Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

El Proceso de Desarrollo RUP-GDIS

Christiane Metzner
Norelva Niño

ND 2012-03

Centro de Ingeniería de Software y Sistemas (ISYS)
Caracas, septiembre 2012

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239
ND 2012-03

El Proceso de Desarrollo RUP-GDIS

Christiane Metzner y Norelva Niño

Centro de Ingeniería de Software y Sistemas (ISYS), Escuela de Computación,
Facultad de Ciencias, Universidad Central de Venezuela
Apartado 40388, Los Chaguaramos, Caracas 1041-A, Venezuela
christiane.metzner@ciens.ucv.ve, norelva.nino@ciens.ucv.ve

Resumen

En este trabajo se describe el proceso de desarrollo de software utilizado a partir del semestre I-2011 en la asignatura de pregrado Ingeniería del Software para el desarrollo de los proyectos. El proceso de desarrollo que se presenta, denominado RUP-GDIS, es una configuración de RUP centrado en las primeras cinco disciplinas correspondientes al núcleo de RUP y se clasifica como un proceso pesado y prescriptivo considerando el número de artefactos de software que se generan. Se basa por una parte en RUP y por otra parte en el trabajo de Hans Admiraal, en el cual se analizan algunas debilidades, ambigüedades o dificultades presentes en el uso de RUP aportando recomendaciones acerca de como tratarlas y superarlas.

Palabras Claves: Proceso de Desarrollo de Software, meta-proceso RUP, artefactos de software.

Contenido

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, Ambiente. 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.

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. El propósito de cada una de las fases se resume a continuación [6]:

- Incepción: establecer una visión general para el negocio, alcance, estimar el 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 iterativamente los requerimientos en orden de prioridades establecidas, preparar la instalación.
- Transición: finalizar, instalar y entregar la versión del sistema. Realizar las pruebas de aceptación del usuario. Examinar el sistema entregado y evaluar que partes satisfacen la visión de acuerdo al documento de visión, desde la perspectiva del negocio.

En cada iteración se realizan las actividades correspondientes a la mayoría o 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 sistema de software con el fin de describir lo que el sistema debe hacer.
- Análisis y Diseño: trasladar los requerimientos a una arquitectura de software con el fin de guiar la implementación del sistema.
- 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 subsistemas de implementación. Implementar clases y objetos en términos de componentes.
- Prueba: realizar una evaluación objetiva del sistema [1]. Esto incluye encontrar y corregir errores, validar que el sistema 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.

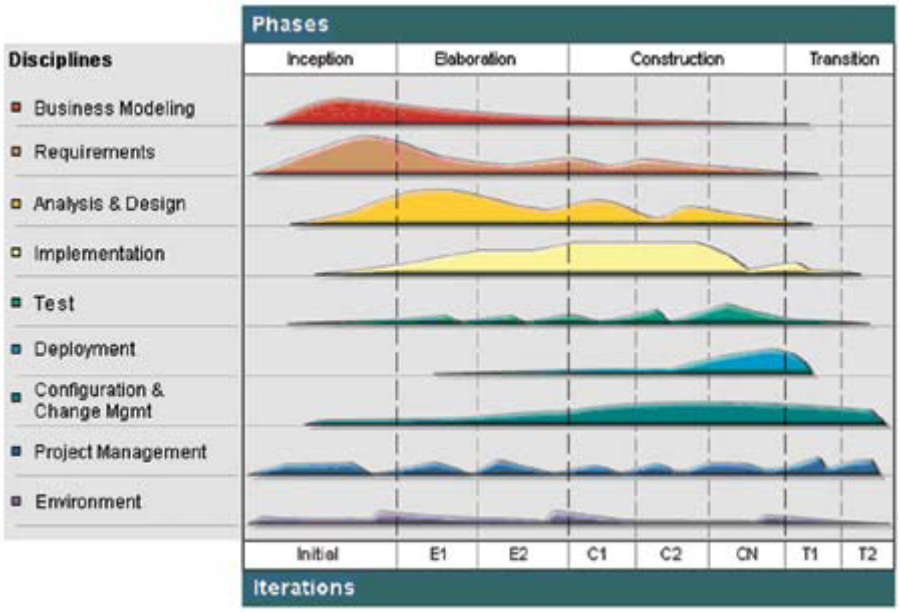

Figura 1. Disciplinas y fases en RUP. Fuente: [8]

