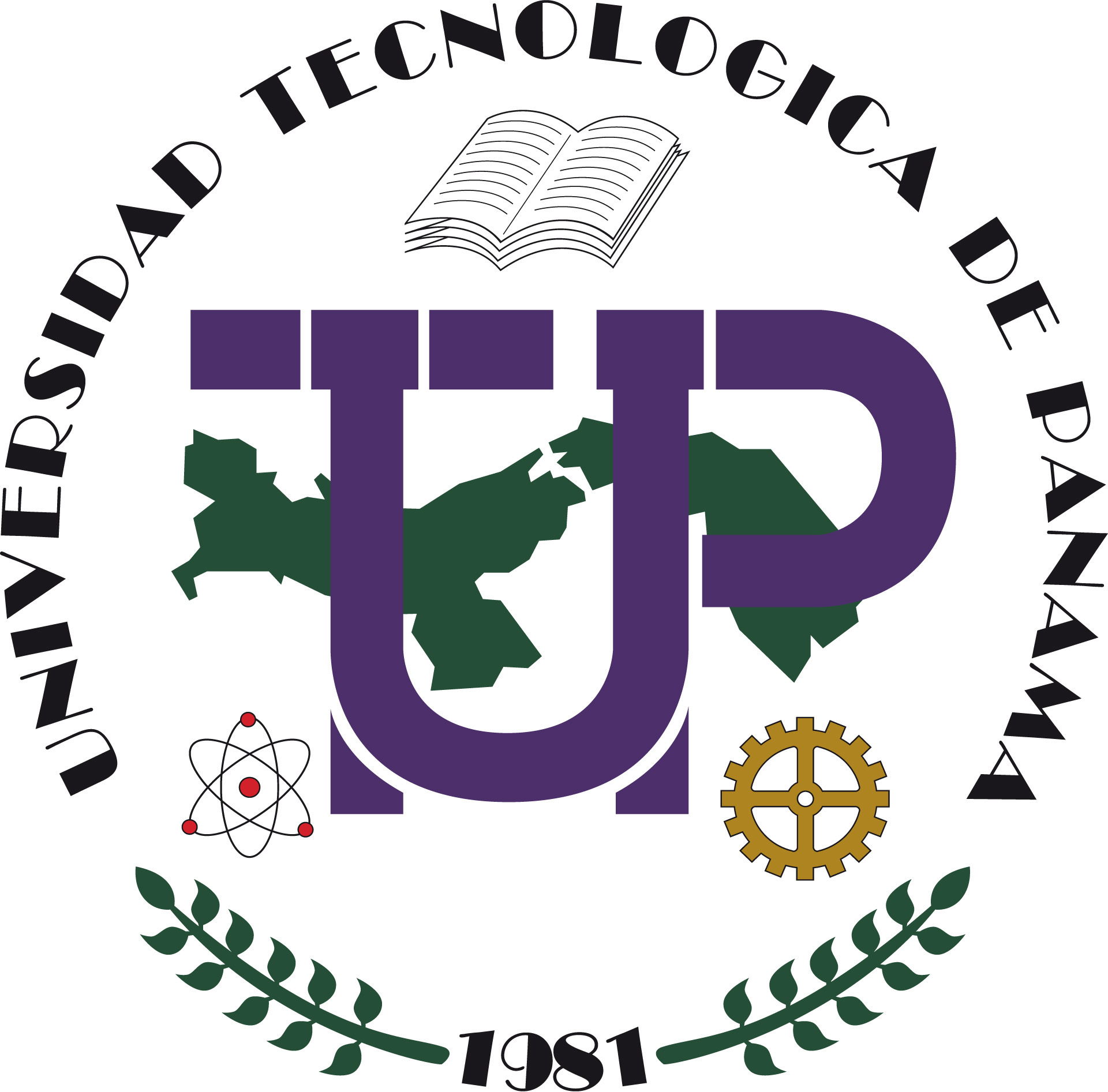
**Universidad Tecnológica de Panamá**

**Facultad de Ingeniería de Sistemas Computacionales**

**Licenciatura en Ingeniería de Sistemas de Información**

**Investigación #1**

**Lenguaje de Modelado UML**

**Curso:**

**Fundamentos de Programación (Java)**

**Facilitadora del curso:**

**Jaramillo, Lourdes**

**Nombres de los Estudiantes:**

**Batista, Johel [8-914-587]**

**Soto, Andrés [20-70-3499]**

**Grupo: 1IF-112**

**Segundo Semestre, Año Académico 2019**

**Comentarios Iniciales sobre UML**

Tal vez uno de los problemas más grandes que ha sufrido históricamente la industria del desarrollo de software es la falta de estandarización y documentación en los sistemas, rutinas y programas que se han ido desarrollando, ya que solamente quienes participaron en el desarrollo inicial, son los que conocen la lógica con la que fueron tomadas ciertas decisiones de diseño arquitectónico informático y en caso de que entre un nuevo miembro al equipo o el sistema pase a ser administrado por un tercero y presente fallos o incluso, se le tenga que dar mantenimiento básico para garantizar su funcionamiento, se encuentran con software altamente incomprensibles y que termina muchas veces siendo reescrito de cero en el peor de los casos.

Es por ello, que a partir de los primeros desarrollos industriales de software vitales, como por ejemplo el que manejaba la computadora del Saturn V (El Cohete que llevó el hombre a la Luna), y por subsecuente el Módulo Lunar Águila, se comenzó a documentar a través de diagramas, una serie de procesos y la lógica que existía detrás de cada una de esas millones de líneas de código que nos dieron la oportunidad de pensar en el espacio como nuestra última frontera.

He aquí, donde años después, especialmente en el caso de la programación orientada a objetos con lenguajes Legacy que han ido evolucionando con el pasar de los años, como lo es el caso de Java y hasta más recientes como PHP y Python, se desarrolló un sistema para “diseñar planos de software” que pudiesen ser fácilmente leídos y comprendidos por personas con los conocimientos básicos en la materia, siendo esta en la que nace el Lenguaje UML.

