RISI 7(1), 27 - 36 (2010)
Revista de Investigación de Sistemas e Informática
Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos
ISSN 1815-0268 (versión impresa)
ISSN 1816-3823 (versión electrónica)

Documentación de Proyectos Con UML 2

Percy E. De la Cruz Vélez de Villa, Daniel Bravo Loayza, Marlene Reyes Huamán

Universidad Nacional Mayor de San Marcos
Facultad de Ingeniería de Sistemas e Informática

pdelacruzv@unmsm.edu.pe, eliasdanielbl@hotmail.com, marcieloreyes@gmail.com

RESUMEN

UML es el lenguaje de modelado estándar en la actualidad, definido por la OMG. Empleado para la documentación de proyectos, actualmente se encuentra en la versión 2, y un número de revisiones menores. Se encuentra organizado en una estructura jerárquica de dos grandes grupos y trece diferentes diagramas, no dispone de una metodología de desarrollo, por lo cual se apoya en metodologías de terceros para el desarrollo de proyectos. Aún cuando se encuentra abundante información acerca de UML, resulta difícil identificar cómo se documenta el desarrollo de sistemas en una propuesta integral. En particular, en las tres primeras fases del desarrollo del proyecto, donde los modelos son muy importantes.

Este artículo pretende mostrar un método para documentar proyectos, y hacer uso de los diagramas del UML 2 para un proyecto de software, bajo las metodologías del UP y del RUP.

Palabras clave: UML 2, UP, RUP.

ABSTRACT

UML modeling language is the standard defined by the OMG. Used for project documentation, is currently in version 2, and a number of minor revisions. It is organized in a hierarchical structure of two groups and thirteen different diagrams, does not have a development methodology, so it relies on third methodologies for development projects. Even when it is abundant information about UML, it is difficult to identify as documented systems development in a comprehensive proposal. In particular, in the first three phases of project development, where the models are very important.

This article demonstrates a method to document projects, and use UML 2 diagrams for a software project, under the methodologies of UP and RUP.

Key words: UML 2, UP, RUP.

1. INTRODUCCIÓN

UML es el lenguaje visual estándar para el modelado de sistemas software. En su nueva versión, UML 2 ofrece nuevas características para facilitar estas tareas, pues ha sido mejorado en numerosas áreas para facilitar la tarea de expresar con mayor claridad y exactitud lo que se desea. Debido a que los proyectos que se abordan con UML muchas veces son amplios y complejos, las soluciones y modelos que se diseñan para la solución pueden llegar a ser también complejos.

Para que ello no llegue a ser una dificultad, se deben conocer los nuevos elementos y características del lenguaje, y que cambios han ocurrido.

UML está diseñado para su uso en proyectos de software

Una primera dificultad para el uso adecuado del UML 2 es identificar, de entre los trece diagramas propuestos por la OMG, cual o cuales son más adecuados o útiles para el desarrollo del proyecto. Este proceso de selección, priorización y énfasis en determinados diagramas del UML 2 a lo largo de las distintas fases y etapas del desarrollo del proyecto software se realizará con el apoyo de las metodologías RUP y UP que constituyen el principal aporte del presente artículo.

Los diagramas fueron modelados en StarUML 2.

2. MARCO TEÓRICO

UML 2

UML 2 presenta una numerosa sintaxis visual nueva. Parte de esta reemplaza y aclara la sintaxis 1.x existente y parte de ella es completamente nueva y representa una nueva semántica añadida al lenguaje [Arlow+2006].

Uno de los cambios más obvios es la introducción de nuevos tipos de diagramas. Diagrama de objetos y diagrama de paquetes que fueron dibujados en versiones anteriores del UML, pero que ahora son oficiales. UML 2 cambia el nombre de diagrama de colaboración a diagrama de comunicación. Asimismo, UML introduce nuevos tipos de diagramas: Diagramas de interacción, diagramas de tiempos y diagramas de estructura compuesta [Fowler2004]. Constituye la primera evolución importante desde la aparición de UML en 1997 [Debrauwer+2005].

Aunque UML 2 realiza muchos cambios sintácticos a UML, comparado con UML 1.x, la buena noticia es que los principios fundamentales permanecen más o menos iguales. Los modeladores que estén acostumbrados a utilizar versiones anteriores de UML deberían experimentar una transición fácil hacia UML 2. De hecho, los cambios más importantes incorporados en UML 2 se han realizado en el metamodelo de UML, y no los encontrarán directamente la mayoría de modeladores [Arlow+2006].

EL UP Y EL RUP

En la actualidad, deberíamos ver UP como el caso general abierto y RUP como una subclase comercial específica que extiende y anula las características de UP. Pero RUP y UP siguen siendo mucho más similares que diferentes. Los workflows básicos de análisis y diseño orientados a objetos son suficientemente similares que una descripción desde una perspectiva UP sería de igual utilidad para los usuarios RUP.

