LECTURA
UNIDAD II

UML

Jaqueline Garcia Luevano ITIC4
Eduardo Flores Gallegos

HORA 1 INTRODUCCIÓN AL UML

El UML (Lenguaje unificado de modelado) ha tomado importancia a lo largo de los años en cuanto al desarrollo de software ya que anteriormente, cuando no se contaba con este, era común que los desarrolladores de sistemas no interpretaran de la manera correcta lo que quería el cliente y por consecuencia este no estaba satisfecho al momento de que se hacia la entrega, por esa razón el UML llegó a proporcionar la organización necesaria para que todas las partes involucradas pudieran entender que es lo que iba a hacer el sistema. Para que la organización antes mencionada se lleve a cabo con éxito existen diagramas, los cuales son: diagrama de clases: el cual nos ayuda a separar objetos por categorías, diagrama de objetos: el cual nos permite darles características específicas a estos, diagrama de casos de uso: el cual nos dice que acciones realizara cada usuario en el sistema, diagrama de estados: que muestra las transiciones de un objeto, de secuencias: en el cual se puede observar las distribuciones de las tareas, de actividades: son las actividades que ocurren dentro del comportamiento de un objeto, de colaboraciones: demuestra la conexión de los objetos, de componentes: representa la estructura del sistema y el diagrama de distribución: el cual muestra la estructura física.

El desarrollo de estos diagramas nos permite organizar la información de tal manera de que cada uno de los involucrados en la creación del sistema (sean expertos o no) puedan entender el funcionamiento de este. A su vez, la creación de diagramas mediante UML, nos facilita el trabajo al momento de desarrollar en equipo ya que un integrante se podría encargar de hacer los diagramas y dado a que estos detallan cada aspecto, y el otro integrante podrá realizar el sistema solo basándose en estos sin problema alguno.

HORA 2 ORIENTACIÓN A OBJETOS

En esta hora se enfoca a la orientación a objetos y lo que es un objeto en sí, principalmente como lo sabemos los objetos están presentes en un sistema como una representación de algo. Estos a su vez tienen atributos (los cuales son características de estos) y pertenecen a una clase (una agrupación de objetos que tiene ciertas características en común). Ahora, la orientación a objetos busca representar el mundo en un software ya que, como lo mencione anteriormente, cada objeto es la representación de algo y ese algo tiene una función dentro del sistema. La orientación a objetos tiene otras características como son la abstracción la cual consiste en dejar solo la información necesaria con respecto a un objeto, la herencia en la cual cada objeto se puede derivar de una clase o de una categoría que se divide por tener ciertas características, el polimorfismo el cual nos indica que varios objetos pueden responder a un mensaje y este mensaje puede significar lo mismo en otras clases, el encapsulamiento permite que se oculten ciertas operaciones que involucra el objeto para así solo mostrar las necesarias, el envío de mensajes permite que los objetos que trabajan en conjunto se comuniquen entre sí, las asociaciones las cuales permiten saber las relaciones que tienen los objetos entre sí y por ultimo está la agregación la cual indica que un objeto puede ser un objeto compuesto, es decir, tener más componentes que lo conforman. La orientación a objetos nos permite tener una cercanía con el mundo real al momento de programar, sin embargo, desde mi punto de vista, es necesario tener en cuenta lo que se quiere abarcar al momento de referirse a la orientación a objetos ya que esta al abarcar “el mundo real” puede llegar a ser muy grande y por consecuencia puede abarcar más de las posibilidades que se pueden manejar. Esta es una de las maneras más comunes de programar ya que nos permite resolver problemas, esto se nos facilita ya que cuenta con todas las características que anteriormente mencioné.