La idea central de las dos dimensiones en RUP es que el desarrollo de un sistema consiste en realizar una serie de *releases* 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 usuario. Cada incremento es el resultado de la iteración a lo largo de cada disciplina. Cada *release* es un sistema que opera. Al enseñar los fundamentos de RUP si se comienza por las fases, usualmente los estudiantes caen en el error de pensar las fases tal como se definen en el modelo de cascada, solo con nombres diferentes. Si se comienza por las iteraciones mostrando como un proyecto 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 resultado final (*release*) entonces esta idea es más fácil de comprender. Lo central son entonces las iteraciones y la estructura de un proyecto y las fases son una conveniencia general para su control.

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 la unidad en *Deployment* se prepare y se entregue para su instalación. El proceso como tal de distribuir e instalar las unidades en *Deployment* no forma parte de RUP, siendo usualmente del dominio de un Departamento de Operaciones y no del grupo de desarrollo. Lo importante es definir el proceso a seguir por el equipo de desarrollo para generar los artefactos de instalación y producir un documento que describa la versión, colocándolo a disposición del Departamento de Operaciones, quien es responsable de la distribución e instalación del sistema final.

En la siguiente sección se especifica el proceso de desarrollo, fundamentado tanto en RUP [6] como en la publicación de Admiraal [2], utilizado a partir del semestre I-2011 en la asignatura de pregrado Ingeniería del Software para el desarrollo de los proyectos.

