El Proceso de Desarrollo RUP-GDIS

Christiane Metzner[1], Norelva Niño[1]
christiane.metzner@ciens.ucv.ve, norelva.nino@ciens.ucv.ve
[1] Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela

Resumen: En este trabajo se presenta y se describe el proceso de desarrollo de software RUP-GDIS y sus conceptos, utilizado actualmente en la asignatura de pregrado Ingeniería de Software de la Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela. RUP-GDIS, es una configuración de RUP centrada en las primeras cinco disciplinas correspondientes al núcleo de RUP, y adaptado para un curso de pregrado en el cual equipos de trabajo de estudiantes deben desarrollar por primera vez un sistema de software. El proceso guía a los estudiantes en "qué", "por qué", "cuándo" y "cómo" realizar exitosamente las diferentes actividades del proceso de desarrollo de software.

Palabras Clave: Ingeniería de Software; Proceso de Desarrollo de Software RUP-GDIS; Artefactos de Software.

Abstract: This paper describes the software development process RUP-GDIS and its concepts that are currently taught in the Software Engineering course at Universidad Central de Venezuela (Ciencias-UCV), Faculty of Sciences, School of Computer Science (http://www.ciens.ucv.ve/ciens) (http://computacion.ciens.ucv.ve/escueladecomputacion/). RUP-GDIS is a configuration of RUP centered on the five core disciplines of RUP, and tailored for an undergraduate course in which student project teams have to develop for the first time a software system. The process guides students in "what", "why", "when" and "how" to perform successfully the different activities defined in the development process.

Keywords: Software Engineering; Software development process RUP-GDIS, Software Artifacts.

1. Introducción

Rational Unified Process (RUP) se define como un meta-proceso que permite configurar procesos iterativos e incrementales y se estructura en dos dimensiones: fases y disciplinas [1]. Las fases son: Incepción, Elaboración, Construcción y Transición. Las disciplinas se categorizan en dos grupos: disciplinas del núcleo de RUP y las disciplinas de soporte al núcleo. Las disciplinas del núcleo de RUP son: Modelado del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, *Deployment*; y las disciplinas de soporte al núcleo son: Gerencia de Configuración y Cambio, Gerencia de Proyecto, Entorno. Cada fase tiene un propósito específico y en cada disciplina se realizan actividades que producen un resultado observable de valor en cada fase. Un proceso configurado a partir de RUP se organiza en términos de iteraciones; cada iteración cubre las disciplinas a lo largo de cada fase y el resultado de cada iteración es un producto ejecutable que se prueba, integra, entrega y se transformará en un sistema final. En la Figura 1, se ilustran las dos dimensiones de RUP donde el área bajo las curvas representa un estimado del esfuerzo de trabajo en cada disciplina cuando se itera a lo largo de las cuatro fases. Se destaca que en este trabajo se mantienen algunos nombres en el idioma inglés, y no su posible traducción al idioma castellano dado que estos son los nombres que se utilizan en el dictado de la asignatura Ingeniería de Software.

Admiraal [2] resume los modelos que se especifican en RUP centrando la atención en las disciplinas de Modelado del Negocio, Requerimientos, Análisis / Diseño e Implementación. En la Figura 2 se presenta un diagrama de paquete que muestra la visión general de los modelos de RUP y las dependencias entre los modelos y las disciplinas en las que Admiraal se centra [2]. Nótese que Admiraal aun cuando no considera la disciplina de *Deployment*, recomienda realizar el Modelo de *Deployment* en la disciplina de Análisis y Diseño.

Como se mencionó previamente, en un proyecto de software que se desarrolla bajo una configuración de RUP, el trabajo se organiza en iteraciones en dos dimensiones: a lo largo de cada fase y a lo largo de cada disciplina.
Es importante entender que no solo las iteraciones de una configuración de RUP, se realizan recorriendo las disciplinas a lo largo de las fases, sino que además, las fases se recorren a lo largo de cada disciplina. El propósito de cada una de las fases se resume a continuación [6]:
Incepción: establecer una visión general para el negocio, alcance, esfuerzo en horas-hombre y costo del proyecto.
Elaboración: refinar la visión, definir la arquitectura, identificar los requerimientos principales, el alcance y los riesgos de la solución.
Construcción: implementar de manera iterativa los requerimientos en el orden de prioridades establecidas, preparar la instalación.
Transición: finalizar, instalar y entregar la versión del *release*. Realizar las pruebas de aceptación. Examinar el *release* y evaluar desde la perspectiva del negocio qué partes satisfacen la visión de acuerdo con el documento de visión.

En cada iteración se realizan las actividades correspondientes a la mayoría o a todas las disciplinas. Iteraciones a lo largo de las fases de Elaboración, Construcción y Transición deberían producir código operativo. Mientras que las iteraciones a lo largo de Incepción, generalmente no producen código. El propósito de las disciplinas en el núcleo de RUP se resume a continuación [6]:
Modelado del Negocio: comprender las necesidades del negocio, describir su funcionamiento y los servicios que ofrece.

Requerimientos: trasladar las necesidades del negocio en comportamientos de un producto de software con el fin de describir lo que el producto debe hacer.
Análisis y Diseño: trasladar los requerimientos a una arquitectura de software con el fin de guiar la implementación.
Implementación: transformar el diseño en código fuente utilizando los mecanismos lingüísticos de un lenguaje de programación, establecer y seguir un estándar de codificación, definir la organización del código en términos de implementación. Implementar clases y objetos en términos de componentes.
Prueba: realizar una evaluación objetiva del producto [1]. Esto incluye encontrar y corregir errores, validar que el producto opere tal como fue diseñado y verificar que los requerimientos hayan sido implementados.
Deployment: producir un *release* del producto y entregar el software a los usuarios finales.

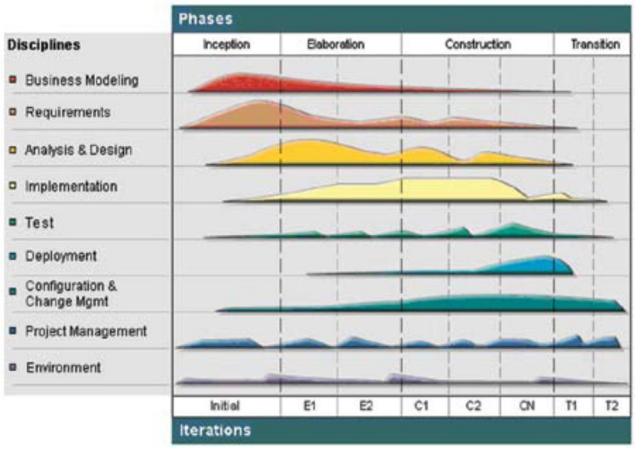

Figura 1: Disciplinas y Fases en RUP [8]

La idea central de las dos dimensiones en RUP es que el desarrollo consiste en realizar una serie de *release* incrementales o incrementos progresivamente más completos. Un *release* puede ser interno o externo. Los internos los utiliza el equipo de desarrollo para demostrar alguna característica o para realizar una presentación. Los externos se le entregan al Negocio. Cada incremento es el resultado de la iteración a lo largo de cada disciplina. Cada *release* es un producto operativo.