HORA 3 USO DE LA ORIENTACIÓN A OBJETOS

En esta hora se muestra más a detalle las estructuras de la orientación a objetos adaptadas al UML en primera estancia el libro nos explica sobre las clases, para que estas estén en UML se pone el nombre de la clase dentro de un rectángulo, a su vez hay paquetes estos se pueden colocar en una figura que a mi parecer simula la de un folder, un ejemplo de lo anterior descrito podría ser acerca de automóviles, aquí el paquete sería Automoviles y una clase que se puede derivar de este podría ser Deportivos, otra podría ser de Lujo, y así sucesivamente, como se puede observar en el anterior ejemplo tanto los nombres de los paquetes como el de las clases comienzan con mayúscula. Un punto para destacar es que en el UML no se usa la acentuación, ni el uso de ñ. Ya que están las clases, ahora se pasa a los atributos, estos no son más que características, mediante los atributos podemos identificar qué es lo que tiene de diferente una clase con otra, los nombres de los atributos se indican en un recuadro en forma en lista, a su vez de que cada atributo inicia con minúscula. Todo objeto que entra en la clase tiene un valor en los atributos, por ejemplo, anteriormente mencioné un paquete llamado Automoviles, la clase seria Deportivos y los atributos seguidos de sus valores serían: color= “Rojo”, marca= “Porsche”, modelo= “Porsche 718 Boxster”. Así mismo se pueden especificar los tipos de los atributos ya sean string, int, etc. Existe una característica como lo es la operación la cual describe lo que una clase puede realizar. Cuando son varias clases, los atributos, valores de los atributos, operaciones, no se dejan tan específicas para no hacer el diagrama tan extenso. Las clases también tienen responsabilidades (es decir lo que debe de hacer la clase) y restricciones (las reglas que debe de seguir la clase). El UML también tiene la posibilidad de agregar notas adjuntas lo cual a mi parecer es de gran ayuda.

HORA 4 USO DE RELACIONES

Ya que en un sistema existen clases, también deben de existir relaciones entre ellas, es por eso que en el UML existen conceptos como lo son las asociaciones, las cuales se usan cuando una clase es dependiente de otra, estas se representan con una línea en la cual se agregan frases como “participa en”, “emplea”, “atiende” y con un triángulo relleno dependiendo a la dirección que vaya. Para indicar la multiplicidad, que no es más que ver cuantos elementos de cada clase participan o se relacionan con la otra, se indica a un costado de la clase y por encima de la línea que los une. Existen diferentes tipos de multiplicidad los cuales son: uno a uno, uno a muchos, uno a uno o más (representado de esta manera 1..*) , uno a ninguno o uno, uno a un intervalo definido, entre otros que nos permiten poner cantidades definidas. Por ejemplo si existiera una clase maestro asociada a materias la relación que tendría seria múltiple ya que un maestro puede impartir una o varias materias, es decir, su relación seria de uno a uno o más (1 a 1..*) . Como se mencionó anteriormente tenemos relaciones las cuales pueden ser de muchos a muchos sin embargo esto puede resultar un problema al momento de buscar, es por eso que las asociaciones calificadas nos permiten cambiar las relaciones de muchos a muchos a una de uno a uno esto mediante la colocación de un calificador. Por otro lado, las asociaciones reflexivas permiten establecer la asociación de una clase consigo misma. La herencia y/o generalización permite, como su nombre lo dice, heredar sus atributos y métodos a otra clase, en UML se representa con una “flecha” con el triángulo vacío hacia la clase principal. La herencia se detecta al momento de que se están describiendo las clases y estas tienen atributos y operaciones en común. Existen otro tipo de relaciones llamadas dependencias, las cuales son cuando una clase utiliza a otra, esta se representa con una línea discontinua y con un triángulo vacío (simula una flecha).