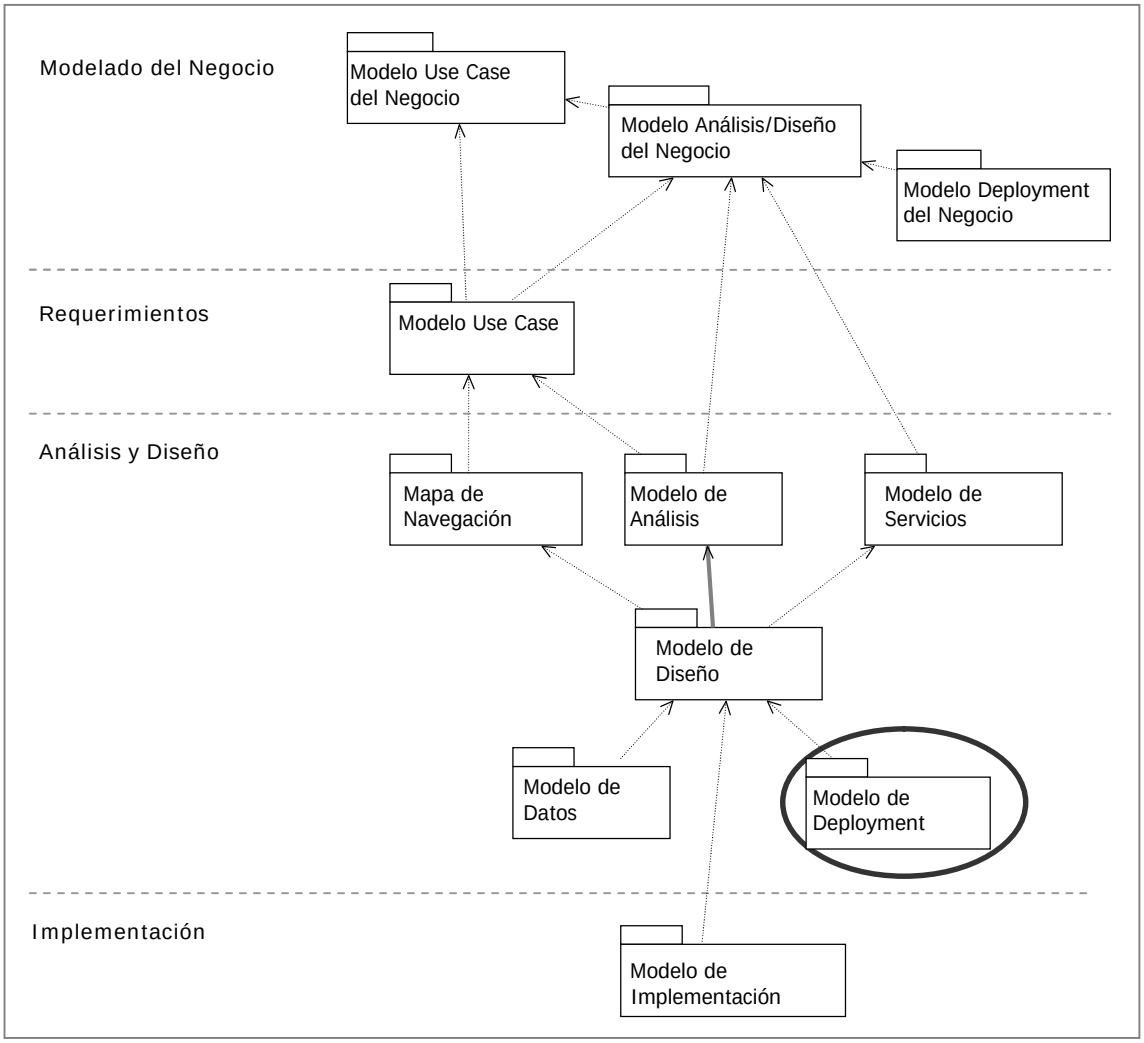

Figura 2. Modelos de RUP y dependencias entre los modelos según Admiraal.

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 tratar de reducir riesgos, como por ejemplo: retraso en las fechas pautadas para las entregas parciales o del sistema final, incumplimiento en lo estipulado a entregar, exceder los costos del proyecto, requerimientos incompletos o el producto de software no cumple con ciertas características de calidad establecidas.

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 cliente 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 la industria, como por ejemplo 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 del Software, siendo esta la primera asignatura en la cual los estudiantes adquieren conocimientos en el área. Adicionalmente, el proceso de desarrollo debe cubrir el contenido del temario en base a los objetivos de la asignatura considerando las limitaciones mencionadas.
El proceso de desarrollo utilizado que se describe a continuación se centra en las disciplinas del núcleo de RUP y se indican los artefactos que se generan para cada modelo considerando las recomendaciones de Admiraal. 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 físico de información usado o producido durante un proceso de desarrollo de software y es el resultado de una actividad realizada por un rol o actor. Ejemplos de artefactos incluyen modelos, código fuente, código ejecutable.

2.1. Disciplina de Modelado del Negocio

Los estudiantes deben utilizar esta disciplina para cualquier aplicación de tipo empresarial. En la disciplina se incorpora el artefacto denominado "Eventos del Negocio" que no está considerado en RUP y se definió con el fin de extraer los eventos de interés a partir de una descripción textual del funcionamiento de un negocio. Este artefacto les facilita a los estudiantes la tarea de identificar los requerimientos funcionales del negocio desde la perspectiva del usuario (clientes del negocio). Los eventos identificados se registran en una tabla de eventos conformada por dos columnas: identificador de evento y la descripción del Evento del Negocio. En base a la tabla de eventos se generan los siguientes modelos de RUP:

- **Modelo Use Case del Negocio**: los use case del negocio presentan lo que el negocio ofrece a sus clientes. El negocio decidirá como realizar sus use case del negocio: puede realizarlos todos manualmente, automatizarlos parcial o totalmente. El modelo describe las interacciones externas de la organización en términos de los use case del negocio: que deben realizar tanto el negocio como los actores del negocio para llevar a buen termino una transacción. 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 cambian sobre periodos extendidos de tiempo si el Negocio cambia 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, las relaciones entre los use case del negocio y sus relaciones con los actores del negocio. Un use case del negocio describe lo que el negocio ofrece a sus clientes y es independiente de las tecnologías. Una técnica que permite evitar funcionalidades de sistema en los use case del negocio consiste en considerar que el negocio usa implementaciones tradicionales como archivadores, carpetas, papel. Si un use case del negocio es independiente de un sistema computarizado es posible implementar la solución utilizando solo estos artefactos.

      Un trabajador del negocio no es un actor del negocio porque forma parte de la organización. Es el representante que el actor del negocio contacta y que se transformará en un actor del sistema en los use case de sistema, donde utiliza el sistema como herramienta para prestarle un servicio a los clientes. 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 a los eventos registrados en la tabla de eventos. Los pasos de un proceso de negocio se realizan en uno ó más use case de sistema en una ó más aplicaciones y el actor del negocio utiliza los use case de sistema para completar el proceso.

