diseño

Del software

- **Niveles**
  - **Arquitectonico** — componentes
    - comportamiento
    - relaciones
  - **nivel** — 👁 — modulos ⚙️
  - **detallado**
    - como implementar — satisfagan requerimientos
    - + cerca codigo
- **Criterios**
  - **no cuantificables**
  - **correctitud** — el diseño
    - implementa requerimientos?
    - es factible? — dadas restricciones
  - **eficiencia** — uso apropiado de los recursos del sistema
  - **simplicidad** — mejor
    - comprension
    - testing
    - debugging
    - modificacion
    - → Mantenibilidad
  - no se llevan
- **Principios**
  - **abstraccion**
    - describe — comportamiento externo
    - detalles — sin
    - 📦
    - importante para
    - divide problema — + facil entender
    - **Tipos**
      - funcional — modulos
        - funciones I/O
        - pre y post condiciones
      - Datos — entidad
        - provee servicios
        - detalles ocultos
        - Solo se 👁 operaciones
  - **Jerarquia y particion**
    - DyC
    - partes manejables
      - solucion
      - Modificacion
      - → separada
    - costo — 📉 complejidad — a veces ❌💶
    - componentes discretas
      - implementacion — separada
      - modificacion — impacto minimo
  - **Modularidad**
    - Provee
    - soporte de
    - reduce costo
      - testing
      - debugging
      - mantenimiento
    - conjuncion de
      - abstraccion
      - particionado
  - **Estrategias**
    - top-down — arranca por componente mas alto nivel
    - bottom-up — arranca por componente mas bajo nivel