HORA 5 AGREGACIÓN, COMPOSICIÓN, INTERFACES Y REALIZACIÓN

Se define como agregaciones cuando una clase está compuesta de otras, estas se representan con una línea y en el extremo que se une con el todo, con un rombo vacío. La composición es un tipo de relación más fuerte ya que depende totalmente de la clase con la que esté ligada y esta se representa como la agregación, pero con el rombo relleno. Los contextos agrupan clases que están relacionadas entre sí, es decir que pertenecen a algo en específico, por lo regular este caso se da más en las composiciones, la manera de agruparla es mediante un rectángulo, este nos permite agrupar las clases que están dentro de una clase en específico. La interfaz es un conjunto de operaciones que hace una clase, esta nos sirve para reutilizar las operaciones que nos hagan falta, en UML se representa con un rectángulo, como si fuera una clase, con la diferencia de que esta no cuenta con atributos además de que se puede especificar con una I, otra manera de representarla es con un circulo. La visibilidad establece en qué nivel se podrán utilizar atributos y operaciones de cierta clase. Es por ello que existen tres niveles de visibilidad, el público permite que las funciones se utilicen por otras clases y para indicar esto se coloca el signo + antes del atributo u operación, en el protegido solo se pueden utilizar las funciones si las clases se heredan de la clase original y se coloca un signo de # antes de los atributos u operaciones, por último, en el privado solo la clase original puede usar los atributos y operaciones y se coloca un – delante del atributo u operación. El ámbito es otra forma de relación de los atributos y operaciones, existen dos tipos de este, el de instancia en el cual cada objeto cuenta con su debido valor, en el ámbito de archivado solo hay un valor para un atributo a través de un conjunto de objetos. Todo esto sirve para complementar las relaciones en nuestras clases, ya que como sabemos hay relaciones de todo tipo y el hecho de que existan más maneras para hacerlo nos facilita un poco el camino.

HORA 6 INTRODUCCIÓN A LOS CASOS DE USO

Para el desarrollo de un sistema el analista debe de prestar suma atención a las necesidades del cliente ya que de esta manera podrá hacer una interpretación de lo que se solicita, para que esto se lleve a cabo con éxito es recomendable tener una entrevista con el cliente para que se aclaren esos puntos. En esta ocasión se profundiza más acerca de los casos de uso, los casos de uso son la descripción de lo que va a hacer cada perfil de usuario e incluso el sistema, es decir describen las funcionalidades. Esto lo hace de una forma gráfica que sea entendible tanto para el analista como el usuario ya que como lo sabemos no todas las personas cuentan con un amplio conocimiento acerca de la realización de sistemas. En este diagrama a las entidades se les conocen como actores. Como anteriormente mencioné, los casos de uso son los que pueden ser aplicados a los usuarios, pero también existen casos adicionales debido a que no solo el cliente es un actor, el sistema también figura como uno y a su vez también tiene procedimientos por realizar, daré por ejemplo mi proyecto, al hacer una entrevista con mi cliente me dio la información necesaria para dar por sentado que necesito tres actores en mi diagrama de casos de uso los cuales son el administrador, el trabajador y el sistema, como se puede observar incluí al sistema como actor, aun sin ser una persona, ya que al momento de que el administrador o trabajador hace el Login este entra en acción realizando el proceso de verificación de usuario. La inclusión en los casos de uso es cuando se incluyen más casos de uso con el fin de complementar el proceso mientras que en la extensión se crean más agregándolos a un existente. El hecho de que se represente de manera simple y detallado hace, a mi parecer, más entendible lo que hará el sistema y no solo para que el cliente lo entienda también el diseñador del sistema tendrá a la mano esta herramienta y por ejemplo si se está trabajando el equipo, cuando un integrante lo diseñe, fácilmente el otro lo puede aplicar. Otro punto a destacar es que este diagrama me parece uno de los más importantes en cuanto a explicación al cliente se refiere.