El UP y RUP modelan el quién, cuándo y qué del proceso de desarrollo de software, pero lo hacen de forma ligeramente diferente. La última versión de RUP tiene algunas diferencias terminológicas y sintácticas de UP, aunque la semántica de los elementos del proceso permanece igual.

Mientras que RUP es un producto de proceso Rational, UP es un proceso de ingeniería de software abierto de los autores de UML [Jacobson+1999].

DIAGRAMAS DE UML 2

Diagramas de estructura

Representan elementos componiendo un sistema o una función. Estos diagramas pueden reflejar las relaciones estáticas de una estructura, como lo hacen los diagramas de clases o de paquetes; o arquitecturas en tiempo de ejecución, tales como diagramas de objetos o de estructura compuesta [SparxSystems2007].

- Diagrama de Clases

  Muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Estos diagramas son los diagramas más comunes en el modelado de sistemas orientados a objetos. Abarcan la vida de diseño estático de un sistema.

- Diagrama de Objetos

  Modelan instancias de los elementos existentes en los diagramas de clases [Booch+2006]. Muestran un conjunto de objetos, y las relaciones entre ellos, en un momento durante la ejecución del sistema [Kendall2004]. Son útiles para entender los diagramas de clases [SparxSystems2007] pudiendo comprenderse con dos o más diagramas de objetos [Fowler2004].

- Diagrama de Estructura Compuesta

  Es un nuevo diagrama en UML 2. Muestra la estructura interna de un clasificador (incluyendo partes y conectores) o de una colaboración [VisualParadigm2007]. Permite mostrarlas como elementos compuestos, exponiendo interfaces y conteniendo puertos y partes [SparxSystems2007]. Como afirma [Fowler2004], gran parte de los conceptos aplicados son también aplicados en los diagramas de componentes, así como los de clases.

- Diagrama de Componentes

  El modelo de componentes ilustra los componentes de software que se usarán para construir el sistema.

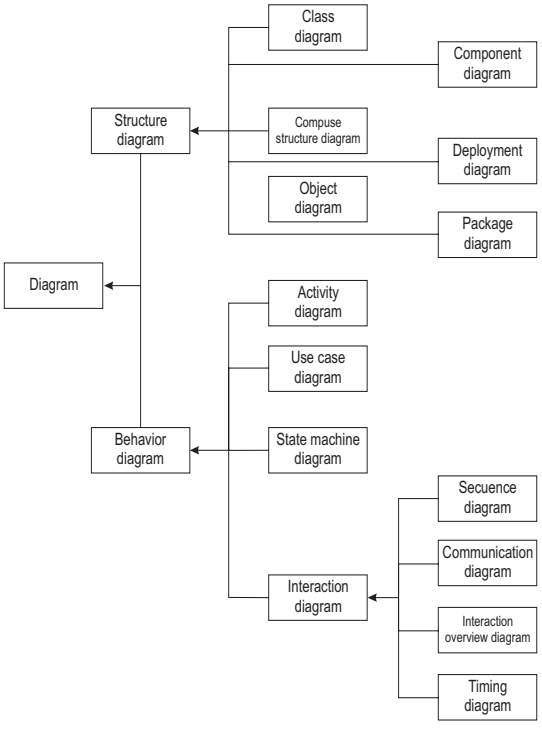

Figura 1. Estereotipos soportados por el modulo diseñado - [Arlow+2006].

  Se pueden construir a partir del modelo de clases y escribir desde cero para el nuevo sistema, o se pueden importar de otros proyectos y de productos de terceros.

  Un componente es una parte lógica y reemplazable de un sistema, que conforma y proporciona la realización de un conjunto de interfaces. Los buenos componentes definen abstracciones precisas, con interfaces bien definidas, y que facilitan la sustitución de los componentes viejos por otros más nuevos y compatibles [Booch+2006].

- Diagrama de Despliegue

  Muestran la configuración de nodos de procesamiento, en tiempo de ejecución, y los artefactos que residen en ellos. Abordan la vista de despliegue estática de una arquitectura.

- Diagrama de Paquetes

  Un paquete es un grupo de elementos de un modelo. Son muy útiles en la gestión de modelos. También muy útiles en temas relacionados con la agrupación, tales como casos de uso, con el fin de facilitar la ruptura entre el trabajo en sub-equipos.

  Un paquete puede contener uno o más tipos de elementos del modelo, cada uno de los cuales debe tener un nombre único dentro del paquete. Un paquete puede ser de clases, CU (…) La única regla es que cada uno de los elementos de un modelo solo puede pertenecer a un único paquete (en términos UML, un modelo es básicamente un paquete que contiene otros paquetes) [Kendall2004].

