomar decisiones importantes de diseño, se incluirá además la realización de las clases del diseño donde se obtendrá el modelo de diseño, el cuál es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, es decir, cómo cumple el sistema sus objetivos. El modelo de datos también será un artefacto que se construirá en este capítulo que permitirá describir la estructura lógica y física de la información persistente.

Diagramas de Clases de Análisis

A continuación se muestran los diagramas de análisis correspondientes a los Casos de Uso del sistema, agrupados en paquetes:

DIAGRAMA DE CLASES DE ANÁLISIS (Paquete de metodología)

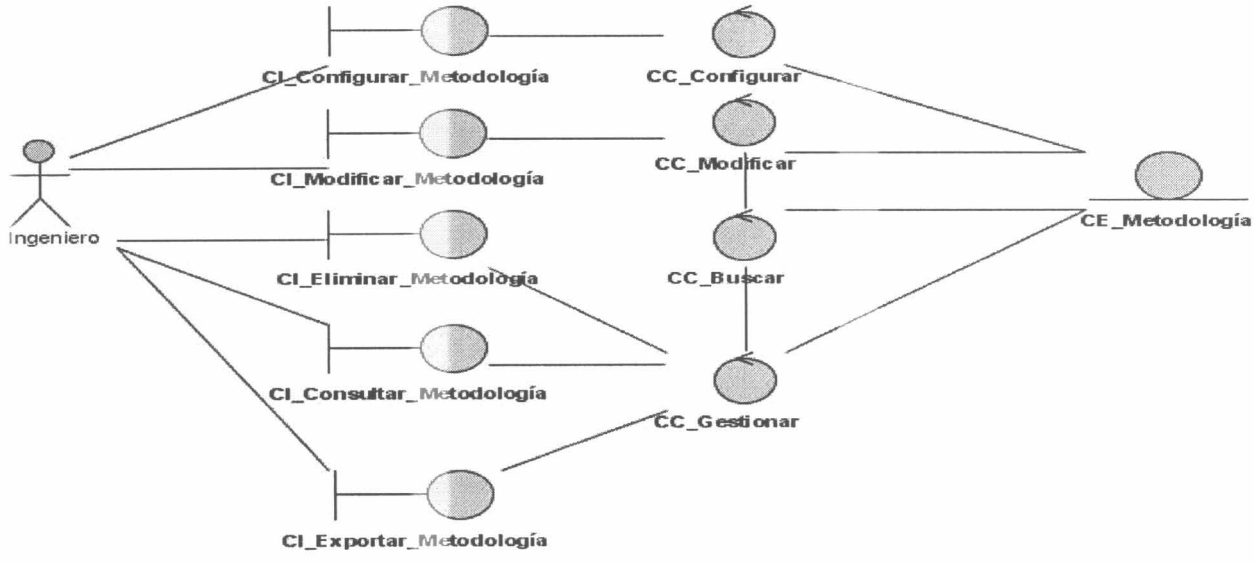


DIAGRAMA DE CLASES DE ANÁLISIS (Paquete de fase)

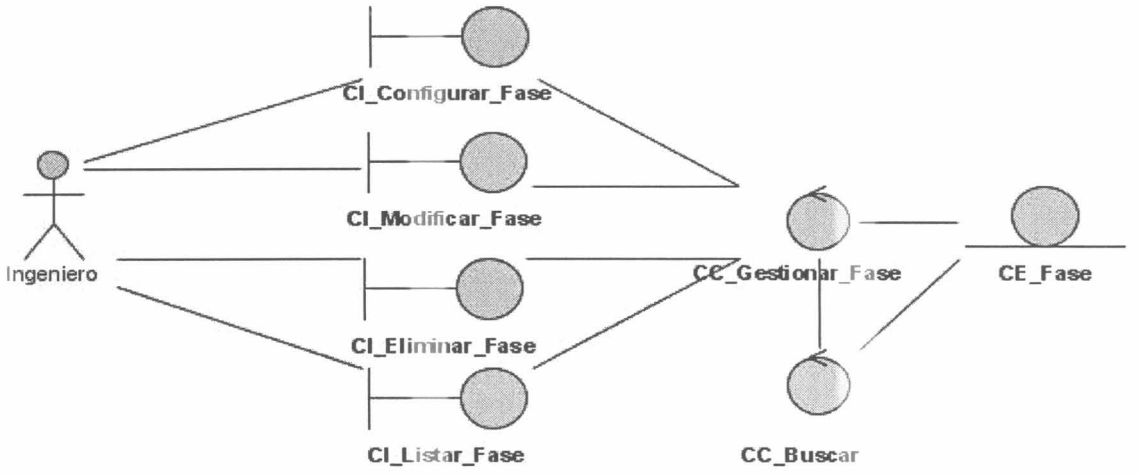


DIAGRAMA DE CLASES DE ANÁLISIS (Paquete de disciplina)

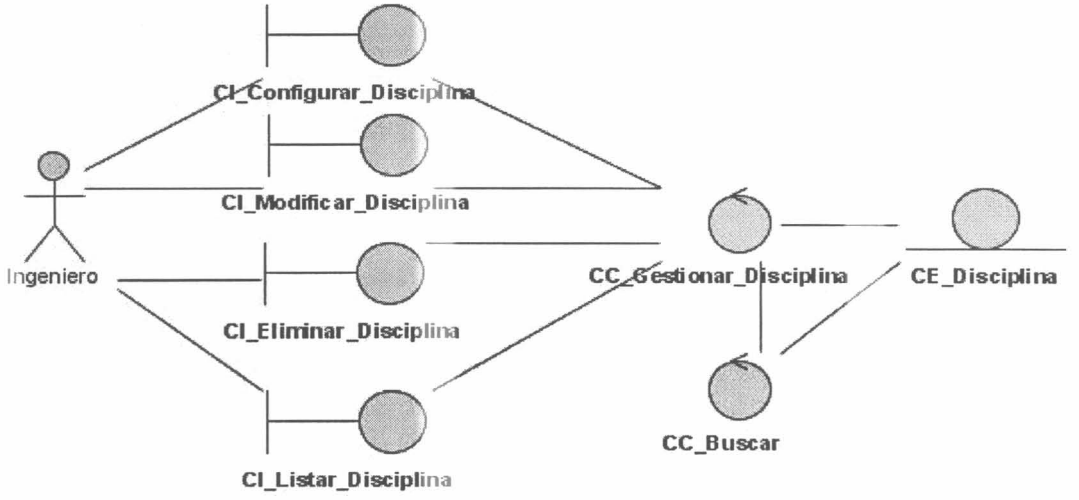


DIAGRAMA DE CLASES DE ANÁLISIS (Paquete de actividad)

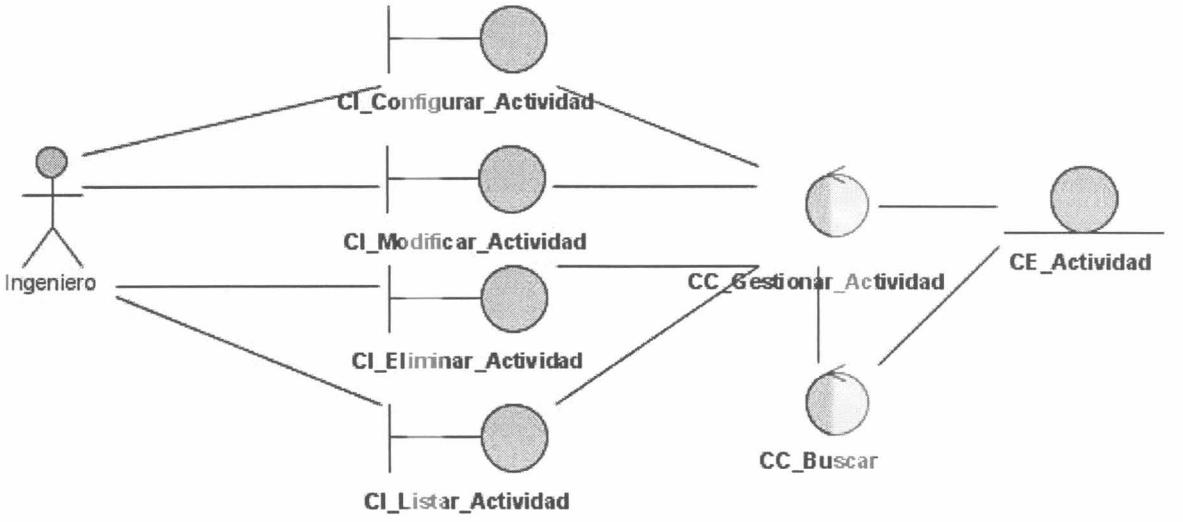


DIAGRAMA DE CLASES DE ANÁLISIS (Paquete de rol)

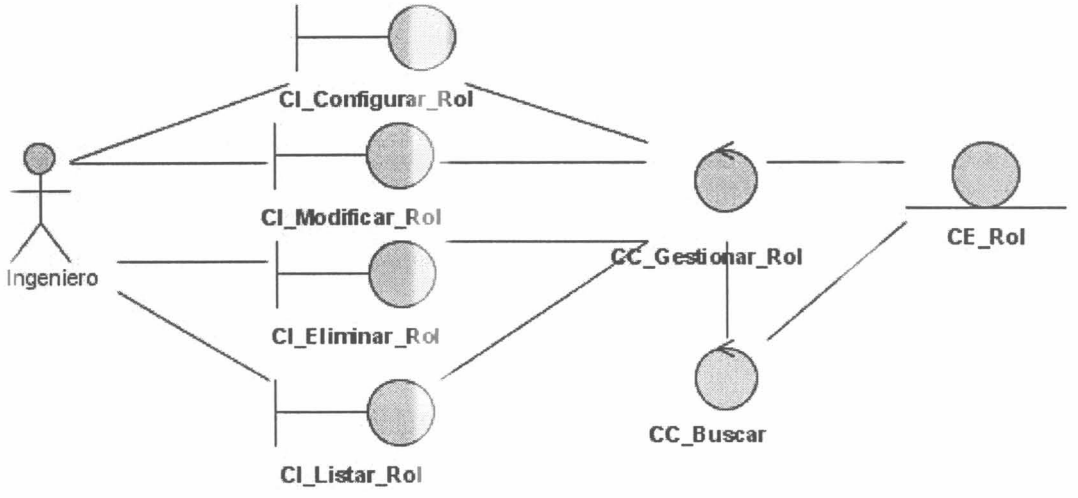


DIAGRAMA DE CLASES DE ANÁLISIS (Paquete de artefacto)

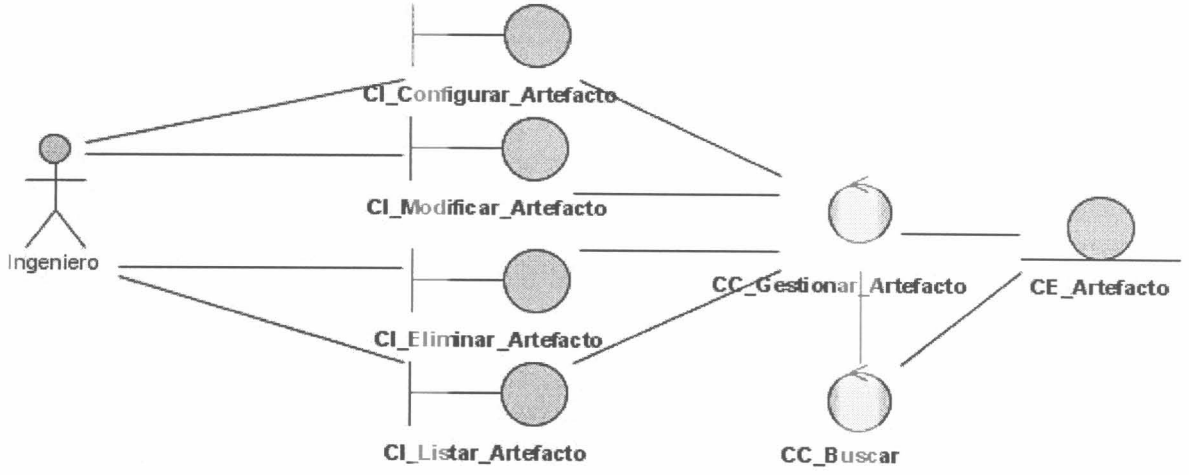
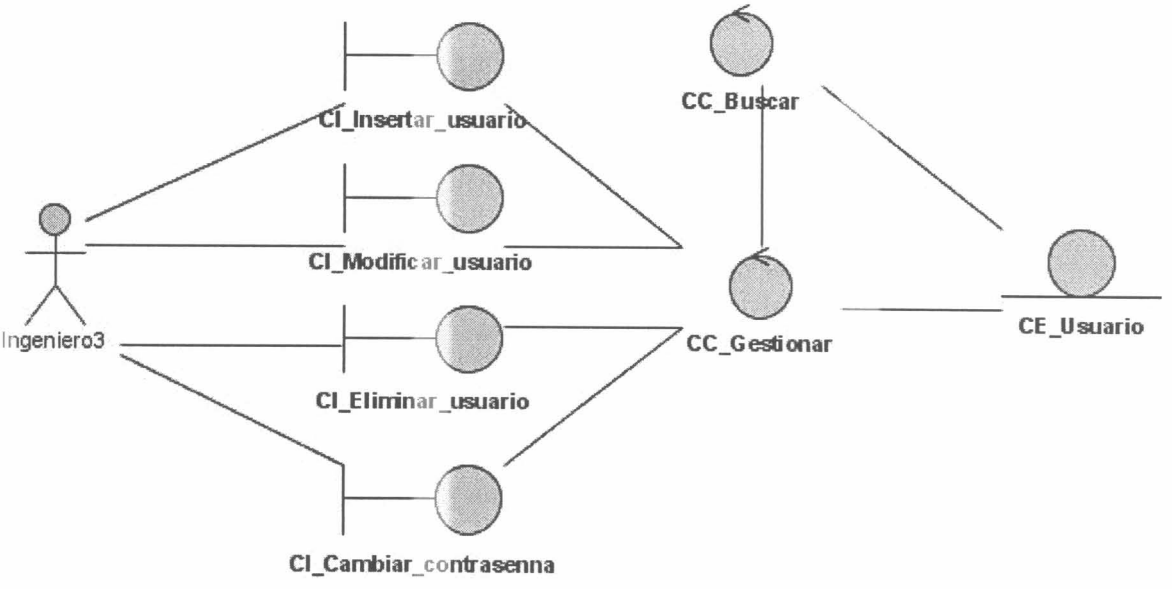


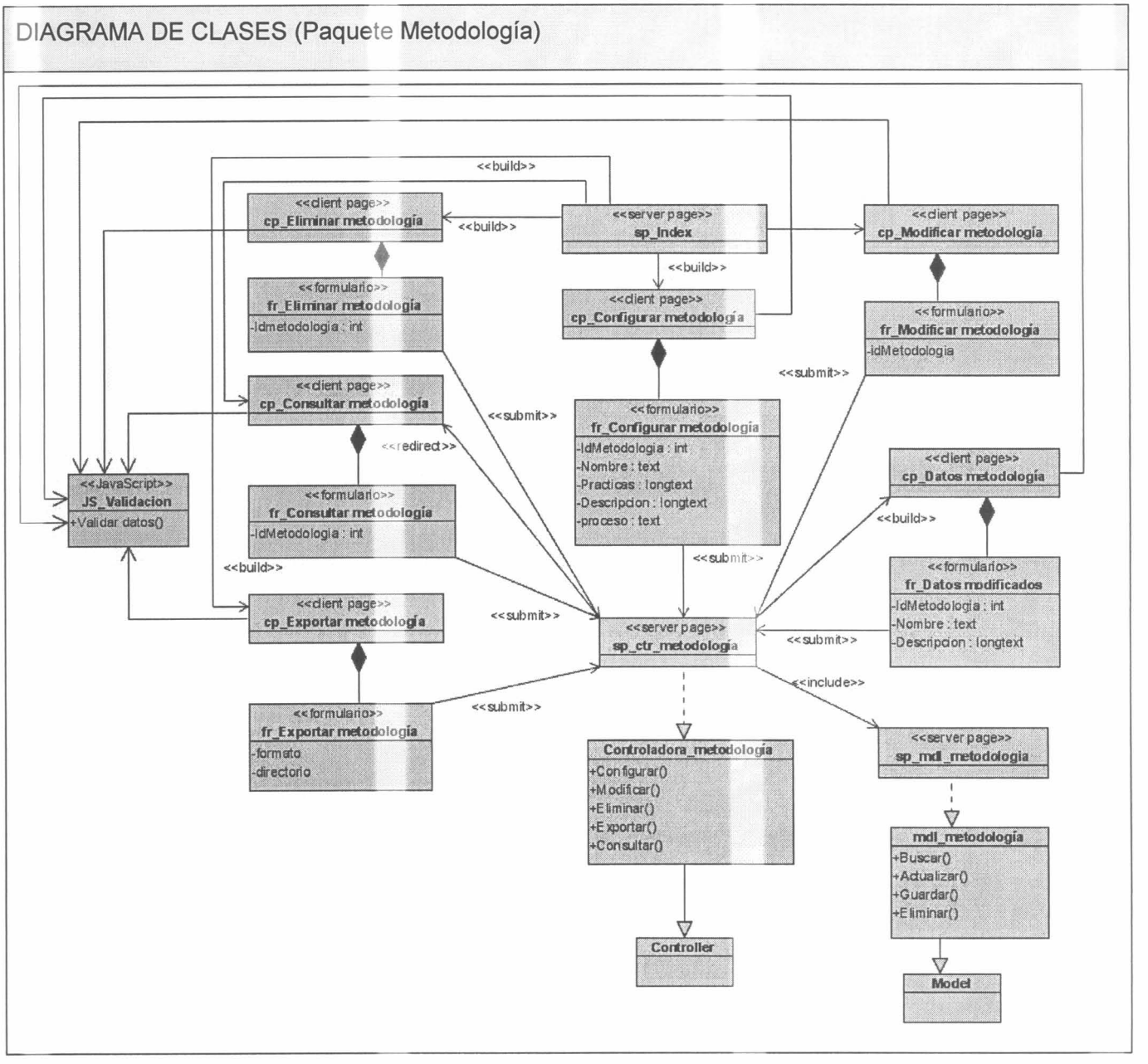
DIAGRAMA DE CLASES DE ANÁLISIS (Paquete de usuario)



Diagramas de Clases de Diseño

A continuación se presenta el diagrama de clases del diseño correspondientes a los Casos de Uso del paquete de Metodología y la explicación de las clases fundamentales, el resto los podrá consultar en el **Anexo 1**:

DIAGRAMA DE CLASES (Paquete Metodología)



Descripción de las clases de Diseño

Descripción de las páginas clientes
Nombre: cp_Configurar metodología, cp_Modificar metodología, cp_Eliminar metodología, cp_Exportar metodología, cp_Consultar metodología.
Tipo de clase: Página Cliente
Descripción: Cada una de estas páginas clientes es construida por la servidora sp_Index, donde el Ingeniero a través de los formularios correspondientes, introducirá los datos requeridos para realizar las acciones sobre la metodología con la cual se está trabajando, estos datos son especificados en el diagrama de clases del diseño expuesto anteriormente. Existe otra página cliente llamada cp_Datos metodología correspondiente al proceso de Modificar metodología, construida por la clase servidora sp_ctr_metodología, en la cual se muestran los datos de la metodología que el usuario ha solicitado para modificar y donde este podrá realizar los cambios.

