PREGRADO

UNIDAD 1 | OVERVIEW

DESIGN FOUNDATION

SI720 | Diseño y Patrones de Software

Al finalizar la unidad de aprendizaje, el aplica los principios SOLID y buenas prácticas de arquitectura de información y datos para el diseño de aplicaciones, bajo el paradigma orientado a objetos.

AGENDA

INTRO

PRINCIPLES

STRATEGIES

Software Design

Defined as

“the process of defining the architecture, components, interfaces, and other characteristics of a system or component

“the result of [that] process”

As a process

Visto como proceso, diseño de software es la actividad del ciclo de vida de ingeniería de software en la que se analizan los requisitos de software, a fin de producir una descripción de la estructura interna del software que servirá de base para su construcción.

As a result

Visto como resultado, el diseño de software describe la arquitectura de software – esto es, cómo es software se descompone y organiza en componentes – así como las interfaces entre dichos componentes.

La descripción debería realizarse a un nivel que permita su construcción.

Blueprint

Durante el diseño de software, los ingenieros de software producen varios modelos que forman una suerte de blueprint de la solución a implementar.

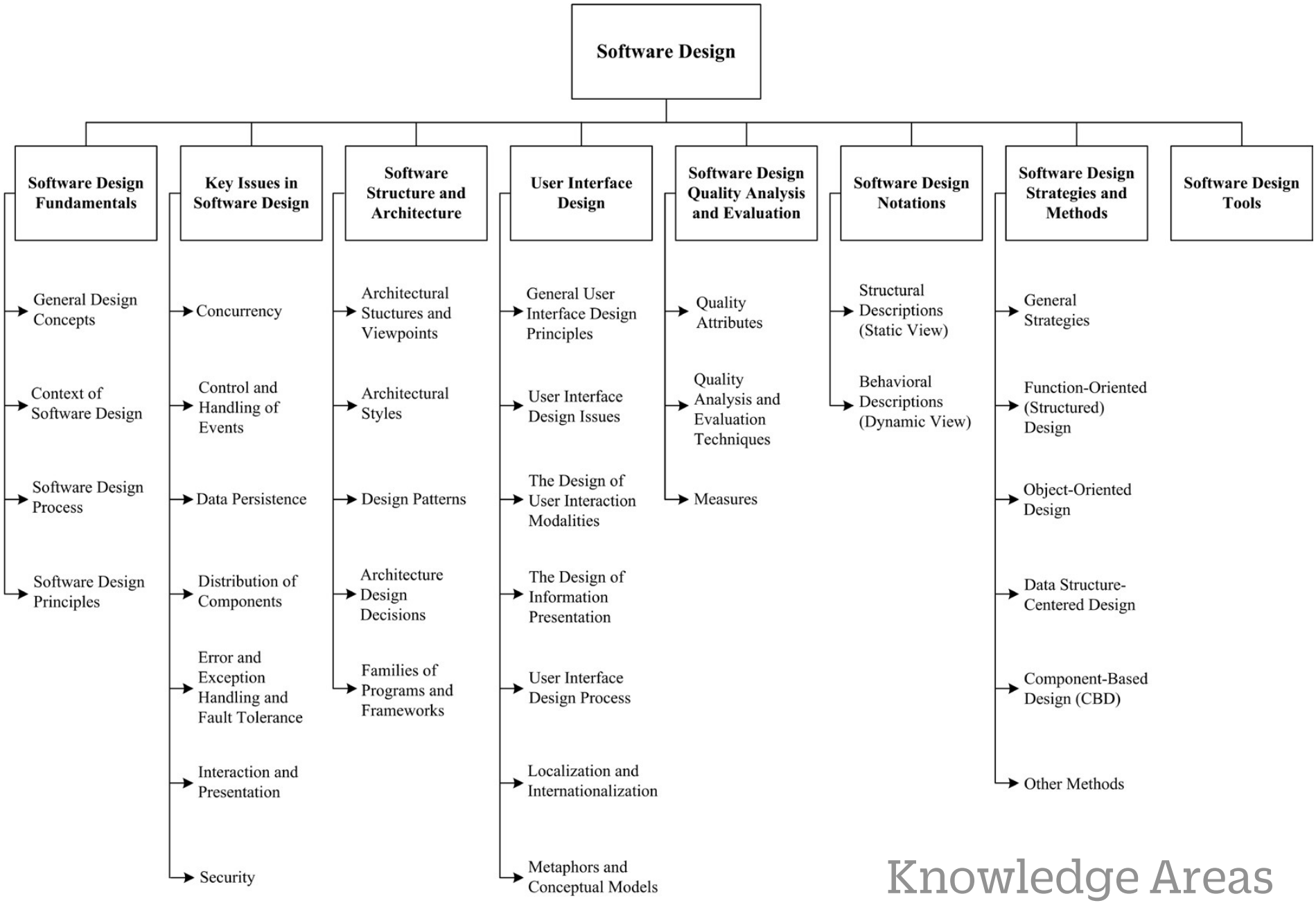
Dichos modelos son susceptibles de ser analizados y evaluados para determinar si permitirán o no cumplir con los diversos requisitos.