DIAGRAMAS DE COMPORTAMIENTO

Los diagramas de comportamiento capturan las variedades de interacción y el estado instantáneo dentro de un modelo mientras se “ejecuta” a través del tiempo [SparxSystems2007].

- Diagrama de Actividades

  Cubren la vista dinámica de un sistema. Son especialmente importantes al modelar el funcionamiento de un sistema y resaltan el flujo de control entre objetos.

- Diagramas de Interacción

  Muestran una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos (…) Los diagramas de interacción se utilizan para modelar

los aspectos dinámicos de un sistema. La mayoría de las veces, esto implica modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento [Booch+2006].

En la realización de CU, se utilizan para modelar interacciones entre objetos que realizan un CU o parte de un CU.

Existen cuatro tipos de diagramas de interacción, cada uno de los cuales enfatiza un aspecto diferente de la interacción.

- Diagrama de secuencia
- Diagrama de comunicación
- Diagrama de visión de interacción
- Diagrama de tiempo

• Diagrama de Secuencia

Muestra un diagrama bidimensional. La dimensión vertical es el eje del tiempo, el tiempo avanza hacia debajo de la página. La dimensión horizontal muestra los roles que representan cada uno de los objetos en la colaboración.

Estos diagramas han mejorado de manera notable en la nueva versión de UML, incluyendo fragmentos combinados y operadores. Un fragmento combinado encapsula porciones del diagrama de secuencia. Estos fragmentos se encuentran rodeados por un frame. El especificador en la esquina superior izquierda describe como el fragmento es manejado [Eriksson+2004]. Las condiciones de protección son expresiones booleanas, y el operando se ejecuta si y solo si la expresión se evalúa en verdadero. Una sola condición de protección se puede aplicar a todos los operandos, o cada operando puede tener su propia condición de protección única. Incluyen instrucciones Alt – Alternativas, break - Pausa, Opt – Opción, par – Paralelo; así como permiten expresar la recursividad. También permiten referenciar a otro diagrama de secuencia.

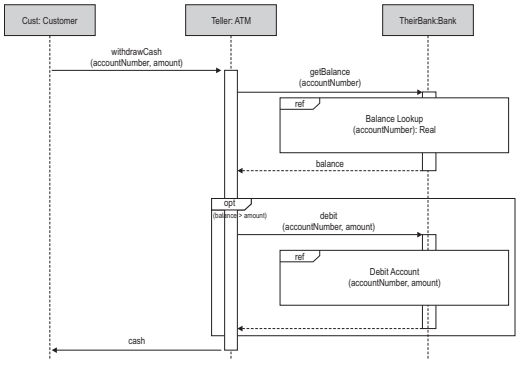

Fig. 2. Un diagrama de secuencia que hace referencia a otros 2 - [Bell2004]

• Diagrama de Comunicación

Inicialmente llamado un diagrama de colaboración, es un diagrama de interacción que muestra información similar a los diagramas de secuencia, pero su foco principal es en la relación de objetos. Los diagramas de secuencia y de comunicación son semánticamente equivalentes, ya que ambos derivan de la misma información del metamodelo de UML. Como consecuencia de esto, se puede partir de un diagrama y convertirlo a otro sin pérdida de información. Ambos comparten el mismo modelo subyacente, pero cada uno puede representar cosas que el otro no representa [Booch+2006].

• Diagrama de Visión Global de Interacción[1]

Son un nuevo diagrama en UML 2 y una variante de los diagramas de actividades [Eriksson+2004]. Fusionan los diagramas de actividades y secuencia para permitir que los fragmentos de interacción sean fácilmente combinados con los puntos y flujos de decisión. Los diagramas de interacción pueden incluir diagramas de secuencia, comunicación, de descripción de la interacción y de tiempos [SparxSystems2007], los nodos representan diagramas de interacción.

• Diagrama de Tiempos

Son un nuevo diagrama en UML 2. [VisualParadigm2007] los describe como: un tipo de diagrama que muestra el comportamiento del objeto(s) en un período determinado de tiempo.

[Hamilton+2006] y [Pilone+2005] los encuentran más asociados a sistemas de tiempo real o siste-

1 En inglés, el término es “interaction overview diagram”; sin embargo, en las traducciones de los autores al español, aparecen con diferentes nombres; así [SparxSystems2007] los denomina “diagrama de descripción de la interacción”; [Arlow+2006], “diagrama visión interacción”; y en el libro del francés [Debrauwer+2005], traducido al español, se les llama “diagrama de vista de conjunto de las interacciones”. Se utiliza el título de Visión Global de Interacción porque es el usado en el libro de los autores del UML [Booch2006] y al cual hacen referencia los otros autores.

mas embebidos, afirmando además que sin embargo no están solo limitados a estos dominios. A su vez [Rumbaugh+2005], afirman que no son necesarios en la mayoría de los diseños conceptuales.