Descripción de las páginas servidoras
Nombre: sp_Index, sp_ctr_metodología
Tipo de clase: Página servidora
Descripción: La página servidora sp_Index tiene la responsabilidad de construir las páginas clientes en el momento que el Ingeniero hace la petición de alguna de las funcionalidades que brinda el sistema para con las metodologías, estas pueden ser: configurar, modificar, eliminar, consultar o exportar. Por otra parte la página servidora sp_ctr_metodología, tiene una instancia de objeto de la clase controladora del framework, además de tener incluida otra página servidora sp_mdl_metodologia la cuál tendrá una instancia de objeto de la clase modelo del framwork .

Diagramas de Secuencia

A continuación se expondrán los diagramas de secuencia (DS) de los Casos de Usos más significativos, el resto de los DS podrán ser consultados en el **Anexo 2**.

DIAGRAMA DE SECUENCIA CU-7 (Configurar metodología)

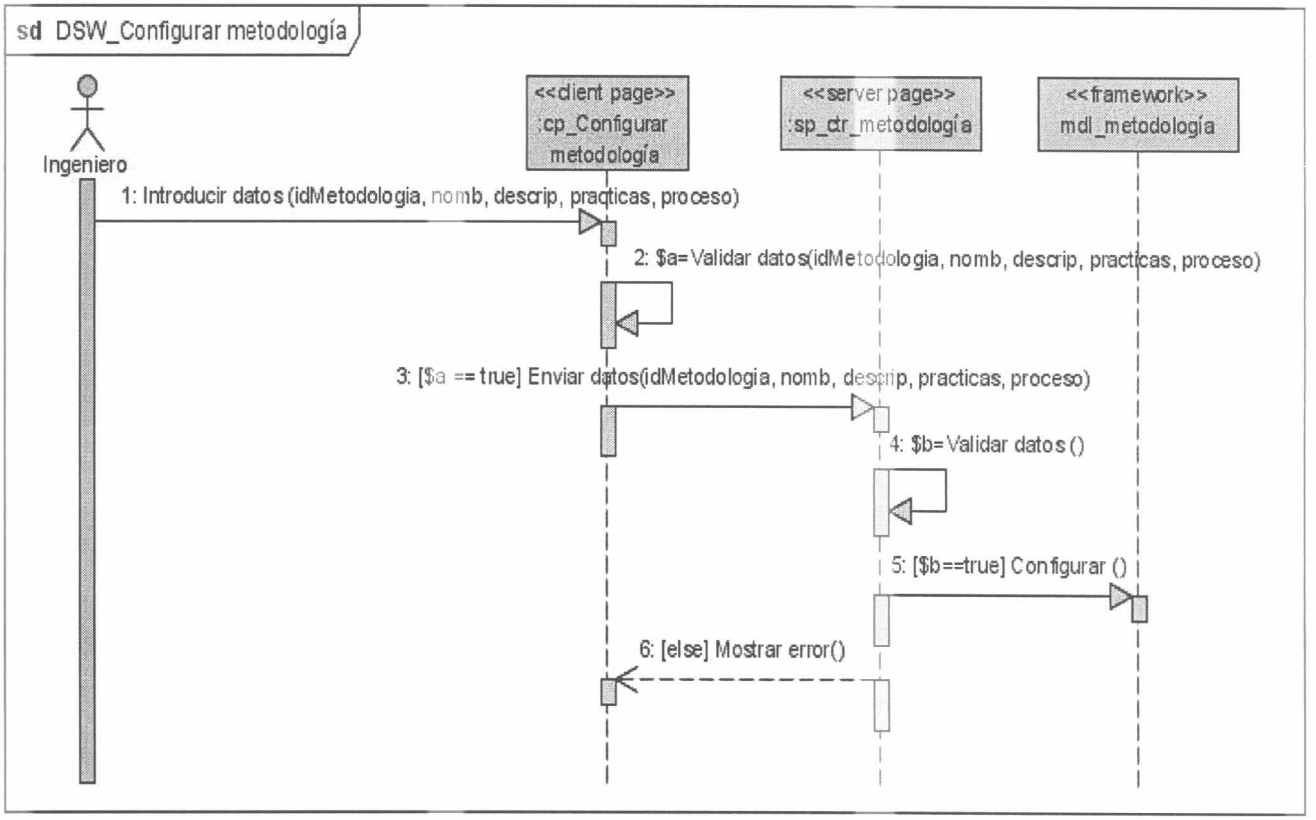


DIAGRAMA DE SECUENCIA CU-10 (Eliminar metodología)

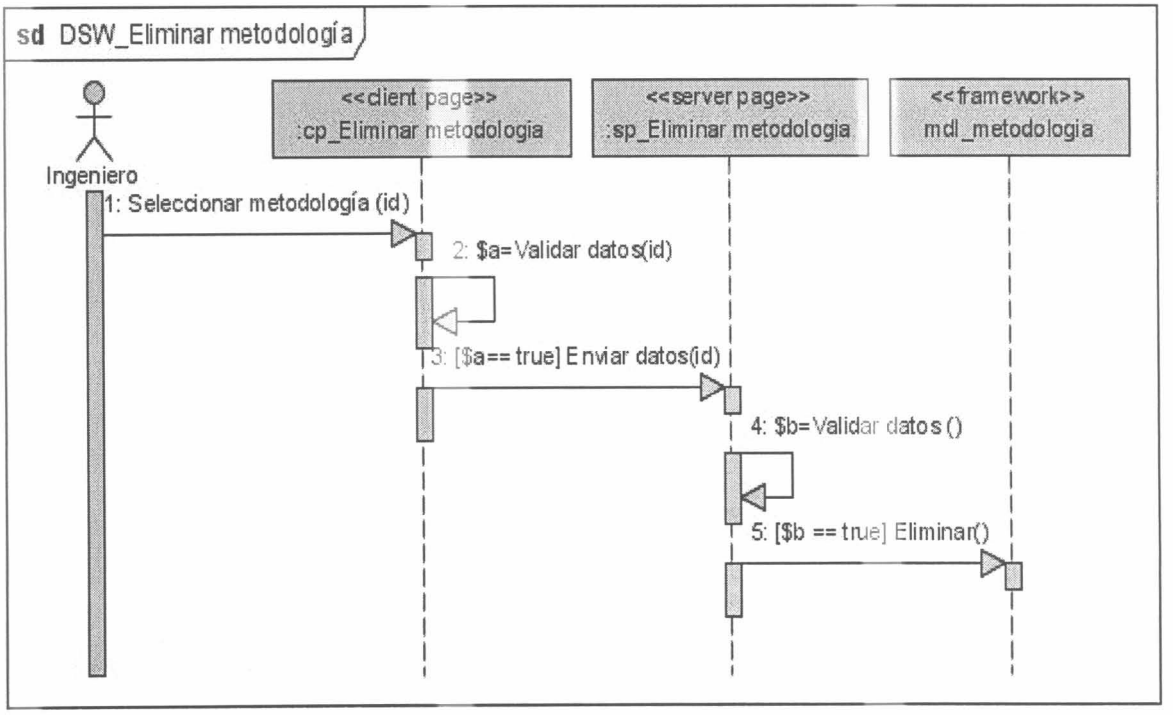


DIAGRAMA DE SECUENCIA CU-8 (Modificar metodología)

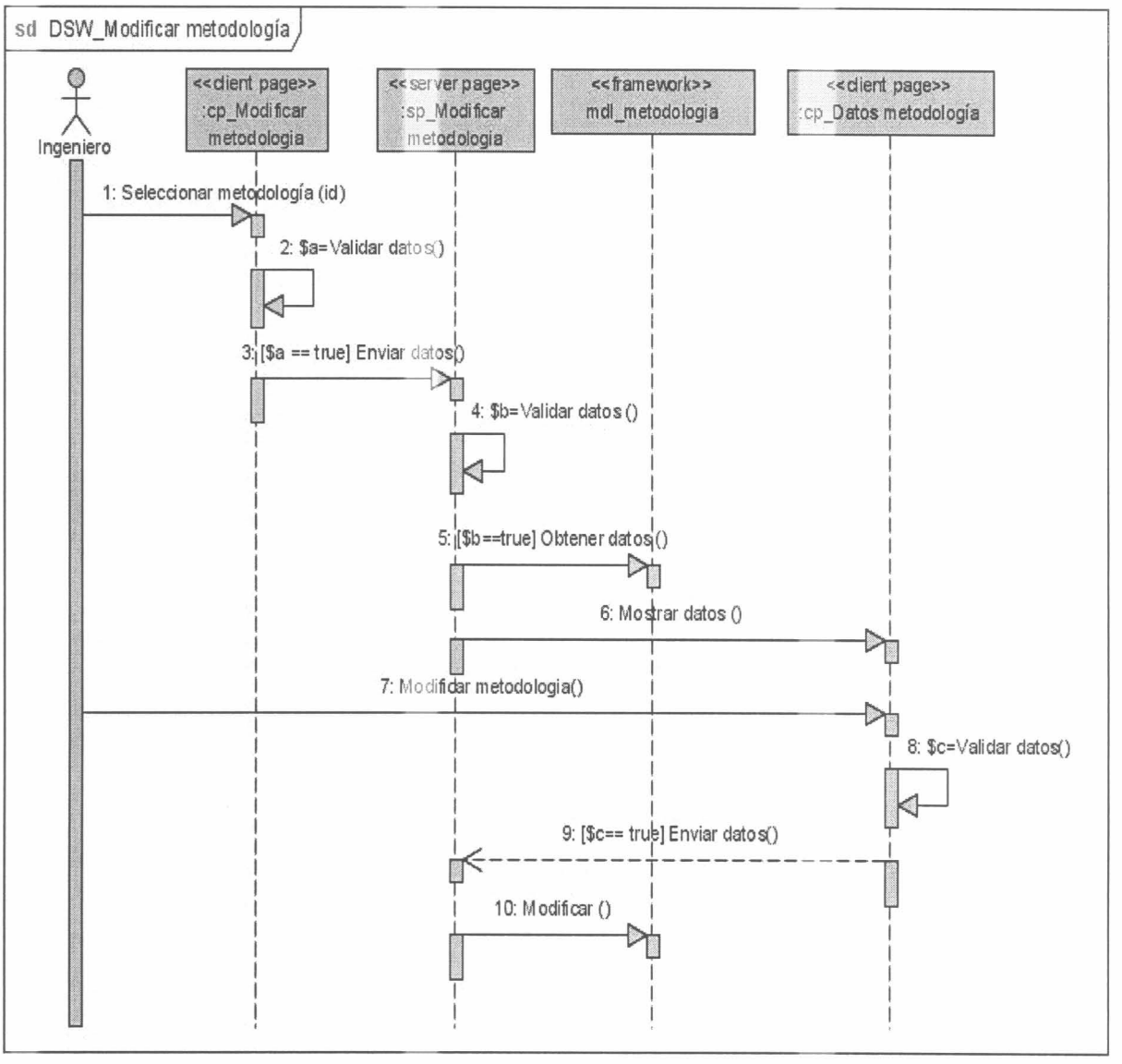


DIAGRAMA DE SECUENCIA CU-11 (Exportar metodología)

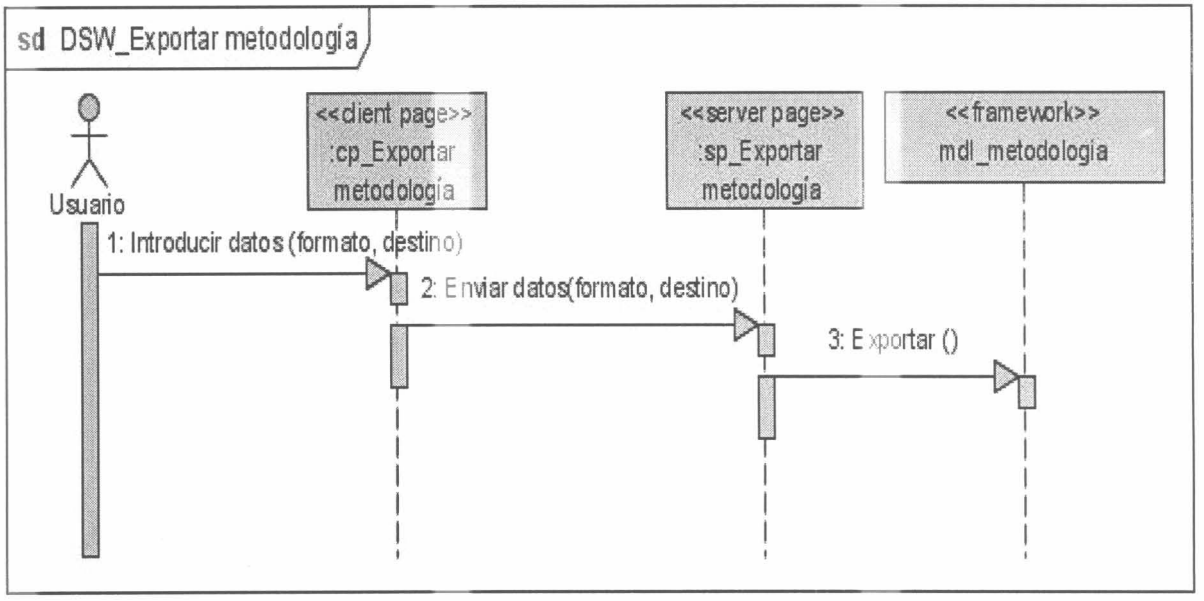


DIAGRAMA DE SECUENCIA CU-9 (Consultar metodología)