En la enseñanza y aprendizaje de los fundamentos de RUP si se comienza por las fases, usualmente los estudiantes caen en el error de interpretar las fases tal como se definen en el modelo de cascada, solo con nombres diferentes. Si se comienza por las iteraciones mostrando como un producto es el resultado de una serie de iteraciones y como el software y sus artefactos evolucionan a lo largo de esta serie de iteraciones hasta lograr un *release* que los estudiantes entregan para la evaluación del docente, esta idea es más fácil de comprender. En consecuencia y a nivel de la asignatura, se enfatiza en las iteraciones y la estructura de un producto.

Las iteraciones iniciales naturalmente tienden a centrarse en las disciplinas de Modelado del Negocio y Requerimientos, mientras que las iteraciones subsiguientes se concentran en la adaptación y retroalimentación. Por otra parte, en la disciplina de *Deployment* se incluyen las actividades necesarias para que las componentes del *release* en *Deployment* se preparen y se entreguen para su instalación. El proceso como tal de distribuir e instalar las componentes en *Deployment* no forma parte de RUP, siendo usualmente del dominio de un Departamento de Operaciones y no del grupo de desarrollo. Lo importante para los estudiantes es definir el proceso a seguir por su equipo de desarrollo para generar los artefactos de instalación y producir un documento que describa la versión.

Este artículo está organizado como se describe a continuación: en la segunda sección se especifica el proceso de desarrollo, fundamentado tanto en RUP [6] como en la publicación de Admiraal [2], y utilizado a partir del semestre I-2011 en la asignatura de pregrado Ingeniería de Software para el desarrollo de proyectos. En la tercera sección se presentan las perspectivas de este trabajo y se indican los resultados alcanzados.

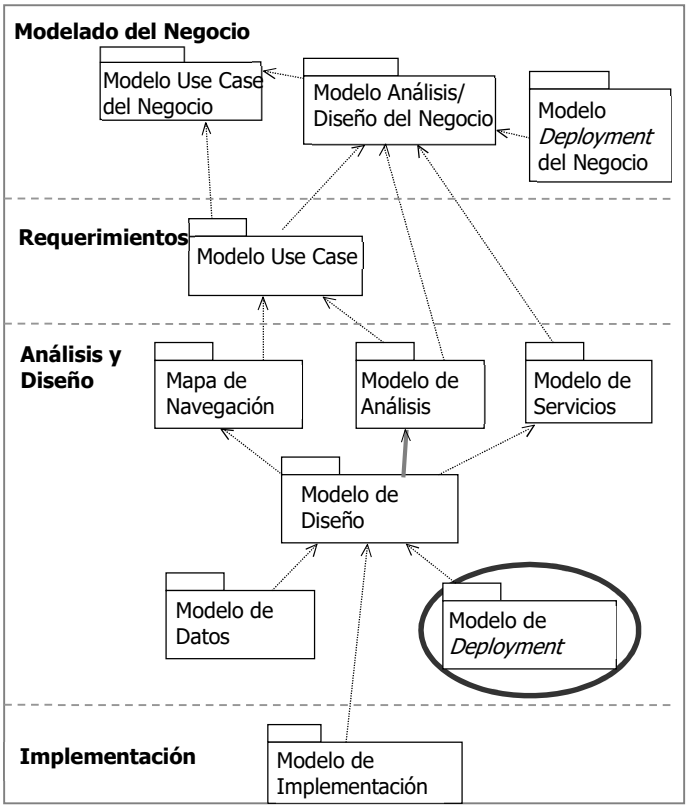

Figura 2: Modelos de RUP y Dependencias entre Modelos [2]

2. Definición del Proceso de Desarrollo

Un proceso puede contener diversos métodos y un método diversas técnicas. Exactamente cuando una técnica se transforma en método es difícil de precisar. El punto importante es que estos conceptos son necesarios. Un método describe como realizar algo. Un proceso es la simulación o ejecución de un método o colección de métodos ("haciendo algo"); refiere a una serie de acciones, cambios de estado o funciones que obtienen un resultado. Es útil recordar que un proceso tiene lugar en el "mundo real" mientras que la descripción de un proceso es lo que se documenta por ejemplo, con RUP.

Una metodología es una colección de métodos relacionados. Refiere al conjunto de prácticas (una práctica es una manera sistemática de realizar y lograr algo), procedimientos y reglas

utilizados por quienes trabajan en una disciplina (por ejemplo, Tecnologías de Información) así como el estudio o análisis teórico de métodos. Un método refiere solo a una de las prácticas. Una técnica es un procedimiento sistemático con el cual realizar una tarea.

La importancia de utilizar un proceso de desarrollo para la construcción de software radica en la necesidad de reducir riesgos, como por ejemplo: retraso en las fechas pautadas para las entregas.

Un proceso de desarrollo guía las actividades a realizar durante el desarrollo de un software. Sin embargo, también es cierto que el hecho de utilizar un proceso de desarrollo no garantiza el éxito de una aplicación, entendiendo por éxito que el software produzca los resultados esperados, el Negocio esté satisfecho con el software producido y los riesgos mencionados anteriormente no hayan impactado el desarrollo.

Actualmente, existen diversos procesos de desarrollo; entre los más conocidos y difundidos se pueden mencionar las configuraciones de RUP [6], UP [9], FDD [10], XP [11], SCRUM [12]. La tendencia en empresas que desarrollan aplicaciones Web o buscan ser competitivas en el mercado, es utilizar procesos livianos o ligeros, es decir se generan solo aquellos artefactos de software necesarios.

En nuestro contexto académico, se tiene la limitación del tiempo y del escaso nivel de conocimiento que tienen los estudiantes acerca de procesos de desarrollo cuando cursan la asignatura Ingeniería de Software, siendo esta la primera asignatura en la cual los estudiantes adquieren conocimientos en el área. En consecuencia es recomendable utilizar un proceso definido que guíe a los estudiantes en el desarrollo de su proyecto en la asignatura. Adicionalmente, es necesario que el proceso de desarrollo utilice los conceptos y técnicas del temario de la asignatura.