HORA 7 DIAGRAMAS DE CASOS DE USO

Como se vio en la hora 6, los casos de uso nos dan una idea de lo que cada actor va a hacer en un sistema, para hacer la representación de este diagrama debe de existir un actor el cual es representado por muñecos de “palito” y el nombre que este reciba va debajo, una elipse representa el caso de uso y el nombre de este puede ir en el centro o debajo de la elipse y para conectar un actor con un caso de uso se utiliza una línea sólida. Para que los casos de usos sean tomados en un sistema estos se encierran en un rectángulo, dejando a los actores fuera para conectarlos con una línea, como anteriormente mencioné. Para la concepción de relaciones entre casos de uso existe la inclusión y la extensión, la inclusión complementa los pasos de un caso de uso permitiendo que sea el proceso un poco más detallado, este se representa con una línea discontinua que une un caso de uso con otro y en medio de esta la palabra “incluir”. La extensión añade algo extra al caso de uso y este se representa de la misma manera que la inclusión solo que con la palabra “extender”. Los casos de uso también se pueden heredar entre sí, de esto surge la generalización, ya que los casos secundarios heredan funciones de los principales. El agrupamiento no es más que agrupar los casos de uso que tengan relaciones entre sí. Al momento de aplicar los casos de uso primero se debe de tener la comprensión del dominio, es decir, hacer un diagrama de clases donde se vea reflejado como es que se trabaja, para así definir los posibles usuarios y cuáles son las posibles acciones que harán estos, después se profundiza para aterrizar la manera en que todo se desea manejar, además de tener la información ordenada y concisa. Dado a lo aprendido en las horas anteriores debemos tener un panorama de que es lo que se va a realizar con respecto a este para posteriormente aplicarlo a nuestros proyectos.

HORA 8 DIAGRAMAS DE ESTADOS

Los diagramas de estados son importantes ya que estos nos permiten conocer y comprender qué es lo que hace un objeto dentro del sistema y no representando sus características físicas, sino que representa su comportamiento. De acuerdo a la lectura, un diagrama de estados en UML captura los cambios que tiene un objeto dentro del sistema, es decir, cada vez que se accione una parte del objeto este cambiara su estado, este tiene la característica que solo muestra las condiciones de un objeto y no de todo el sistema. La simbología de este consiste en un círculo relleno (el cual representa el punto inicial) apuntando con una fecha hacia un rectángulo con esquinas redondeadas para finalizar con una flecha que apunta hacia un círculo con solo una parte rellena (el cual representa el punto final). A este diagrama también se le pueden agregar detalles, en este es posible dividir el símbolo (el rectángulo con esquinas redondeadas), en tres partes, en la superior contendría el nombre del estado, después las variables las cuales son como contadores y por último las acciones las cuales pueden ser entrada, salida y hacer. Estos diagramas también incluyen sucesos y acciones los cuales simplemente nos dicen cuándo se va a realizar un proceso y que acción lleva consigo. Las condiciones de seguridad dentro de estos diagramas nos permiten establecer que es lo que se pude hacer mientras un estado no se esté ejecutando. Los sub estados no son más que estados dentro de otros estados y de estos hay dos tipos: los secuenciales los cuales ocurren uno detrás de otro y los concurrentes que es cuando dos secuencias de sub estados se están ejecutando al mismo tiempo. El estado histórico nos permite que nuestros sucesos tengan “memoria” es decir que vuelvan al lugar en donde se quedó después de una pausa y su símbolo es un “H” encerrada en un círculo. Un mensaje es una señal que se manda al momento de que se hace la transición de un objeto a otro. A mi parecer este es un diagrama complejo ya que, como anteriormente mencione, representa el comportamiento de un objeto y para que no haya confusión es necesario seguir estrictamente la simbología y así lograr con éxito la descripción de tal comportamiento.