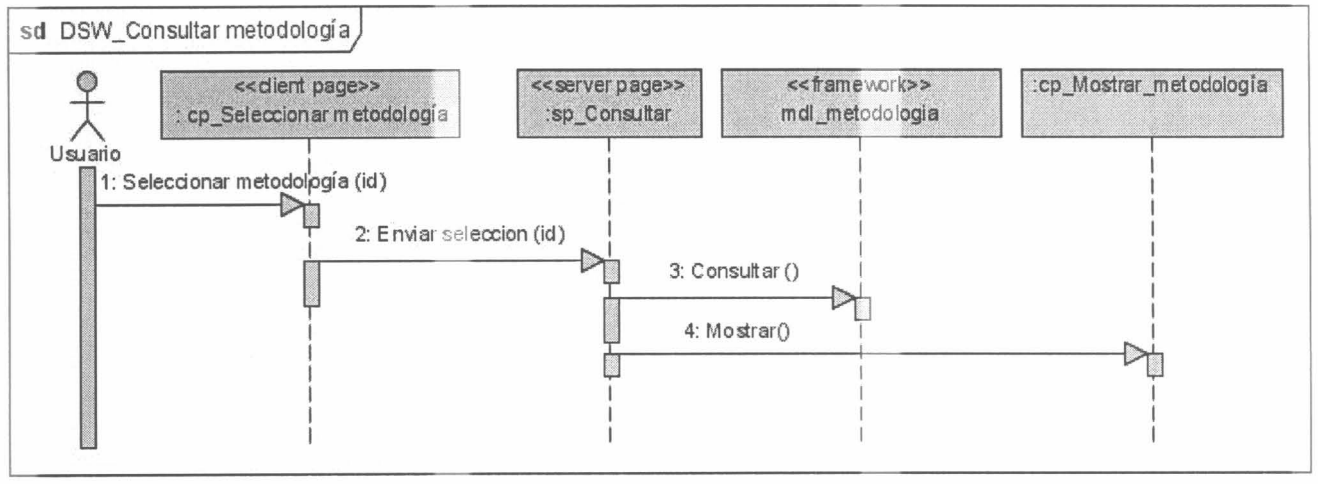
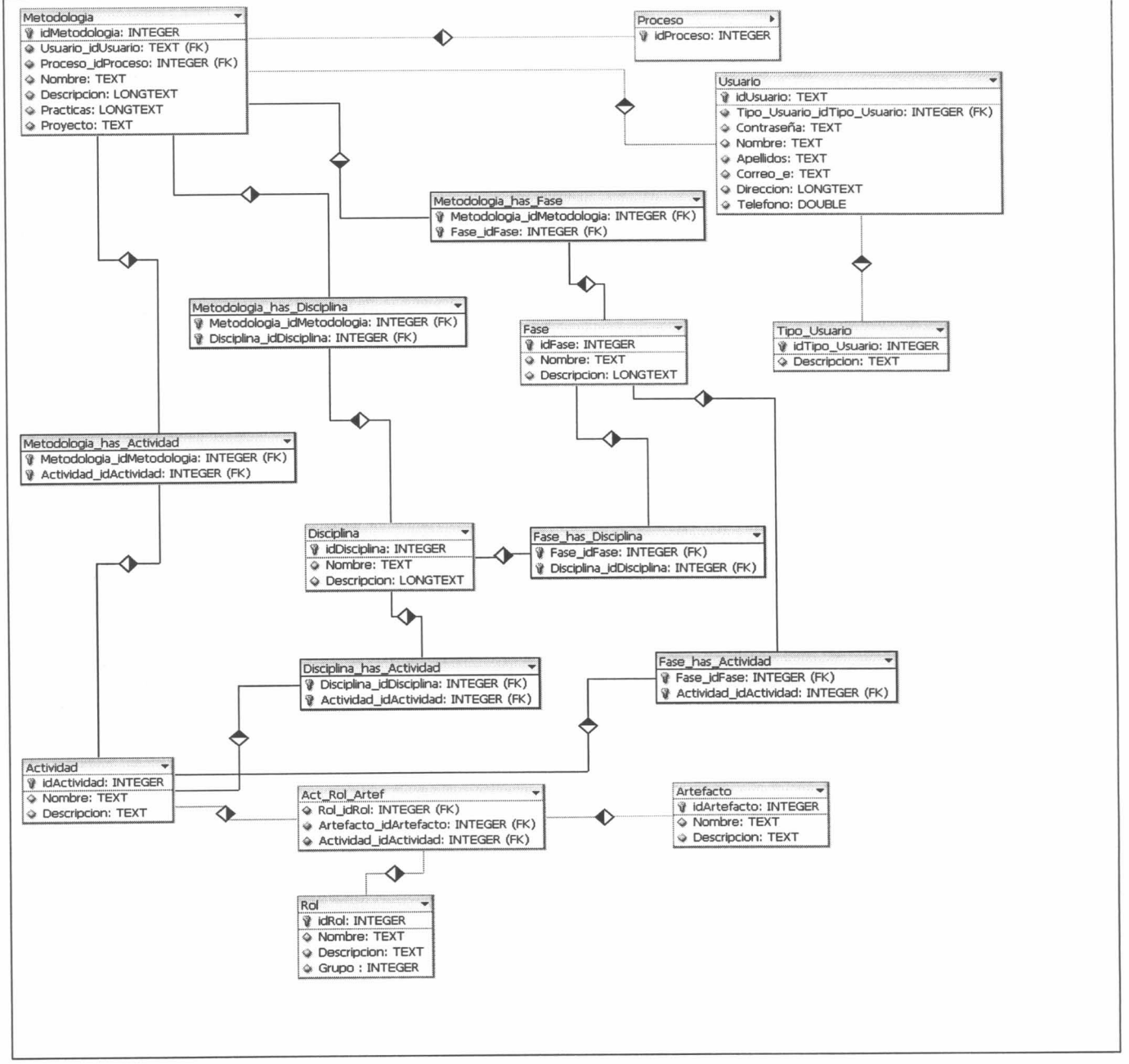


Diagrama Entidad Relación

A continuación se representa el Diagrama Entidad Relación del sistema, así como la descripción de cada una de las tablas que lo componen.

DIAGRAMA ENTIDAD RELACIÓN



Nombre: Usuario		
Descripción: Almacena todos los datos referentes a los usuarios que son creados en el sistema.		
Atributo	**Tipo**	**Descripción**
IdUsuario	Integer	Identificador del usuario
Nombre	text	Nombre del usuario
Apellidos	text	Apellidos del usuario
Contraseña	double	Contraseña para entrar al sistema
Correo_e	text	Correspondiente a la dirección de correo electrónico
Direccion	Longtext	Dirección del usuario
Telefono	double	Teléfono del usuario

Nombre: Metodologia		
Descripción: Almacena todos los datos referentes a las metodologías que son creadas en el sistema.		
Atributo	**Tipo**	**Descripción**
IdMetodologia	Integer	Identificador de la metodología
Nombre	text	Nombre de la metodología
Descripcion	longtext	Descripción breve de las características que posee la metodología, sobre todo las

		características del sistema que se desea modelar bajo los principios de esta.
Practicas	longtext	Se especificarán las prácticas o principios que el Ingeniero considere que ayudarán al mejor desarrollo del software.

Nombre: Procesos		
Descripción: Almacena todos los datos referentes a los procesos que estarán disponibles en el sistema.		
Atributo	**Tipo**	**Descripción**
IdProceso	Integer	Identificador del proceso
Nombre	text	Nombre del proceso por el cual se regirá el desarrollo del software.

Nombre: Fase		
Descripción: Almacena todos los datos referentes a las fases que son creadas en el sistema.		
Atributo	**Tipo**	**Descripción**
IdFase	Integer	Identificador de la fase
Nombre	text	Nombre de la fase
Descripcion	longtext	Descripción de las principales características de la fase.

Nombre: Disciplina		
Descripción: Almacena todos los datos referentes a las disciplinas que son creadas en el sistema.		
Atributo	**Tipo**	**Descripción**
IdDisciplina	Integer	Identificador de la disciplina
Nombre	text	Nombre de la disciplina
Descripcion	longtext	Descripción de lo que englobará la disciplina a almacenar.

Nombre: Actividad		
Descripción: Almacena todos los datos referentes a las actividades que son creadas en el sistema.		
Atributo	**Tipo**	**Descripción**
IdActividad	Integer	Identificador de la actividad
Nombre	text	Nombre de la actividad
Descripcion	longtext	Descripción de lo que se realizará durante dicha actividad.

Nombre: Rol
Descripción: Almacena todos los datos referentes a los roles que son creadas en el sistema.

Atributo	Tipo	Descripción
IdRol	Integer	Identificador del rol
Nombre	text	Nombre del rol
Descripcion	longtext	Descripción de la(s) responsabilidades del rol.

Nombre: Artefacto		
Descripción: Almacena todos los datos referentes a las metodologías que son creadas en el sistema.		
Atributo	**Tipo**	**Descripción**
IdArtefacto	Integer	Identificador de artefacto
Nombre	text	Nombre del artefacto
Descripcion	longtext	Descripción breve de lo que representa el artefacto.

Conclusiones

El proceso de desarrollo de software representa para los ingenieros un desafío en muchos aspectos, entre estos, lo relacionado con la selección de la metodología a utilizar, dado fundamentalmente por su diversidad. Al concluir el desarrollo de este trabajo se llegó a las siguientes conclusiones:

Se realizó un estudio del estado del arte sobre los procesos y metodologías para desarrollo de software, en el cual se determinaron los principales elementos que las conforman, este estudio nos permitió ubicar a los lectores en este contexto.

Se realizó un estudio de las herramientas análogas a la solución propuesta, lo que permitió hacer una caracterización de las mismas, mediante la documentación de sus principales características y funcionalidades. Además se investigó todo lo referente a la arquitectura de RINDE, el cual permitió realizar el análisis y diseño del sistema propuesto, definiendo las principales clases que lo componen puesto que la herramienta a desarrollar será integrada a esta plataforma.

Se entrevistó a los desarrolladores de RINDE para determinar el framework a utilizar en el desarrollo de la herramienta, en este caso el CodeIgniter, arrojando ser el más óptimo en la creación de sistemas con características similares al nuestro.

Lo anteriormente expuesto, la creación del modelo de dominio, el levantamiento de requisitos tanto funcionales como no funcionales, y el resto de los artefactos enmarcados en la fase de Elaboración permitió la confección del análisis y diseño del Habilitador Metodológico.

Recomendaciones

Con vistas al desarrollo futuro de este proyecto se recomienda:

- Motivar el uso del Habilitador Metodológico en el proceso de desarrollo de software
- Continuar desarrollando el sistema propuesto a partir del análisis y diseño realizado.

Referencias bibliográficas

Jacobson, I. B. (1999). *El Proceso Unificado de Desarrollo de Software.*

Pressman, R. (1997). *Un enfoque práctico.*

Reynoso, C. (2004). *Métodos Heterodoxos en Desarrollo de Software.*

The Philosophy of Scrum. http://www.controlchaos.com/old-site/philo.htm (Accessed June 1, 2008).

Pressman, Roger . 1997. *Ingeniería de Software. Un enfoque Práctico.*

Booch, Grady, James Rumbaugh, and Ivar Jacobson. 1999. *El Proceso Unificado de Desarrollo de Software.*

Canós, José Hilarios, Patricio Letelier, and María del Carmen Panadés. 2004. Metodologías Ágiles en el Desarrollo del Software.

Fernández Escribano, Gerardo. 2002. Introducción a Extreme Programing.

Highsmith, Jim. 1997. Messy, Exciting, and Anxiety-Ridden: Adaptive Software Development. http://www.jimhighsmith.com/articles/messy.htm (Accessed June 1, 2008).

IBM. 2006. Eclipse Process Framework (EPF) « Alec the Geek. http://alecthegeek.wordpress.com/2006/10/19/eclipse-process-framework/ (Accessed June 1, 2008).

IBM. 2007. IBM Rational Method Composer V7.2 strengthens and expands IBM IT process management platform in sup. http://www-01.ibm.com/cgi-bin/common/ssi/ssialias?infotype=an&subtype=ca&htmlfid=897/ENUS207-222 (Accessed June 1, 2008).

Marcelo Hernán, Shenone . 2004. Diseño de una metodología ágil de desarrollo de software.

Taber, Cara, and Martin Fowler. 2001. Planning and Running an XP Iteration. http://martinfowler.com/articles/planningXpIteration.html (Accessed June 1, 2008).

Bibliografía

All About Agile: How To Implement Scrum In 10 Easy Steps. http://www.agile-software-development.com/2007/09/how-to-implement-scrum-in-10-easy-steps.html (Accessed June 1, 2008).

Desarrollo en espiral - Wikipedia, la enciclopedia libre. http://es.wikipedia.org/wiki/Desarrollo_en_espiral (Accessed June 1, 2008).

2006. Eclipse Process Framework (EPF) « Alec the Geek. http://alecthegeek.wordpress.com/2006/10/19/eclipse-process-framework/ (Accessed June 1, 2008).

El Modelo Espiral. http://www.ldc.usb.ve/~vtheok/cursos/ci3711/apuntes/99-01-14/Info/Modelo%20Espiral.htm (Accessed June 1, 2008).

Ingeniería del Software - Monografias.com. http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml (Accessed June 1, 2008).

Portal for Agile Methodologies and Practices. http://agile.csc.ncsu.edu/ (Accessed June 1, 2008).

2007. Programación Extrema. http://www.programacionextrema.org/ (Accessed June 1, 2008).

2004. The Enterprise Single Sign-On System. http://book.itzero.com/read/microsoft/0505/Sams.Microsoft.BizTalk.Server.2004.Unleashed.Nov.2004.eBook-LiB_html/0672325985/ch08lev1sec3.html (Accessed June 1, 2008).

The Philosophy of Scrum. http://www.controlchaos.com/old-site/philo.htm (Accessed June 1, 2008).

2004. XP Agile Universe - The premier conference on Agile software development processes.Frequently Asked Questions. http://www.xpuniverse.com/faq (Accessed June 1, 2008).

Pressman, Roger . 1997. *Ingeniería de Software. Un enfoque Práctico.*

Arboleda Jiménez, Hugo F. 2005. Modelos de ciclos de vida en desarrollo de software. http://www.acis.org.co/index.php?id=551 (Accessed June 1, 2008).

Booch, Grady, James Rumbaugh, and Ivar Jacobson. 1999. *El Proceso Unificado de Desarrollo de Software.*

Cáceres, Paloma , and Esperanza Marcos. 2003. Hacia un proceso metodológico dirigido por modelos para el desarrollo ágil de sistemas.

Canós, José Hilarios, Patricio Letelier, and María del Carmen Panadés. 2004. Metodologías Ágiles en el Desarrollo del Software.

Colado, César , and Alfonso Franco. 2003. Métricas de seguridad: una visión actualizada. *SIC* 57.

Díaz, María Irma . 2006. La incertidumbre y la ingeniería de software.

Fernández Escribano, Gerardo. 2002. Introducción a Extreme Programing.

GACITÚA BUSTOS, RICARDO A. 2003. Métodos de desarrollo de Software: El desafío pendiente de la estandarización. 12.

Highsmith, Jim. 1997. Messy, Exciting, and Anxiety-Ridden: Adaptive Software Development. http://www.jimhighsmith.com/articles/messy.htm (Accessed June 1, 2008).

IBM. 2007. IBM Rational Method Composer V7.2 strengthens and expands IBM IT process management platform in sup. http://www-01.ibm.com/cgi-bin/common/ssi/ssialias?infotype=an&subtype=ca&htmlfid=897/ENUS207-222 (Accessed June 1, 2008).

Krebs, Joe. 2005. RUP in the dialogue with Scrum. http://www-128.ibm.com/developerworks/rational/library/feb05/krebs/ (Accessed June 1, 2008).

Manuel Calero Solis. 2003. Una explicación de la programación extrema (XP).

Marcelo Hernán, Shenone . 2004. Diseño de una metodología ágil de desarrollo de software.

Mendoza Sanchez, María A. 2004. Metodologías de Desarrollo de Software.

Millington, Don, and Jennifer Stapleton. 1995. Developing a RAD Standar. http://www.agilealliance.com/system/article/file/963/file.pdf (Accessed June 1, 2008).

Reynoso, Carlos . 2004. Métodos Heterodoxos en Desarrollo de Software.

Rising, Linda. 2001. Agile Methods: What's it All About? http://ddci.com/NewsArchive/news_vol2num9.php#Agile (Accessed June 1, 2008).

Taber, Cara, and Martin Fowler. 2001. Planning and Running an XP Iteration. http://martinfowler.com/articles/planningXpIteration.html (Accessed June 1, 2008).

Wake, William C. 2002. Métodos y Herramientas. 10. http://www.methodsandtools.com/PDF/dmt0402.pdf (Accessed June 1, 2008).

Anexo 1. Diagramas de clases de diseño

DIAGRAMA DE CLASES (Paquete Usuario)

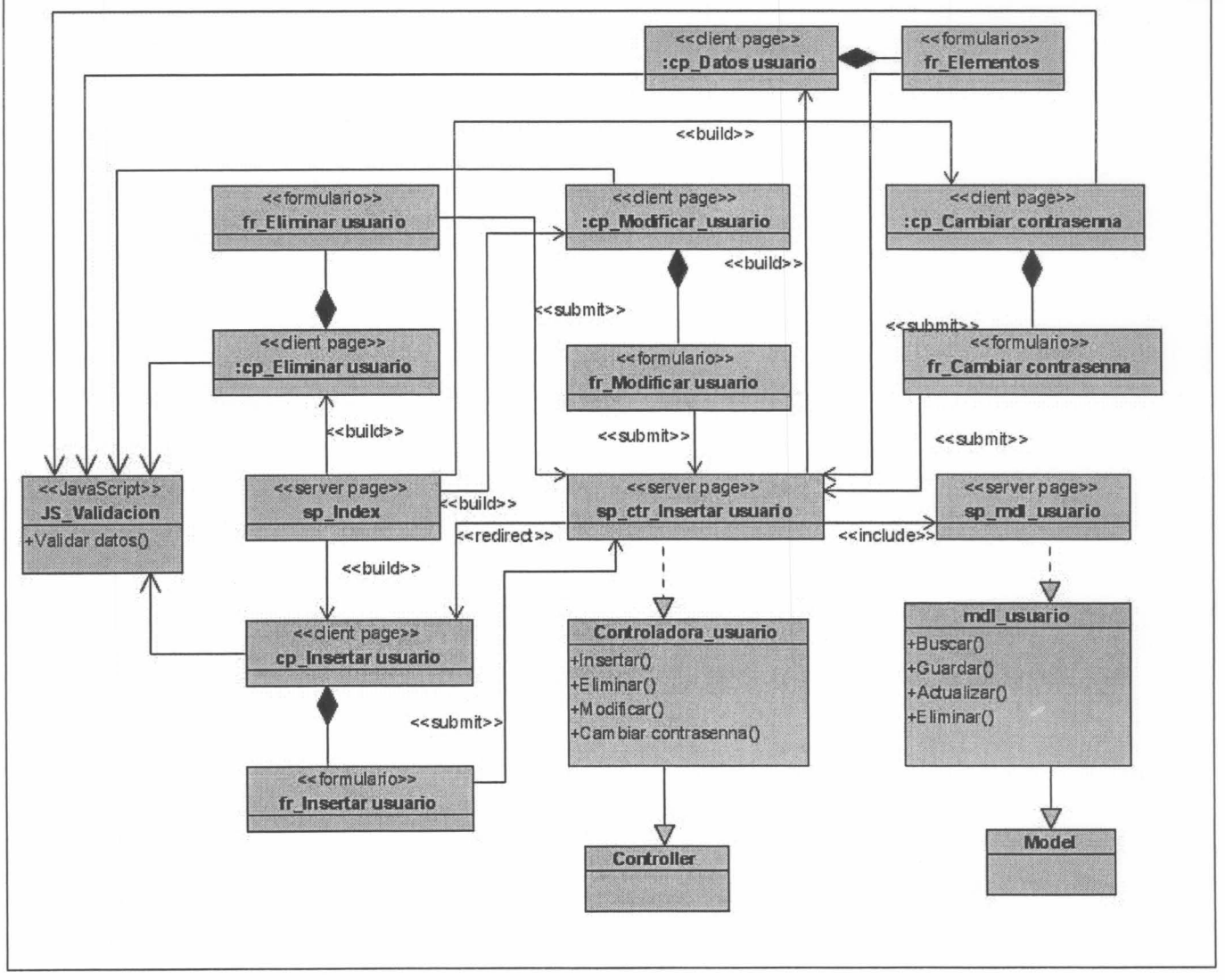


DIAGRAMA DE CLASES (Paquete Fase)

