Проектиране на компонентите

- **Проектиране на компонентите**

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 7/e

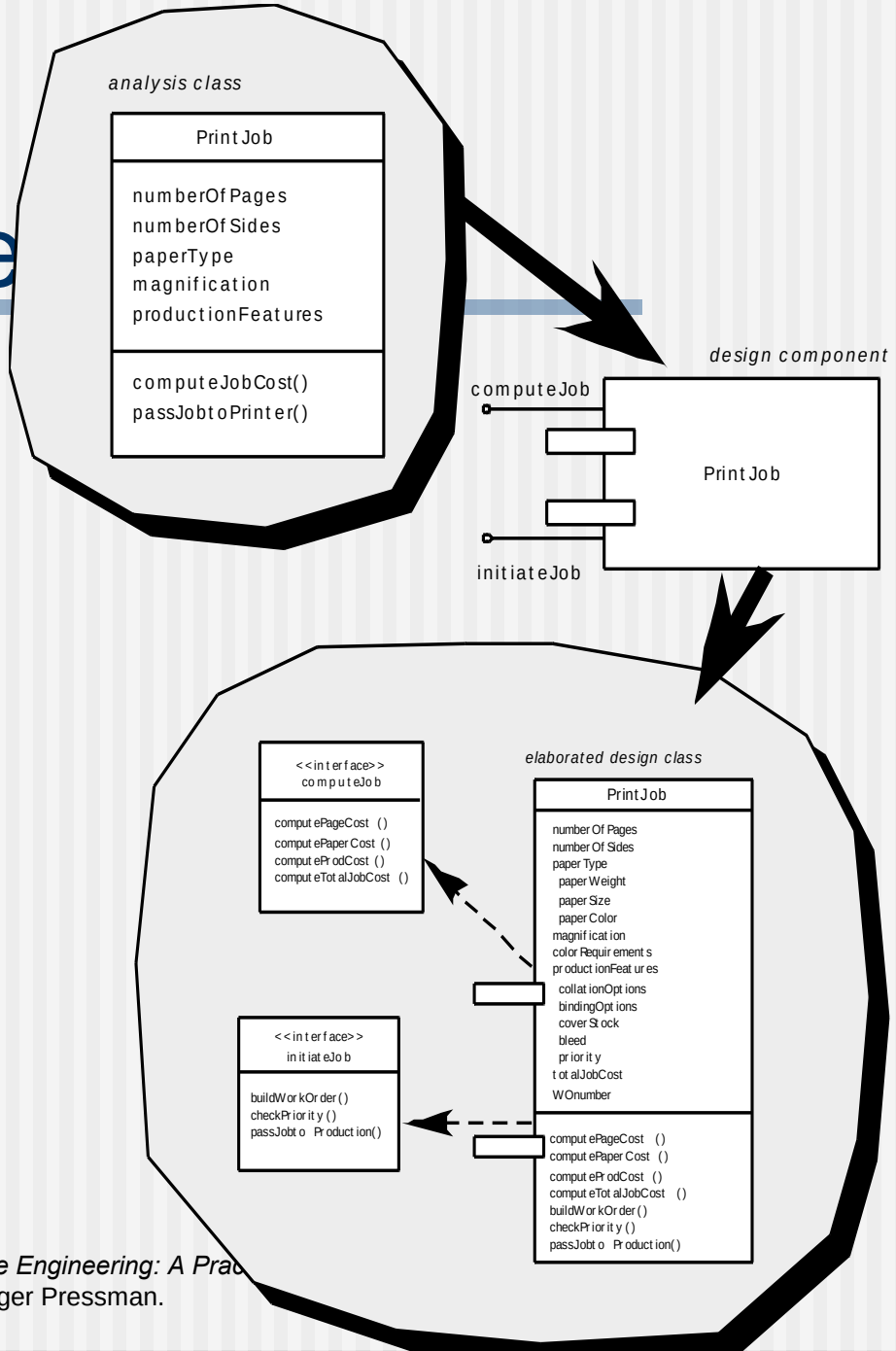
by Roger S. Pressman

Лектор: Доц. д-р Ася Стоянова-Дойчева

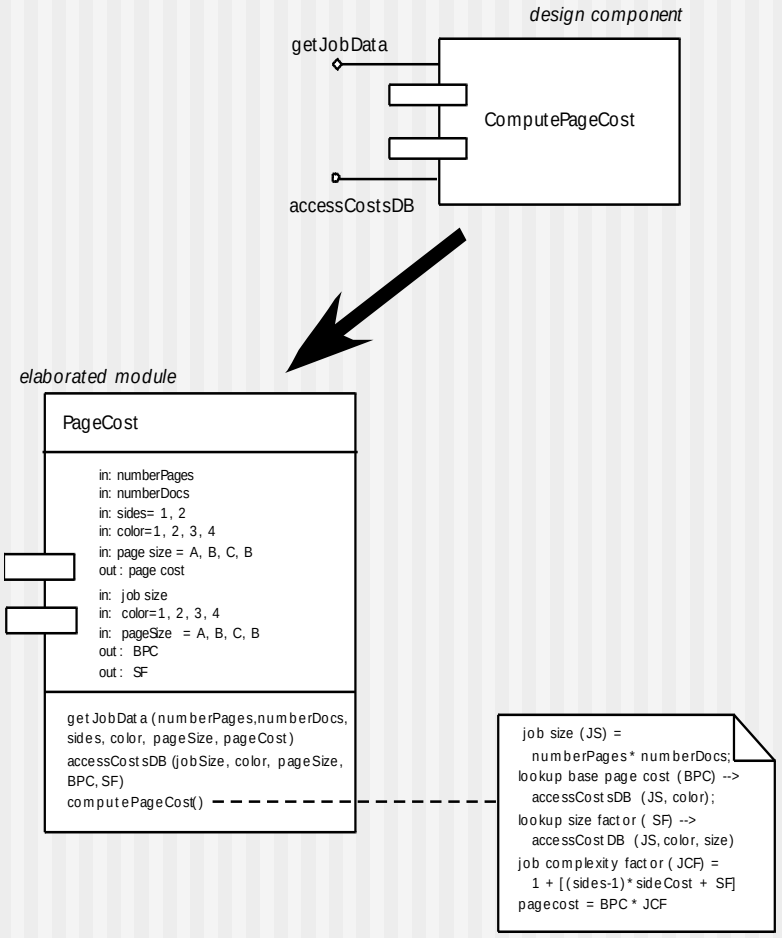
Какво е компонент?

- *OMG Unified Modeling Language Specification* [OMG01] дефинира компонента като:
  - "… модулна, възможна за реализация и заменяема част от системата, която капсулира имплементация и множество от интерфейси"
- *ОО изглед:* компонента съдържа множество от сътрудничещи си класове
- *Конвенционален изглед:* компонента съдържа логика, вътрешни структури от данни, които отговарят за имплементацията на логиката и интерфейси, които позволяват на компонента да бъде извикван и да бъдат прехвърляни данни към него.

OO компоне



Конвенционален компонент



Основни проектни принципи ООП (SOLID)

- **S**ingle Responsibility Principle (SRP). *„Класът трябва да има една единствена отговорност.“*
- **O**pen-Closed Principle (OCP). *“Модулът (компонента) трябва да бъде отворен за разширения, но затворен за модификации.“*
- **L**iskov Substitution Principle (LSP). *“Подкласовете трябва да бъдат заменими с техните базови класове.“*
- **I**nterface Segregation Principle (ISP). *“Много специфични клиентски интерфейси са по-добре от един с обща цел.“*
- **D**ependency Inversion Principle (DIP). *“Зависимост от абстракции. Не зависимост от конкретики.”*

Насоки за проектиране

- Компоненти
  - Трябва да бъде зададена конвенция за именуване на компонентите, които са специфицирани като част от архитектурния модел и след това усъвършенствана и приложена върху модела от компонентно ниво.
- Интерфейси
  - Интерфейсите предоставят важна информация за комуникацията и сътрудничеството на компонента
- Зависимости и наследяване
  - Добра идея е да моделираме зависимостите от ляво на дясно, а наследяванията от долу нагоре.