- Diagrama de máquina de Estados[2]

  Muestran una máquina de estados, que consta de estados, transiciones, eventos y actividades. Un diagrama de estados muestra la vista dinámica de un objeto.

DOCUMENTACIÓN DE PROYECTOS CON UML 2

La documentación de los sistemas es un aspecto sumamente importante, tanto en el desarrollo de la solución como en el mantenimiento del mismo. Asimismo, es un elemento importante durante los procesos de evaluación o auditorías de sistemas, ya sea bajo COBIT, ISO o la norma peruana NTP-ISO/IEC 12207. Esta documentación de proyectos se realiza con el apoyo de alguna metodología y en base a modelos, los cuales son desarrollados casi exclusivamente con UML, que se encuentra actualmente en la versión 2.

En esta sección, se describen brevemente las fases y los artefactos producidos y, en particular, los diagramas generados e incluidos en el proyecto. Quizá se perciba poca relación secuencial entre los distintos artefactos descritos en cada fase. Ello, debido a que si bien los artefactos son producidos en dichas fases, lo son en distintas iteraciones y niveles de madurez del proyecto.

1. Inicio
   1.1. Modelado del negocio. Documento de glosario del negocio, documento reglas del negocio. Documento de visión del negocio. Modelo de casos de uso del negocio. Documento de realización de casos de uso del negocio. Documento de especificaciones suplementarias del negocio. Documento de objetivos del negocio. Documento de reglas del negocio. Diagrama de actividades (opcional), diagrama de secuencia (opcional) y colaboración[3] (opcional).
   1.2. Requisitos. Modelo de análisis del negocio. Requerimientos de los stakeholder, plan de gestión de requerimientos, especificaciones suplementarias, modelo de casos de uso (especificaciones de casos de uso), modelo de dominio (clases de dominio), glosario, especificación de requerimientos de software, storyboard, diagrama de paquetes, prototipos de interfaz de usuario (prototipos de bajo hechos a mano alzada y alto nivel hecho en alguna herramienta especifica o directamente en el IDE).
2. Elaboración
   2.1. Análisis y diseño. Diagrama de actividades, diagrama de visión global de la interacción, diagrama de comunicación, diagrama de secuencia, diagrama de objetos, diagrama de clases (clases de análisis y de diseño), diagrama de máquina de estados (para las clases de análisis y de diseño, en las clases más relevantes), documento de arquitectura de software, modelo de desplazamiento, modelo de análisis, modelo de diseño, realización de casos de uso, paquetes de diseño (de análisis y de diseño), modelo de datos, definición de datos.
   2.2. Implementación. Modelo de implementación, diagrama de componentes, diagrama de artefactos, diagrama de despliegue (primeras versiones).
3. Construcción

   Se actualizan los artefactos que fueron corregidos durante la fase de construcción.
   3.1. Implementación. Modelo de implementación, diagrama de artefactos, plan de integración de la construcción, diagrama de componentes.

3. ESTRUCTURA PROPUESTA

Tal como se describió en la sección anterior, se presenta la estructura propuesta para el Modelado del sistema, basado en el modelo arquitectural 4+1 de [Kruchten2000] y soportado por el UP y el RUP:

2 Los diagramas de máquina de estados reciben distintos nombres en combinaciones singular/plural, [Kendall2004], [Eriksson+2004] y [Arlow+2006]; por ejemplo, los denominan máquinas de estado; [Pilone+2005] y [Fowler2004], los denomina máquina de estado; todos haciendo referencia al mismo diagrama.

3 En el perfil BM, aún se los llama diagrama de colaboración a los ahora llamados, diagramas de secuencia.

4 La organización está basada en la propuesta por el Rational Rose, sustentándola con otras bibliografías e incluyéndola en la plantilla propuesta.