El proceso de desarrollo utilizado que se describe a continuación se centra en las disciplinas del núcleo de RUP indicándose los artefactos que se generan para cada modelo, considerando las recomendaciones de Admiraal y en base a la experiencia adquirida en el uso de este proceso. Un modelo es una representación descriptiva grafica o textual, ejecutable o estática, de un subconjunto de propiedades de un sistema. Un artefacto puede ser un modelo, un elemento de un modelo o un documento. Un documento puede contener a su vez otros documentos. Es un producto usado o producido durante un proceso de desarrollo de software y es el resultado de una actividad. Ejemplos de artefactos incluyen modelos, código fuente, código ejecutable.

2.1 Disciplina de Modelado del Negocio

Los estudiantes deben utilizar esta disciplina para desarrollar su proyecto. En la disciplina se incorpora el artefacto denominado "Tabla de Eventos del Negocio", que no está definido en RUP, con el fin de extraer de una descripción textual los eventos de interés para un actor del negocio. Un evento del negocio es algo que un actor del negocio puede solicitarle al negocio. Es importante destacar que un evento es instantáneo, mientras que el Negocio tiene que realizar una serie de acciones o actividades para atender un evento. La Tabla de Eventos del Negocio les facilita a los estudiantes la tarea de identificar los use case del negocio desde la perspectiva de un actor del negocio. Los eventos identificados se registran en una tabla de eventos conformada por dos columnas: Identificador de Evento y Descripción del Evento del Negocio. En base a la tabla de eventos se generan los siguientes modelos de RUP:

Modelo Use Case del Negocio: un use case del negocio presenta lo que el Negocio ofrece a los actores. El Negocio decidirá cómo realizarlos, bien sea manualmente, o bien sea automatizarlos parcial o totalmente.

Los documentos y productos que intervienen en un flujo de trabajo se denominan entidades del negocio. Las entidades del negocio pueden representarse como clases del negocio.

El modelo describe en términos de los use case del negocio las interacciones externas de la organización, lo que deben realizar tanto el Negocio como los actores del negocio para llevar a buen término un flujo de trabajo. Los elementos de este modelo son estables y facilitan el desarrollo de modelos subsecuentes que pueden no ser estables. Permite adquirir conocimientos acerca del dominio e identificar posibles soluciones a problemas del negocio. Estos modelos evolucionan sobre periodos extendidos de tiempo, si el Negocio modifica la manera en que opera y/o sus productos. Los diagramas a utilizar son:

Diagrama Use Case: identifica los use case del negocio, los actores del negocio y sus relaciones. Un use case del negocio modela los servicios que el Negocio ofrece a los actores del negocio, y es independiente de las tecnologías. Una estrategia que permite evitar la consideración de funcionalidades de sistema en los use case del negocio consiste en considerar que el negocio implementa sus servicios de manera no computarizada.

Un trabajador del negocio no es un actor del negocio, porque es parte del negocio. Es un representante del negocio con el cual un actor del negocio interactúa y que se transformará en un actor en los use case de sistema, donde utiliza el sistema como herramienta para prestarle un servicio a los actores del negocio. Es importante tener en cuenta que en los use case del negocio, no se diferencia entre actores primarios y secundarios. Es decir, solo hay actores del negocio. Para identificar los use case del negocio, se consideran únicamente los eventos registrados en la tabla de eventos. En la especificación de cada uno de los use case del negocio se presenta tanto el flujo básico como los flujos alternativos de trabajo del negocio, es decir, qué hace el negocio para ofrecerle algo de valor al actor dado que éste ha generado uno o más eventos en la interacción con el negocio. Los pasos de un proceso de negocio se realizan en uno o más use case de sistema.

El diagrama use case no describe los procesos del negocio; para esto se utilizan diagramas de actividad. Para especificar los use case del negocio y establecer los pasos en el flujo de eventos, se puede utilizar alguno de los siguientes artefactos:

Diagrama de Actividad: describe las acciones y los resultados asociados a un flujo de eventos de un use case del negocio.

Plantilla de Especificación de los use case: describe de manera rigurosa que se hace cuando un actor interactúa con el use case (ver Apéndice A1).

Modelo de Análisis / Diseño del Negocio: modela el funcionamiento interno de la organización para realizar los use case del negocio. Se modela también la estructura organizacional y el flujo de la información en caso de que se considere relevante. Los diagramas a utilizar son:

Diagrama de Clase: identifica las clases del negocio y las relaciones entre ellas. Se corresponde con la estructura de la organización y de la información. Es de destacar que el modelado orientado a objetos (OO) específicamente modela el comportamiento. Los objetos existen en un sistema computacional y están sujetos a restricciones y acciones del sistema, poseen un ciclo de vida: se crean y se destruyen, pueden ser asignados a otros objetos y tienen un comportamiento. Al definir un objeto/clase asegúrese de considerar que existe en el sistema y que tiene comportamientos para su existencia. Los trabajadores del negocio en principio no son clases del negocio. Pueden llegar a serlo dependiendo de las necesidades de persistencia de la información. Algunos criterios que pueden ayudar a decir si se representan como clases se presentan a continuación:

¿El trabajador del negocio tiene un comportamiento identificable en el dominio del problema? Esto es, ¿se pueden nombrar servicios o funciones necesarias en el dominio del problema que son propias del trabajador y que este provee? Indíquelos.

¿El trabajador del negocio tiene relaciones con otros trabajadores del negocio? Analícelas e incorpórelas.

¿El trabajador del negocio actúa dentro de los límites del sistema? Si no lo hace, puede ser un actor del sistema.

¿El trabajador del negocio posee una estructura identificable? Esto es, ¿es posible identificar algún conjunto de atributos que el trabajador posee y que deben administrarse? Agréguelos.

Diagrama de Actividad: se usa principalmente para modelar el flujo de trabajo y es útil para analizar los use case describiendo las acciones que necesitan realizarse, cuándo se realizan y quién es el responsable de realizarlas; no da el detalle de cómo cooperan o colaboran los objetos, por eso no substituye un diagrama de secuencia. Los trabajadores del negocio y los actores del negocio se representan en las particiones del diagrama y éstas contienen las acciones o actividades. Las entidades del negocio son las entradas y / o salidas de las acciones o actividades. Las notas se utilizan para indicar que un objeto del negocio se genera en una acción o actividad para un actor del negocio. Una acción o actividad puede corresponderse con un use case del negocio. Se diferencian los objetos físicos de los objetos de información [3]. El estereotipo *<<Information>>* se utiliza para indicar que un objeto es de tipo información y el estereotipo *<<Physical>>* se utiliza para indicar que un objeto representa un objeto físico real. No deben existir transiciones entre objetos de información que representen un flujo de control.