**Una breve Historia sobre UML**

Como un lenguaje formal, UML comenzó a gestarse en octubre de 1994, cuando Rumbaugh se unió a la compañía Rational fundada por Booch (dos reputados investigadores en el área de metodología del software). El objetivo de ambos era unificar dos métodos que habían desarrollado: el método Booch y el OMT (Object Modelling Tool).

Su primer borrador fue encontrado en octubre de 1995. En esa misma época otro reputado investigador, Jacobson, se unió a Rational y se incluyeron ideas suyas. Estas tres personas son conocidas como los “tres amigos creadores del UML”.

Todas estas colaboraciones condujeron a la definición formal de la primera versión de UML, con lo que se estaba necesitando un lenguaje no sólo para comunicar las ideas a otros desarrolladores sino también para servir de apoyo en los procesos de análisis de un problema, especialmente a nivel de metodologías de desarrollo de software.

Con este objetivo se creo el Lenguaje Unificado de Modelado (UML: Unified Modeling Language).

UML se ha convertido en ese estándar tan ansiado para representar y modelar la información con la que se trabaja en las fases de análisis y, especialmente, de diseño.

**Evolución de UML**

El OMG (Object Management Group) gestiona estándares relacionados con la tecnología orientada a objetos (metodologías, bases de datos objetuales, CORBA, etc.), propuso una serie de modificaciones y una nueva versión de UML (la 1.1), que fue adoptada por el OMG como estándar en noviembre de 1997.

Un modelo es una simplificación de la realidad. El objetivo del modelado de un sistema es capturar las partes esenciales del sistema. Para facilitar este modelado, se realiza una abstracción y se plasma en una notación gráfica. Esto se conoce como modelado visual. El modelado visual permite manejar la complejidad de los sistemas a analizar o diseñar.

UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten. Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se pueda implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos, como por ejemplo el caso más específico podría ser Java y todos sus derivados).

Adicional a todo esto, UML es un método formal de modelado utilizado en los proyectos informáticos más importantes a nivel mundial y siendo enseñado en las principales universidades.