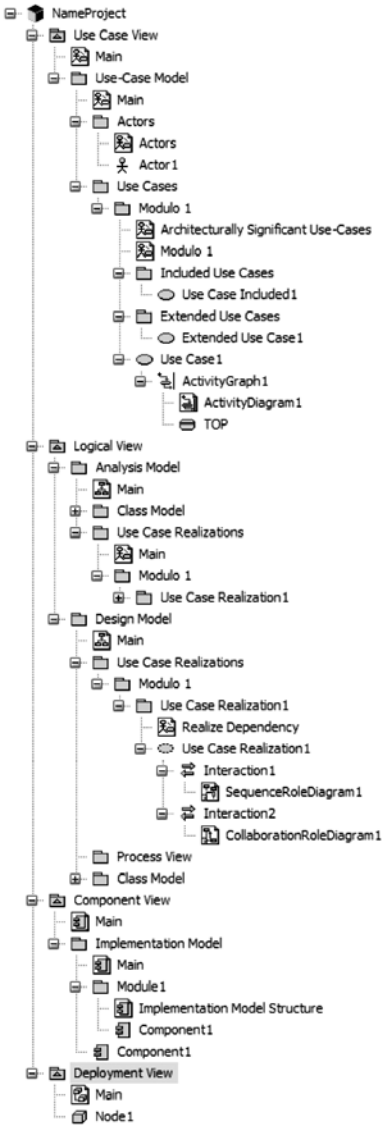

Figura 3: Vista Casos de Uso.

A. Use Case View (Vista de casos de uso)[4]. Se muestra en la Fig. 3:
   a. Como un primer punto, es la identificación de los requisitos, mediante el modelo de requisitos, no es un elemento visual, por tanto, no se incluye en la Fig. 3.

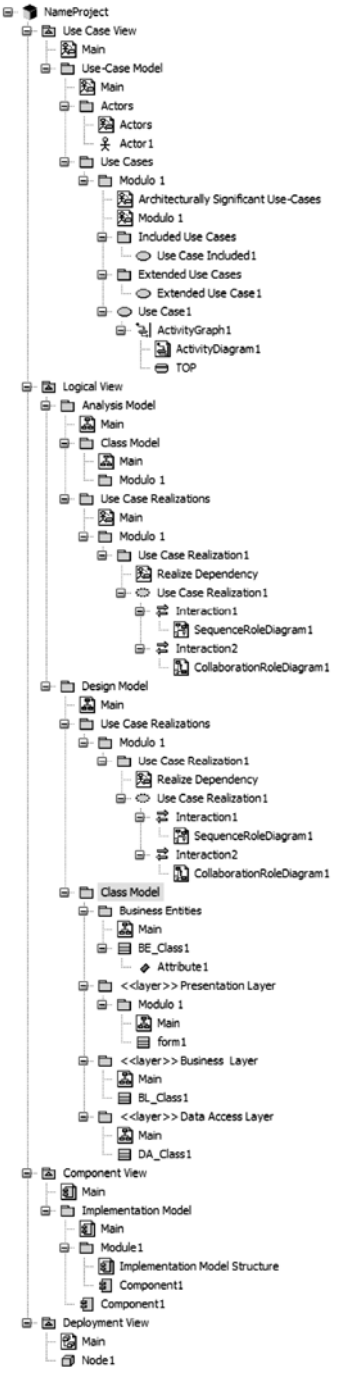

Figura 4: Vista Lógica.

5 [Arlow+2006] y otro autores identifican como segundo punto y primer punto a modelar en la vista de Casos de Uso la identificación de los paquetes.

6 Son un nuevo diagrama "oficial" en UML2, [Arlow+2006] y [Windle+2002] describen ubicarlos a este nivel del desarrollo del proyecto. Sin embargo [Boggs+2002], ya eran descritos en versiones de anteriores de UML, pero de manera informal.

7 La organización está basada en la propuesta por el Rational Rose.

b. Se agrupa a los actores en un paquete y un diagrama de CU con relación de los actores entre sí[5] .

c. En el paquete Casos de uso se agrupan los casos de uso en paquetes[6] o módulos, de acuerdo a la organización del proyecto.

d. Los CU include y extends se agrupan en paquetes[7].

e. Cada CU relevante debe tener su propio diagrama de actividades[8] (o diagrama de visión global de la interacción[9] , si es que corresponde).

f. Cada módulo tiene al menos dos diagramas de CU, el Architecturally Significant Use-Case, que incluye a los CU, actores y demás artefactos relevantes del módulo. Además, un diagrama que muestra todos los artefactos del mismo.

B. Logical View (Vista lógica)[10]. Se muestra en la Fig. 4. Agrupa al modelo de análisis y modelo de diseño. Los elementos del Modelado del Negocio se incluye en otro archivo de proyecto.

a. Analysis Model: Incluye a su vez a:

i. Class Model (Modelo de clases). De análisis[11], usando los estereotipos MVC en UML.

ii. Use Case Realization (Realización de Casos de Uso). De análisis. Puede incluirse, y es muy recomendable el diseño de diagrama de actividades, o una nueva iteración a los mismos desarrollados en la vista anterior. Puede incluir a los diagramas de secuencia, comunicación o ambos[12].

iii. Diagrama de paquetes (de análisis)[13].

b. Design Model[14]. Incluye a su vez a:

i. Use Case Realization (Realización de casos de uso)[15]. De Diseño. Puede incluir a los diagramas de secuencia, colaboración o ambos. La principal diferencia es el nivel de detalle, se tiene en consideración al patrón de diseño a utilizar, así como la arquitectura.

ii. Diagrama de paquetes (de diseño). Diseño de subsistemas (paquetes).

iii. Class Model (Modelo de clases). De diseño. Modela las clases, bajo las directivas del patrón de diseño elegido[16]. Si el sistema desarrollado es web, se puede usar la extensión WAE[17] para las clases de la capa o nivel presentación.