Diagrama de Secuencia: representa el flujo de trabajo centrado en el intercambio de mensajes entre entidades del negocio. Las entidades del negocio representan las instancias de las clases del negocio identificadas en el diagrama de clase del negocio. Los trabajadores del negocio y actores del negocio se representan con el icono de actor y el estereotipo *<<user>>*. Los trabajadores del negocio intercambian mensajes con las entidades del negocio.

2.2 Disciplina de Requerimientos

Se utilizan los modelos generados en la disciplina de Modelado del Negocio para elaborar los requerimientos del sistema para el Negocio. En la práctica, el Negocio establece cuáles de los use case del negocio que han sido identificados y especificados van a ser automatizados actualmente; a futuro se considerará y planificará la automatización de otros use case del negocio. Los use case del negocio se identifican durante la disciplina de Modelado del Negocio y describen un proceso del negocio. Los use case (de sistema) se identifican durante la disciplina de Requerimientos y describen un requerimiento funcional desde la perspectiva de un actor; muestran las funcionalidades del sistema para que los actores logren sus objetivos.

La experiencia en la realización del proyecto del semestre II-2010, motivó la incorporación del artefacto denominado "Tabla de Eventos del Sistema" que no está definido en RUP. Este artefacto facilita la tarea de identificar los requerimientos funcionales (lo que el producto debe hacer), no funcionales (ciertas características de calidad que el producto debe poseer) y las restricciones (del negocio o las del uso de herramientas que apoyan la generación de artefactos de software). Los eventos del sistema se registran en una tabla de eventos.

Para la identificación de los requerimientos funcionales del sistema se consideran las especificaciones de los use case del negocio. Se utiliza la tabla de eventos para registrar los eventos del sistema.

Una estrategia para comenzar a identificar eventos del sistema es trasladar cada una de las acciones o actividades que están en los pasos del flujo básico en la respuesta del negocio de los use case del negocio y colocarlos en cada fila de la tabla. Luego se examinan y se transforman utilizando terminología de sistema. Además se analiza si existen eventos que puedan ser generalizados o si existen eventos que forman parte de otros eventos. Por otra parte, se identifica y analiza cuáles otros eventos pueden ocurrir a nivel de sistema, eventos que no se identificaron y/o se consideraron en el Modelo del Negocio por ser internos al negocio.

La tabla de eventos del sistema está conformada por siete columnas: Identificador de Evento, Descripción del Evento del Sistema, Restricciones, Actor, Prioridad, Identificador del Use Case del Negocio con el cual se relaciona, e Identificador del Use Case del Sistema (se indica una vez generado el Modelo Use Case del Sistema). La prioridad indica el orden en que los eventos del sistema van a ser diseñados e implementados; su rango de valores va de 1 al número máximo de eventos. El valor 1 se asocia al evento de mayor importancia para el Negocio (o para el Desarrollo si el evento debe considerase antes de diseñar o implementar otro evento) y el valor máximo de prioridad se le asigna al evento que se considere de menor importancia. Un evento del sistema que ya se implementó no tiene prioridad y se indica con el identificador "IMP".

Una vez identificados los eventos del sistema, se genera el siguiente modelo de RUP:

Modelo Use Case: describe las interacciones entre los actores y el sistema, y la meta de los actores al usar el sistema (use case).

Para identificar los use case del sistema se utiliza la tabla de eventos y generalmente la correspondencia no es uno a uno. Se utilizan los siguientes diagramas:

Diagrama Use Case del Sistema: describe lo que debe hacer el sistema para automatizar uno o más pasos de la realización del use case de negocio. Se representan los use case del sistema, los actores del sistema y las relaciones entre los use case y sus actores. Un actor puede corresponderse con un actor del negocio, en caso de que el actor del negocio acceda al sistema. Un actor primario es aquel que inicia un use case y obtiene un beneficio cuando se obtiene el propósito del use case, un actor secundario participa en obtener el propósito. Si se identifica un use case sin actor, esto probablemente es el resultado de una descomposición funcional. No debería asociarse más de un actor con un use case debido a que la especificación se redacta desde la perspectiva de un solo actor que tiene un propósito especifico. Tratar de describir un flujo de eventos desde más de una perspectiva es confuso y lleva a descripciones sobrecargadas y difíciles de comprender. Por convención, los actores primarios se colocan del lado izquierdo y los actores secundarios del lado derecho en el diagrama use case [15].

Los diagramas use case son una herramienta que comunica el alcance de un negocio o sistema; la información importante está en la especificación de estos. Es de destacar que los use case deben ser cajas negras, describiéndose solamente el comportamiento que es visible para los actores. Para especificar los use case y establecer los pasos en el flujo de eventos, se puede utilizar alguno de los siguientes artefactos:

Diagrama de Actividad: documenta flujos de trabajo del negocio ante las solicitudes del actor. Puede ser: (1) detallado cuando es necesario comprender un proceso complejo del negocio (en la disciplina de Modelado del Negocio) o (2) simplificado para el actor y el sistema. El caso 2 documenta los detalles del use case y describe las acciones y los resultados asociados a un flujo de eventos de un use case del sistema.

Plantilla de Especificación de use case del sistema: la descripción se presentó en la subsección 2.1. A nivel de sistema se distingue entre actores primarios y secundarios.

Un diagrama de actividad describe las actividades de un actor o conjunto de actores, mientras que un use case describe las interacciones con un sistema que permiten realizar las actividades.

2.3 Disciplina de Análisis y Diseño

El Modelo de Análisis presenta un "diseño preliminar" de un conjunto de requerimientos; el Modelo de Diseño muestra como las tecnologías seleccionadas realizan el Modelo de Análisis. Admiraal [2] recomienda no realizar un Modelo de Análisis, argumentando que el Modelo de Análisis del Negocio y el Modelo Use Case proveen suficiente información que permiten hacer un primer esbozo de la arquitectura de componentes en el Modelo de Diseño y para comenzar a hacer las realizaciones de los use case en términos de las componentes que interactúan. Esta sugerencia fue considerada en los semestres I-2011 y II-2011, sin embargo se detectó dificultad por parte de los estudiantes al momento de comprender y trabajar con el diagrama de clase de diseño. Por supuesto que la experiencia de Admiraal es distinta a la de los estudiantes de Ingeniería de Software, pero consideramos conveniente desarrollar tanto el Modelo de Análisis como el Modelo de Diseño que comprenden los siguientes modelos.