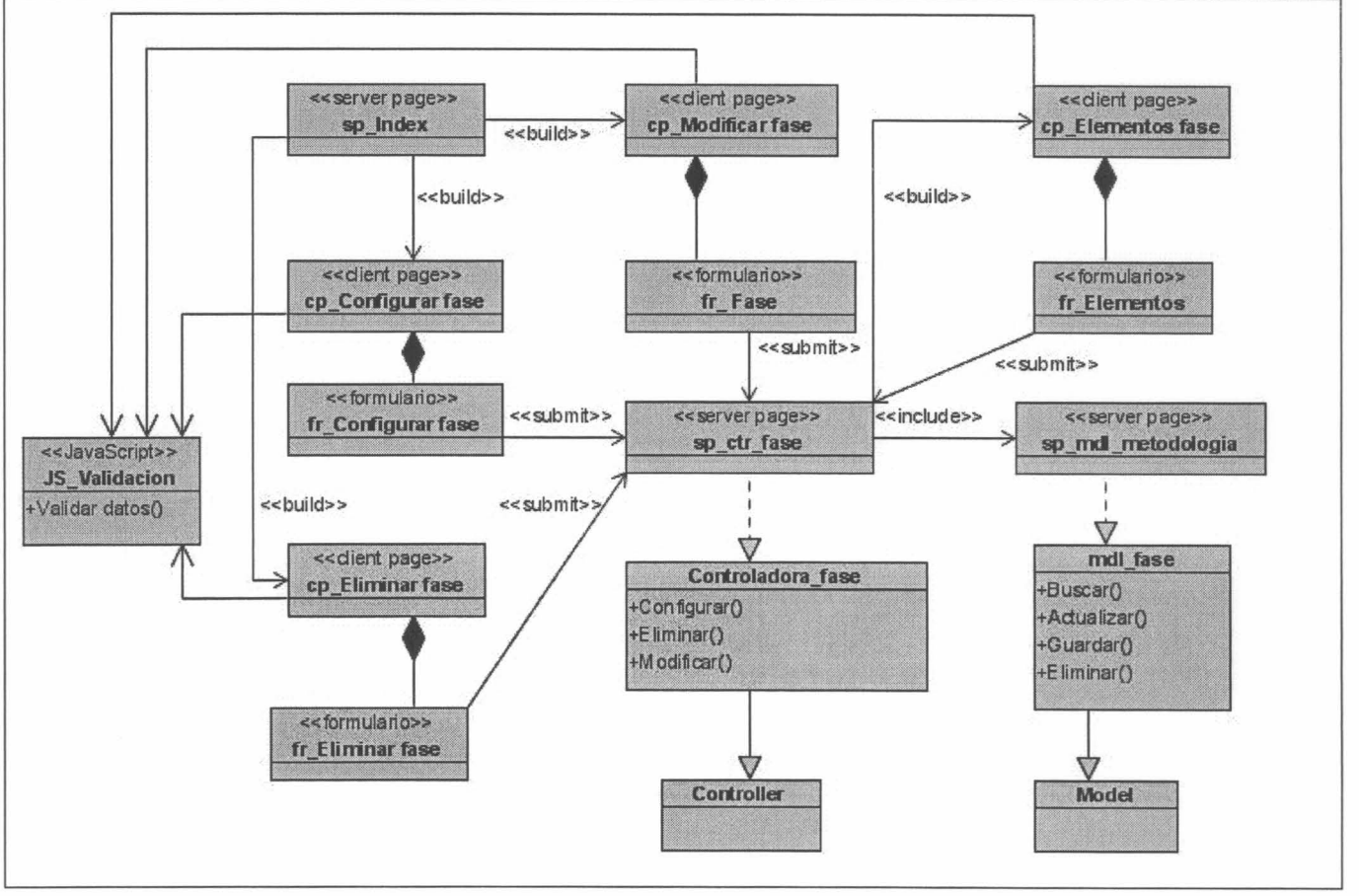


DIAGRAMA DE CLASES (Paquete Disciplina)

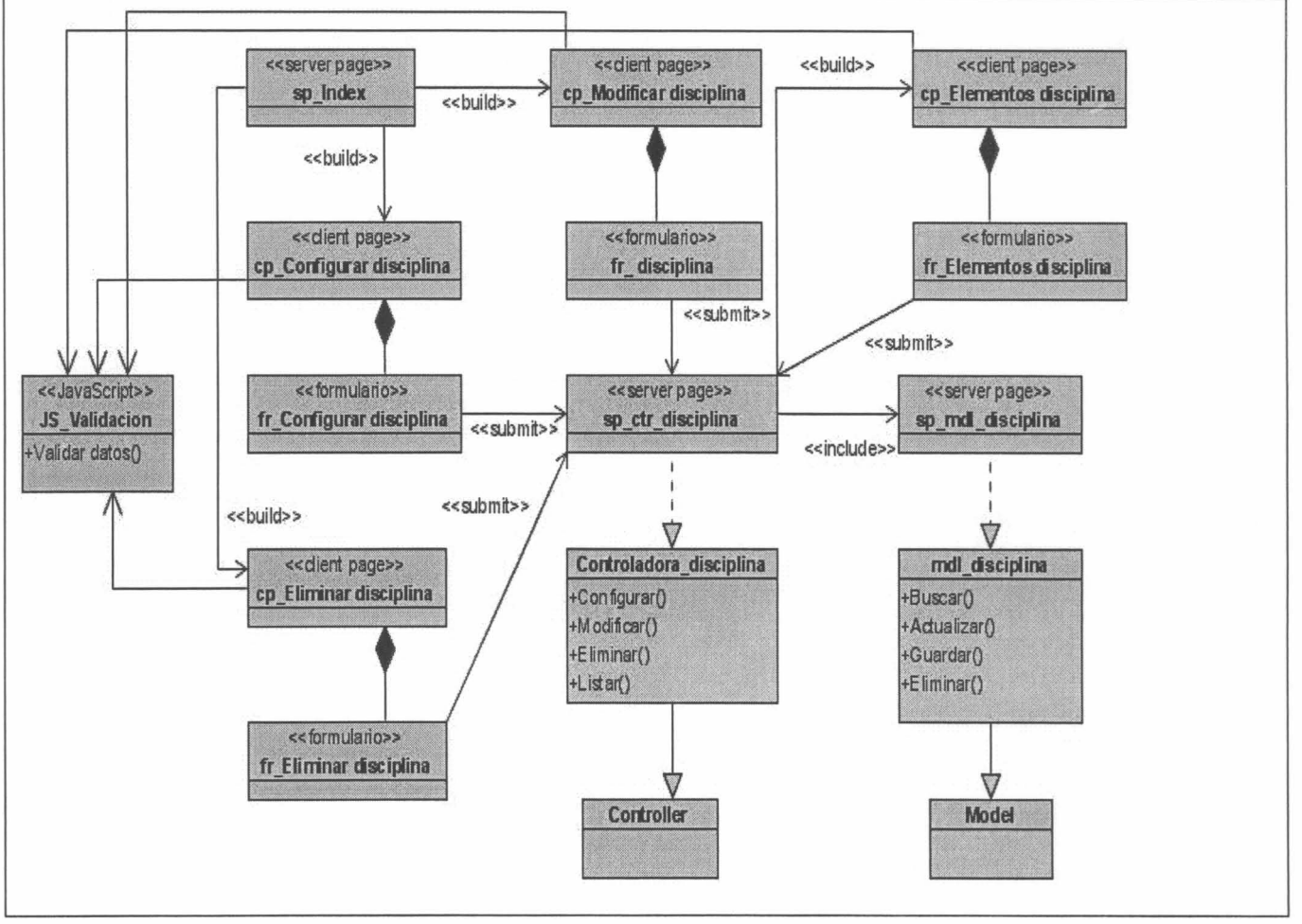


DIAGRAMA DE CLASES (Paquete Actividad)

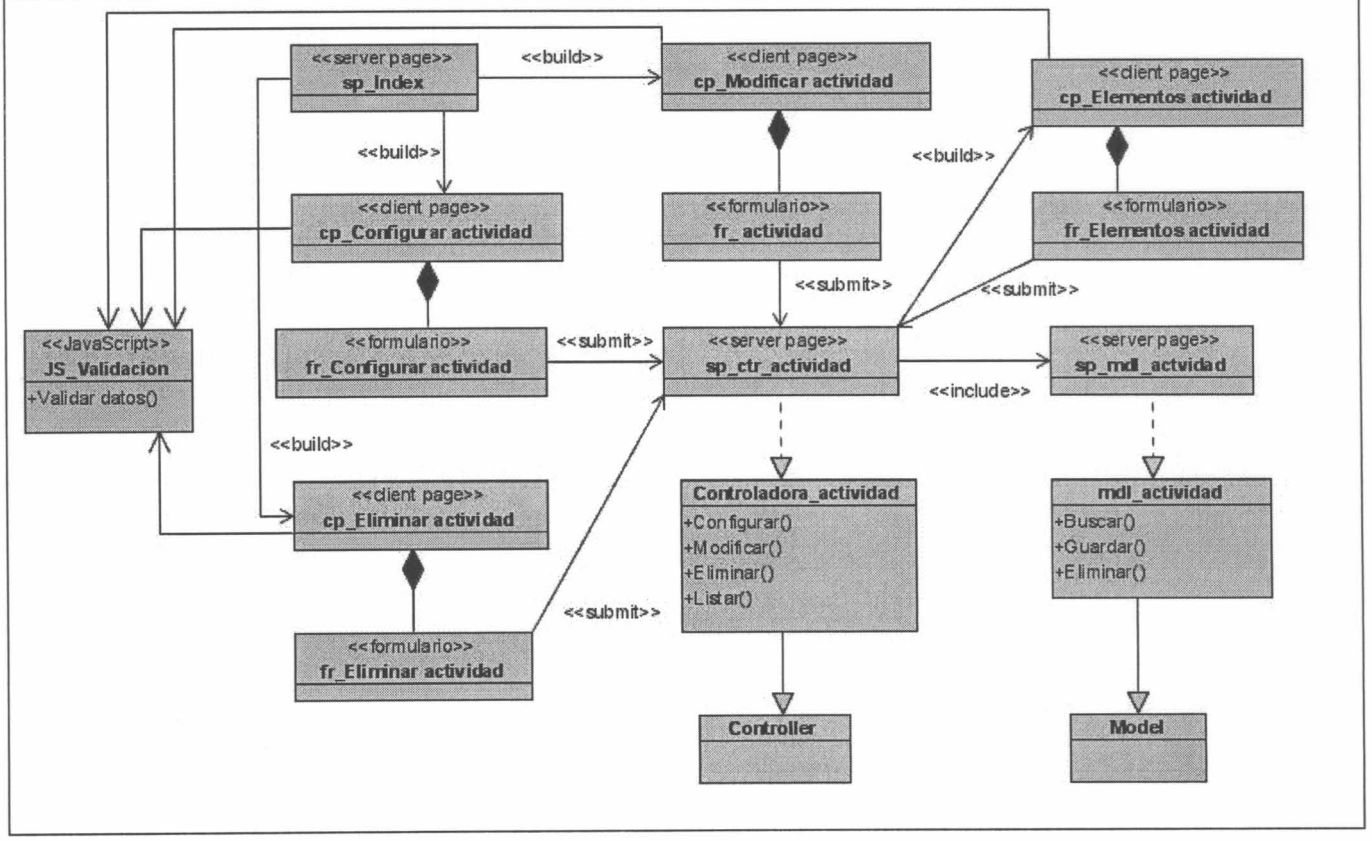


DIAGRAMA DE CLASES (Paquete Artefacto)

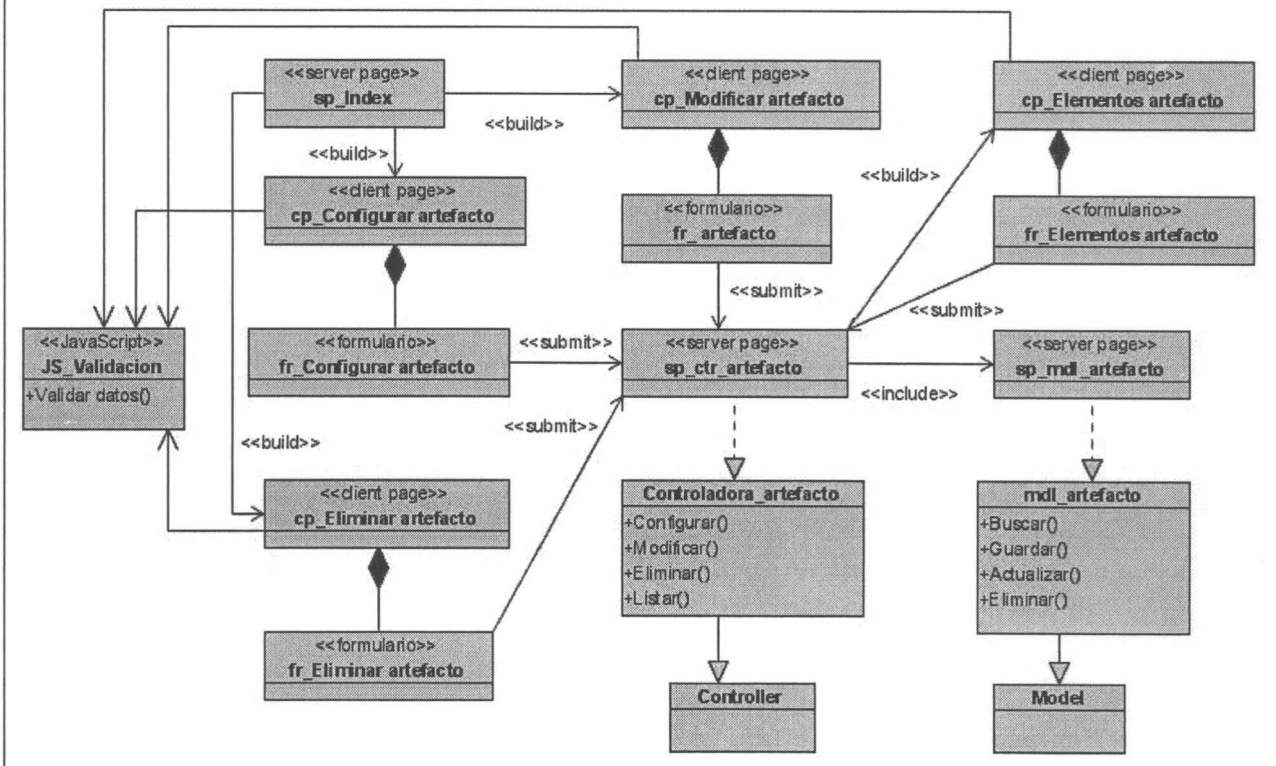


DIAGRAMA DE CLASES (Paquete Rol)

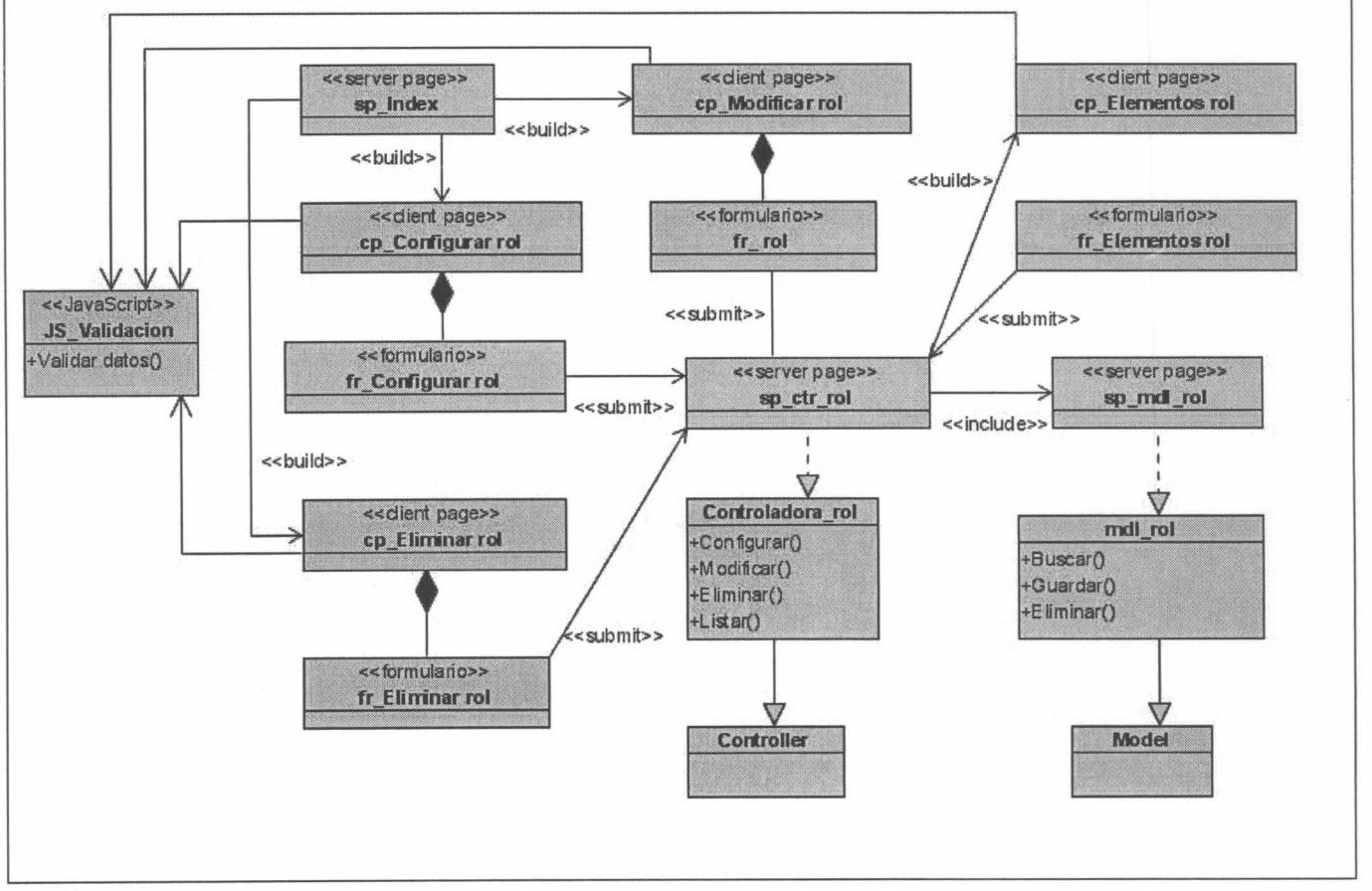


DIAGRAMA DE CLASES (Asociar Elemento)