8 [Windle+2002] y [Boggs+2002] recomiendan desarrollar en este punto el desarrollo de estos diagramas, aunque no describen a qué nivel de detalle o cuan amplios se los debe desarrollar. Aquí usamos el criterio de "comunicación"; es decir, solo hasta el punto de comunicar el objetivo del CU.

9 Si bien en la bibliografía consultada no se identificó de manera explícita el uso de estos diagramas; por su semejanza en uso y funcionalidad con los diagramas de actividades, se los implementa a este nivel dentro del proyecto.

10 La organización de la vista lógica se encuentra viene documentada en [Jacobson+1999], dedicándole un capitulo tanto apara el análisis como el diseño. [Kruchten2000] describe la importancia del análisis y del diseño en el desarrollo de un sistema. El RUP los trata de forma integrada, como una disciplina, aunque son dos técnicas separadas [Shuja+2008].

11 A nivel de diagrama de clases se pueden usar los estereotipos frontera, controlador, entidad [Boggs+2002] y [Quatrani2002]. En este nivel no se describe la arquitectura [Shuja+2008]. Sin embargo, [Arlow+2006] no hace ninguna recomendación especial sobre el tipo de estereotipos, en todo caso queda a criterio del equipo de desarrollo su uso.

12 [Quatrani2002] ofrece una serie de argumentos acerca del uso de los diagramas de secuencia y comunicación (detallados), usados en la realización de los CU, recordemos que con StarUML se puede generar automáticamente uno a partir del otro.

13 Se identifican si hay anidamiento de paquetes, dependencias, relaciones <<import>>, <<trace>> y otras, dependiendo del nivel de detalle que requiere el proyecto.

14 El elemento más importante es el diagrama de clases. También incluye información de cómo las clases colaboran entre sí [IBM1998]. Asimismo, [Quatrani2002] describe un método para identificar las relaciones, atributos y métodos de las clases.

15 A este nivel, se trabajan con las clases que van a formar parte del proyecto, es decir, aquellas que se van a programar [Jacobson+1999].

16 En la actualidad los patrones de diseño en entorno web más usados son Layers y MVC. A nivel de web services es SOA. Como se describió anteriormente, el proyecto está implementado con el patrón de diseño Layers y siguiendo la organización propuesta por [Eeles2001]. En este nivel, se diseñan aquellas clases ques poseerán el atributo de persistencia, o dicho de otra manera, que formarán parte de la base de datos [Shuja+2008].

17 [Conallen2002] propone una extensión para modelar proyectos desarrollados en entorno web. Los detalles de esta extensión son descritos en el Anexo D.

18 Tal como [Arlow+2006] afirman, para las maquinas de estados el UP (y el RUP) no propone un lugar específico para modelar el ciclo de vida de las clases. El criterio utilizado es: si añade valor el diagrama, ayuda a entender el ciclo de vida del proyecto, entonces utilizarlo. Son más útiles al final de la fase de elaboración y al principio de la fase de construcción.

19 Puede ser realizado directamente con diagrama de clases, con el DSL E-R (entidad-relación), o con el perfil UML propuesto por el RUP denominado "UML Profile for Database Design" y descrito en [Naiburg+2001].