orientado a Objetos

- **Conceptos**
  - **Objetos**
    - datos + comportamiento
    - proposito diseño — definir
      - clases
      - relaciones
  - **Encapsulado**
    - propiedad basica objetos
    - oculta — datos
    - provee interfaces — acceder
    - brinda
      - abstraccion
      - ocultado informacion
  - **Estado**
    - mantiene informacion
    - define comportamiento — servicios + estado
    - otorga identidad
  - **Clases**
    - es una — plantilla crear objetos
    - define
      - estructura
      - servicios
    - tiene
      - interfaz — que puede accederse — exterior
      - cuerpo — implementa operaciones
      - variables de instancia — retener estado
    - operaciones
      - publicas — visible exterior
      - privadas — visible clase
      - protegidas — visible clase y subclase
  - **Relaciones**
    - asociacion — objeto vinculado con otro — durante un tiempo
    - agregacion
      - objeto es parte de otra clase
      - ciclo de vida — no relacionado
      - unidireccional
    - composicion
      - Está compuesto por otros
      - ciclo de vida — muy relacionado
  - **Herencia**
    - "Y" hereda de "X"
      - Y toma
        - atributos
        - operaciones
        - → X
      - Y = subclase
      - X = superclase, clase base
      - Y tiene
        - parte derivada — herencia
        - parte incremental — nueva
    - Crea relacion — "es un"
    - Tipos
      - Estricta
        - solo agrega
        - menos efectos colaterales
      - No estricta — redefine algo
  - **Polimofismo**
    - objeto puede — ser distintos tipos
    - objeto tiene
      - tipo estatico
      - tipo dinamico
    - vinculacion dinamica — codigo asociado — determina en ejecucion
- **Acoplamiento**
  - **caracteristicas**
    - intermodular
    - captura — fuerza interconexion modulos
  - **Por interaccion**
    - metodos clase — invocan metodos — otra clase
    - Ejemplos
      - acceso partes internas — otros metodos
      - manipulacion directa variables — otra clase
      - paso informacion — variables temporales
      - → 😀
      - Comunicacion por parametros
        - con menor
          - numero parametros posible
          - pasaje informacion posible
        - pasando solo datos — y no control
        - → 😀
  - **Por componentes**
    - Clase A — variables clase C
    - A acoplado C — => — A acoplado subclases C
    - Ejemplos
      - A tiene
        - Variables instancia C
        - Parametros C
        - metodo con variables locales C
        - → 😀
      - apariciones C en A son
        - atributos
        - parametros metodo
        - → 😀 — Osea, son visibles
  - **Por herencia**
    - A subclase C — => — A acoplado C
    - subclase modifica metodos — ☠️
    - subclase solo agrega
      - variables instancia
      - Metodos
      - → 🙂
- **Cohesion**
  - **caracteristicas**
    - intramodular
    - captura — cuanta relacion elementos modulo
  - **de metodo**
    - similar a — cohesion funciones
    - porque elementos juntos mismo metodo?
    - cada metodo — funcion claramente definida — todos elementos contribuyendo — 😀
    - se describe con — oracion simple — 😀
  - **de clase**
    - Porque
      - atributos
      - metodos
      - → misma clase?
    - clase = unico concepto — elementos contribuyendo — concepto
    - multiples conceptos — 😀
    - metodos separables grupos — acceden distintos atributos — 😀
  - **de herencia**
    - porque clases — misma jerarquia?
    - razones subclases
      - generalizacion-especializacion — 😀
      - Reuso
- **OMT**
  - **Caracteristicas**
    - punto partida — analisis OO
    - produce — modelo detallado final
  - **Conceptos**
    - evento: — solicitud operacion
    - Escenario:
      - identifican
      - secuencia eventos — ejecucion particular
      - como casos de uso
  - **Pasos**
    - 1. Producir el diagrama de clases. — diagrama obtenido en analisis
    - 2. Producir el modelo dinámico y usarlo para definir operaciones de las clases.
      - objetivo — especificar — cambio estado por eventos
      - modelar
        - escenarios iniciados — eventos externos
        - escenarios
          - exitosos
          - excepcionales
      - descubrimientos modelado — expandir diagrama clases
    - 3. Producir el modelo funcional y usarlo para definir operaciones de las clases.
      - objetivo
        - describe — operaciones sistema
        - especifica — computar valores salida — con valores entrada
      - hacer dfd — transformadores representan — operaciones objetos
    - 4.Identificar las clases internas y sus operaciones.
      - objetivo — considerar cuestiones implementacion
      - por cada clase — evaluar si es necesaria
      - considerar — implementacion operaciones clase
    - 5. Optimizar y empaquetar.
      - agregar asociaciones redundantes — optimizar acceso datos
      - guardar atributos derivados — para evitar — calculos complejos
      - usar tipos genericos
      - Ajustar herencia
        - subir en jerarquia — operaciones comunes
        - generacion clases abstractas
- **Metricas**
  - **WMC**
    - complejidad clase depende
      - cantidad metodos
      - su complejidad
    - formula
      - Mi, ..., Mn — metodos clase C
      - C(Mi) — complejidad metodo i
        - longitud estimada
        - complejidad interfaz
        - complejidad flujo datos
        - etc
      - $$WMC = \sum_{i=1}^{n} C(M_i)$$
    - uso — valor alto — clase propensa errores
    - acronimo? — Weighted Method Class
  - **DIT**
    - mide — profunidad jerarquia clases
    - DIT de C — profunidad desde raiz jerarquia
    - herencia multiple? — camino mas largo
    - uso — mayor DIT — mayor probabilidad errores
    - acronimo? — Depth Inheritance Tree
  - **NOC**
    - mide — cantidad subclases inmediatas
    - acronimo? — Number Of Children
    - uso
      - mayor NOC — mayor reuso
      - da idea influencia clase — mayor influencia — mayor importancia correccion diseño
  - **CBC**
    - acronimo? — Class Based Coupling
    - mide — cantidad clases acoplada
    - uso — mayor CBC — mayor probabilidad error
  - **RFC**
    - acronimo? — Responses For Class
    - mide — grado acoplamiento
    - es la — cantidad — metodos de C — + — metodos otras clases reciben mensaje de C
    - uso — RFC alto
      - complicado testear
      - clase propensa errores

