**Что это такое?**

**Шаблон проектирования** или **паттерн** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *design pattern*) в [разработке программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) —  [архитектурная конструкция](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), представляющая собой решение проблемы [проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) в рамках некоторого часто возникающего [контекста](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82).

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в [код](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0); это лишь пример решения задачи, который можно использовать в различных ситуациях. [Объектно-ориентированные](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) шаблоны показывают отношения и [взаимодействия](https://ru.wikipedia.org/wiki/%D0%92%D0%B7%D0%B0%D0%B8%D0%BC%D0%BE%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D0%B5) между [классами](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) или [объектами](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), без определения того, какие конечные классы или объекты приложения будут использоваться.

«Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования, называются [идиомами](https://ru.wikipedia.org/wiki/%D0%98%D0%B4%D0%B8%D0%BE%D0%BC%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.

На наивысшем уровне существуют **архитектурные шаблоны**, они охватывают собой архитектуру всей [программной системы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0).

[Алгоритмы](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) по своей сути также являются шаблонами, но не проектирования, а [вычисления](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5), так как решают вычислительные задачи.

**Немного истории**

* В [1970-е](https://ru.wikipedia.org/wiki/1970-%D0%B5) годы [архитектор](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D0%BE%D1%80) [Кристофер Александр](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%B0%D0%BD%D0%B4%D0%B5%D1%80,_%D0%9A%D1%80%D0%B8%D1%81%D1%82%D0%BE%D1%84%D0%B5%D1%80) составил набор шаблонов проектирования. В области архитектуры эта идея не получила такого развития, как позже в области программной разработки.
* В [1987 году](https://ru.wikipedia.org/wiki/1987_%D0%B3%D0%BE%D0%B4) [Кент Бэк](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%BA,_%D0%9A%D0%B5%D0%BD%D1%82) (Kent Beck) и [Вард Каннингем](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%BD%D0%B8%D0%BD%D0%B3%D0%B5%D0%BC,_%D0%92%D0%B0%D1%80%D0%B4) (Ward Cunningham) взяли идеи Александра и разработали шаблоны применительно к разработке программного обеспечения для разработки графических оболочек на языке [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk).
* В [1988 году](https://ru.wikipedia.org/wiki/1988_%D0%B3%D0%BE%D0%B4) [Эрих Гамма](https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%BC%D0%B0,_%D0%AD%D1%80%D0%B8%D1%85) (Erich Gamma) начал писать докторскую диссертацию при цюрихском университете об общей переносимости этой методики на разработку программ.
* В 1989—1991 годах Джеймс Коплин (James Coplien) трудился над разработкой идиом для программирования на [C++](https://ru.wikipedia.org/wiki/C%2B%2B) и опубликовал в 1991 году книгу Advanced C++ Idioms.

В этом же году Эрих Гамма заканчивает свою докторскую диссертацию и переезжает в [США](https://ru.wikipedia.org/wiki/%D0%A1%D0%A8%D0%90), где в сотрудничестве с Ричардом Хелмом (Richard Helm), Ральфом Джонсоном (Ralph Johnson) и Джоном Влиссидсом (John Vlissides) публикует книгу [Design Patterns — Elements of Reusable Object-Oriented Software](https://ru.wikipedia.org/wiki/Design_Patterns). В этой книге описаны 23 шаблона проектирования. Также команда авторов этой книги известна общественности под названием «Банда четырёх» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Gang of Four*, часто сокращается до *GoF*). Именно эта книга стала причиной роста популярности шаблонов проектирования.

**Классификация**

* [3. Паттерны проектирования классов/обьектов](http://citforum.ru/SE/project/pattern/index.shtml#3)

Описание системы в терминах классов/обьектов следует считать низшим уровнем ее представления. В свою очередь, при моделировании системы на уровне классов/обьектов обычно проводят дополнительную типологизацию в двух аспектах, а именно, описывают структуру системы в терминах микроскопических элементов (см. [раздел 2](http://citforum.ru/SE/project/pattern/index.shtml#2)) и то, каким образом такая система обеспечивает требуемый функционал. Соответственно, среди паттернов проектирования выделены **структурные паттерны**, см. [раздел 3.1](http://citforum.ru/SE/project/pattern/index.shtml#3.1) и **паттерны распределения обязанностей между классами/объектами**, 3.2. Поскольку отдельные объекты создаются и уничтожаются в процессе работы системы, выделена еще одна большая группа **паттернов проектирования**, которые служат для создания обьектов, 3.3.