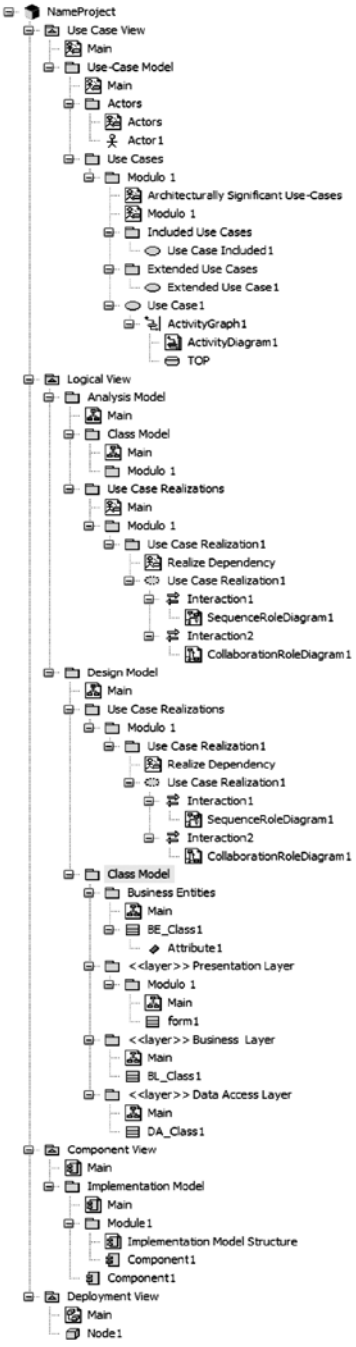

Figura 5. Vista de componentes.

iv. Diagrama de estados para las clases más relevantes[18].

v. Modelo de datos (modelo de base de datos)[19].

C. Component View (Vista de componentes)[20]. Se muestra en la Fig. 5.

Incluye a:

i. Implementation Model[21]. Describe la implementación de cada módulo o subsistema a desarrollar. Si se considera necesario, se puede incluir un diagrama de artefactos[22].

D. Deployment View[23] (Vista de desplazamiento). Se muestra en la Fig. 5. Muestra el desplazamiento físico del sistema, organizado por módulos o subsistemas (si es que los hubiere). Existe un único diagrama de desplazamiento por sistema.

4. CONCLUSIONES

La estructura estandarizada del UML permite un gradual proceso de adopción, los usuarios del mismo solo necesitan adoptar los elementos del lenguaje que necesitan, y de acuerdo a su experiencia y criterio, puede, ir adoptando el resto de características que les ofrece el lenguaje. Solo se ha añadido un pequeño número de características nuevas al lenguaje para hacerlo más claro, coherente y compatible con los nuevos avances del desarrollo de software.

Se deben identificar cuáles son los diagramas relevantes en un proyecto, de acuerdo a las características del mismo, el objetivo principal de los modelos es comunicar y ese debe ser el criterio para priorizar un diagrama sobre otro, sin hacer un uso excesivo e innecesario de los recursos del proyecto en la documentación.

Algunos diagramas existentes en versiones anteriores de UML, como los diagramas de secuencia, se han visto muy mejorados en sus funcionalidades y su capacidad de comunicar. Otros nuevos, como los diagramas de visión global de la interacción, permiten representar con mayor claridad escenarios en los cuales los diagramas de actividades se encontraban algo limitados. También,

20 En la vista de componentes se despliegan los "ensamblados" de que consta el sistema, organizados por módulos o sub-sistemas [Booch+2006] y [Shuja+2008].

21 El modelo de implementación está muy relacionada con la forma como se va a implementar el sistema. En entorno web se sugiere usar el perfil WAE, propuesto por [Conallen2002], que tiene estereotipos específicos para representar mejor los elementos web.

22 [Booch+2006] no define un punto específico para este tipo de diagramas, pero por afinidad con los diagramas de componentes se los incluye en la vista de componentes.

23 Incluye procesadores, dispositivos, conexiones y procesos [Boggs+2002].

los diagramas de estructura compuesta facilitan la tarea de modelar APIs o componentes, de forma clara y visual. Con ello, vemos que si bien se han ampliado los tipos de diagramas y la forma de representar modelos, ello ha permitido trabajar en escenarios que antes hubiesen sido muy dificultosos o poco claros trabajar con los diagramas existentes.

Si bien en esta investigación se desarrolló bajo las metodologías RUP y UP, se pueden desarrollar propuestas con otras metodologías, tales MSF, XP, Scrum u OpenUp también podrían ser temas como metodologías para el desarrollo de proyectos.

5. REFERENCIAS

[Arlow+2006] Arlow, Jim y Neustadt, Ila. 2006. *UML2.* [ed.] Victor Manuel Ruiz Calderón y Susana Krahe Pérez-Rubín. [trad.] Beatriz Parra Fernández. Madrid : ANAYA MULTIMEDIA, 2006. pág. 609. ISBN: 84-415-2033-X.

[Bell2004] Bell, Donald. 2004. UML's Sequence Diagram. [En línea] 16 de Febrero de 2004. [Citado el: 18 de Abril de 2009.] http://www.ibm.com/developerworks/rational/library/3101.html.

[Boggs+2002] Boggs, Wendy y Boggs, Michael. 2002. *Mastering UML with Rational Rose 2002.* [ed.] Donna Crossman. Alameda: SYBEX Inc., 2002. pág. 702. ISBN: 0−7821−4017−3.