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 para especificar los 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 Actividad***: se usa principalmente para modelar el flujo de trabajo y es útil para analizar los use case describiendo las acciones que necesitan realizarse, cuando se realizan y quien es el responsable de realizarlas; no da el detalle de como 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 estas contienen las acciones. Los documentos y productos se denominan entidades del negocio. Las entidades del negocio son las entradas y / o salidas de las acciones. Las notas se utilizan para indicar que algo se genera en una acción para un actor 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 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. Los trabajadores del negocio se representan con el icono de actor y el estereotipo de *<<user>>* e intercambian mensajes con las entidades del negocio.
  - ***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. Las entidades del negocio que son manipuladas en un flujo de trabajo pueden representarse como clases del negocio. Es de destacar que el modelado OO específicamente es el modelado de comportamiento. Los objetos existen en un sistema computacional y están sujetos a restricciones y acciones del sistema, poseen un ciclo de vida: se construyen 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 actúa dentro de los limites del sistema? Si no lo hace, puede ser un actor del sistema.
    - 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?
    - 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?
    - El trabajador del negocio tiene relaciones con otros trabajadores 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 "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 (cierta característica 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, los diagramas de actividad y de secuencia. Se utiliza la tabla de eventos para registrar los eventos del sistema desde la perspectiva del usuario y está conformada por siete columnas: identificador de evento, Descripción del Evento del Sistema, Restricciones, Actor, Prioridad, identificador del UCN con el cual se relaciona e identificador del UC del sistema que 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 desde 1 hasta el número máximo de eventos en el sistema. El valor 1 se asocia al evento de mayor importancia para el negocio (o para el desarrollo si el elemento es necesario para 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). Se utilizan los siguientes diagramas:
    - ***Diagrama Use Case del sistema***: describe lo que debe hacer el sistema para automatizar uno ó 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 alcanza la meta del use case, un actor secundario participa en lograr la meta. Si se identifica un use case sin actor, esto es el resultado de una descomposición funcional. Tampoco debería asociarse mas de un actor con un use case debido a que la especificación se redacta desde la perspectiva de uno solo que tiene una meta especifica. Tratar de escribir 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. Para identificar los use case del sistema se utiliza la tabla de eventos y generalmente la correspondencia no es uno a uno; una acción en el diagrama de actividad del Modelo de Análisis / Diseño del Negocio puede corresponderse con un use case.

      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 usuario. 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 para especificar los use case del sistema. (ver Apéndice A1).

      El diagrama de actividad describe las actividades de un actor o conjunto de actores; el 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" para 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 del Software. Por lo tanto, consideramos conveniente desarrollar el Modelo de Análisis y posteriormente el Modelo de Diseño. En el proceso de desarrollo a partir del semestre I-2012 se realiza tanto el Modelo de Análisis como el Modelo de Diseño. Se generan los siguientes modelos:

- **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. Se utilizan los siguientes artefactos:
    - ***Prototipos***: muestra los elementos de la interfaz gráfica de las ventanas del sistema. Los prototipos pueden generarse con una herramienta o manualmente. Un prototipo está enfocado a un actor humano, dado que presenta a los elementos gráficos de las ventanas del sistema, y no a un actor no humano. Por lo tanto, el artefacto Prototipos es de interfaz usuario. 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>>.
    - ***Diagrama de Estado***: representa la secuencia de navegación entre las ventanas del sistema. Las ventanas del sistema se representan con los estados en el diagrama y las transiciones representan los posibles caminos de navegación.
- **Modelo de Análisis**: se analizan y refinan los requerimientos que han sido descritos en el modelo use case para alcanzar una visión detallada de los requerimientos del sistema. El modelo de análisis se describe en un lenguaje para los desarrolladores y proporciona una visión general y conceptual del sistema respecto a lo que se tiene que hacer y no como se va a hacer [7]. Por este motivo es que resulta ser un modelo útil y conveniente en caso de que se incorporen nuevos desarrolladores en un equipo de desarrollo ya que les facilita comprender el sistema sin que existan detalles de alternativas de diseño que pueden variar y están atadas al ambiente de implementación. El Modelo de Análisis se considera como una primera versión 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, se etiquetan las clases con un estereotipo para categorizar las clases como interfaz, entidad o de 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 del estereotipo 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 los prototipos del sistema. Por lo general, cada prototipo de interfaz del sistema se corresponde con una clase interfaz. Las clases interfaz se utilizan en el Modelo de Análisis para modelar las interacciones entre el sistema y sus actores.