Съгласуваност

- Конвенционален изглед:
  - the “single-mindedness” of a module
- ОО изглед:
  - *съгласуваността* означава, че компонента или класа имат само атрибути и операции, които са тясно свързани един с друг и към класа или компонента

Свързване

- Конвенционален изглед:
  - Степента, до която компонент е свързан с други компоненти и с външния свят
- ОО изглед:
  - Количествена мярка на степента, до която класовете са свързани един с друг

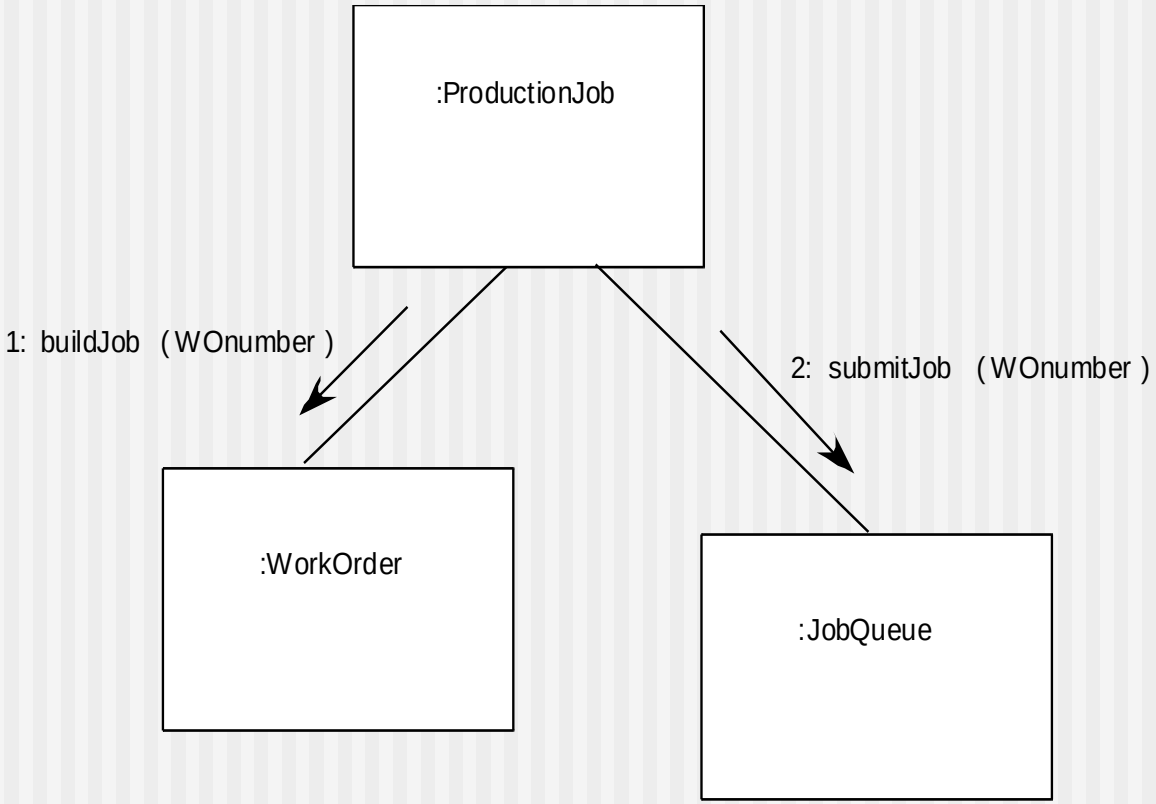
Проект на компонентите

- Стъпка 1. Идентифициране на всички проектни класове от проблемния домейн.
- Стъпка 2. Идентифициране на всички проектни класове от инфраструктурния домейн.
- Стъпка 3. Разработване на всички проектни класове, които не са компоненти за повторно използване.
- Стъпка 3a. Специфициране на детайли на съобщения, когато класовете или компонентите си комуникират.
- Стъпка3b. Идентифициране на подходящи интерфейси за всеки компонент.