Modelo de Análisis: se analizan y refinan los requerimientos del modelo use case para obtener una visión detallada de los requerimientos del sistema. El modelo de análisis se describe en un lenguaje para desarrolladores y proporciona una visión general y conceptual del sistema respecto a lo que se tiene qué hacer y no cómo se va a hacer [7]. Por este motivo es que es un modelo útil y conveniente ya que facilita comprender el sistema sin mostrar detalles de alternativas de diseño que pueden variar y están atadas al entorno de implementación. El Modelo de Análisis se considera como una versión inicial del Modelo de Diseño. Los diagramas a utilizar son:

Diagrama de Clase: representa la estructura estática del sistema con las clases, atributos, operaciones y relaciones que van a ser diseñadas e implementadas. Se incorporan en las clases del Modelo de Análisis / Diseño del Negocio atributos y responsabilidades u operaciones a un nivel alto de abstracción, y se etiquetan las clases con un estereotipo para categorizar las clases como interfaz, entidad o control. Puede ocurrir que ciertas clases entidad del negocio se conviertan en atributos de otras clases o no sean consideradas como clases entidad. El uso de estereotipos se omite en el caso de las clases entidad para no sobrecargar visualmente al diagrama. Las clases entidad se utilizan en el Modelo de Análisis para modelar la persistencia de información.

Las clases interfaz se indican con el estereotipo *<<boundary>>* o *<<b>>* y se identifican a partir de las interacciones entre los actores del sistema y los use case del sistema. Los atributos de las clases interfaz se identifican a partir del flujo básico y/o alternativo de las especificaciones de los use case en los que un actor interactúa con objetos del sistema y el sistema realiza acciones sobre objetos como consecuencia de las interacciones. Las clases interfaz se utilizan en el Modelo de Análisis para modelar las interacciones entre el sistema y sus actores.

Las clases control se definen para evitar que las clases interfaz tengan relación de asociación con las clases entidad. Decisiones respecto al orden en que se instancian las clases interfaz, o acciones a realizar cuando un elemento gráfico se presiona, por ejemplo un botón, deben implementarse en clases control y no en las clases interfaz ya que por lo general esas acciones involucran la consulta, recuperación o almacenamiento de data en las clases entidad. Sin embargo, las validaciones de datos capturados por un objeto interfaz pueden definirse en su clase interfaz. Las clases control se indican con el estereotipo *<<control>>* o *<<ctrl>>*. Es útil y recomendable realizar un pseudocódigo para identificar las operaciones en las clases control así como operaciones adicionales en las clases entidad e interfaz.

Diagrama de Secuencia: representa el orden de envío de mensajes entre instancias de clases que sean de interés, y para identificar nuevas operaciones de las clases. En análisis el diagrama documenta solo las interacciones que son entradas y resultados. Pueden evolucionar a lo largo de un proyecto cuando se agregan instancias que representan decisiones de diseño. Se muestra el actor y los objetos del sistema así como los mensajes de interacción. El diagrama de secuencia permite describir un

comportamiento que es más complejo de lo que se ve a simple vista, así como identificar las asociaciones y las operaciones que se requieren. Es importante tener en cuenta que un actor del sistema puede enviar mensajes solo a objetos interfaz y no a objetos control ni a objetos entidad.

Modelo de Mapa de Navegación: describe la secuencia de navegación que puede recorrer un actor del sistema. Una relación entre un use case y un actor implica la existencia de una interfaz. Si el actor es humano, es una interfaz usuario; si es un sistema es una interfaz de sistema. La interfaz de sistema se especifica en el diagrama de clase de diseño donde se definen los métodos de la clase con el estereotipo <<sistema>>. Se utilizan los siguientes artefactos:

Prototipos: muestra los elementos de la interfaz gráfica de usuario. Los prototipos pueden generarse con una herramienta o manualmente. Por lo general, cada prototipo de interfaz usuario se corresponde con una clase interfaz.

Diagrama de Estado: representa la secuencia de navegación entre las instancias de las clases interfaz del sistema. Las instancias de las clases interfaz del sistema se representan con estados en el diagrama y las transiciones representan los posibles caminos de navegación, resultado de interacciones.

Modelo de Diseño: El lenguaje de programación utilizado en la asignatura Ingeniería de Software es *Java*™ [13] con el entorno de desarrollo *NetBeans* [14]. Estas herramientas se seleccionaron dado que por una parte *Java* es un lenguaje orientado a objetos, por lo tanto los conceptos cubiertos en el programa de la asignatura corresponden directamente a propiedades que se implementan en *Java*, por ejemplo, clase/objeto. Por otra parte, es un lenguaje ampliamente documentado, utilizado generalmente en las organizaciones. El uso de *NetBeans* facilita el diseño e implementación de los prototipos de interfaz y se cumple con uno de los objetivos de la asignatura respecto a utilizar entornos de desarrollo.

El Modelo de Diseño especifica el diseño detallado de las clases. Los diagramas a generar son:

Diagrama de Clase: se especifican propiedades de las clases entidad, interfaz y control: tipo de datos y visibilidad de los atributos y operaciones especificados en las clases del Modelo de Análisis. Una operación especificada en una clase del Modelo de Análisis se convierte en uno o más métodos en el Modelo de Diseño. Se analiza si es posible generalizar operaciones de las clases de análisis. Atributos especificados en una clase de análisis se pueden convertir en clases de diseño. Las clases de diseño pueden etiquetarse con estereotipos para reflejar decisiones de implementación en un lenguaje de programación, por ejemplo, una clase interfaz que va a ser implementada bajo el entorno de desarrollo *NetBeans* se puede etiquetar con el estereotipo *<<form>>*. Si se utilizan patrones de diseño [5], generalmente se agregan atributos, métodos, relaciones y eventualmente clases para dar solución a un problema específico de diseño. Puede ser de utilidad incorporar notas con pseudocódigo en los métodos de clases.

Es útil y recomendable generar el pseudocódigo para identificar si existen los métodos en las clases de control que permiten realizar las funcionalidades o si es necesario definir otros métodos en las clases entidad e interfaz. Este pseudocódigo facilita determinar las funcionalidades de los métodos, así como definir y comprender el flujo de control de la aplicación. Se especifican estructuras de datos para implementar las relaciones 1:N.

Diagrama de Secuencia: se muestran los intercambios de mensajes entre los objetos de diseño y son más detallados que los diagramas de secuencia que se generan en el Modelo de Análisis.

2.4 Disciplina de Implementación

Se establece el estándar de codificación en cuanto a nombramiento de clases, métodos y atributos. En la asignatura se utiliza el estándar que se presenta en el Apéndice A2.

