Sesión 3
Principios del Modelamiento Visual

Unidad 1

Mg. Gustavo G. Delgado Ugarte

DEFINICIÓN DE MODELAMIENTO

Definición de modelamiento

- Una organización exitosa es aquella que constantemente implementa software de calidad del software que satisfaga las necesidades de sus usuarios
- Una organización que puede desarrollar software en forma oportuna y previsible, con uso eficiente y eficaz de los recursos, tanto humanos como materiales, es la que tiene un negocio sostenible
- Un buen software, es aquel que satisfaga las necesidades cambiantes de los usuarios y el negocio

Definición de modelamiento

- Para implementar un software que cumple con sus metas, tiene que conocer y comprometer a los usuarios de una forma disciplinada, para exponer las necesidades reales de su sistema
- Para desarrollar software de calidad perdurable, tiene que diseñar una arquitectura básica sólida que es resistente al cambio
- Para desarrollar software de forma rápida, eficiente y eficaz, con un mínimo de desechos y retrabajo de software, es necesario tener las personas adecuadas, las herramientas adecuadas y el enfoque correcto
- Para hacer todo esto de forma consistente y predecible, con una apreciación de los costos en la vida útil del sistema, debe tener un proceso de desarrollo robusto que puede adaptarse a las necesidades cambiantes de su negocio y la tecnología

Definición de modelamiento

- El modelamiento es parte central de todas las actividades que conducen a la implementación de un buen software
- Construimos modelos para:
  - Comunicar la estructura deseada y el comportamiento de nuestro sistema
  - Visualizar y controlar la arquitectura del sistema
  - Comprender mejor el sistema que estamos construyendo, con frecuencia la exposición de las oportunidades para la simplificación y la reutilización
  - Gestionar el riesgo

LOS CUATRO PRINCIPIOS DEL MODELAMIENTO VISUAL

Los cuatro principios del modelamiento visual

1. La elección de qué modelos crear tiene una profunda influencia sobre la forma en que es atacado un problema y la forma una solución
2. Cada modelo puede ser expresado en diferentes niveles de precisión
3. Los mejores modelos se conectan a la realidad
4. Un único modelo o punto de vista es insuficiente. Cada sistema no trivial es abordado mejor a través de un pequeño conjunto de modelos casi independientes con múltiples puntos de vista

La elección de qué modelos crear tiene una profunda influencia sobre la forma en que es atacado un problema y la forma una solución

- En otras palabras, elegir bien los modelos
- Los modelos correctos
  - Iluminan brillantemente los problemas de desarrollo más “preversos”
  - Ofrecen una visión que simplemente no podría obtenerse de otra manera
- Los modelos errados inducen al error, lo que nos enfoca en cuestiones irrelevantes

"La elección de qué modelos crear tiene una profunda influencia sobre la forma en que es atacado un problema y la forma una solución"

- Rigurosas y continuas pruebas de los modelos, nos proveerán de un nivel mucho más alto de confianza en que el sistema que se ha modelado se comportará como lo espera en el mundo real
  - Ej. Comportamiento de Vientos de un edificio, mediante un modelo probado en un túnel de viento

“La elección de qué modelos crear tiene una profunda influencia sobre la forma en que es atacado un problema y la forma una solución”

- En el software, los modelos que elija puede afectar su visión del mundo
  - Si construye un sistema a través de los ojos de un desarrollador de bases de datos, es probable que se centrará en los modelos de entidad-relación que impulsan el comportamiento en los procedimientos almacenados y disparadores(triggers)
  - Si construye un sistema a través de los ojos de un analista de estructura, es probable que terminará con los modelos centrados en algoritmia, con flujos de datos de un proceso a otro
  - Si construye un sistema a través de los ojos de un desarrollador orientado a objetos, que terminará con un sistema cuya arquitectura está centrada en un mar de clases y patrones de interacción que dirigen cómo las clases trabajan juntas
  - Modelos ejecutables puede ayudar mucho a la prueba
- Cada visión del mundo conduce a un tipo diferente de sistema, con diferentes costos y beneficios