Orientado a funciones

- **Modulo**
  - **Que es?**
    - Logicamente separado
    - Discreta
  - **Acoplamiento**
    - **Independencia**
      - funcionan sin la presencia de otro
      - Beneficios
        - implementacion
        - Testing
        - modificacion
        - → separada — menos 💵
      - total — No existe 🦄
    - **Caracteristicas**
      - Intermodular
      - No puede — reducirse en implementacion
    - **tipos conexiones**
      - interfaces
      - datos compartidos
    - **Complejidad interfaces**
      - minimizar
        - cantidad
        - Complejidad
        - → 😀
      - parametros al pedo? — 😀
      - interfaz compleja — 😀
    - **flujo informacion**
      - Tipos
        - Control
          - Accion depende informacion
          - Indica — necesidad separacion
        - Dato — funciones I/O
    - **Determinacion**
      - modulos
        - A
        - B
        - → entender A => entender B — Acoplamiento
    - **Ejemplos**
      - tipos conexiones
        - por entradas especificas — 😀
        - A elementos internos — 😀
      - Complejidad interfaces
        - simples, obvias — 😀
        - complicadas, oscuras — 😀
      - flujo informacion
        - datos — 😀
        - control — 🤓
        - datos+control — 😀
  - **Cohesion**
    - **Definicion** — componentes modulo — fuerza vinculo
    - **Caracteristicas** — -acoplamiento — +cohesion
      - intra-modular
    - **Niveles**
      - **casual**
        - sin significado
        - codigo cortado en 🧩
        - evitar codigo repetido
      - **logica** — funciones dentro — misma clase logica
      - **temporal** — logica — +
        - relacionados en el tiempo
        - se ejecutan juntos
      - **procedural** — misma unidad procedural
        - ciclo
        - secuencia decisiones
      - **comunicacional** — operan sobre el mismo dato
      - **secuencial**
        - salida de uno — entrada otro
        - caracteristicas
          - buena cohesion
          - facil mantener
          - dificil reusar
      - **funcional** — todos elementos — una funcion
    - **Determinacion**
      - oracion compuesta
        - ejemplo — comas o mas de un verbo
        - secuencial
        - comunicacional
      - Describir proposito en una oracion
        - palabras tiempo
          - ejemplo
            - primero
            - luego
            - cuando
            - despues
          - secuencial
          - temporal
        - Sin unico objeto especifico despues de verbo
          - ejemplo — "editar los datos"
          - logica
        - inicializar/limpiar — temporal
      - describen con — modulos cohesivos — oracion simple
  - **MDE**
    - **estructura**
      - se decide durante diseño
      - implementacion no la cambia
      - afecta mantenimiento
    - **Para que?**
      - Controlar estructura
      - Auxiliar diseñador — con pautas
      - Objetivo — especificar
        - modulos funciones
        - conexiones
        - → siguiendo
          - estructura jerarquica
            - bajo acoplaimiento
            - alta cohesion
    - **Donde?**
    - **Pasos**
      - 1. Reformular problema DFD.
        - identificar
          - entradas
          - salidas
          - fuentes
          - sumideros
        - arrancar desde
          - entrada
          - salida
        - etiquetar todo
        - hacer varios — para elegir uno
      - 2. Identificar las entradas y salidas más abstractas.
        - MAI — dato distante entrada — sigue considerandose entrada
        - MAO — dato distante salida — sigue considerandose salida
        - objetivo — separar transformadores centrales — realizan transformacion básica
        - como hallar? — ir mas lejos posible sin perder naturaleza
          - entrante
          - saliente
        - sepera sistema en
          - proceso entrada
          - transformaciones
          - proceso salida
      - 3. primer nivel factorización.
        - especificar
          - modulo principal
          - modulo entrada subordinado
            - cada item dato MAI
            - proposito — enviar principal
          - modulo salida subordinado — cada item dato MAO
          - transformador subordinado — cada transformador central
        - funciones
          - modulo principal — coordinador
          - modulos subordinados — entregar entradas logicas
          - modulos transformadores — obtener salidas logicas
          - modulos de salida — consumir
      - 4.1 Factorizar los módulos de entrada
        - transformador dato MAI — transformador central
        - repite proceso (3) — modulo entrada = modulo principal
      - 4.2 Factorizar los módulos de salida
        - transformador dato MAO — transformador central
        - repite proceso (3) — modulo salida = modulo principal
      - Paso 4.3: Factorizar los transformadores centrales
        - sin reglas
        - usar refinamiento top-down
      - 5. Mejorar la estructura
        - Usar heuristicas
          - tamaño modulo
          - flechas
            - salida
            - entrada
          - Alcance del efecto
          - Alcance del control
  - **Metricas**
    - **De red**
      - diagrama estructuras
        - 1 modulo invocador
        - menos arbol — 😀
        - → reduce — acoplamiento
      - formula
        - n - e - 1
          - n = nodos
          - e = aristas
        - no tiene en cuenta — uso comun rutinas
        - + chico — 😀
    - **De Estabilidad**
      - Estabilidad
        - trata de
          - capturar impacto cambios — sobre diseño
        - de un modulo
          - cantidad suposiciones otros modulos — sobre este
        - depende de
          - interfaz
          - datos globales
    - **de Flujo de informacion**
      - tiene en cuenta
        - complejidad intra-modulo — LOC
        - complejidad inter-modulo
          - inflow — info entrante
          - outflow — info saliente
      - formula
        - (Para un modulo C)
        - DC = tamaño * (inflow * outflow)²
          - define complejidad en
            - cantidad info
              - fluye adentro
              - fluye afuera
            - tamaño
        - DC =fan_in * fan_out + inflow * outflow
          - fan_in — modulos que llaman
          - fan_out — modulos llamados
          - importante cantidad modulos — fluye info — <= — tamaño no importa
      - Uso
        - determinar modulos
          - propenso error — DC > complej_media + desv_std
          - complejos — complej_media < DC < complej_media + desv_std
        - compara — modulos mismo sistema

Presented with xmind