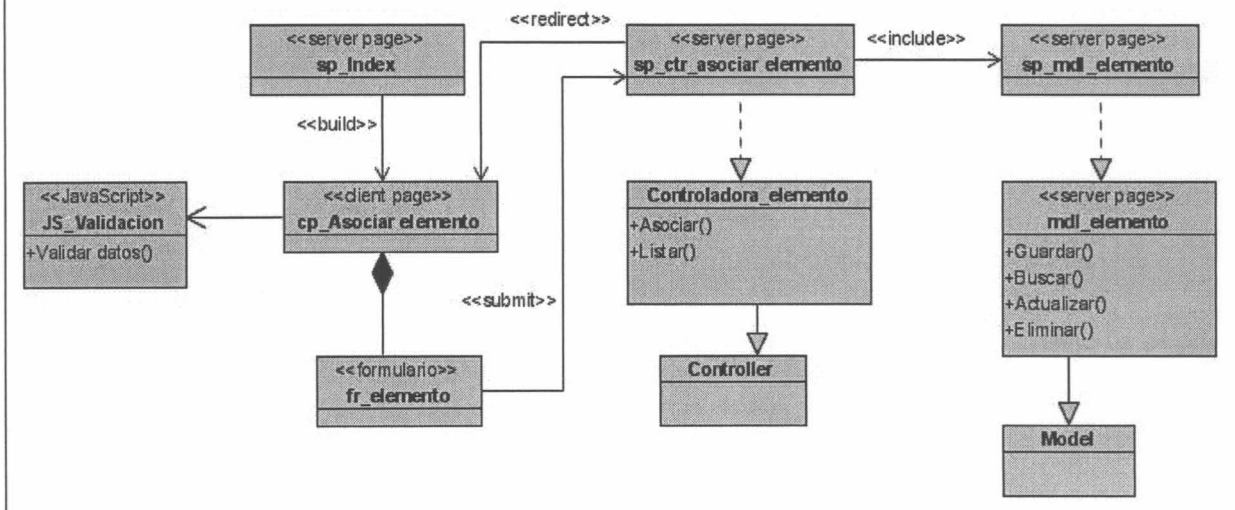


DIAGRAMA DE CLASES (Iniciar Sesión)

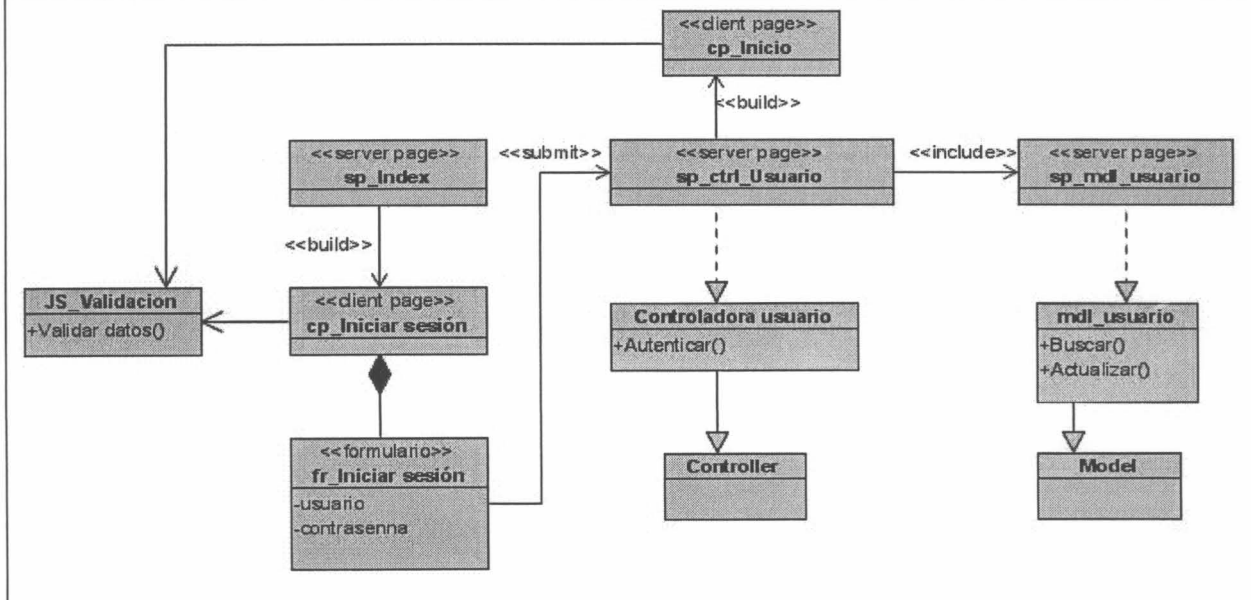


DIAGRAMA DE CLASES (Finalizar Sesión)

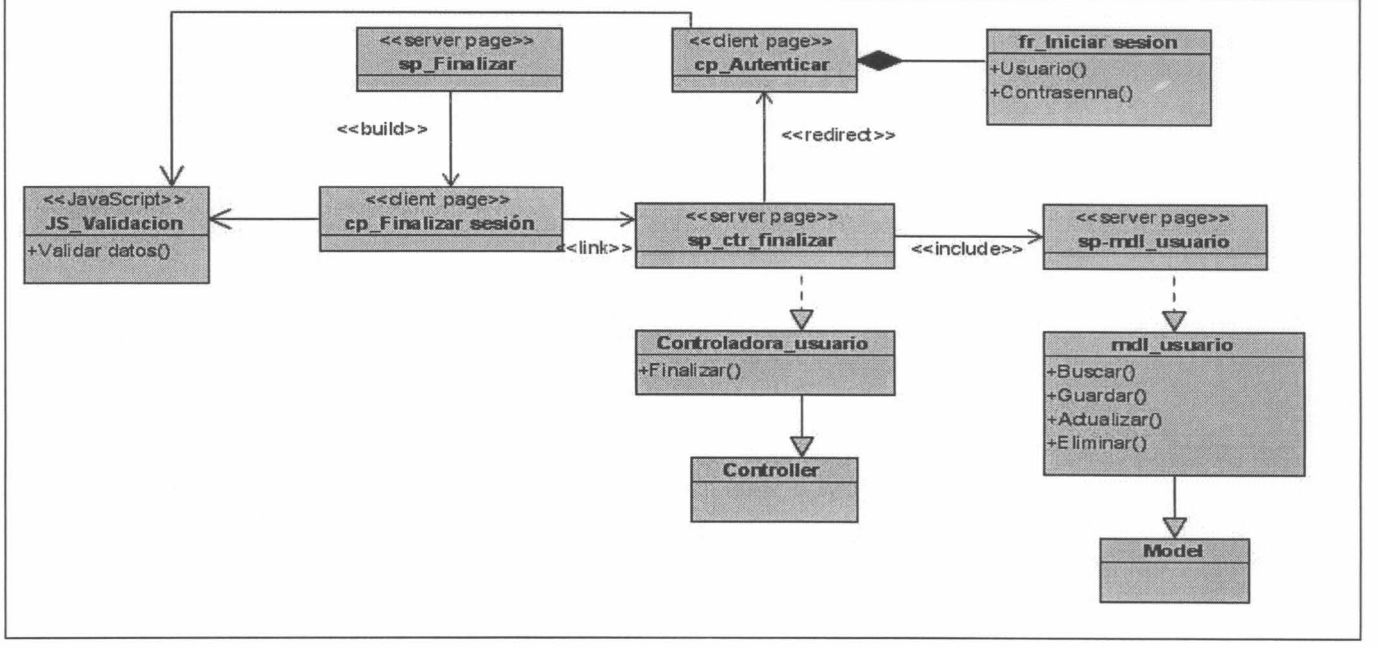
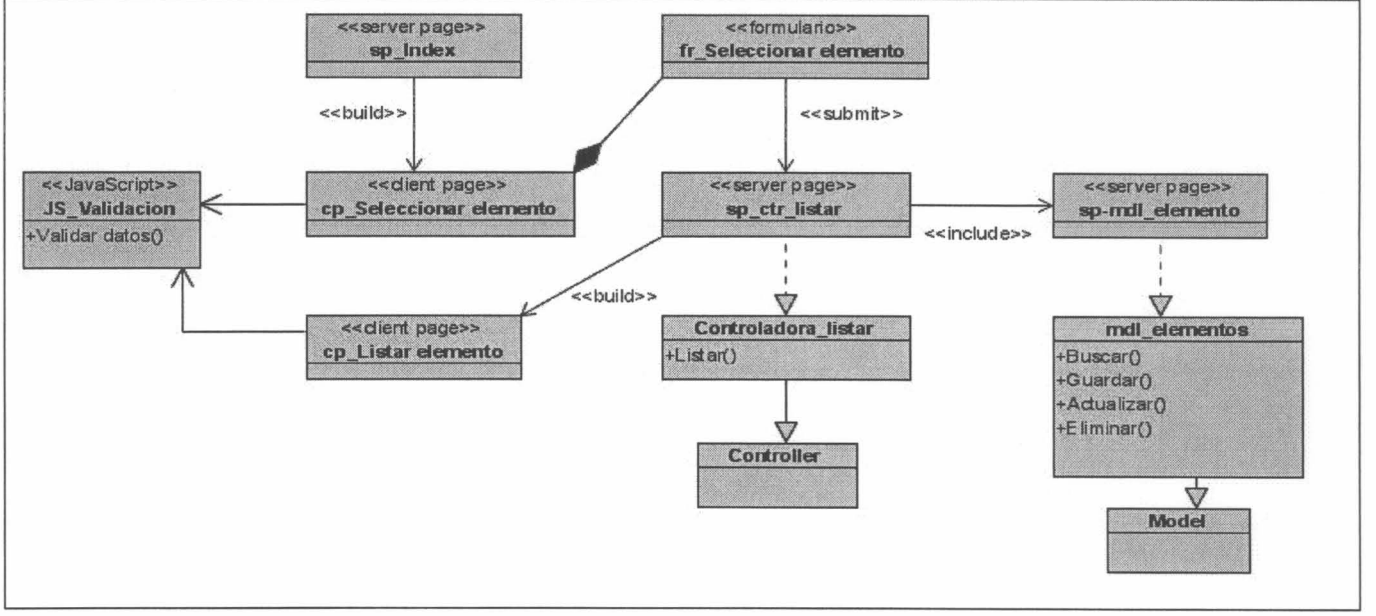


DIAGRAMA DE CLASES (Listar elemento)



Anexo 2. Diagramas de Secuencia

DIAGRAMA DE SECUENCIA CU-1 (Iniciar sesión)

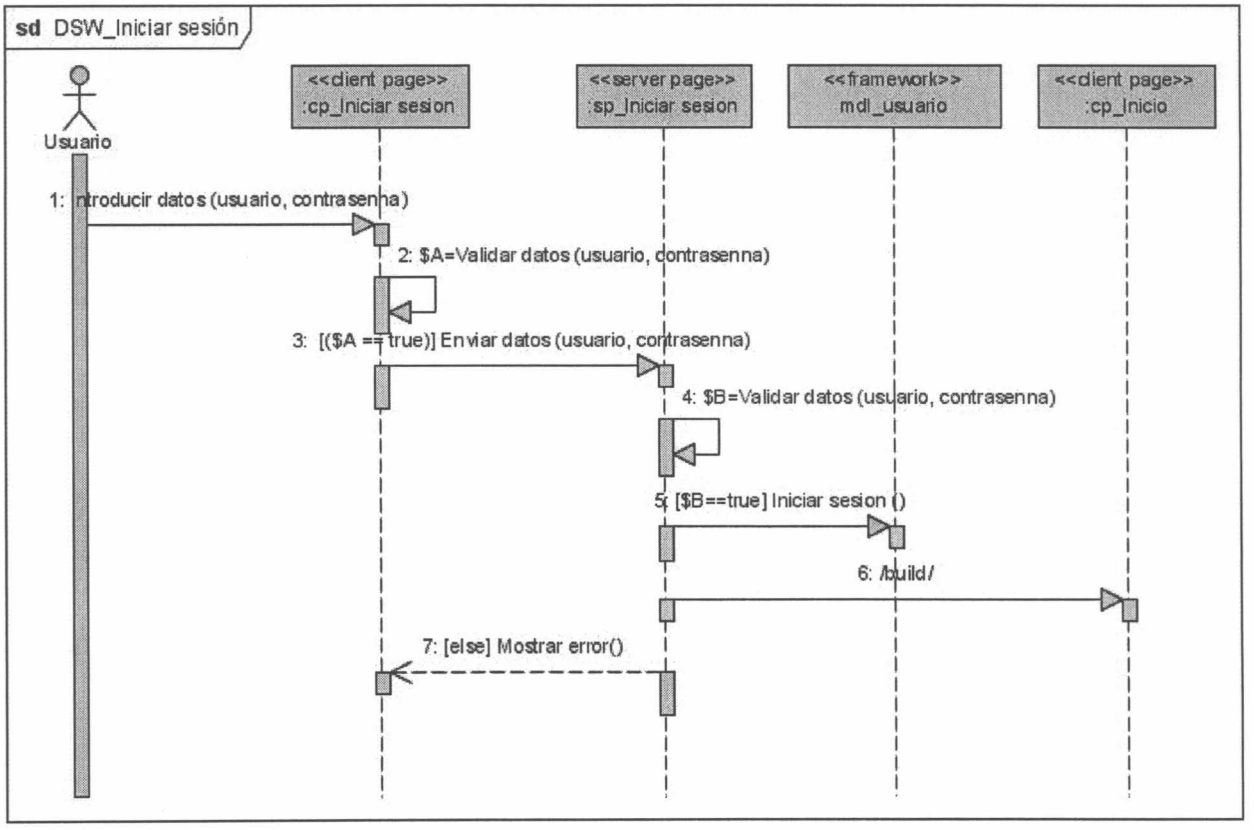


DIAGRAMA DE SECUENCIA CU-2 (Finalizar sesión)

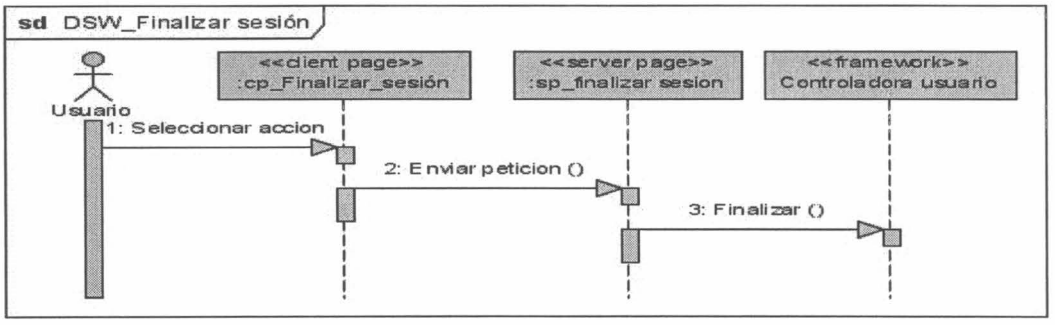


DIAGRAMA DE SECUENCIA CU-3 (Insertar usuario)

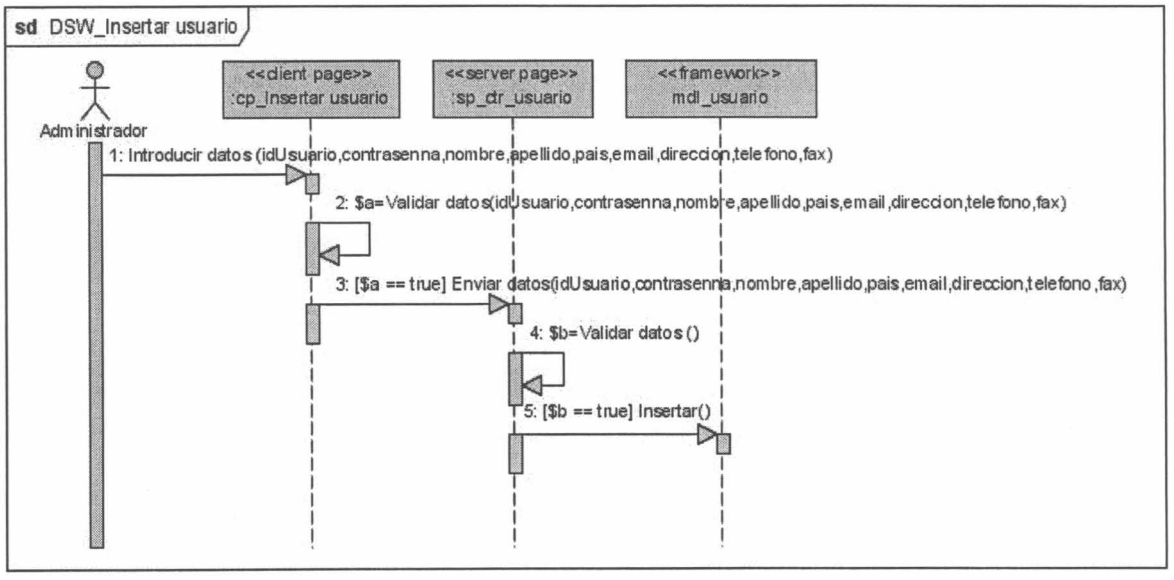


DIAGRAMA DE SECUENCIA CU-4 (Eliminar usuario)