“Cada modelo puede ser expresado en diferentes niveles de precisión”

- Si está construyendo un rascacielos
  - A veces se necesita un punto de vista de 30.000 pies, para ayudar a los inversionistas visualizar su apariencia
  - A veces se necesita bajar al nivel del suelo, cuando hay un tramo de tubería difíciles o un elemento estructural inusual

“Cada modelo puede ser expresado en diferentes niveles de precisión”

- Lo mismo ocurre con los modelos de software
  - A veces, un modelo ejecutable rápido y simple de la interfaz de usuario es exactamente lo que necesita
  - Otras veces, se tiene que ir a nivel de bits, como cuando se va a especificar las interfaces entre sistemas o cuando se está luchando con cuellos de botella en redes
- Los mejores tipos de modelos son los que permiten elegir el grado de detalle, dependiendo de quién está visualizando qué y por qué
  - Un analista o un usuario final tendrá que centrarse en las cuestiones del qué
  - Un desarrollador tendrá que centrarse en las cuestiones del cómo.
  - Ambos quieren visualizar el sistema en diferentes niveles de detalle en diferentes momentos

“Los mejores modelos se conectan a la realidad”

- Un modelo físico de un edificio que no responde de la misma manera como lo hacen los materiales reales tiene sólo un valor limitado
- Un modelo matemático de una aeronave que asume sólo las condiciones ideales y la fabricación perfecta puede enmascarar algunas de las características potencialmente fatal de la aeronave real
- Es mejor tener modelos que tienen una clara conexión con la realidad, y donde la conexión es débil, saber exactamente cómo estos modelos se han divorciado del mundo real.
- Todos los modelos simplifican la realidad, el truco es asegurarse de que sus simplificaciones no enmascarar los detalles importantes

“Los mejores modelos se conectan a la realidad”

- El talón de Aquiles de las técnicas de análisis estructurado es que el modelo de análisis y el modelo de diseño están desconectados
  - El sistema concebido y el sistema construido divergen con el tiempo
- En los sistemas orientados a objetos, es posible conectar casi todos los puntos de vista independientes de un sistema en un todo semántico

“Un único modelo o punto de vista es insuficiente”

- En la construcción de un edificio, no existe un solo conjunto de planos que muestre todos sus detalles.
- Se necesitará los planos de planta, elevaciones, planos eléctricos, planos de calefacción y planos de Sanitarios.
- **Dentro de cualquier tipo de modelo, necesita múltiples puntos de vista para captar la amplitud del sistema**

“Un único modelo o punto de vista es insuficiente”

- Para entender la arquitectura de un sistema, se necesita varios puntos de vista complementarios entre sí:
  - Una visión de casos de uso (la exposición de los requisitos del sistema)
  - Una vista de diseño (capturar el vocabulario del espacio del problema y el espacio de soluciones)
  - Una visión de la interacción (mostrando las interacciones entre las partes del sistema y entre el sistema y el medio ambiente)
  - Un punto de vista de aplicación (abordando la realización física del sistema)
  - Una vista de despliegue (centrándose en cuestiones de ingeniería de sistemas)
- Cada uno de estos puntos de vista pueden tener aspectos estructurales, así como de comportamiento
- En conjunto, estas vistas representan los planos de software

EL LENGUAJE UNIFICADO DE MODELADO (UML)

UML

- El UML es un lenguaje para la
  - Visualización
  - Especificación
  - Construcción
  - Documentación

  de los artefactos de un sistema de software

UML es un Lenguaje

- Un lenguaje proporciona un vocabulario y las reglas para combinar palabras en dicho vocabulario con el fin de comunicarnos
- Un lenguaje de modelado es un lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema
- Un lenguaje de modelado como el UML es un lenguaje estándar para los modelos de software