[Booch+2006] Booch, Grady, Rumbaugh, James y Jacobson, Ivar. 2006. *El Lenguaje Unificado de Modelado: Guía de usuario.* [ed.] Martín-Romo Miguel. [trad.] Jesús J. Garcia Molina y José Sáez Martínez. Segunda edición. Rivera del Loira : PEARSON EDUCACIÓN S.A., 2006. pág. 527. ISBN 10: 84-7829-076-1.

[Conallen2002] Conallen, Jim. 2002. *Building Web Applications with UML Second Edition.* [ed.] Inc. Pearson Education. Segunda edición. Boston : Addison Wesley, 2002. pág. 496. ISBN: 0-201-73038-3.

[Debrauwer+2005] Debrauwer, Laurent y Van der Heyde, Fien. 2005. *UML 2, Iniciación, ejemplos y ejercicios corregidos.* s.l. : Ediciones ENI, 2005. ISBN: 978-2-7460-4741-9.

[Eeles2001] Eeles, Peter. 2001. Layering Strategies. [En línea] 15 de Octubre de 2001. [Citado el: 15 de Diciembre de 2009.] http://www.ibm.com/developerworks/rational/library/4699.html.

[Eriksson+2004] Eriksson, Hans-Erik, y otros. 2004. UML™ 2 Toolkit. [ed.] Kevin Kent. Indianapolis, Indiana, EE.UU. : Wiley Publishing, Inc., 2004. ISBN: 0-471-46361-2.

[Fowler2004] Fowler, Martin. 2004. *UML Distilled: A Brief Guide to the Standard Object Modeling Language.* Third Edition. Boston: Addison Wesley, 2004. p. 159. A Brief Guide to the Standard Object Modeling Language. ISBN 0-321-19368-7.

[Hamilton+2006] Hamilton, Kim y Miles, Russell. 2006. *Learning UML 2.0.* [ed.] Brett McLaughlin McLaughlin y Mary T. O'Brien. California : O'Reilly, 2006. pág. 286. ISBN: 0-596-00982-8.

[IBM1998] *Best Practices for Software Development Teams.* IBM. 1998. TP026B, Lexington: s.n., Noviembre 01, 1998.

[Jacobson+1999] Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 1999. *El Proceso Unificado de Desarrollo de Software.* [ed.] Andrés Otero. [trad.] Salvador Sánchez, y otros. s.l.: Pearson Educación S.A., 1999. pág. 464. ISBN: 84-7829-036-2.

[Kendall2004] Kendall, Scott. 2004. *Fast Track UML 2.0.* Tennessee: Apress TM, 2004. ISBN:1590593200.

[Kruchten2000] Kruchten, Philippe. 2000. The Rational Unified Process an Introduction. Massachusetts, EE.UU.: Addison Wesley Longman, Inc, 14 de Marzo de 2000. ISBN: 0-201-70710-1.

[Naiburg+2001] Naiburg, Eric J. and Maksimchuk, Robert A. 2001. *UML for Database Design.* Primera Edición. NJ: Addison Wesley, 2001. p. 320. ISBN: 0-201-72163-5.

[Pilone+2005] Pilone, Dan y Pitman, Neil. 2005. *UML 2.0 in a Nutshell.* [ed.] Gennick Jonathan. First Edition. s.l.: O'Reilly Media, Inc., 2005. pág. 234. ISBN: 0-596-00795-7.

[Quatrani2002] Quatrani, Terry. 2002. *Visual Modeling with Rational Rose 2002 and UML.* s.l.: Addison Wesley, 2002. p. 288. ISBN: 0-201-72932-6.

[Rumbaugh+2005] Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2005. *The Unified Modeling Language: Reference Manual.* Second Edition. Boston : Pearson Education, Inc., 2005. pág. 742. ISBN 0-321-24562-8.

[Shuja+2008] Shuja, Ahmad K. y Krebs, Jochen. 2008. *IBM Rational Unified Process Reference and Certification Guide—Solution Designer.* [ed.] Tara Woodman y Ellice Uffer. Boston: IBM Press, 2008. ISBN: 0-13-156292-4.

[SparxSystems2007] Sparx Systems Pty Ltd. 2007. UML Tutorial. [En línea] 19 de Enero de 2007. [Citado el: 26 de diciembre de 2008.] http://www.sparxsystems.com.au/uml-tutorial.html.

[VisualParadigm2007] Visual Paradigm. 2007?. UML 2 Diagrams. [En línea] 2007? [Citado el: 01 de Enero de 2009.] http://www.visual-paradigm.com/VPGallery/diagrams/index.html.

[Windle+2002] Windle, Daniel R. and Abreo, Rene L. 2002. *Software Requirements Using the Unified Process: A Practical Approach.* Primera edición. New Jersey : Pearson Education, Inc., 2002. p. 288. ISBN: 0-13-096972-9.