Se realiza el siguiente modelo:

Modelo de Implementación: describe la implementación del diseño del sistema y se utilizan los siguientes artefactos de software:

Código Fuente Documentado en el lenguaje de programación *Java*™.

Diagrama de Paquete: se utiliza para organizar clases en los subdirectorios de un proyecto bajo *NetBeans* de acuerdo con un criterio.

2.5 Disciplina de Prueba

En RUP se distinguen cuatro tipos de prueba [6]: unitaria, de integración, de sistema y de aceptación. Las pruebas que realizan los estudiantes de la asignatura en el proceso RUP-GDIS son pruebas unitarias y de integración. Las pruebas de sistema y de aceptación se realizan en el contexto académico con la entrega que hacen los estudiantes a los miembros del grupo docente. Se realiza el siguiente modelo:

Modelo de Prueba: describe las pruebas. Debe indicarse el identificador de clase, el identificador del caso de prueba, su descripción, y un reporte del resultado de la prueba.

Especificación de Casos de Prueba: describe cuáles son los datos con los que se ejecuta el caso de prueba (ver Apéndice A3).

En la Tabla I se resumen, por cada una de las disciplinas, los modelos que se consideran en el proceso RUP-GDIS y los artefactos de software que se pueden utilizar, y en la Figura 3 se resumen los modelos considerados en el proceso RUP-GDIS y las dependencias entre los modelos y las disciplinas.

Tabla I: Artefactos de RUP-GDIS

Disciplina	Modelos a generar en la disciplina	Artefactos de software utilizados	Observación
Modelado del Negocio		- Tabla de Eventos del Negocio	
	Modelo Use Case del Negocio	- Diagrama use case - Plantilla para especificar use case, o - Diagrama de actividad	La especificación de cada use case del negocio se realiza utilizando la plantilla o un diagrama de actividad

Disciplina	Modelos a generar en la disciplina	Artefactos de software utilizados	Observación
	Modelo de Análisis/Diseño del Negocio	- Diagrama de clase - Diagrama de actividad - Diagrama de secuencia	
Requerimientos		- Tabla de Eventos del Sistema	
	Modelo Use Case	- Diagrama use case - Plantilla para especificar use case, o - Diagrama de actividad	
Análisis y Diseño	Modelo de Análisis	- Diagrama de clase - Diagrama de secuencia	
	Modelo de Mapa de Navegación	- Prototipos - Diagrama de estado	
	Modelo de Diseño	- Diagrama de clase - Diagrama de secuencia	
Implementación	Modelo de Implementación	- Código fuente operativo y documentado - Diagrama de paquete	
Prueba	Modelo de Prueba	- Especificación de casos de prueba	Identificador de clase, identificador de caso de prueba y su descripción, reporte del resultado de la prueba

No se considera a la disciplina de *Deployment* dado que un producto de software desarrollado como proyecto en la asignatura no se entrega a usuarios finales. La entrega se realiza al grupo docente de la asignatura para su evaluación. Los modelos especificados en el proceso RUP-GDIS y sus dependencias se basan en Admiraal [2] y están adaptados para ser generados y utilizados por estudiantes del 3er semestre.

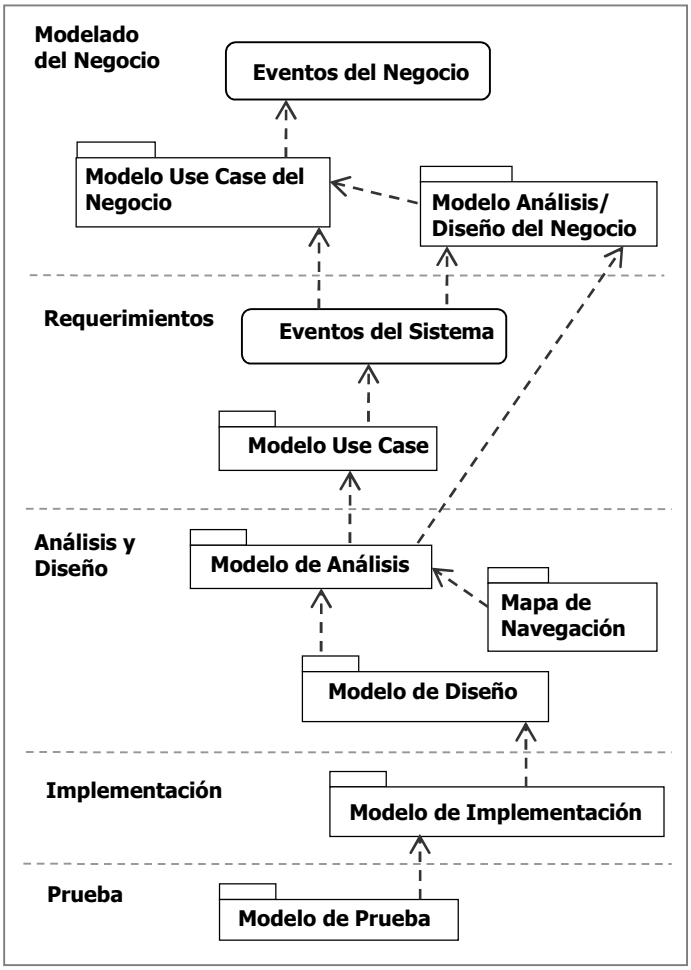

Figura 3: Modelos en RUP-GDIS y sus Dependencias

2.6 Sintesis de Recomendaciones

A continuación en la Tabla II se resumen las indicaciones que se recomiendan seguir al utilizar el proceso RUP-GDIS.

Tabla II: Recomendaciones

Modelos	Artefactos de software	Qué hacer?
Modelo Use Case del Negocio	Tabla de Eventos del Negocio	Identificar eventos de interés para los actores del negocio
	Diagrama UCN	UCN se identifican únicamente a partir de los eventos del negocio Evitar considerar funcionalidades de sistema –> pensar que todo se realiza manualmente Un trabajador del negocio no es un actor del negocio Un UCN no describe un proceso del negocio
	Plantilla para especificar use case	Eventos del negocio son entrada del actor indicar acciones/actividades que realiza el Negocio para atender un evento No usar términos que refieran a un sistema automatizado Usar predicados y lógica de primer orden