Aportando entonces, las siguientes ventajas:

1. Permite un mayor rigor en la especificación de conceptos e ideas a expresar en el desarrollo lógico de una aplicación informática
2. Ayudar a realizar la verificación y validación del modelo lógico que ha sido desarrollado.
3. Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos).

**Tipos de Diagramas y su aplicación**

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. Lo que se busca con los diagramas es tener la oportunidad de presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Recordemos que un modelo es una representación simplificada de la realidad; el modelo UML describe lo que supuestamente hará un sistema, sin embargo no dice cómo se va a implementar en producción dicho sistema.

Estos son los diagramas más comunes que podemos encontrar dentro del lenguaje UML:

1. **Diagrama de Clases:** Los diagramas de clases describen la estructura estática (Rígida, sin ningún tipo de movimiento) que se puede encontrar dentro de un sistema. Las cosas que existen y que nos rodean se agrupan naturalmente en categorías. Una clase es una categoría o grupo de cosas que tienen atributos (propiedades) y acciones similares. Un ejemplo puede ser la clase “Aviones” que tiene atributos como el “modelo de avión”, “la cantidad de motores”, “la velocidad de crucero” y “la capacidad de carga útil”.
   1. **Diagrama de Objetos**: Los Diagramas de Objetos se encuentran fuertemente vinculados con los Diagramas de Clases, de hecho algunos autores tienden a considerarlos como un derivado de los mismos. Un objeto es una instancia de una clase, por lo que un diagrama de objetos puede ser visto como una instancia de un diagrama de clases. Los diagramas de objetos describen la estructura estática de un sistema en un momento particular y son usados para probar la precisión de los diagramas de clases.
2. **Diagrama de Casos de Uso:** Son descripciones de las acciones de un sistema desde el punto de vista del usuario final del mismo. Se consideran como una muy herramienta valiosa dado que es una técnica de aciertos y errores para obtener los requerimientos del sistema, justamente desde el punto de vista del usuario. Los diagramas de caso de uso modelan la funcionalidad del sistema usando actores y casos de uso. Los casos de uso son servicios o funciones provistas por el sistema para sus usuarios.
3. **Diagrama de Estados:** En cualquier momento, un objeto se encuentra en un estado particular, ya sea por ejemplo el caso de si una lámpara se encuentra encendida o apagada, el auto en movimiento o detenido, la persona leyendo o cantando, etc.. El diagrama de estados UML captura esa pequeña realidad, desde el punto de vista de soluciones binarias 0 (apagado o negativo) y 1 (encendido o positivo).
4. **Diagrama de Secuencias:** Los diagramas de clases y los de objetos representan información estática. No obstante, en un sistema funcional, ante la constante interacción de los objetos que forman parte del mismo entre sí, resulta que tales interacciones suceden con el tiempo. El diagrama de secuencias UML muestra la mecánica de la interacción con base en tiempos.
5. **Diagrama de Actividades:** Ilustra la naturaleza dinámica de un sistema mediante el modelado del flujo ocurrente de actividad en actividad. Una actividad representa una operación en alguna clase del sistema y que resulta en un cambio en el estado del sistema. Típicamente, los diagramas de actividad son utilizados para modelar el flujo de trabajo interno de una operación.
6. **Diagrama de Colaboraciones:** Describe las interacciones entre los objetos en términos de mensajes secuenciados. Los diagramas de colaboración representan una combinación de información tomada de los diagramas de clases, de secuencias y de casos de uso, describiendo el comportamiento, tanto de la estructura estática, como de la estructura dinámica de un sistema.
7. **Diagrama de Componentes:** Describe la organización de los componentes físicos de un sistema.
8. **Diagrama de Distribución:** Nos muestra la arquitectura física de un sistema informático. Puede representar a los equipos y a los dispositivos, y también mostrar sus interconexiones y el software que se encontrará en cada máquina.

**Herramientas CASE que ayudan al Modelado**

### **ERwin**

Es una herramienta de diseño de bases de datos. Proporciona productividad en el diseño, generación y mantenimiento de aplicaciones. Desde un modelo lógico de los requisitos de información, hasta el modelo físico perfeccionado para las características específicas de la base de datos diseñada, ERwin permite visualizar la estructura, los elementos importantes y optimizar el diseño de la base de datos. Genera automáticamente las tablas y miles de líneas de procedimientos almacenados y disparadores para los principales tipos de bases de datos.