UML es un Lenguaje para Visualización

- Problema 1
  - Para muchos programadores, la distancia entre el pensar una aplicación y luego codificarla es cercana a cero. (Lo piensas, lo codificas)
    - El programador se sigue haciendo modelos pero mentalmente
    - La comunicación de los modelos conceptuales a los demás está propenso a errores, a menos que todos los involucrados hablen el mismo idioma.
  - Los proyectos y las organizaciones desarrollan su propio lenguaje, y es difícil de entender lo que está pasando, si es usted un extraño o nuevo en el grupo

UML es un Lenguaje para Visualización

- Problema 2
  - Hay algunas cosas sobre un sistema de software que no se pueden entender, al menos que construyamos modelos que trasciendan al lenguaje de programación textual
    - El significado de una jerarquía de clases se puede deducir, pero no directamente, observando detenidamente el código para todas las clases en la jerarquía
    - La distribución física y la posible migración de los objetos en un sistema basado en la Web se puede deducir, pero no directamente, mediante el estudio de código del sistema.

UML es un Lenguaje para Visualización

- Problema 3
  - Si el desarrollador que cortar el código, nunca escribió los modelos que están en su cabeza, esta información se perderá para siempre o, en el mejor de los casos, sólo será parcialmente recreable a partir de la aplicación una vez que el desarrollador haya avanzado
- Escribir modelos en UML aborda el tercer problema: **Un modelo explícito facilita la comunicación**

UML es un Lenguaje para Visualización

- Algunas cosas son mejores en modelo textual, mientras que otros son mejores modelados de forma gráfica.
  - En todos los sistemas interesante, hay estructuras que trascienden lo que se pueda representar en un lenguaje de programación
- Esto aborda al segundo problema: **UML es un lenguaje gráfico**

UML es un Lenguaje para Visualización

- UML es algo más que un montón de símbolos gráficos; detrás de cada símbolo en la notación UML hay una semántica bien definida
  - Un programador puede escribir un modelo en UML, y otro desarrollador **puede interpretar ese modelo sin ambigüedad**
- Esto resuelve el primer problema

UML es un Lenguaje para Especificación

- **Especificación** significa construir **modelos precisos, sin ambigüedades y completos**
- UML aborda la especificación de todas las decisiones importantes de análisis, diseño e implementación que se deben tomar en el desarrollo y despliegue de un sistema de software

UML es un Lenguaje para Construcción

- UML no es un lenguaje de programación visual, pero sus modelos se pueden conectar directamente a una variedad de lenguajes de programación
  - Es posible mapear un modelo en UML a un lenguaje de programación como Java, C++ o Visual Basic, o incluso a las tablas de una base de datos relacional o una base de datos orientada a objetos

UML es un Lenguaje para Construcción

- Este mapeo permite a la ingeniería directa, la generación de código desde un modelo UML a un lenguaje de programación
- Es posible reconstruir un modelo a partir de una implementación de nuevo en UML (La ingeniería inversa)
- La combinación de estos dos caminos, la generación de código y la ingeniería inversa, permite tener la capacidad de trabajar ya sea en una vista gráfica o textual, mientras las herramientas mantienen coherentes las dos vistas

UML es un Lenguaje para Construcción

- UML es suficientemente expresivo y sin ambigüedades para permitir
  - La ejecución directa de modelos
  - La simulación de sistemas
  - La instrumentación de los sistemas que se ejecutan

UML es un Lenguaje para Documentación

- Una “sana” organización de software produce todo tipo de artefactos, además de código ejecutable. Estos artefactos incluyen (pero no limitados a)
  - Requisitos
  - Arquitectura
  - Diseño
  - El código fuente
  - Planes de Proyecto
  - Pruebas
  - Prototipos
  - Liberaciones