      Las clases de control se definen para evitar que las clases interfaz tengan relación de asociación con las clases entidad. Decisiones respecto al orden de despliegue de los prototipos o acciones a realizar cuando un elemento gráfico se presiona, por ejemplo un botón, deben ser implementadas por las clases de control y no por las clases interfaz. Sin embargo las validaciones de datos capturados por un objeto interfaz pueden definirse en su clase interfaz. Las clases de control se indican con el estereotipo *<<control>>* o *<<ctrl>>*.
    - ***Diagrama de Secuencia***: representa el orden de invocación de operaciones 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, descubrir las asociaciones y las operaciones que se requieren.
- **Modelo de Diseño a nivel de componentes**: un componente es un segmento de código compilado y puede contener especificaciones de interfaz, clases que implementan un conjunto de interfaces o clases que utilizan ciertas interfaces. Son un mecanismo que permiten reducir relaciones entre clases mediante el agrupamiento de clases cohesivas y limitando las dependencias con interfaces. En el contexto del proceso RUP-GDIS una componente se refiere a una clase. El lenguaje de programación utilizado en la asignatura Ingeniería del Software es *Java*™ [13] con el ambiente de desarrollo *NetBeans* [14]. El Modelo de Diseño especifica el diseño detallado de las clases.

  Los diagramas a generar son:

- ***Diagrama de Clase***: se le incorporan detalles a las clases entidad, interfaz y de control. Se incorporan los tipos de datos y las visibilidades de los atributos y operaciones a las clases del Modelo de Análisis, se agregan nuevas clases no identificadas previamente. Una operación de clase del Modelo de Análisis se convierte en uno ó más métodos de la clase en el Modelo de Diseño. Atributos 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 ambiente *NetBeans* se puede etiquetar con el estereotipo *<<form>>*. Si se utilizan patrones de diseño [5], generalmente se agregan atributos, operaciones, relaciones y eventualmente clases para dar solución a un problema específico de diseño. Puede ser de utilidad incorporar notas con pseudocódigo en las operaciones de clases.
- ***Diagrama de Secuencia***: son más detallados que los diagramas de secuencia que se generan en el Modelo de Análisis. En el diseño, se muestran los intercambios de mensajes entre los objetos de diseño.

2.4. Disciplina de Implementación

Se establece el estándar de codificación en cuanto a nombramiento de clases, métodos y atributos que se presenta en el Apéndice A2. Los subsistemas de implementación son los paquetes (*package*) que agrupan las componentes siguiendo algún criterio y las componentes son las clases (archivos con extensión *.java)*. 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*.

2.5. Disciplina de Prueba

De acuerdo con RUP se distinguen 4 tipos de prueba: unitaria, de integración, de sistema y de aceptación. Las pruebas consideradas en el proceso RUP-GDIS son: unitarias, de integración o a nivel del sistema. Se realiza el siguiente modelo:

- **Modelo de Prueba**: describe las pruebas del código. Debe indicarse el identificador de clase o componente, el identificador del caso de prueba, su descripción, y un reporte del resultado de la prueba.
  - ***Especificación de Casos de Prueba***: describe cuales son los datos con los que se ejecuta el caso de prueba (ver Apéndice A3).

En la Tabla 1 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.

Tabla 1. Resumen del proceso RUP-GDIS.

Disciplina	Modelos a generar en la disciplina	Artefactos de software utilizados	Observación
Modelado del Negocio		- Eventos del Negocio	
	Modelo Use Case del Negocio	- Diagrama use case - Plantilla para especificar los use case o - Diagrama de actividad	La especificación de cada use case del negocio se realiza utilizando la plantilla o el diagrama de actividad.
	Modelo de Análisis/Diseño del Negocio	- Diagrama de actividad - Diagrama de secuencia - Diagrama de clase	
Requerimientos		- Eventos del Sistema	
	Modelo Use Case	- Diagrama use case. - Plantilla para especificar los use case o - Diagrama de actividad	La especificación de cada use case se realiza utilizando la plantilla o el diagrama de actividad.