HORA 9 DIAGRAMAS DE SECUENCIAS

Los diagramas de secuencias nos muestran cómo es que un objeto se comunica con otro, todo esto en una secuencia y en cierto periodo de tiempo. En este los objetos se representan con un rectángulo, el nombre de este se coloca en el centro y subrayado. Así mismo este tiene una línea discontinua en vertical que representa la vida de un objeto y a su vez entre esta se encuentra un pequeño rectángulo que representa la operación de un objeto. En este diagrama también se incluyen elementos como lo son los mensajes, los cuales se pasan a través de las líneas de vida y el tiempo (el cual es claro lo que indica) y comienza en la parte superior y termina en la parte inferior del diagrama. En un diagrama de secuencias también se puede mostrar la forma en que la interfaz gráfica se relaciona con otros objetos como el sistema operativo o la CPU. En un diagrama de secuencias de instancias se enfoca solo en un proceso en específico, por ejemplo, en un caso de uso. Un ejemplo aplicado a mi proyecto podría ser cuando el trabajador consulta los apartados (siendo este un caso de uso), se desarrolla el diagrama de secuencias de instancias de tal manera que se siga el flujo de este proceso. Por otro lado, el diagrama de secuencias genérico muestra las otras posibles opciones a un caso de uso, lo cual me atrevería a comprar con un "if" ya que si un escenario no ocurre puede ocurrir el otro. En ocasiones en estos diagramas puede aparecer un objeto y este se pude añadir de la misma manera, con un rectángulo horizontal y con la palabra crear(). En estos diagramas también se tiene la recursividad que no es más que cuando una operación se llama a sí misma.

HORA 10 DIAGRAMAS DE COLABORACIONES

En este diagrama se muestran las relaciones de los objetos entre sí, es similar al diagrama de secuencias con la diferencia de que en el diagrama de colaboraciones se enfoca en el contexto y en la organización. Este también permite el envío de mensajes entre objetos y evita la multiplicidad. Este se representa con los objetos representados en un rectángulo y sus relaciones mediante líneas que unen a cada uno de ellos además de que con una flecha se indica el receptor del mensaje, en adición a esto el mensaje se muestra mediante una etiqueta a un costado de la flecha. Como ya lo sabemos este sistema se puede aplicar al ejemplo de la interfaz de usuario, ya que realiza una serie de pasos para poder mostrarle al usuario lo que este necesita ver y no todo el proceso que conlleva eso. En este diagrama también es posible agregar los cambios de estado de un objeto esto mediante la creación de otro rectángulo con el estado y a su vez ligándolo al siguiente mediante una línea discontinua. Por otro lado, se tiene que en este diagrama también es posible que un objeto envíe un mensaje a varios objetos de una clase y este se representa con una pila de rectángulos y cuando un mensaje pide a un objeto que haga algo se representa como una expresión a lo que es igual. Cuando se dice que un objeto es activo, esto no quiere decir más que ese objeto es el que controla el flujo del diagrama y este se representa de la misma manera que un objeto común, pero con el borde más grueso. La sincronización entre objetos se da cuando un mensaje se debe de ejecutar siempre y cuando uno antes lo haya mandado, es decir, que los objetos se organicen en un determinado orden. A pesar de que el diagrama de secuencias y el de colaboraciones se parezcan es de utilidad incluir ambos ya que al tener ambos nos permite tener una visión más amplia de nuestro sistema y a la vez que sea más entendible. Este diagrama nos permite tener una visión clara de las relaciones entre objetos, como mencione al principio además de que este se organiza de acuerdo al espacio.

HORA 11 DIAGRAMAS DE ACTIVIDADES

Un diagrama de actividades muestra, como su nombre lo dice, el flujo de actividades que están dentro del diagrama de estados. La representación de este en UML es: para comenzar, se coloca un circulo relleno, posteriormente cada actividad va encerrada en un rectángulo con las esquinas redondeadas y cambia de una actividad a otra con una flecha que indica el flujo de estas mismas, para terminar con este se coloca un circulo con solo el centro relleno. En estos diagramas también es posible representar decisiones por si en algún momento se requiere que tome cierto camino en específico, esto se puede representar de dos formas, la primera consiste en seguir el flujo de las actividades mostrando las opciones y al final uniéndose, es decir, que independientemente del camino que se haya tomado ambas llegan al final, la segunda se puede representar con un rombo del cual se desprenden las opciones y al igual que el anterior se unen al final. Este diagrama también cuenta con rutas concurrentes, las cuales no son más que cuando dos actividades se ejecutan al mismo tiempo y se representa separando estas con líneas gruesas; las indicaciones también se aplican y estas se representan con un pentágono convexo para enviar y con un pentágono cóncavo para recibir. A su vez, dentro de este diagrama existen los marcos de responsabilidad, los cuales nos indican quienes son los responsables de cada actividad que se va a realizar, para representar esto se divide el número de marcos que sea necesario por cada persona que va a realizar las actividades y por consiguiente se acomoda el diagrama de tal manera que aun siga siendo visible su flujo. Para crear diagramas híbridos dentro de este basta con agregar símbolos de otros diagramas (siempre y cuando se relacionen a lo que se está elaborando). Como observación, podría agregar que este diagrama es un tanto parecido al diagrama de flujo (de hecho, en la lectura lo menciona) y esto permitió que fuera un poco más entendible para mí.

