# Builder pattern

Строител (Builder) е създаващ шаблон за дизайн. За разлика от методите абстрактна и метод фабрика, които имат за цел да усигорят полиморфизъм, Строител има за цел да реши проблемите с телескопичния конструктор (telescoping constructor anti-pattern), който възниква, когато увеличението на комбинациите от параметри в конструктора води до експоненциален списък на конструктори. Вместо използването на много конструктори, шаблонът използва обект, строител, който получава параметрите един по един и връща готовият обект.

Шаблонът има и друго предимство. Може да бъде използван за обекти, които съдържат информация, която трудно може да се промени (html код, SQL заявка). Този вид данни не могат да бъдат редактирани стъпка по стъпка, а трябва да бъдат обработени на веднъж. Най-добрият вариант за изработване на такъв обект, е чрез използване на строител.

Целта на този шаблон е да раздели създаването на сложни обекти от тяхното представяне. По този начин процесът на създаване може да имплементира различни видове обекти.

Строител (Builder pattern) често изгражда Компоситор (Composite pattern). Често, проекти започват с помощта на шаблона Фабрика Метод (Factory Method), като по-прост и адаптивен и се развиват към Абстрактна Фабрика (Abstract Factory), Прототип (Prototype), или Строител (Builder) (по-гъвкави, по-сложни), където е необходима по-голяма гъвкавост. Понякога създаващите шаблони се допълват: Builder може да използва един от другите модели за имплементиране на различни обекти.

# Composite pattern

Композиторният (Composite) шаблон описва, че група от обекти се третира по същия начин, както една инстанция на обект. Намерението на Композитор е да "композира" обекти в дървовидна структури. Прилагане на Композитор шаблона ни позволява да се отнасяме към отделните обекти и композиции по еднакъв начин.

Когато става въпрос за данни в дървовидна структура, програмиста често трябва да прави разлика между листо-възел и клон. Това прави кода по-сложен и следователно податлив на грешки. Решението е интерфейс, който позволява отнушение към сложни и примитивни обекти еднакво. В обектно-ориентираното програмиране, Композитор е обектно проектиран като композиция от един-или-повече подобни обекти, всички проявяващи сходна функционалност. Това е известно като "има" (“has-a”) зависимост между обектите. Основната концепция е, че можете да манипулирате един единствен екземпляр на обекта, точно както бихте манипулират група от тях.

Композитор трябва да се използва, когато може да се пренебрегне разликата между съставните и отделните обекти. Ако програмисти смятат, че те използват множество обекти по сходен начин, и често имат почти идентичен код, за да се справят с всеки един от тях, Композитор ще е подходящ за целта.

# Strategy pattern

Стратегически шаблон е поведенчески шаблон за дизайн, който позволява динамична замяна на поведението на алгоритъм, по време на работа на приложението. Този шаблон дефинира семейство от алгоритми, всеки един от тях е капсулиран като отделен обект и прави всеки отделен алгоритъм взаимно заменяем с останалите. Шаблонът позволява превключването от един към друг алгоритъм да става независимо от клиентския код, който ги използва.

Поведението на конкретен клас не се наследява. Използват се интерфейси, който дефинират външният вид на класовете , които ще го наследяват и имплементират. За целта се използва друг шаблон (композиция). По този начин се прави разделяне на поведението и класа, който използват това поведение. При използване на Стратегически шаблон, често се постига и намаляване на използваната памет.

Шаблонът прилича на Състояние (State pattern). Докато стратегическия шаблон промения поведението на даден обект, Състояние промения вида или състоянието на самия обект.