Disciplina	Modelos a generar en la disciplina	Artefactos de software utilizados	Observación
Análisis y Diseño	Modelo de Mapa de Navegación	- Prototipos - Diagrama de estado	
	Modelo de Análisis	- Diagrama de clase - Diagrama de secuencia	
	Modelo de Diseño a nivel de componentes	- Diagrama de clase - Diagrama de secuencia	
Implementación	Modelo de Implementación	- Código fuente documentado - Diagrama de paquete	
Prueba	Modelo de Prueba	- Especificación de casos de prueba	Identificador de clase o componente, identificador de caso de prueba y su descripción, reporte del resultado de la prueba.

En la Figura 3 se resumen los modelos considerados en el proceso RUP-GDIS y las dependencias entre los modelos y las disciplinas.
En el proceso RUP-GDIS no se considera a la disciplina de *Deployment* dado que un sistema 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.

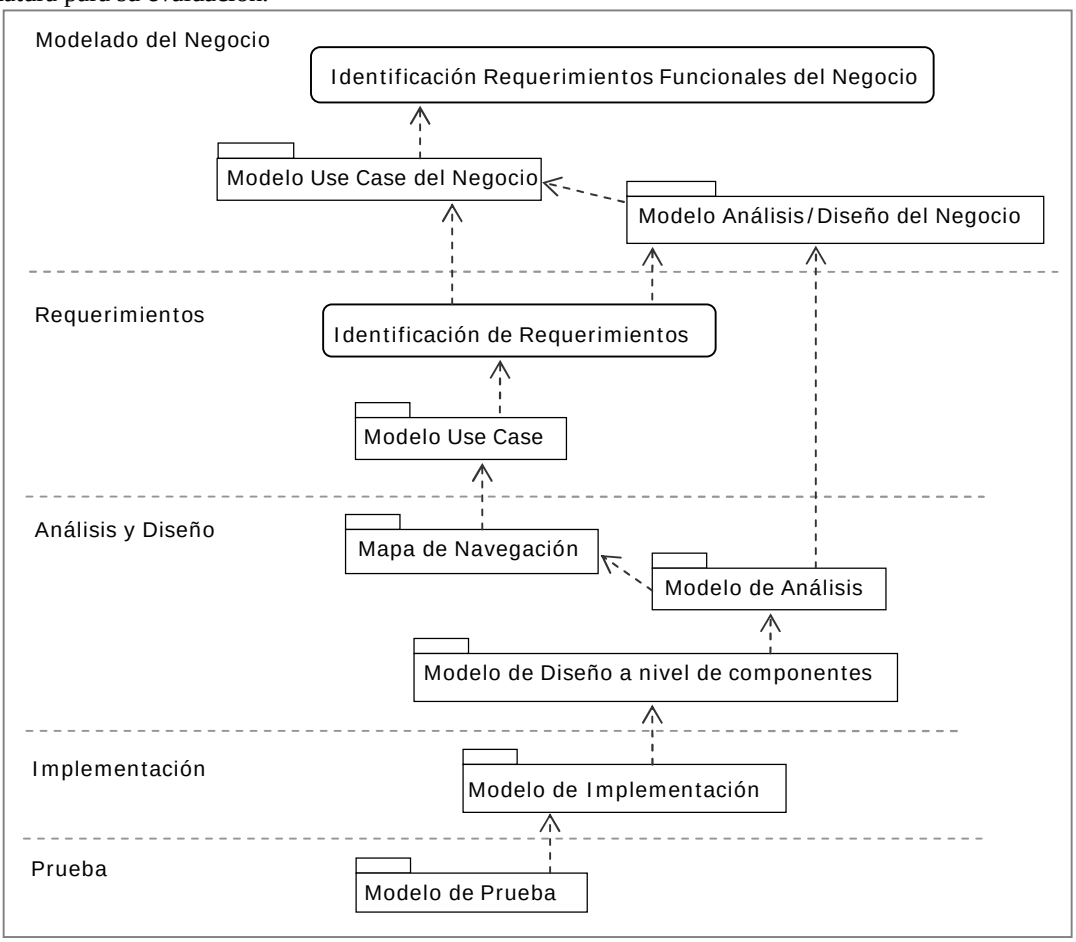

Figura 3. Resumen de los modelos del proceso RUP-GDIS y dependencias entre los modelos.