Modelos	Artefactos de software	Qué hacer?
		No colocar elementos que informa un actor Considerar propiedades de entidades del negocio
	Diagrama de clase	Identificar clases/objetos involucrados en las especificaciones de los use case Actores del negocio y trabajadores del negocio en principio no se modelan como clases del negocio Asegúrese que un objeto/clase existe en el sistema y tiene comportamiento ¿El trabajador del negocio tiene un comportamiento identificable en el dominio del problema? Esto es, ¿se pueden nombrar servicios o funciones necesarias en el dominio del problema que son propias del trabajador y que este provee? Indíquelos ¿El trabajador del negocio tiene relaciones con otros trabajadores del negocio? Analícelas e incorpórelas ¿El trabajador del negocio actúa dentro de los límites del sistema? Si no lo hace, puede ser un actor del sistema ¿El trabajador del negocio posee una estructura identificable? Esto es, ¿es posible identificar algún conjunto de atributos que el trabajador posee y que deben administrarse? Agréguelos
Modelo de Análisis / Diseño del Negocio	Diagrama de Actividad	Solo para los UCN donde el diagrama representa algo de valor Usar estereotipos No deben existir transiciones entre objetos de información
	Diagrama de secuencia	Un diagrama de secuencia por cada UCN No se genera si no existen interacciones entre dos o más objetos en el UCN
	Tabla de Eventos del Sistema	Utilizar especificaciones de UCN Trasladar acciones y/o actividades del flujo básico del negocio a la tabla Examinar y transformar usando terminología de sistema Generalizar o combinar eventos Identificar y analizar nuevos eventos a nivel de sistema Eventos se consideran operaciones en pseudo-lenguaje, cercanos al lenguaje de programación
Modelo Use Case	Diagrama use case	UC se identifican a partir de los eventos del sistema Actores del sistema pueden ser actores del negocio o trabajadores del negocio Quién inicia un UC? Quién participa en el UC? No usar relaciones entre UC sino después de varias iteraciones No debería asociarse más de un actor con un use case que tiene un propósito especifico
	Plantilla para especificar use case	Utilizar como base las especificaciones de los UCN Incluir funcionalidades de sistema
Modelo de Análisis	Diagrama de clase	Clases entidad del negocio pueden representarse como atributos de otras clases Qué tiene que hacer el sistema, no cómo se hace Atributos de clases interfaz se identifican a partir del flujo básico y/o alternativo de las especificaciones Clases control se definen para evitar asociar clases interfaz con clases entidad Decisiones respecto al orden en que se instancian las clases interfaz, o acciones a realizar cuando un elemento gráfico se presiona deben implementarse en clases control Validaciones de datos capturados por un objeto interfaz pueden definirse en su clase interfaz
	Diagrama de secuencia	Un actor del sistema puede enviar mensajes solo a objetos interfaz Identificar nuevas asociaciones y operaciones
Modelo Mapa de Navegación	Prototipos	No agregar botones para capturar data de *text field*
	Diagrama de estado	Nombrar estados según acción No referir a elementos gráficos en el nombre de eventos en transiciones Usar transiciones internas en los estados para el ingreso de valores
Modelo de Diseño	Diagrama de clase	Agregar propiedades a clases entidad, interfaz y control Incorporar métodos Especificar tipo, estructuras de datos y visibilidad Usar estereotipos Usar Patrones de diseño Incorporar notas con pseudocódigo en los métodos de clases
	Diagrama de secuencia	Solo si se especifican intercambios de mensajes que no se especifican en el UC correspondiente
Modelo de Implemen-tación	Código fuente	Respetar el estándar de codificación
	Diagrama de paquete	Definir un criterio para agrupar clases
Modelo de Prueba	Especificación de casos de prueba	Considerar pre y post condiciones especificadas en los UC Utilizar pruebas caja blanca y caja negra

3. PERSPECTIVAS

El grupo docente de la asignatura Ingeniería de Software de la Licenciatura en Computación debe enfrentar el reto de plantear de manera consciente el "qué", "por qué", "cuándo" y "cómo"

realizar las diferentes actividades de un proceso de desarrollo de software. En este trabajo se describió el proceso de desarrollo RUP-GDIS que ha sido y está siendo utilizado para el desarrollo del proyecto de la asignatura desde el semestre I-2011. RUP-GDIS es una configuración de RUP que se centra en sus primeras cinco disciplinas.

A través de la utilización de RUP-GDIS durante un semestre se estudian seis diagramas UML a saber: diagrama use case, diagrama de actividad, diagrama de secuencia, diagrama de clase, diagrama de estado y diagrama de paquete, que se correlacionan con los objetivos de la asignatura, siendo la premisa subyacente que el desarrollo del proyecto provee a los estudiantes de una base para el desarrollo de futuros proyectos. La evolución del proyecto a lo largo de varios semestres fomenta en los estudiantes la habilidad de desarrollar componentes que deben incorporar en un software existente, promoviendo asimismo técnicas de resolución de problemas que podrán aplicar en situaciones y problemas similares.

La utilización de RUP-GDIS se evalúa al final de cada semestre con un cuestionario anónimo para identificar los elementos en los cuales los estudiantes presentan dificultades. Es importante destacar que a medida que transcurren los semestres el proceso está sujeto a modificaciones en base al análisis de las respuestas que indican los estudiantes en el cuestionario.

Por ejemplo, para identificar las dificultades al elaborar diagramas de clase, se presentan las siguientes alternativas: a) Identificar clases; b) Identificar asociaciones; c) Uso de Generalización/Especialización; d) Evitar asociaciones redundantes; e) Especificación de multiplicidades en asociaciones; f) Identificar atributos en las clases; g) Identificar operaciones en las clases; h) Otras razones, especifique; i) NO tuvo dificultades.

La evaluación de los resultados del cuestionario permite determinar las dificultades que enfrentan los estudiantes al utilizarlo y guía las posibles modificaciones que pueden ser pertinentes de realizar tanto al proceso como a las actividades de docencia. Las evaluaciones realizadas en los semestres I-2012 y II-2012 pueden ser consultadas en Niño [17] y en los informes de docencia correspondientes a los semestres II-2014 y I-2015 que se encuentran en el Departamento de la Escuela de Computación. En el Apéndice A4 se presentan las modificaciones más significativas que se han realizado al proceso. Por ejemplo, se amplió la explicación del concepto "evento del negocio" en la disciplina Modelado del Negocio y se incluyó una estrategia para identificar eventos del sistema en la disciplina de Requerimientos. Las modificaciones al proceso definido originalmente en el año 2011 se realizaron con la finalidad de resolver las dificultades que enfrentan los estudiantes con el concepto de evento, en particular evento del negocio, así como aportar estrategias, lineamientos y recomendaciones para la elaboración de ciertos artefactos de software.