Los modelos son utilizados para planificar otras actividades como la verificación y validación, siendo también la base para la construcción y pruebas.

Actividades

Software architectural design (high-level design)

Software detailed design



Knowledge Areas

AGENDA

INTRO

PRINCIPLES

STRATEGIES

Principle

"A comprenhensive and fundamental law, doctrine, or assumption"

- Merriam-Webster's Collegiate Dictionary

Software design principles

Nociones clave que proporcionan las bases para diversas aproximaciones y conceptos de diseño de software.

Software design principles

Abstraction

Coupling and cohesion.

Decomposition and modularization.

Encapsulation/information hiding.

Separation of interface and implementation.

Sufficiency, completeness, and primitiveness.

Separation of concerns.

Abstraction

Es una vista de un objeto, la cual se enfoca en la información relevante para un propósito particular e ignora el resto de información.

Mecanismos de abstracción

Parametrización: Se abstrae de los detalles de representación de datos, representando éstos como parámetros nombrados.

Especificación: Tres clases: procedimental, de datos, de control o iteración.

Coupling and Cohesion

Coupling: “A measure of the interdependence among modules in a computer program”.

Cohesion: “A measure of the strength of association of elementos within a module”.

Decomposition and modularization

Descomponer en módulos significa que el software de gran envergadura se divide en un número de componentes más pequeños los cuales se pueden nombrar y que tienen interfaces que describen las interacciones entre dichos componentes.

La meta usualmente es ubicar las funcionalidades y responsabilidades en diferentes componentes.

Encapsulation and information hiding

Agrupar y empaquetar los detalles internos de una abstracción haciéndolos inaccesibles para entidades externas.

Separation of interface and implementation

Separar la interfaz de la implementación implica definir un componente especificando una interfaz pública (conocida por los clientes) que esté separada de los detalles de cómo están implementados dicho componente.

Sufficiency, completeness, and primitieveness

Alcanzar suficiencia y completez significa asegurarse que un componente de software captura todas las características importantes de una abstracción y nada más. Primitivismo significa que el diseño debería basarse en patrones que sean fáciles de implementar.

Separation of concerns

Un *concern* es un área de interés con respecto al diseño de un software que es relevante para uno o más de sus stakeholders. Cada vista de la arquitectura enmarca uno o más concerns. Separar los concerns en vistas permite que los stakeholders interesados se enfoquen en un conjunto reducido de cosas en un momento y ofrece un medio para administrar la complejidad.

SOLID software design principles

	Principle	Description
S	Single responsibility	Una clase debería tener una y solo una razón para cambiar, lo cual quiere decir que debería tener solo una función.
O	Open/closed	Los objetos de software deberían ser abiertos a la extension pero cerrados a la modificación.
L	Listkov substitution	Los objetos del mismo tipo deberían poder reemplazarse por otros de la misma categoría sin alterar la función del programa.
I	Interface segregation	Ningún cliente debería ser forzado a depender de métodos que no usa. Las interfaces del programa siempre deberían mantenerse pequeñas y separadas entre sí.
D	Dependency inversion	Los módulos de alto nivel no deberían depender de los módulos de bajo nivel, sin embargo ambos deberían depender de abstracciones.

AGENDA

INTRO

PRINCIPLES

STRATEGIES

Key Issues

Quality concerns (i.e. Performance, security, reliability, usability)

How to decompose, organize, and package software components.

Concurrency.

Control and Handling of Events.

Data Persistence.

Distribution of Components.

Error and Exception Handling, and Fault Tolerance.

Interaction and Presentation.

Security.

Software Structure and Architecture

Una arquitectura de software es el conjunto de estructuras necesarias para razonar sobre el sistema, que involucra elementos de software, relaciones entre ellos y propiedades de ambos.

Conceptos útiles en diferentes niveles de abstracción:

Architectural styles (Architetural Design)

Design Patterns (Detailed Design)

Software Design Strategies and Methods

General Strategies

Divide-and-conquer, stepwise refinement,

top-down vs. bottom-up,

heuristics, patterns, pattern language based,

Iterative and incremental approach.

Software Design Strategies and Methods

Function-Oriented (Structured) Design

Object-Oriented Design

Data Structure-Centered Design

Component-Based Design (CBD)

Aspect-oriented Design

Domain-Driven Design (DDD)

RESUMEN

Recordemos

Software Design

Design principles

SOLID

Strategies

REFERENCIAS

Para profundizar

https://www.computer.org/education/bodies-of-knowledge/software-engineering

PREGRADO

Ingeniería de Software

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería

UPC
Universidad Peruana
de Ciencias Aplicadas

Prolongación Primavera 2390,
Monterrico, Santiago de Surco
Lima 33 - Perú
T 511 313 3333
https://www.upc.edu.pe

exígete, innova