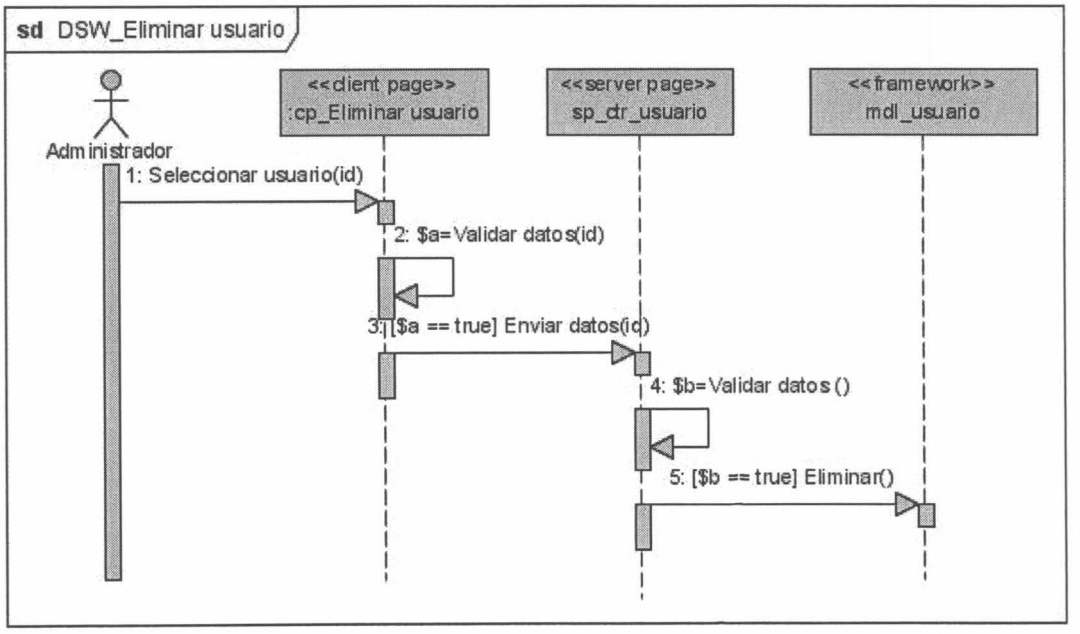


DIAGRAMA DE SECUENCIA CU-5 (Modificar usuario)

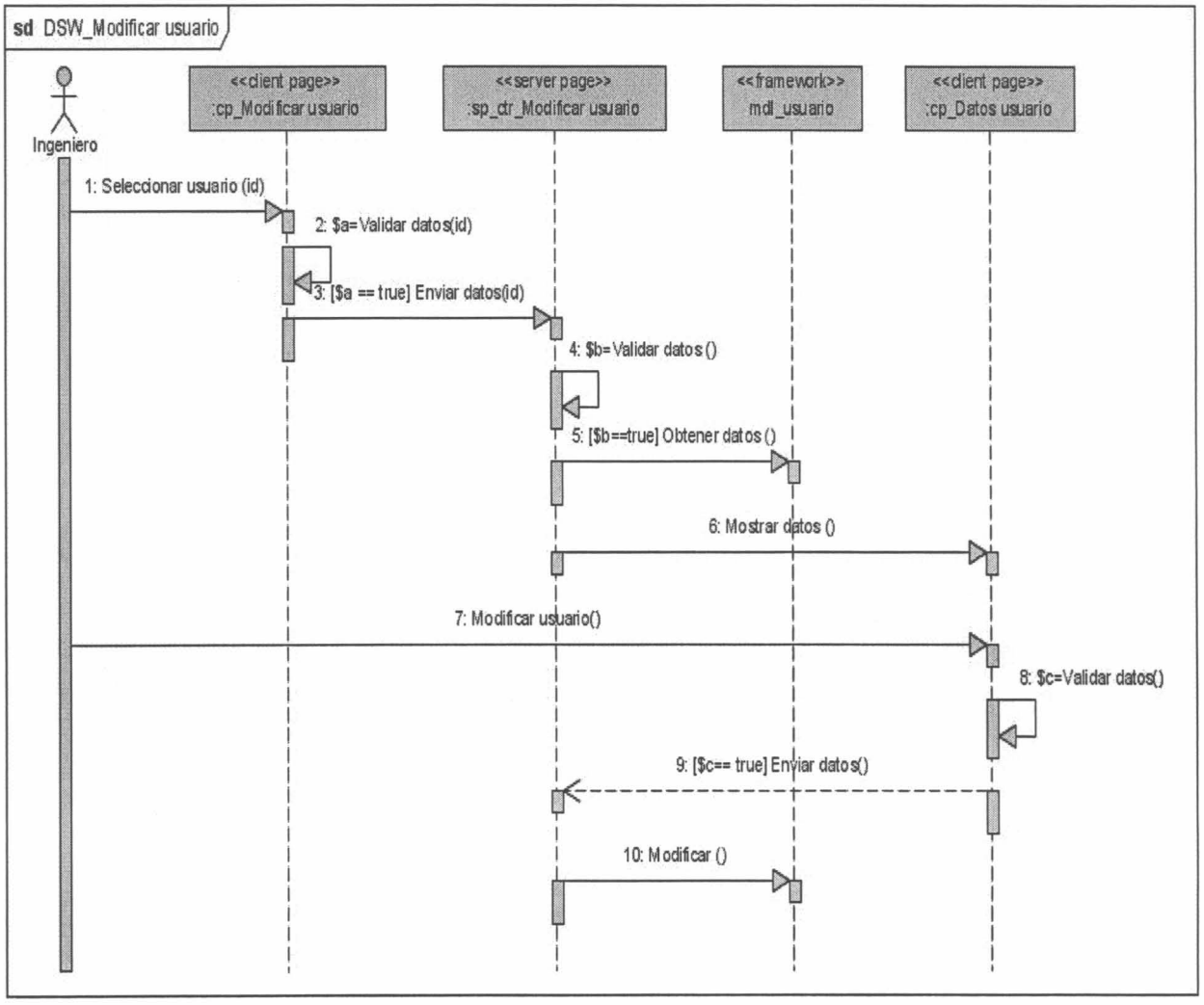


DIAGRAMA DE SECUENCIA CU-6 (Cambiar contraseña)

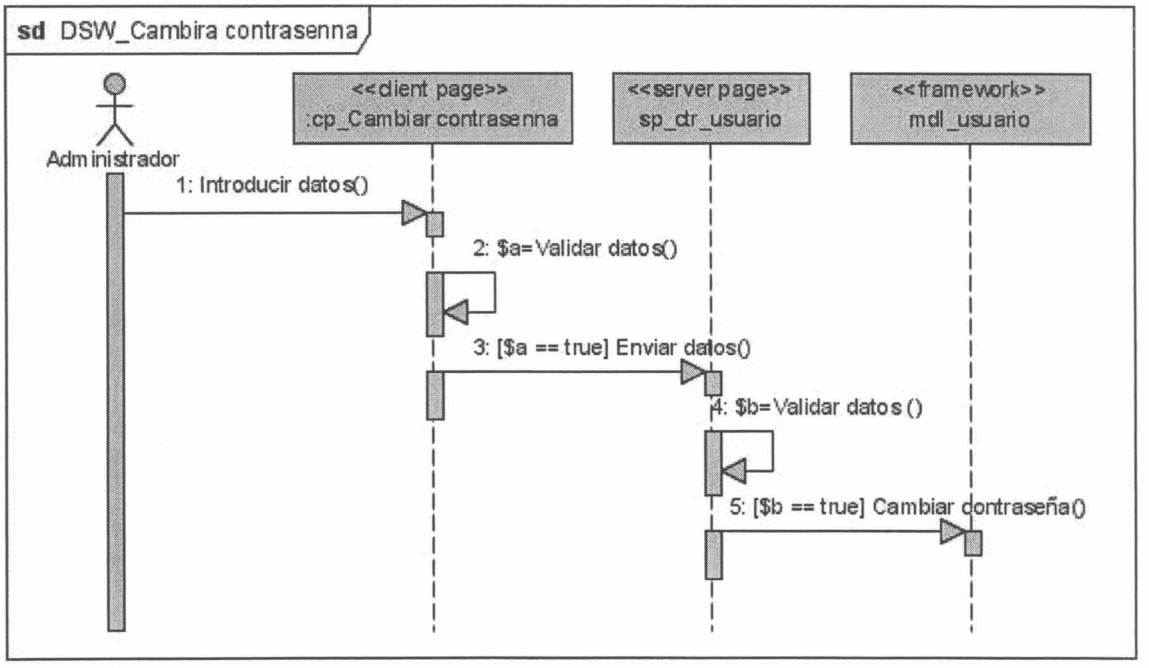


DIAGRAMA DE SECUENCIA CU-18 (Configurar actividad)

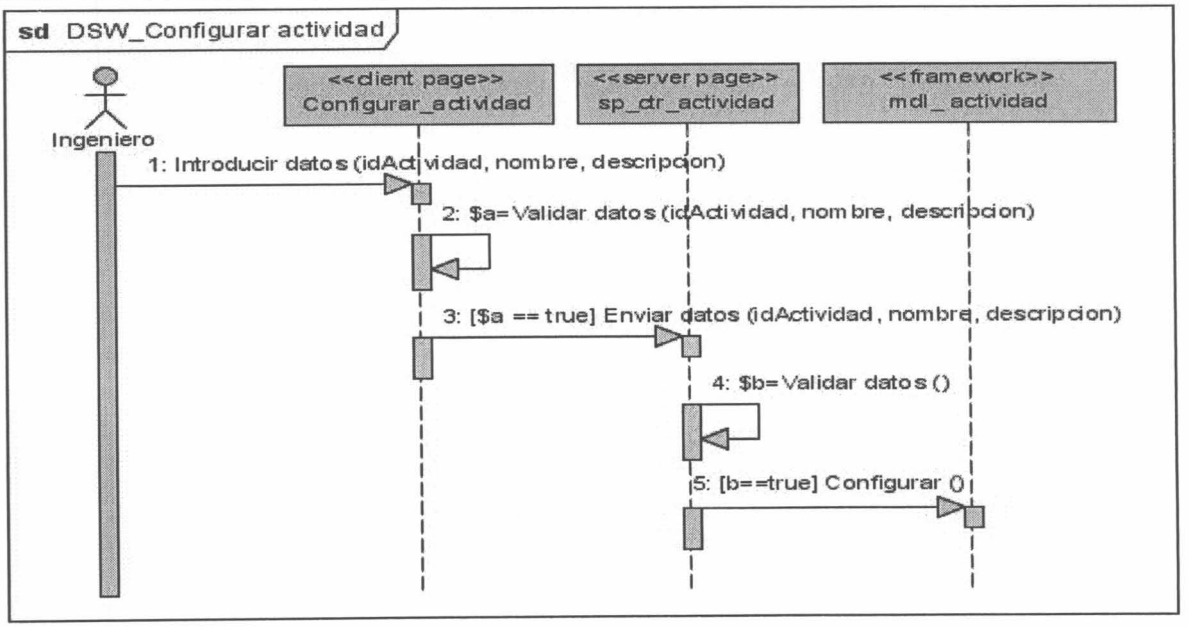


DIAGRAMA DE SECUENCIA CU-24 (Configurar artefacto)

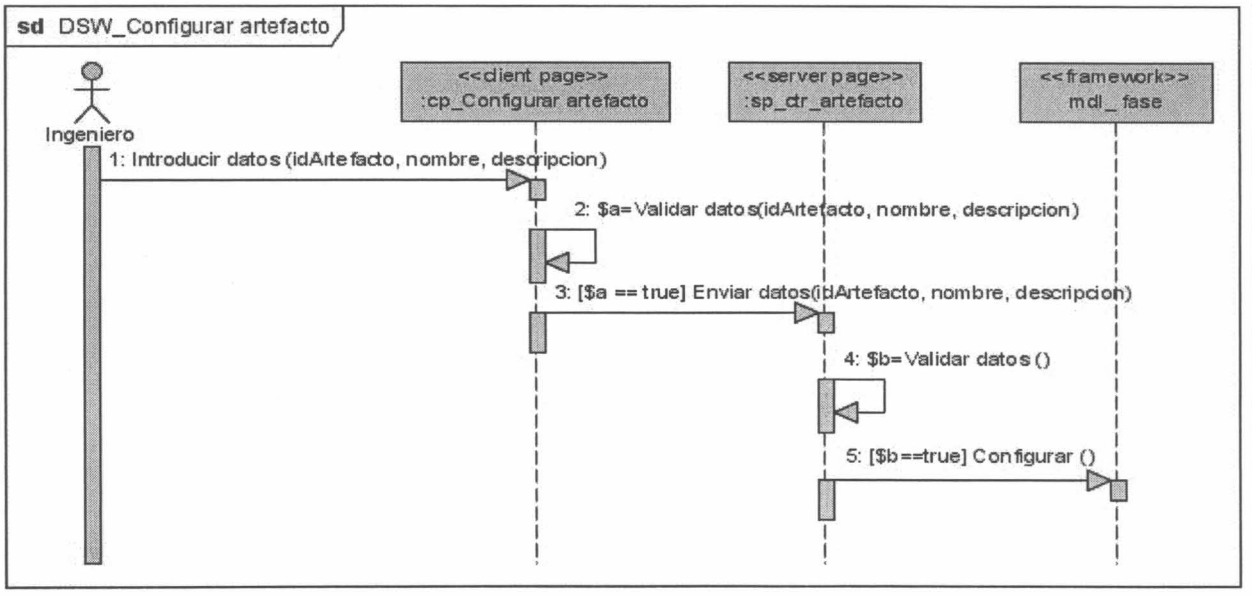


DIAGRAMA DE SECUENCIA CU-15 (Configurar disciplina)

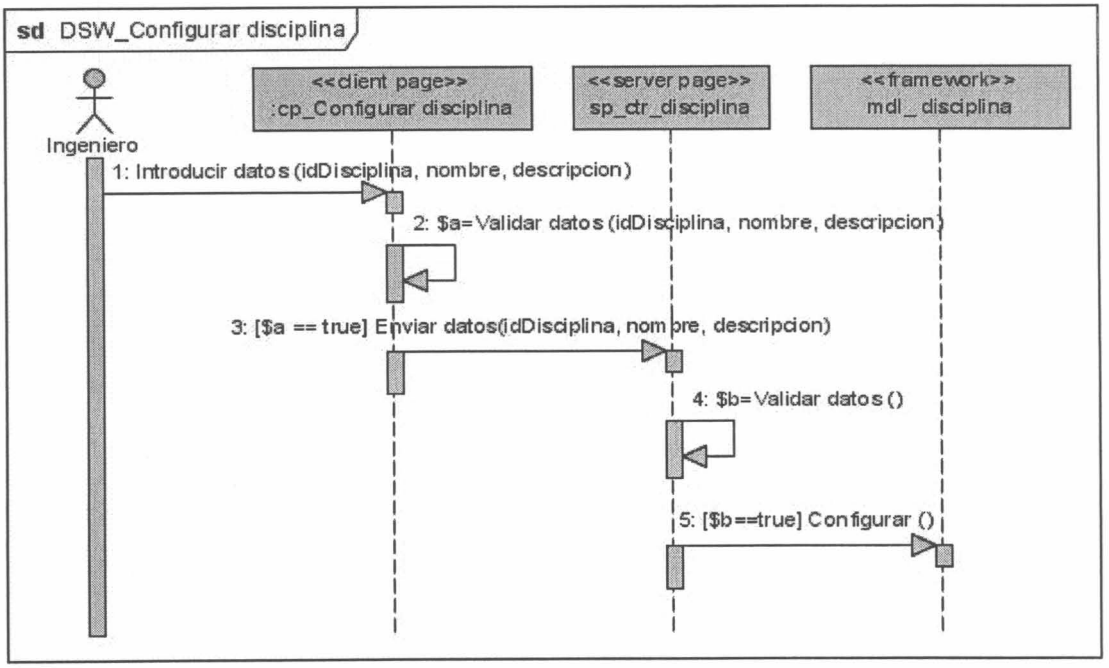


DIAGRAMA DE SECUENCIA CU-12 (Configurar fase)

