Strategy шаблонът е от типа шаблони които определят вид поведение.

Motivation – В практиката много често няколко различни класа се различават единствено по поведението си. В тези случаи е добра идея алгоритмите които се използват да се определят в различни класове, което позволява различната им имплементация да се избира at runtime.

Intent – Дефиниране на семейство алгоритми, които се енкапсулират и се извежда един общ интерфейс, който прави алгоритмите взаимнозаменяеми (един вид Dependency Injection). Strategy позволява алгоритмите да се различават независимо от класовете които ги използват (interchangeability).

Applicability – Да речем, че имаме клас който изпълнява някакъв вид валидация върху постъпващите данни. Тук може да се използва Strategy за да позволи на класа да избере подходящия тип валидация at runtime, спрямо подадения тип данни.

Structure – Strategy interface който енкапсулира реалното поведение на всеки тип алгоритъм – да речем тип калкулатор който в една ситуация трябва да събира числа, в друга да ги изважда. Client който имплементира Strategy interface-a и използва неговата инстанция. В някои случаи разработчика подава конкретната инстанция на поведение към клиента.

Related Patterns – В някои случаи Strategy шаблонът може да се използва заедно с Factory (дефиниран в самия Client) който намалява coupling-а между Client и специфичните поведения на Strategy.  
Strategy и Bridge шаблоните имат сходна структура (UML диаграма), но се различават в това, че Strategy дефинира тип заменяемо поведение, докато Bridge управлява структурата.

Sources:

<http://en.wikipedia.org/wiki/Strategy_pattern>  
<http://www.oodesign.com/strategy-pattern.html>  
<http://stackoverflow.com/questions/1710809/when-and-why-should-the-strategy-pattern-be-used>