El resultado de este trabajo es un proceso de desarrollo de software definido y documentado, a disposición en la Escuela de Computación, proceso que actualmente se utiliza sistemáticamente en la asignatura Ingeniería de Software. Esperamos que esta experiencia aporte a los estudiantes conocimientos y habilidades esenciales para su desempeño en sus estudios y en su vida profesional.

REFERENCIAS

[1] S. Ambler, *A Manager's Introduction to The Rational Unified Process (RUP)*, 2005

[2] H. Admiraal, *Pitfalls using UML in RUP*, 2007

[3] H. Erickson and M. Penker, *UML Toolkit*, John Wiley & Sons, INC. 1998

[4] Essi-Scope, *Quality Characteristics*, http://www.cse.dcu.ie/essiscope/index.html

[5] E. Gamma; R. Helm; R. Johnson and J. Vlissides, *Dessign Patterns. Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995

[6] Rational Unified Process, *Rational Unified Process. Best Practices for Software Development Teams*, Rational Software Corporation White Paper, TP026B, Rev 11/01, 2011

[7] http://people.cs.uchicago.edu/~matei/CSPP523/lect4.ppt

[8] http://www.ibm.com/developerworks/library/ws-soa-term2/index.html

[9] I. Jacobson; G. Booch and J. Rumbaugh, *El Proceso Unificado de Desarrollo de Software*, Addison-Wesley, 2000

[10] http://www.featuredrivendevelopment.com

[11] D. Wells, *Extreme Programming: A Gentle Introduction*, http://www.extremeprogramming.org, 2009

[12] K. Schwaber and J. Sutherland, *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*, Scrum.org., 2011, http://www.scrum.org

[13] Java™, http://www.java.com/en

[14] NetBeans, http://netbeans.org

[15] C. Larman, *Chapter 6: Use-Case Model: Writing Requirements in Context* en *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Designs and The Unified Process*, Second Edition, Prentice Hall, 2001

[16] C. Metzner and N. Niño, *El Proceso de Desarrollo RUP-GDIS*, Lecturas en Ciencias de la Computación, Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela, ISSN 1316-6239, ND 2012-03, pp 1–14, 2012

[17] N. Niño, *Evaluación Cuantitativa de la Enseñanza y el Aprendizaje de un Proceso de Desarrollo de Software en el Pregrado de la Licenciatura en Computación, UCV*, Universidad Central de Venezuela, Trabajo de Ascenso para optar a la categoria de Asociado, 2014

APÉNDICES

En esta sección se incluyen en el Apéndice A1 la plantilla para especificar tanto los use case del negocio como los use case del sistema. En el Apéndice A2 se presentan algunos elementos del estándar de codificación a usar en la disciplina *Implementación*. En el Apéndice A3 se presenta la plantilla utilizada para especificar casos de prueba y en el Apéndice A4 se presentan las modificaciones realizadas a la publicacion ND-2012-03 [16].

Apéndice A1: Plantilla para Especificar Use Case

1. **Identificador y Nombre:** UC[N]# - Nombre del Use Case
   1.1. **Breve Descripción:** descripción y propósito
   1.2. **Actores:** lista de actores que participan. A nivel de sistema se distingue entre actores primarios y actores secundarios, a nivel de negocio no se hace esta distinción
   1.3. **Flujo de Eventos:**
      1.3.1. **Flujo Básico:** describe el proceso normal

	Entrada del actor	Respuesta del Negocio / Sistema
1.-	Eventos que realiza un actor cuando interactúa con el Negocio	Secuencia de acciones o actividades que realiza el Negocio para atender un evento que genera el actor
2.-	…	…

		El use case finaliza

1.3.2. **Flujos Alternativos:** describe procesos alternativos
Alternativa 1: nombre de la alternativa. Nota: las pre y post condiciones se satisfacen solo para el flujo básico

	Entrada del actor	Respuesta del Negocio / Sistema
1.-		Secuencia de acciones o actividades que realiza el Negocio cuando ocurre la condición bajo la cual se realiza el flujo alternativo

1.4. **Requerimientos Especiales:** identifique requerimientos adicionales. Incorpore en esta sección requerimientos no funcionales
1.5. **Pre-condición:** predicados que deben satisfacerse antes de realizar el use case
1.6. **Post-condición:** predicados que se satisfacen cuando el use case finaliza exitosamente (flujo básico)
1.7. **Puntos de Extensión:**
1.7.1. **Include:** lista de identificador y nombre de los use case con los cuales mantiene relación de inclusión
1.7.2. **Extend:** lista de identificador y nombre de los use case con los cuales mantiene relación de extensión

Apéndice A2: Estándar de Codificación

El estándar de codificación utilizado en cuanto a nombramiento de clases, métodos y atributos es:

- clases se colocan tipo título, por ejemplo: AgendaDeCitas
- clases control se inician con la palabra Ctrl, por ejemplo: CtrlCentral
- la primera letra de los atributos y métodos se coloca en minúscula
- el nombre de la asociación entre clases se coloca con la inicial de las tres o cuatro primeras letras de la clase y en minúscula.

Apéndice A3: Plantilla para Especificar Casos de Prueba

Id Prueba: #
Tipo de Prueba: se indica el tipo de prueba a ser considerada
Descripción: breve descripción del propósito de la prueba
Clase: nombre de la clase a ser probada
Método: signatura del método a ser probado
Tipo de retorno: tipo de retorno del método
Pre-condición: predicados que deben satisfacerse antes de realizar la prueba
Post-condición: predicados que deben satisfacerse cuando se realiza la prueba
Casos de prueba: conjunto de datos con los que se va a ejecutar el software
Valor esperado: valor(es) que se debe(n) obtener después de la ejecución
Resultado: valor(es) que se obtienen como salida una vez realizada la prueba (experimentación)

Apéndice A4: Modificaciones

Sección	Descripción
2.1	Se amplió la explicación del concepto “evento del negocio”
2.1	Se modificó el orden en que se generan los diagramas del Modelo de Análisis / Diseño del Negocio
2.2	Se incluyó una estrategia para identificar eventos del sistema
2.3	Se modificó el orden en que se generan los modelos en la disciplina de Análisis y Diseño
2.3	Se destaca la utilidad de generar pseudocódigo
2.3	Se incluyó un lineamiento respecto al envío de mensajes de un actor
2.3	Se incluyó un lineamiento para identificar métodos en la disciplina Análisis y Diseño
2.5	Se indican los tipos de pruebas que utilizan los estudiantes de la asignatura
Tabla I	Se modificó el orden en que se generan los artefactos de software de los modelos en la disciplina Modelado del Negocio y en la disciplina Análisis y Diseño
Figura 3	Se modificaron etiquetas y dependencias entre los modelos Mapa de Navegación y Modelo de Análisis