**EasyCASE**

El centro de productos para procesos, modelado de datos y eventos, e Ingeniería de bases de datos, es un producto para la generación de esquemas de bases de datos e ingeniería inversa, trabaja para proporcionar una solución comprensible para el diseño, la consistencia y la documentación del sistema como un todo.

**Oracle Designer**

Es un juego de herramientas para guardar las variables que el usuario necesita y automatizar la construcción rápida de aplicaciones cliente / servidor flexibles y gráficas. Integrado con Oracle Developer, Oracle Designer ofrece una solución para desarrollar sistemas empresariales cliente / servidor de segunda generación.

**PowerDesigner**

Es un conjunto de aplicaciones Powersoft para la construcción, diseño y modelado de datos en varias aplicaciones. Es la herramienta para el análisis, el diseño inteligente y la construcción sólida de una base de datos y un desarrollo orientado a modelos de datos a nivel físico y conceptual, que proporcionan a los desarrolladores de aplicaciones Cliente / Servidor la base más sólida para aplicaciones de alto rendimiento.

**System Architect**

Tiene un repositorio único que integra todas las herramientas y metodologías utilizadas. En la elaboración de los diagramas, el Arquitecto de Sistemas se conecta directamente con el diccionario de datos, los elementos asociados, los comentarios, las reglas de validación, la normalización, etc. Tiene control automático de diagramas y datos, estandarizaciones y equilibrio entre el "Padre y Hijo "diagramas. Además del equilibrio horizontal, que funciona integrado con el diccionario de datos, asegurando la compatibilidad entre el modelo de datos y el modelo funcional.

**Snap**

Es un CASE para el desarrollo de aplicaciones en sistemas IBM AS / 400. Proporciona el entorno de trabajo integral, ofreciendo la posibilidad de construir sistemas de calidad inmejorable, adheridos a los protocolos de IBM SAA., Completamente documentados y ajustados a los requisitos específicos de la organización, en una fracción del tiempo y costo en que ser invertido , si se usaran herramientas tradicionales.

**Ventajas y Desventajas**

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| UML se puede usar para diferentes tipos de sistemas | UML no es un método de desarrollo |
| UML consolida muchas de las notaciones y conceptos más usadas orientados objetos | UML al no ser un método de desarrollo es independiente del ciclo de desarrollo |
| UML es fácilmente entendible | UML no se presta con facilidad al diseño de sistemas distribuidos |

**Comentarios Finales**

Una de las principales ventajas del uso de las ya mencionadas herramientas de tipo UML, es la estandarización de los procesos y diagramación de los mismos, de forma que estos puedan ser entendidos por cualquier desarrollador se encuentre o no relacionado con el trabajo que se está llevando a cabo, es decir que son como los planes que fundamentan los cimientos lógicos de cualquier desarrollo de software, los cuales podrán ser interpretados por personal idóneo en la materia.

Esta analogía proviene principalmente de hace muchísimos años, donde la estandarización dentro del diseño de sistemas (Civiles, arquitectónicos, eléctricos y mecánicos) requirió que se diera un proceso de desarrollo de los mismos a nivel conceptual que pudiese ser compartido bajo normas con diversos profesionales de la industria.

Es por ello, que especialmente en los lenguajes orientados a objetos, como es el caso de Java, el cual estaremos estudiando a fondo este semestre; se convierte en un proceso imperativo, antes de conocer cualquier tipo de sintaxis básica de este lenguaje, el conocer los procesos lógicos y su desarrollo que se da detrás de algunas líneas de códigos, volviéndose los diagramas UML en uno de nuestros más grandes aliados estratégicos para la realización de este fin.

**Infografía/Bibliografía**

1. Enrique Hernández Orallo, "El Lenguaje Unificado de Modelado (UML)", 2002
2. Teatro Abadía, Cátedra de Proyectos, “Diagramas de UML”, 2008