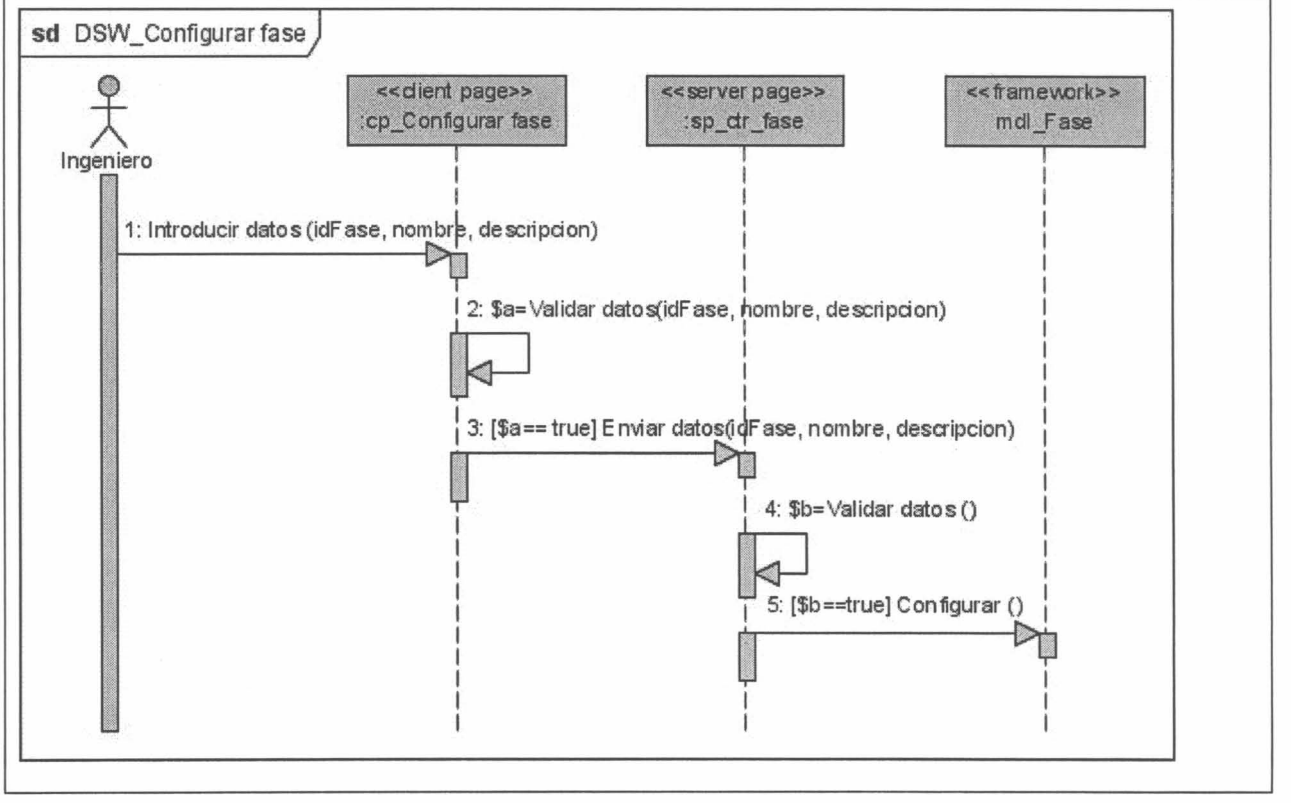


DIAGRAMA DE SECUENCIA CU-21 (Configurar rol)

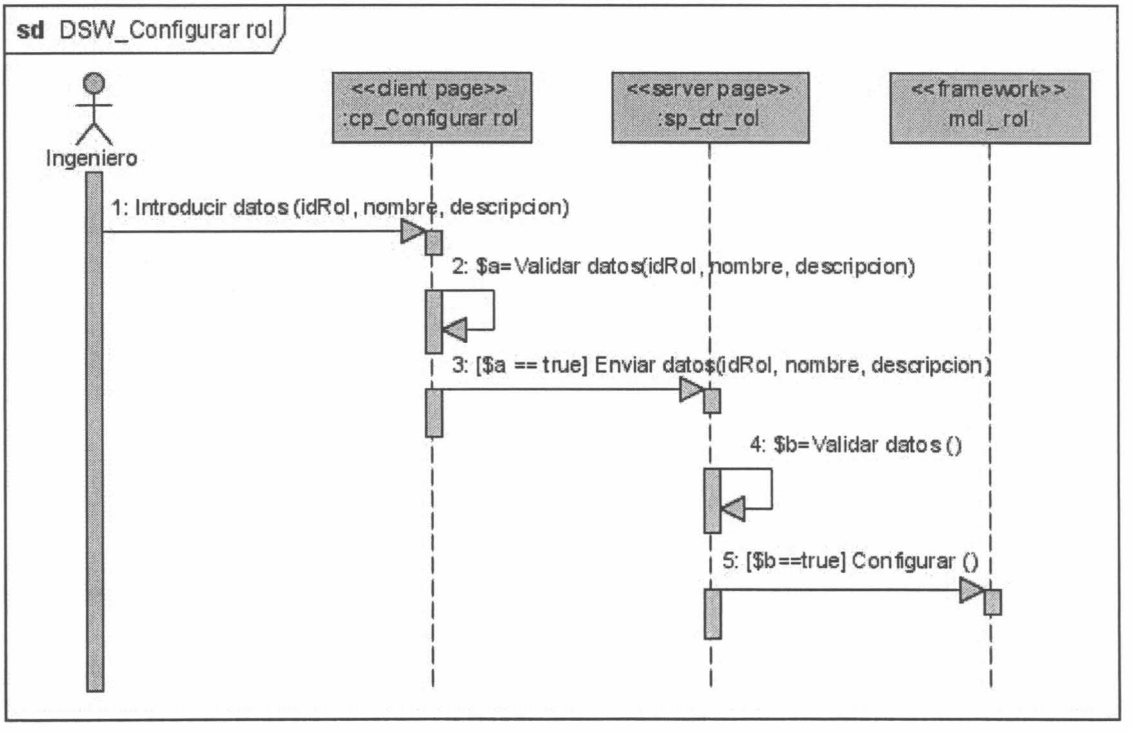


DIAGRAMA DE SECUENCIA CU-20 (Eliminar actividad)

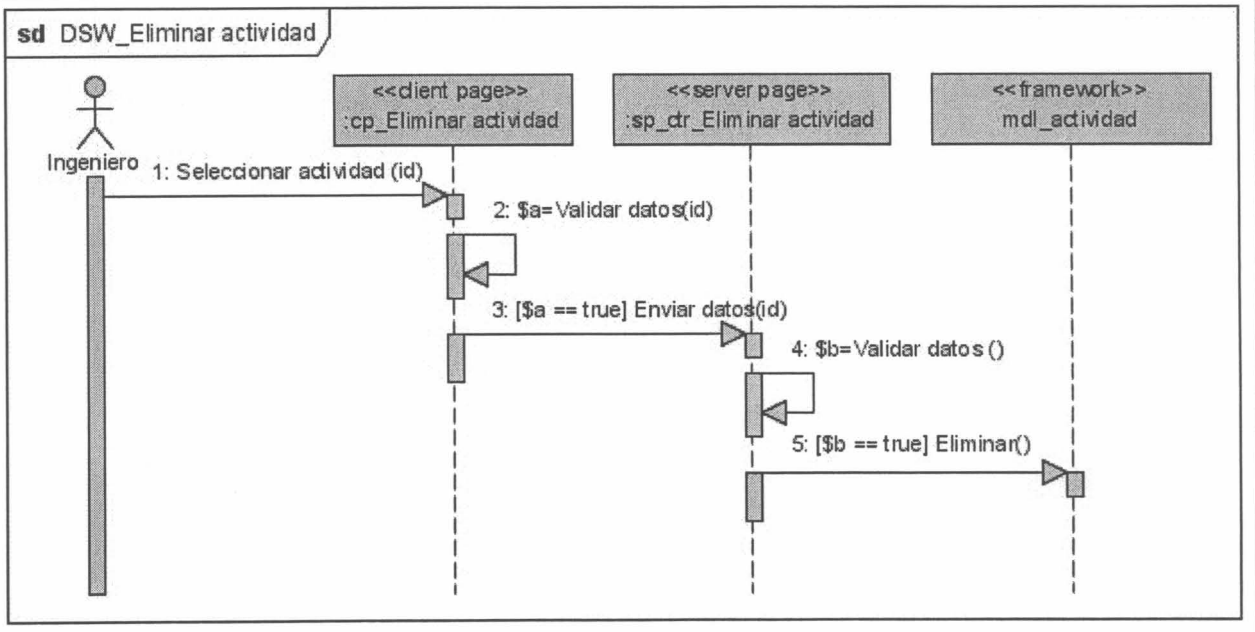


DIAGRAMA DE SECUENCIA CU-26 (Eliminar artefacto)

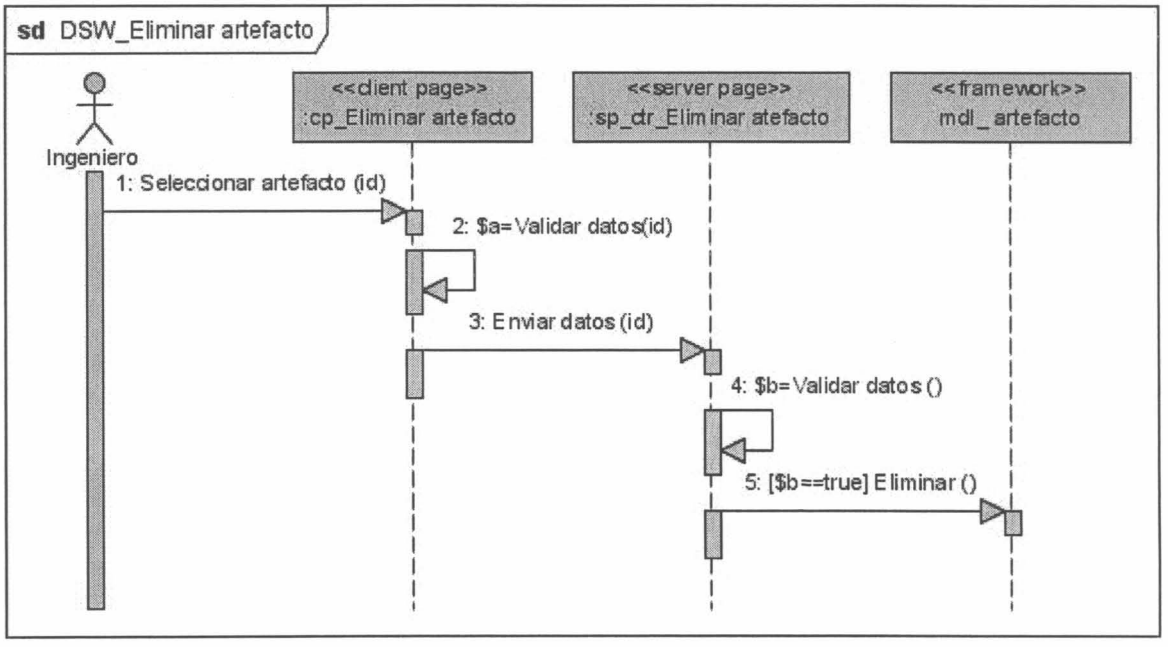


DIAGRAMA DE SECUENCIA CU-17 (Eliminar disciplina)

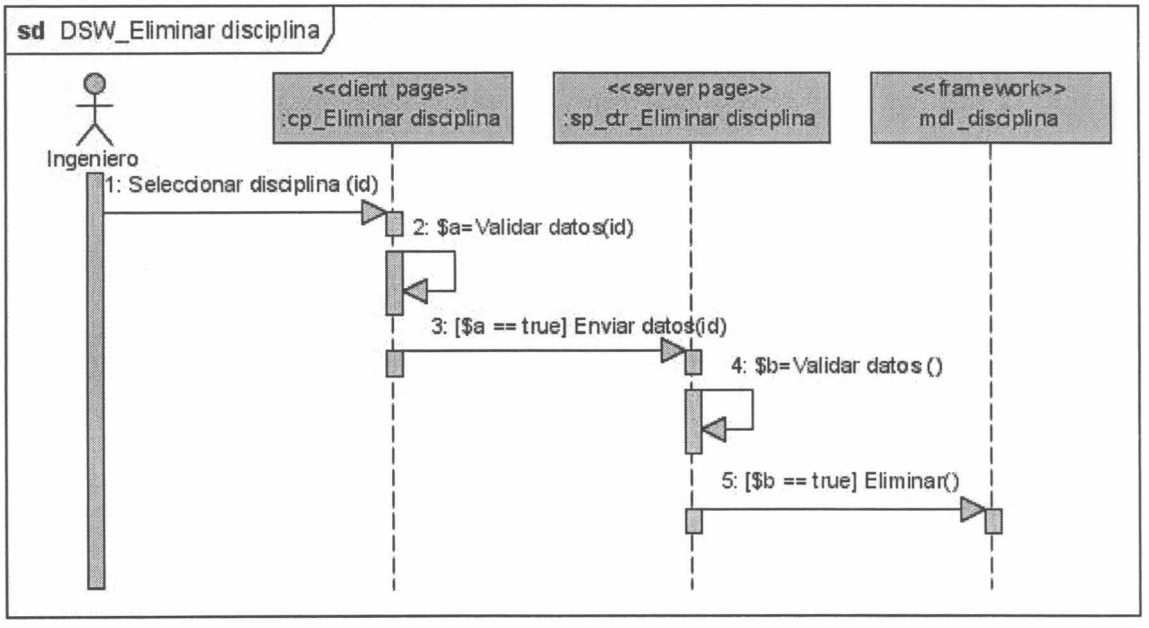


DIAGRAMA DE SECUENCIA CU-14 (Eliminar Fase)

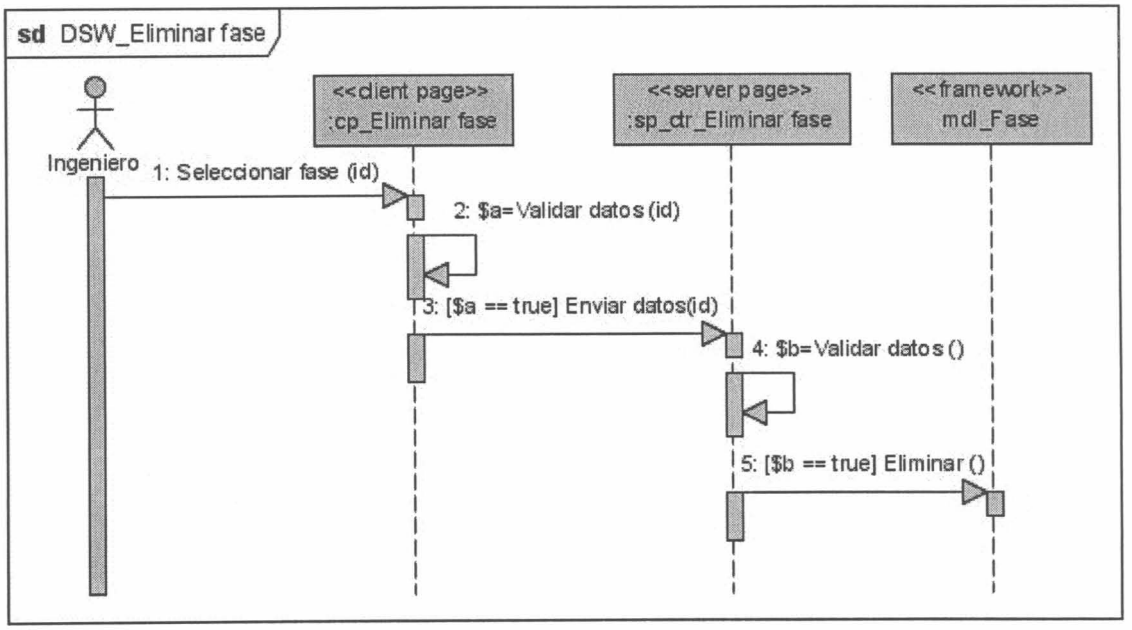


DIAGRAMA DE SECUENCIA CU-23 (Eliminar rol)

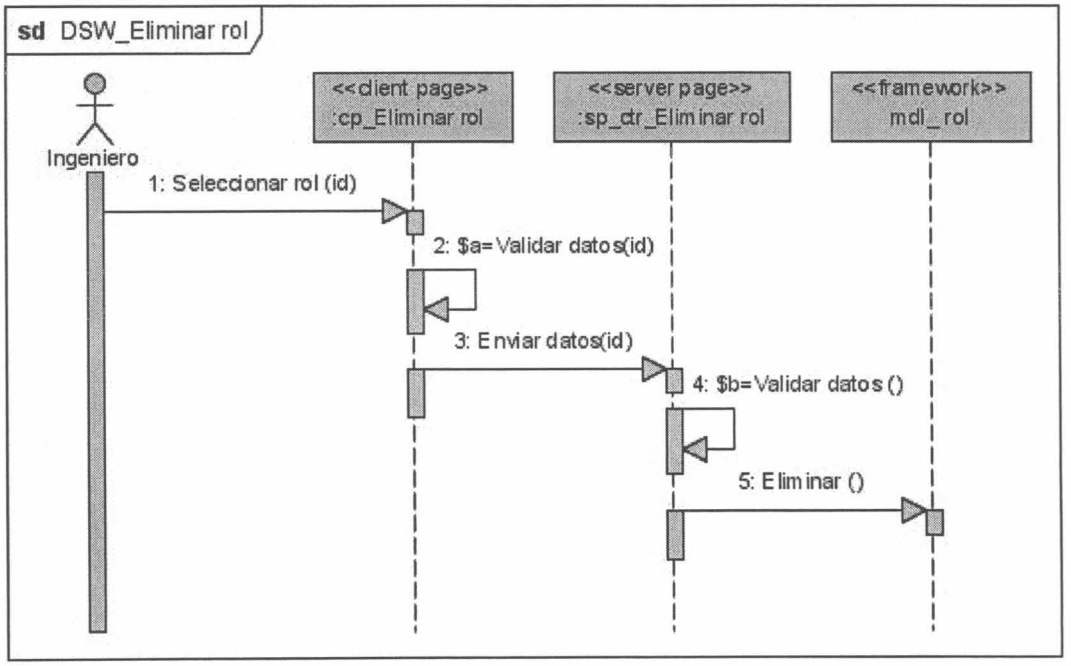


DIAGRAMA DE SECUENCIA CU-27 (Listar elementos)