* + [3.1 Структурные паттерны проектирования классов/обьектов](http://citforum.ru/SE/project/pattern/index.shtml#3.1)
    - [3.1.1 Адаптер (Adapter) - GoF](http://citforum.ru/SE/project/pattern/index.shtml#3.1.1)
    - [3.1.2 Декоратор (Decorator) или Оболочка (Wrapper) - GoF](http://citforum.ru/SE/project/pattern/index.shtml#3.1.2)
    - [3.1.3 Заместитель (Proxy) или Суррогат (Surrogate) - GoF](http://citforum.ru/SE/project/pattern/index.shtml#3.1.3)
    - [3.1.4 Информационный эксперт (Information Expert)- GRASP](http://citforum.ru/SE/project/pattern/index.shtml#3.1.4)
    - [3.1.5 Компоновщик (Composite) - GoF](http://citforum.ru/SE/project/pattern/index.shtml#3.1.5)
    - [3.1.6 Мост (Bridge), Handle (описатель) или Тело (Body) - GoF](http://citforum.ru/SE/project/pattern/index.shtml#3.1.6)
    - [3.1.7 Низкая связанность (Low Coupling) - GRASP](http://citforum.ru/SE/project/pattern/index.shtml#3.1.7)
    - [3.1.8 Приспособленец (Flyweight) - GoF](http://citforum.ru/SE/project/pattern/index.shtml#3.1.8)
    - [3.1.9 Устойчивый к изменениям (Protected Variations) - GRASP](http://citforum.ru/SE/project/pattern/index.shtml#3.1.9)
    - [3.1.10 Фасад (Facade) - GoF](http://citforum.ru/SE/project/pattern/index.shtml#3.1.10)
  + [3.2 Паттерны проектирования поведения классов/обьектов](http://citforum.ru/SE/project/pattern/p_1.shtml#3.2)
    - [3.2.1 Интерпретатор (Interpreter ) - GoF](http://citforum.ru/SE/project/pattern/p_1.shtml#3.2.1)
    - [3.2.2 Итератор (Iterator) или Курсор (Cursor) - GoF](http://citforum.ru/SE/project/pattern/p_1.shtml#3.2.2)
    - [3.2.3 Команда (Command), Действие (Action) или Транзакция (Транзакция) - GoF](http://citforum.ru/SE/project/pattern/p_1.shtml#3.2.3)
    - [3.2.4 Наблюдатель (Observer), Опубликовать - подписаться (Publish - Subscribe) или Delegation Event Model - GoF](http://citforum.ru/SE/project/pattern/p_1.shtml#3.2.4)
    - [3.2.5 Не разговаривайте с неизвестными (Don't talk to strangers) - GRASP](http://citforum.ru/SE/project/pattern/p_1.shtml#3.2.5)
    - [3.2.6 Посетитель (Visitor) - GoF](http://citforum.ru/SE/project/pattern/p_1.shtml#3.2.6)
    - [3.2.7 Посредник (Mediator) - GoF](http://citforum.ru/SE/project/pattern/p_1.shtml#3.2.7)
    - [3.2.8 Состояние (State) - GoF](http://citforum.ru/SE/project/pattern/p_2.shtml#3.2.8)
    - [3.2.9 Стратегия (Strategy) - GoF](http://citforum.ru/SE/project/pattern/p_2.shtml#3.2.9)
    - [3.2.10 Хранитель (Memento) - GoF](http://citforum.ru/SE/project/pattern/p_2.shtml#3.2.10)
    - [3.2.11 Цепочка обязанностей (Chain of Responsibility) - GoF](http://citforum.ru/SE/project/pattern/p_2.shtml#3.2.11)
    - [3.2.12 Шаблонный метод (Template Method) - GoF](http://citforum.ru/SE/project/pattern/p_2.shtml#3.2.12)
    - [3.2.13 Высокое зацепление (High Cohesion) - GRASP](http://citforum.ru/SE/project/pattern/p_2.shtml#3.2.13)
    - [3.2.14 Контроллер (Controller) - GRASP](http://citforum.ru/SE/project/pattern/p_2.shtml#3.2.14)
    - [3.2.15 Полиморфизм (Polymorphism) - GRASP](http://citforum.ru/SE/project/pattern/p_2.shtml#3.2.15)
    - [3.2.16 Искусственный (Pure Fabrication) - GRASP](http://citforum.ru/SE/project/pattern/p_2.shtml#3.2.16)
    - [3.2.17 Перенаправление (Indirection) - GRASP](http://citforum.ru/SE/project/pattern/p_2.shtml#3.2.17)
  + [3.3 Порождающие паттерны проектирования](http://citforum.ru/SE/project/pattern/p_2.shtml#3.3)
    - [3.3.1 Абстрактная фабрика (Abstract Factory, Factory), др. название Инструментарий (Kit) - GoF](http://citforum.ru/SE/project/pattern/p_2.shtml#3.3.1)
    - [3.3.2 Одиночка (Singleton) - GoF](http://citforum.ru/SE/project/pattern/p_2.shtml#3.3.2)
    - [3.3.3 Прототип (Prototype) - GoF](http://citforum.ru/SE/project/pattern/p_2.shtml#3.3.3)
    - [3.3.4 Создатель экземпляров класса (Creator) - GRASP](http://citforum.ru/SE/project/pattern/p_2.shtml#3.3.4)
    - [3.3.5 Строитель (Builder) - GoF](http://citforum.ru/SE/project/pattern/p_2.shtml#3.3.5)
    - [3.3.6 (Фабричный метод) Factory Method или Виртуальный конструктор (Virtual Constructor) - GoF](http://citforum.ru/SE/project/pattern/p_2.shtml#3.3.6)
* [4 Архитектурные системные паттерны](http://citforum.ru/SE/project/pattern/p_3.shtml#4)