UML es un Lenguaje para Documentación

- Dependiendo de la cultura de desarrollo, algunos de estos artefactos son tratados más o menos formalmente que otros.
- **Tales artefactos** no son sólo los entregables de un proyecto, también **son críticos para controlar, medir y comunicar** acerca de un sistema durante su desarrollo y después de su implementación

UML es un Lenguaje para Documentación

- UML aborda la documentación de la arquitectura de un sistema y todos sus detalles
- UML proporciona un lenguaje para expresar los requisitos y las pruebas
- UML proporciona un lenguaje para el modelado de las actividades de la planificación del proyecto y gestión liberaciones

PROCESO DEL MODELADO VISUAL

Proceso del modelado visual

- El UML es en gran parte independiente del proceso, lo que significa que no está vinculado a ciclo de vida de desarrollo de software en particulpar.
- Sin embargo, para obtener el mayor beneficio de la UML, se debe considerar un proceso que:
  - Esté orientado a Casos de Uso
  - Esté centrado en la arquitectura
  - Sea Iterativo e incremental

Orientado a Casos de Uso

- Significa usar Casos de Uso como artefacto principal para establecer el comportamiento deseado del sistema, para comprobar y validar la arquitectura del sistema, para las pruebas, y para la comunicación entre los stakeholders del proyecto

Centrado en la arquitectura

- Significa que la arquitectura de un sistema se utiliza como un artefacto principal de la conceptualización, la construcción, gestión y evolución del sistema en desarrollo.

Iterativo e incremental

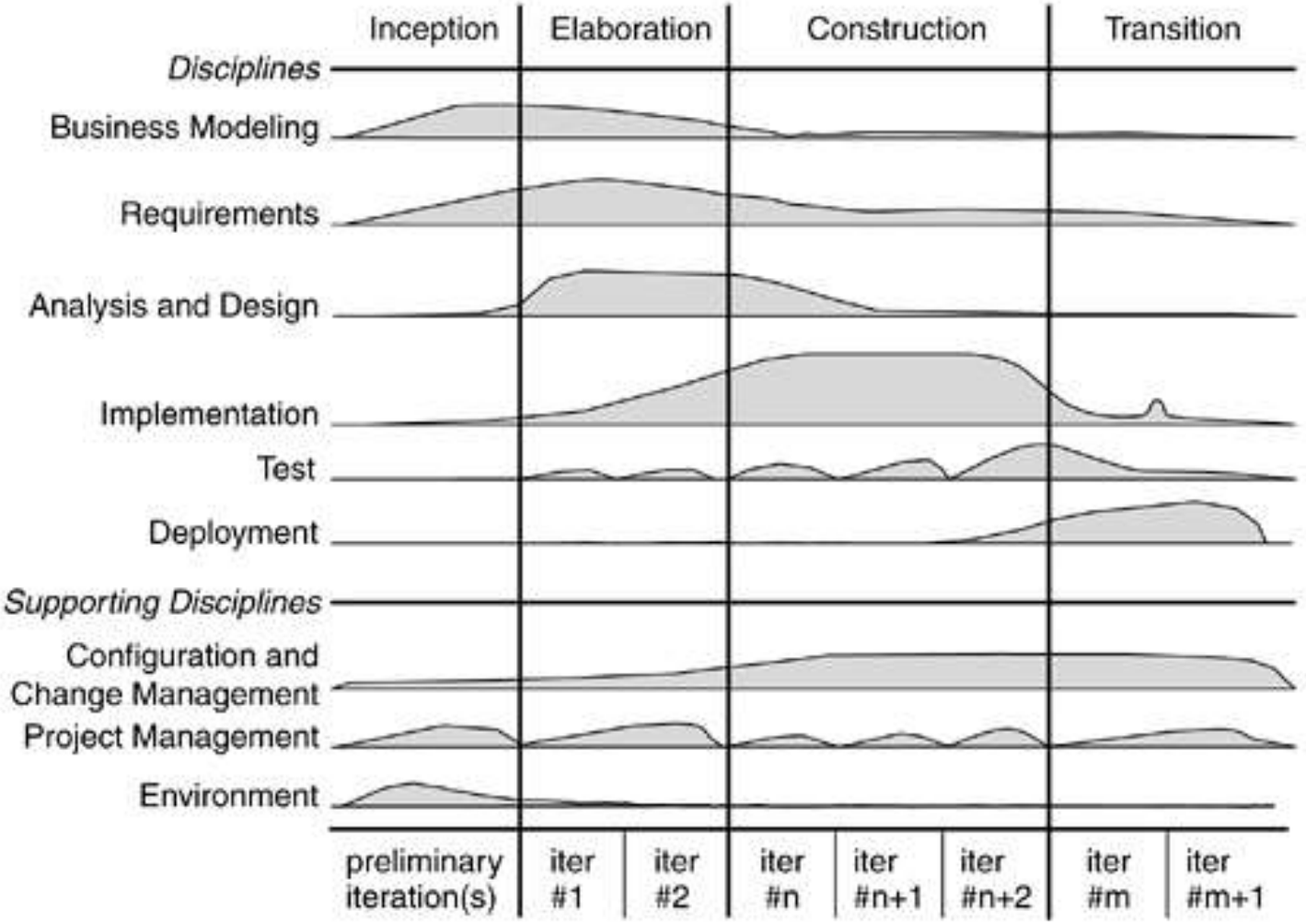
- Un proceso iterativo implica la gestión de una constante liberación de ejecutables
- Un proceso incremental implica la integración continua de la arquitectura del sistema para producir estas liberaciones, en cada nueva liberación se incorpora mejoras constantes de las otras
- En conjunto, un proceso iterativo e incremental es orientado a riesgos, lo que significa que cada nueva liberación se centra en atacar y reducir los riesgos más importantes para el éxito del proyecto