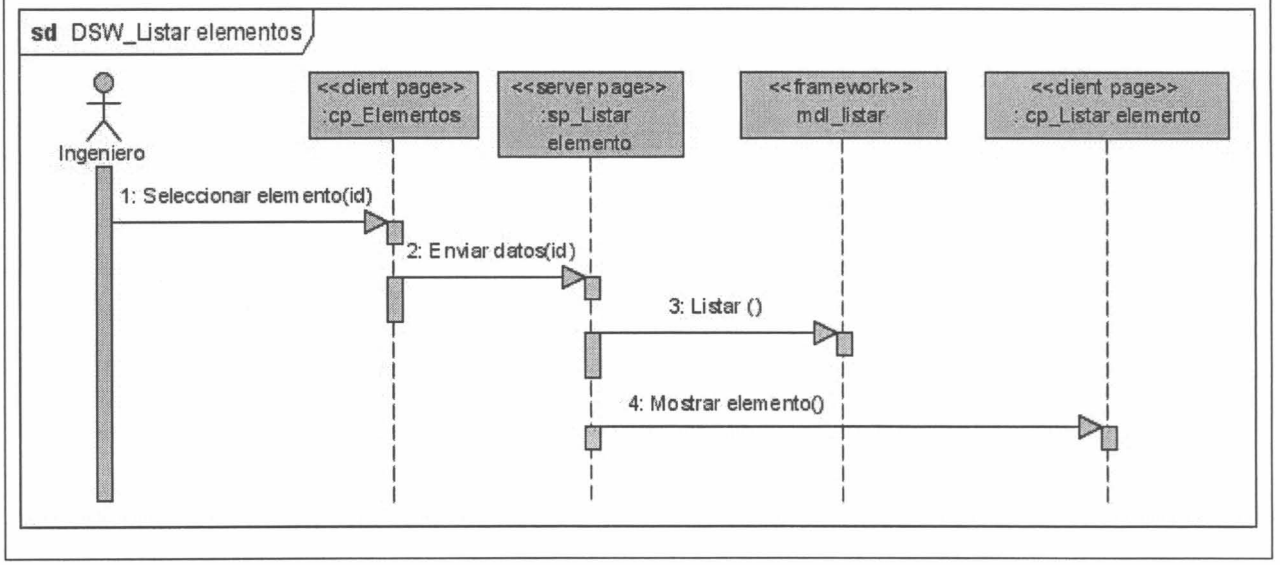


DIAGRAMA DE SECUENCIA CU-19 (Modificar actividad)

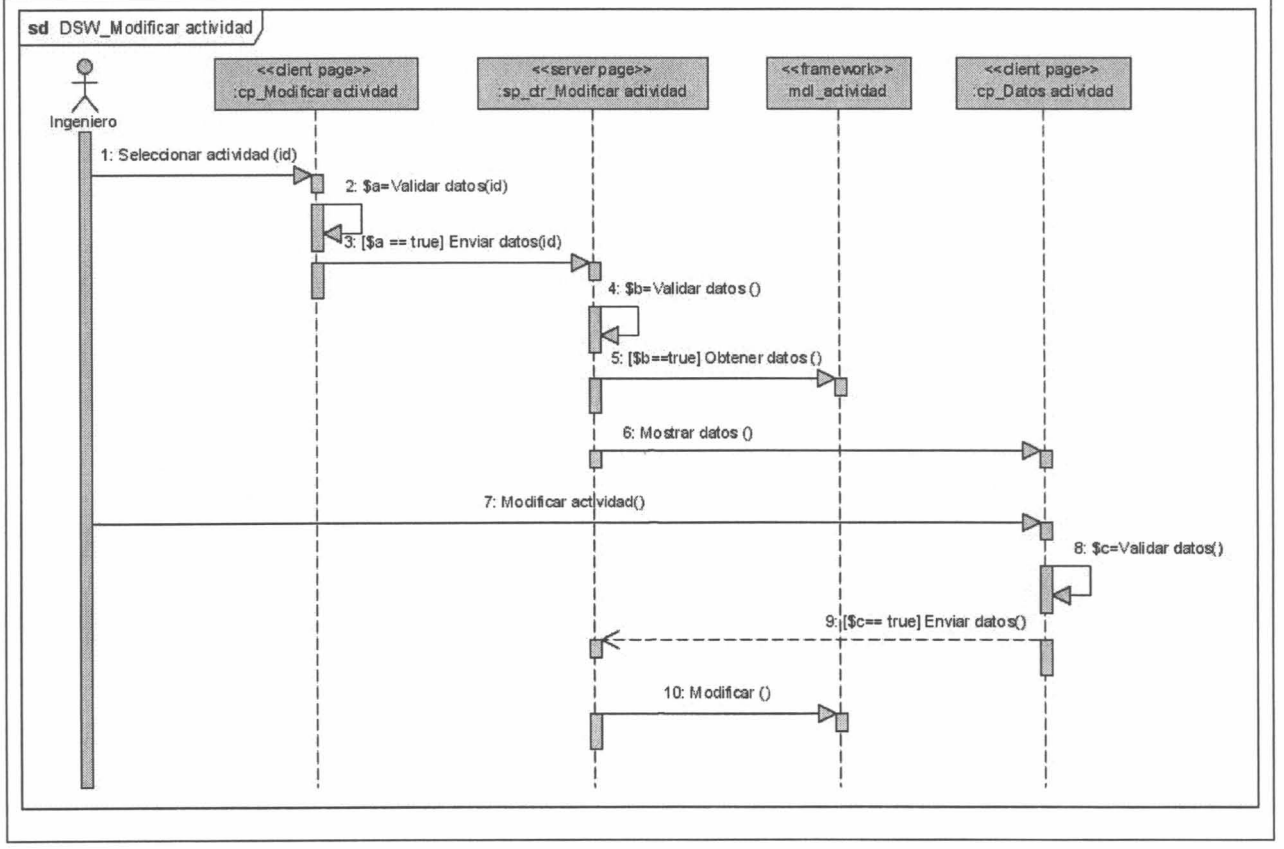


DIAGRAMA DE SECUENCIA CU-25 (Modificar artefacto)

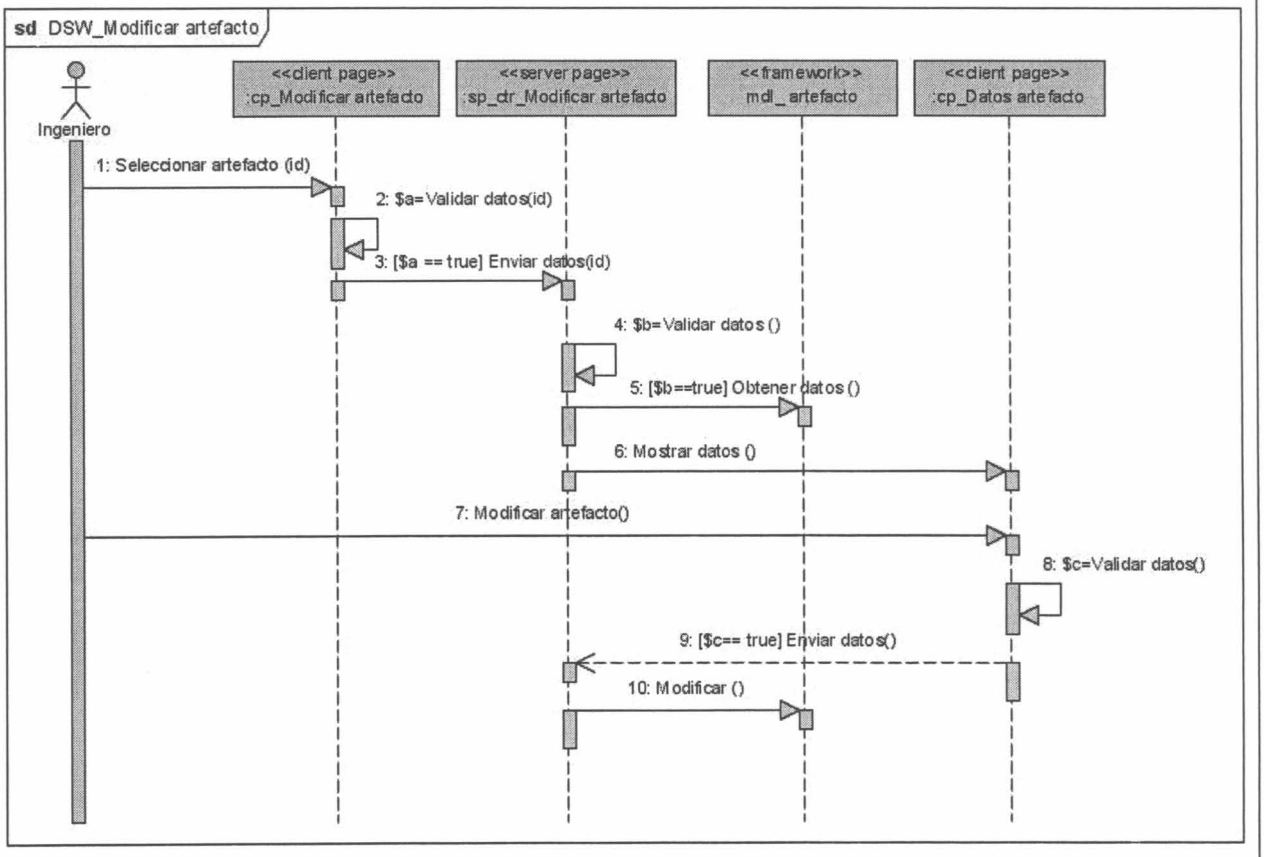


DIAGRAMA DE SECUENCIA CU-16 (Modificar disciplina)

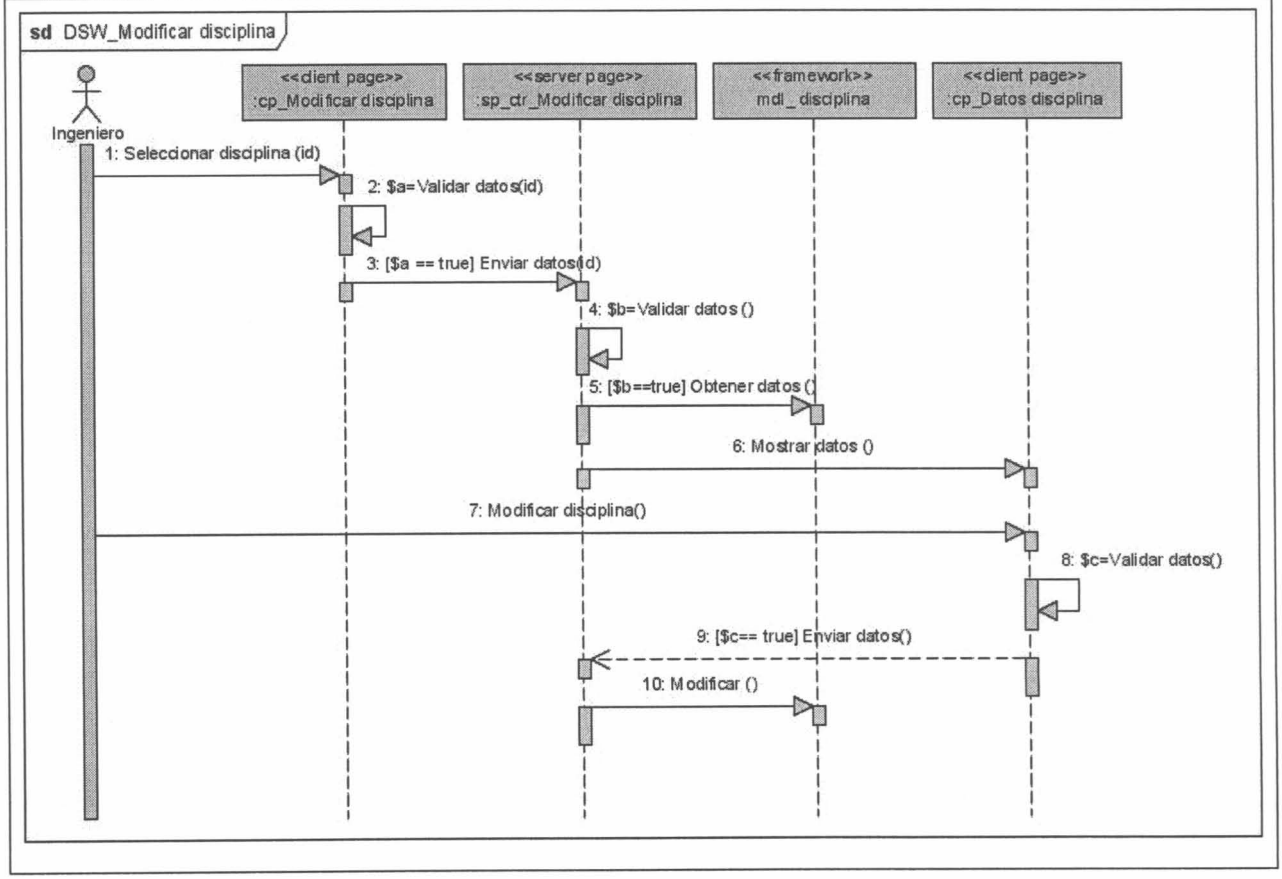
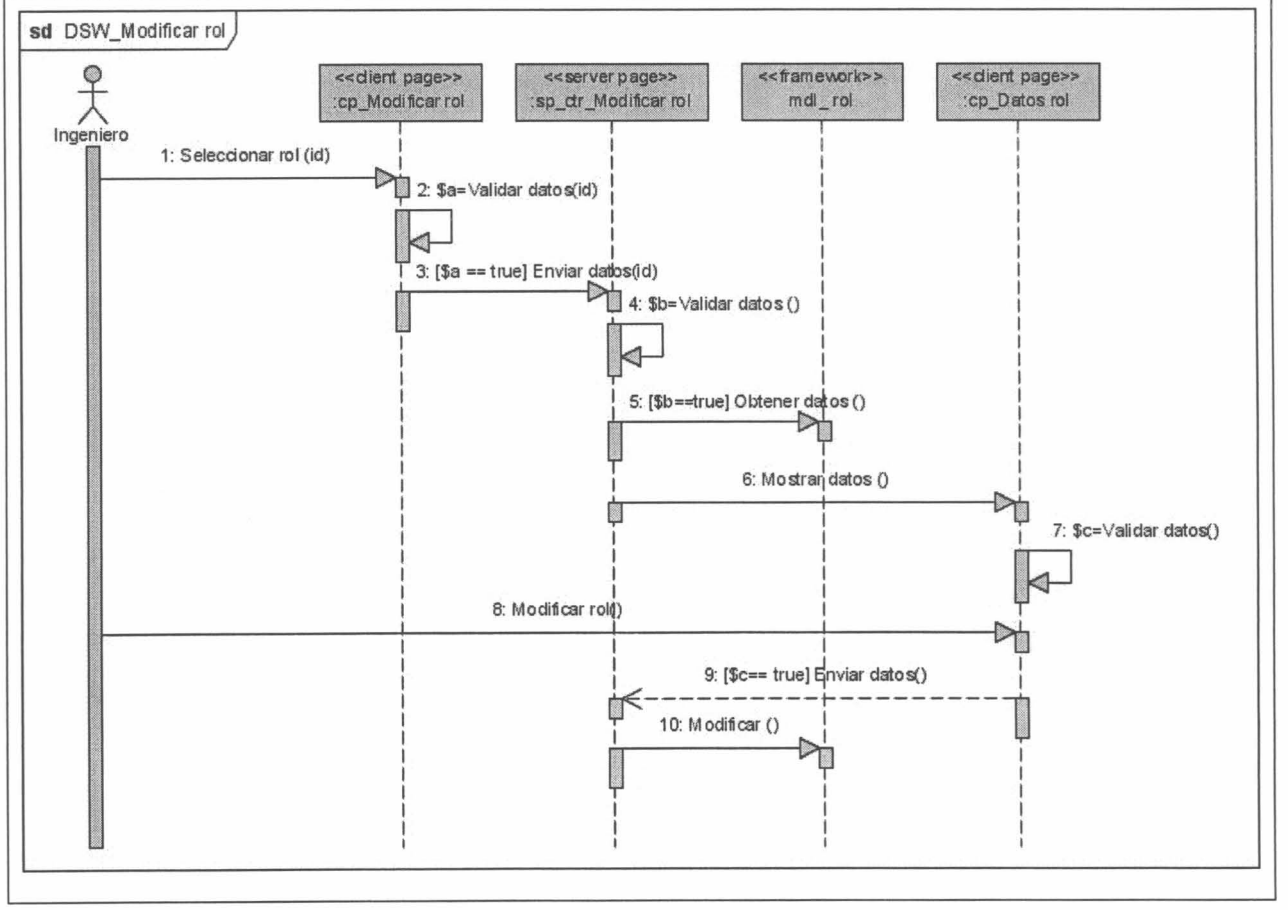


DIAGRAMA DE SECUENCIA CU-22 (Modificar rol)



Glosario de términos

Caso de uso: Proceso que se realiza en un sistema.

Estado: Es algún modo externamente observable del comportamiento.

Habilitador Metodológico: Herramienta accesible vía Web para la gestión de las metodologías de desarrollo.

Herramienta: Aplicación empleada para la construcción de otros programas o aplicaciones.

Ingeniería de software: es una disciplina que integra procesos, métodos y herramientas para el desarrollo de software de computadora.

Metodología: Conjunto de procedimientos o principios que guían particularmente el correcto desarrollo del software.

Plataforma: sistema operativo o sistemas complejos, ya sea de hardware o software, sobre el cual un programa pueda ejecutarse. Ejemplos típicos incluyen: arquitectura de hardware, lenguajes de programación y sus librerías de tiempo de ejecución. Ejemplos de plataformas son PC (Windows) y Macintosh (Mac).

Proceso: Conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software.

Release: Lanzamiento o liberación de una versión de un programa de software.

Requisitos: Características que se desea que posea un sistema o un software.

Software: Se refiere a todos aquellos programas y procedimientos que la computadora es capaz de leer ya que están escritos en lenguaje máquina y que por eso permite que ésta pueda operar.

Wiki: Sitio web cuyas páginas pueden ser editadas por múltiples lectores a través del navegador web.