Es importante entender que no solo las iteraciones del proceso de desarrollo se realizan recorriendo a las disciplinas a lo largo de las fases sino que además, las fases se recorren a lo largo de cada disciplina.

3. Perspectivas

El grupo docente de la asignatura Ingeniería del Software de la Licenciatura en Computación debe enfrentar el reto de plantear de manera conciente el "que", "porque", "cuando" y "como" 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 software existente promoviendo asimismo técnicas de resolución de problemas que podrán aplicar en situaciones y problemas similares. Esperamos que esta experiencia aporte a los estudiantes de conocimientos y habilidades esenciales para su desempeño en sus estudios y en su vida profesional.

Un estudio de caso actualmente en desarrollo permitirá establecer los factores que afectan el éxito en la utilización de RUP-GDIS y las posibles dificultades que encuentran en su aplicación.

Referencias

[1] Scott Ambler: "A Manager's Introduction to The Rational Unified Process (RUP)". 2005

[2] Hans Admiraal: "Pitfalls using UML in RUP". 2007

[3] Hans-Erick Erickson; Magnus Penker: "UML Toolkit". John Wiley & Sons, INC. 1998

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

[5] Erich Gamma; Richard Helm; Ralph Johnson; John 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] Disponible en: http://people.cs.uchicago.edu/~matei/CSPP523/lect4.ppt

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

[9] Ivar Jacobson; Grady Booch; James Rumbaugh: "El Proceso Unificado de Desarrollo de Software". Addison-Wesley. 2000

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

[11] Don Wells: "Extreme Programming: A gentle introduction", http://www.extremeprogramming.org/, 2009

[12] Ken Schwaber; Jeff 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/

Apéndice

Apéndice A1: Plantilla para especificar los use case del negocio y del sistema.

1. Identificador y Nombre: UC[N]# - Nombre Del Use Case

1.1. Breve Descripción: breve descripción del use case indicando lo que le permite hacer a los actores.

1.2. Actores: lista de actores que interactúan con el use case. Puede ser relevante distinguir entre actores primarios y actores secundarios.

1.3. Flujo de Eventos:

1.3.1. Flujo Básico

	Entrada del actor	**Respuesta del Negocio / Sistema**
1.-	Se describe la secuencia de pasos que realiza un actor cuando interactúa con el use case.	Por lo general, el primer paso en el flujo indica cuando se inicia el use case.
2.-	**…**	El último paso en el flujo indica cuando finaliza el use case.

1.3.2. Flujos Alternativos

Alternativa 1: nombre de la alternativa. Nota: en general en las publicaciones sobre RUP no se alerta sobre el

hecho de que Flujos Alternativos pueden causar que la(s) post-condicione(s) del use case sean diferentes a las post-condiciones del flujo básico.

	Entrada del actor	Respuesta del Negocio / Sistema
1.-	Se describe la secuencia de pasos eventuales que realiza un actor cuando interactúa con el use case.	

1.4. **Requerimientos Especiales:** identifique cualquier requerimiento adicional e incorpore en esta sección requerimientos no funcionales.
1.5. **Pre-condición:** condiciones que deben satisfacerse antes de realizar el use case.
1.6. **Post-condición:** condiciones que se satisfacen cuando el use case finaliza.
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 en cuanto a nombramiento de clases, métodos y atributos es:

- las clases se colocan tipo titulo, por ejemplo: *AgendaDeCitas*
- las clases de control se inician con la palabra *Ctrl*, por ejemplo: *CtrlCentral*
- los atributos y métodos se inician en minúscula la primera letra
- los roles de asociación que representan las relaciones entre clases, se inicia con la palabra *rol* y a continuación el tipo de la clase de la cual se va a guardar la referencia.

Apéndice A3: Plantilla para especificar los casos de prueba.

Id Prueba: #
Tipo de Prueba: se indica si el tipo de prueba a ser considerada es unitaria, de integración o de sistema.
Descripción: breve descripción del propósito de la prueba.
Clase: nombre de la clase que va a ser probada.
Método: signatura del método a ser probado. **Tipo de retorno:** tipo de retorno del método.
Pre-condición: condiciones que deben satisfacerse antes de realizar la prueba.
Post-condición: condiciones 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).