Fases

- Este proceso se puede dividir en fases
- Una fase es el lapso de tiempo entre dos hitos importantes del proceso, cuando un conjunto bien definido de los objetivos se cumplen, los artefactos se han completado, y se toman las decisiones si se debe pasar a la siguiente fase
- Hay cuatro fases en el ciclo de vida del software de desarrollo: inicio, elaboración, construcción y transición

Fases

Figure 2-24. Software Development Life Cycle



Fases

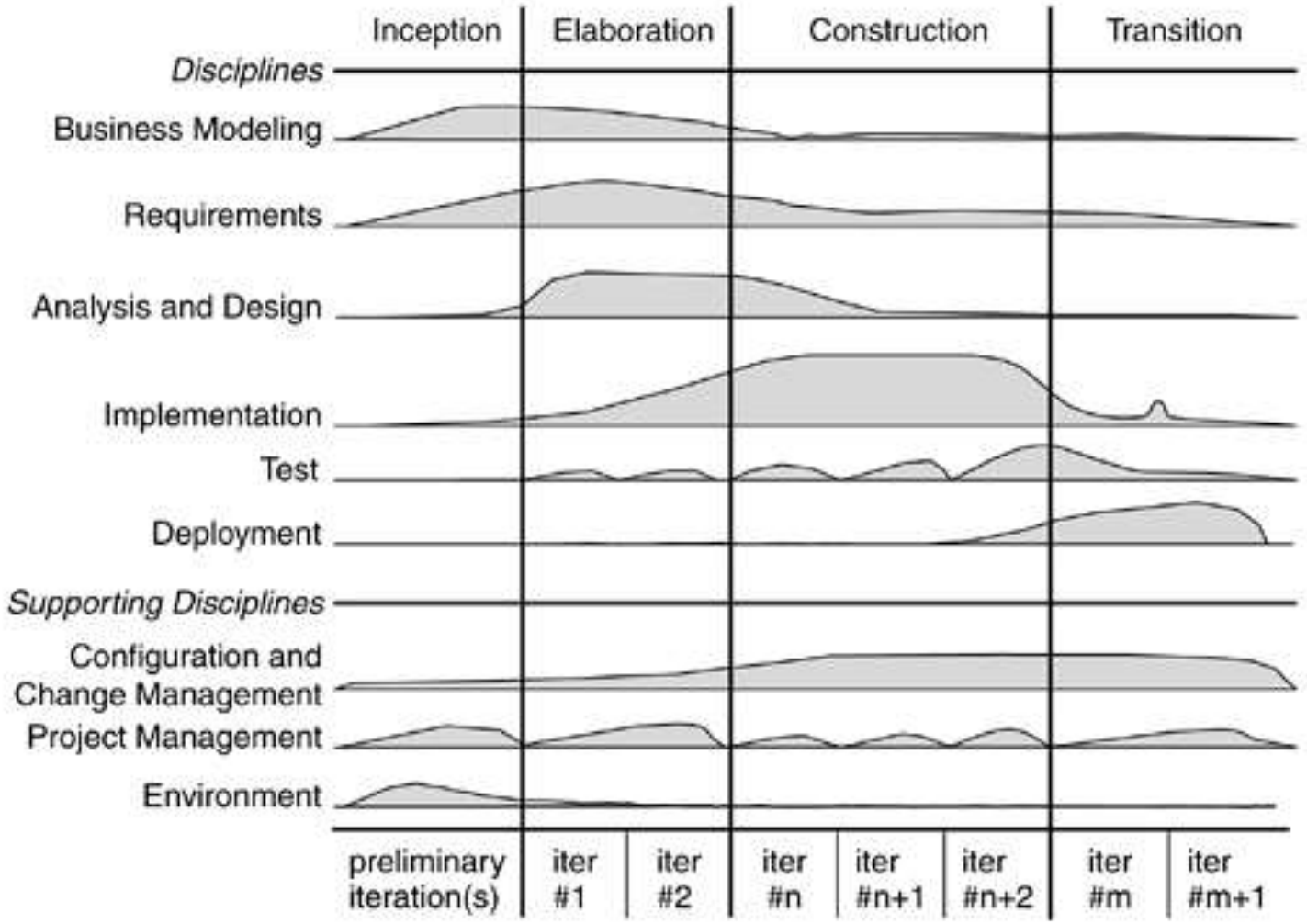
- **Inicio.-** Establece la visión, el alcance, y el plan inicial del proyecto
- **Elaboración.-** Diseña, implementa y prueba una arquitectura sólida y completa el plan de proyecto
- **Construcción.-** Construye la primera versión del sistema operativo
- **Transición.-** Entregar el sistema a sus usuarios finales

Fases

- Inicio y Elaboración se enfocan más en las actividades creativas y la ingeniería del ciclo de vida de desarrollo
- Construcción y Transición se enfocan más en las actividades de producción

Disciplinas

Figure 2-24. Software Development Life Cycle



Disciplinas

- El Rational Unified Process(RUP) consiste en nueve disciplinas
  - **Modelo de Negocio**.- Describe la estructura y la dinámica de la organización del cliente
  - **Requisitos.-** Obtiene requisitos usando una variedad de enfoques
  - **Análisis y diseño.-** Describe los múltiples puntos de vista arquitectónicos

Disciplinas

- **Aplicación.-** Toma en cuenta el desarrollo de software, pruebas unitarias y la integración
- **Prueba.-** Describe las secuencias de comandos, la ejecución de pruebas, y las métricas de seguimiento de defectos
- **Implementación.-** Incluye lista de materiales, notas de la versión, entrenamiento y otros aspectos de la entrega de una aplicación

Disciplinas

- **Gestión de configuración.-** Controla los cambios y mantiene la integridad de los artefactos de un proyecto y las actividades de gestión
- **Gestión de Proyectos.-** Describe las diversas estrategias de trabajo con un proceso iterativo
- **Medio Ambiente.-** Cubre la infraestructura necesaria para desarrollar un sistema