HORA 12 DIAGRAMAS DE COMPONENTES

En los diagramas de componentes representa, como su nombre lo dice los componentes (los cuales son piezas ejecutables de software) de sistema además de las interfaces. Estos son importantes ya que nos permite saber con qué estamos trabajando, muestra una vista física del sistema ya que muestra los elementos que componen al sistema, es decir, los componentes. Otro punto importante a considerar es la interfaz ya que ese es el medio por el cual el sistema y el usuario se comunican. Las interfaces tienen la característica de que se pude reutilizar o sustituir un componente siempre y cuando tengan la misma y esto nos facilita el trabajo a la hora del modelado. Existen tres tipos de componentes: los de distribución, los que son para trabajar en el producto y los de ejecución. Para representar un componente en este diagrama se usa un rectángulo con el nombre de este al centro además de que se colocan dos rectángulos pequeños sobrepuestos al lado izquierdo, también se pude mostrar cuando este es parte de un paquete o tiene más información. Se pueden mostrar relaciones y estas solo con una línea uniendo los componentes. Para representar a un componente y sus interfaces existen dos maneras, la primera consiste en el componente y unido a este con una flecha discontinua un rectángulo con la información de la interfaz la segunda manera, y a mi parecer más sencilla, es en la que está el componente y unido a un lado está un pequeño círculo, esto representa la realización y para representar la dependencia solo se une la interfaz al componente con una flecha discontinua. Estos diagramas se pueden utilizar en una página web con una applet en Java.

HORA 13 DIAGRAMAS DE DISTRIBUCIÓN

En este diagrama se muestra la estructura física, es decir los recursos de cómputo y a estos recursos se les llaman nodos; su representación es un tipo caja con el nombre del nodo. Para agregar información extra basta con dividir la caja y en la parte de abajo colocar esta, en la parte de arriba seguiría el nombre del nodo; otra manera más detallada puede ser colocando dentro del nodo las relaciones de dependencia. Los nodos también se pueden relacionar entre sí, y para esto se coloca una línea entre estos. Un punto importante es que se pueden conectar tanto nodos alámbricos como inalámbricos. Estos diagramas son, a mi parecer, fáciles de deducir en qué lugares se pueden utilizar, ya que como anteriormente mencioné, representan los recursos de computo esto quiere decir que en cualquier lugar donde se use uno será posible realizar este diagrama. Unos ejemplos son: en una casa que tenga un equipo de cómputo, en el libro se nos muestra el ejemplo el cual tiene 5 nodos: el procesador ISP, modem, procesador PC, impresora y monitor, también cada uno con su respectiva información adicional o con sus relaciones de dependencia. Otro ejemplo un poco más complejo son las redes token-ring (una red LAN) ya que en esta se conectan varios nodos, porque al ser una red y en forma de anillo, como su nombre lo dice, es necesario, la manera en que se conectan y gracias a los token permite saber a cada equipo cuándo puede transmitir información. Así como hay un diagrama para los dos ejemplos anteriores también existen otros como ARCnet, thin ethernet y Ricochet, los cuales varían en la forma en que funcionan las redes.

BIBLIOGRAFÍA

Schmuller Joseph. (SA). Recuperado de: https://trello-attachments.s3.amazonaws.com/5e1647e1e290f274a873acd2/5e58644b2de3593d7b9117a1/ba5b1a9962d815a291f51e93140cf4ea/Prentice_Hall_Aprendiendo_UML_en_24_horas.pdf