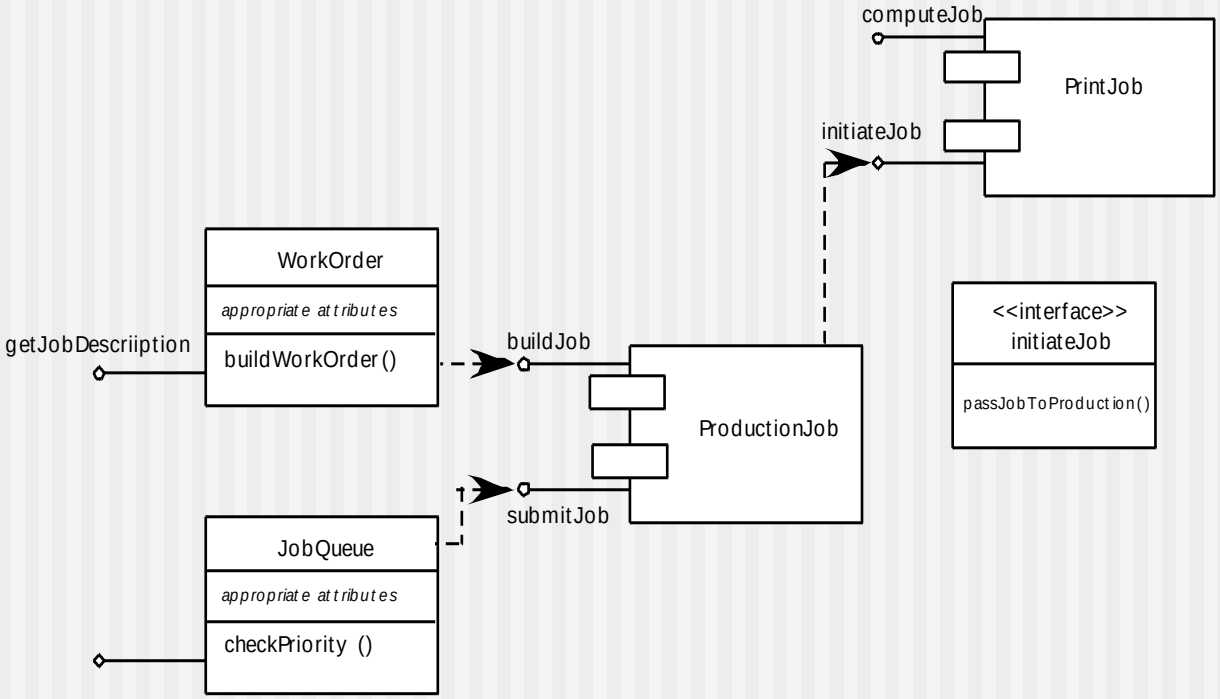
Проект на компонентите

- Стъпка 3c. Определяне на атрибути и дефиниране на типове данни и структури от данни необходими за имплементацията им.
- Стъпка 3d. Описание на потока на обработка за всяка операция.
- Стъпка 4. Описание на постоянните източници на данни (бази данни и файлове) и идентифициране на класовете, които ги управляват.
- Стъпка 5. Разработване и определяне на поведенческото представяне на класа или компонент.
- Стъпка 6. разработване на deployment диаграми за предоставяне на допълнителна информация за имплементацията.
- Стъпка 7. Управляване на проекта на компонентите и обмисляне на алтернативи.

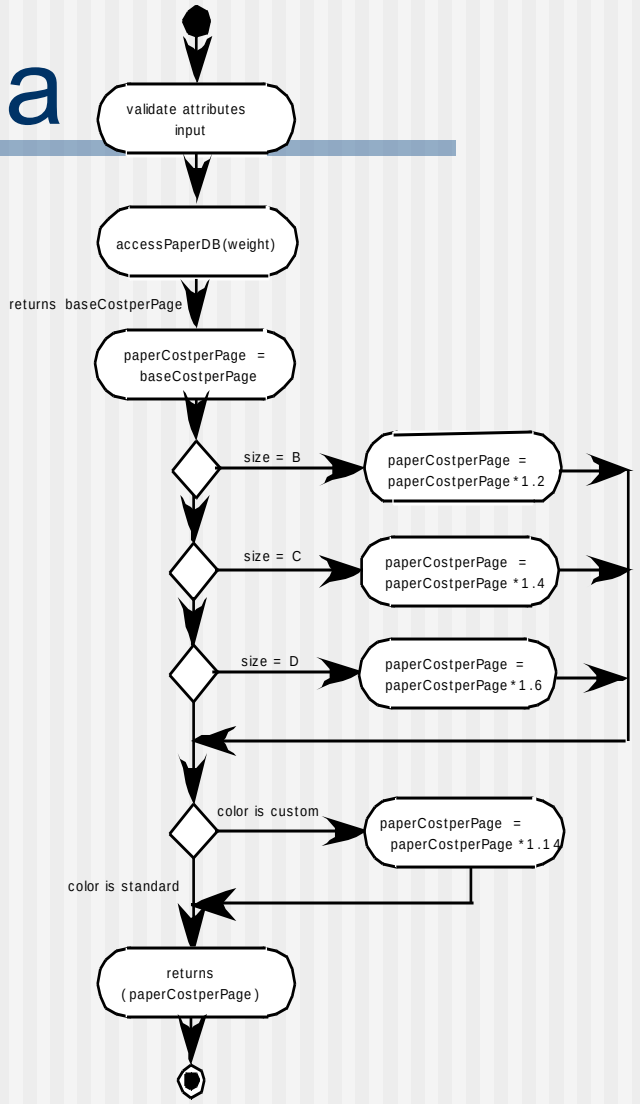
Collaboration диаграма



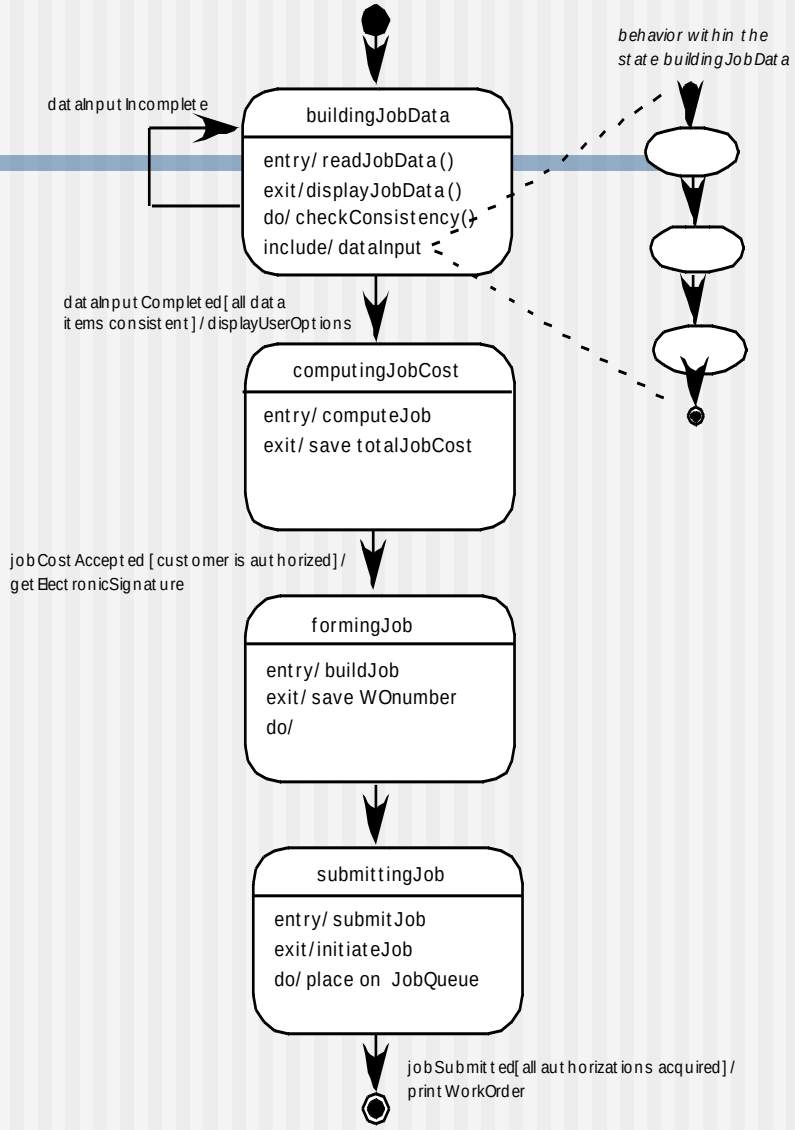
Refactoring



Activity диаграма



Statechart



Проектиране на компонентите при WebApps

- WebApp компонент е
  - (1) добре дефинирана съгласувана функция, която манипулира съдържание или обработва данни за крайния потребител или
  - (2) съгласуван пакет на съдържание и функционалност, което предоставя на крайния потребител искани възможности.
- Следователно проектирането на компоненти в WebApps често включва елементи на съдържанието и функционалността.

Проект на съдържанието за WebApps

- Фокусиране върху съдържателните обекти и начина, по който те могат да бъдат пакетирани за представяне на крайния потребител на WebApp

Проект на функционалностите за WebApps

- Съвременните уеб приложения предоставят все по-усъвършенствани функции за обработка като:
  - (1) изпълнява локализирана обработка за динамично генериране на съдържание и навигация;
  - (2) изпълнява изчисления или обработка на данни , подходящи за домейна на уеб приложението;
  - (3) предоставя сложни заявки към базата и достъп, или
  - (4) предоставя интерфейси за данни с външни корпоративни системи.

MVC-архитектура