архитектурные системные паттерны объединены в группы в соответствии с теми задачами, для решения которых они разработаны. Для организации классов или обьектов системы в базовые подструктуры используются **структурные** архитектурные паттерны, см. [4.1](http://citforum.ru/SE/project/pattern/p_3.shtml#4.1). С другой стороны, для обеспечения требуемого системного функционала первостепенное значение имеет адекватная организация взаимодействия отдельных архитектурных элементов системы - этой цели служат **паттерны управления**, см. [раздел 4.2](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2).

* + [4.1 Структурные паттерны](http://citforum.ru/SE/project/pattern/p_3.shtml#4.1)
    - [4.1.1 Репозиторий](http://citforum.ru/SE/project/pattern/p_3.shtml#4.1.1)
    - [4.1.2 Клиент/сервер](http://citforum.ru/SE/project/pattern/p_3.shtml#4.1.2)

[4.1.4 Многоуровневая система (Layers) или абстрактная машина](http://citforum.ru/SE/project/pattern/p_3.shtml#4.1.4)

* + - [4.1.5 Потоки данных (конвейер или фильтр)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.1.51)

Можно MVC

* + [4.2 Паттерны управления](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2)
    - [4.2.1 Паттерны централизованного управления](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.1)
      * [4.2.1.1 Вызов - возврат (сценарий транзакции - частный случай).](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.1.1)
      * [4.2.1.2 Диспетчер](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.1.2)
    - [4.2.2 Паттерны управления, основанные на событиях](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.2)
      * [4.2.2.1 Передача сообщений](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.2.1)
      * [4.2.2.2 Управляемый прерываниями](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.2.2)
    - [4.2.3 Паттерны, обеспечивающие взаимодействие с базой данных](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3)
      * [4.2.3.1 Активная запись (Active Record)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.1)
      * [4.2.3.2 Единица работы (Unit Of Work)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.2)
      * [4.2.3.3 Загрузка по требованию (Lazy Load)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.3)
      * [4.2.3.4 Коллекция обьектов (Identity Map)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.4)
      * [4.2.3.5 Множество записей (Record Set)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.5)
      * [4.2.3.6 Наследование с одной таблицей (Single Table Inheritance)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.6)
      * [4.2.3.7 Наследование с таблицами для каждого класса (Class Table Inheritance)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.7)
      * [4.2.3.8 Оптимистическая автономная блокировка (Optimistic Offline Lock)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.8)
      * [4.2.3.9 Отображение с помощью внешних ключей](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.9)
      * [4.2.3.10 Отображение с помощью таблицы ассоциаций (Association Table Mapping)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.10)
      * [4.2.3.11 Пессимистическая автономная блокировка (Pessimistic Offline Lock)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.11)
      * [4.2.3.12 Поле идентификации (Identity Field)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.12)
      * [4.2.3.13 Преобразователь данных (Data Mapper)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.13)
      * [4.2.3.14 Cохранение сеанса на стороне клиента (Client Session State)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.14)
      * [4.2.3.15 Cохранение сеанса на стороне сервера (Server Session State)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.15)
      * [4.2.3.16 Шлюз записи данных (Row Data Gateway)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.16)
      * [4.2.3.17 Шлюз таблицы данных (Table Data Gateway)](http://citforum.ru/SE/project/pattern/p_3.shtml#4.2.3.17)
      * Еще DAO и DTO
* [5 Паттерны интеграции корпоративных информационных систем](http://citforum.ru/SE/project/pattern/p_4.shtml#5)

Паттерны интеграции информационных систем представляют собой , как это описано в разделе 2, **верхний уровень классификации** паттернов проектирования. Аналогично паттернам более низких уровней классификации, среди паттернов интеграции выделена группа **структурных паттернов**, см.[раздел 5.1](http://citforum.ru/SE/project/pattern/p_4.shtml#5.1). Структурные паттерны описывают основные компоненты единой интегрированной метасистемы. В свою очередь, для описания взаимодействия отдельных корпоративных систем, включенных в интегрированную метасистему, организована группа паттернов, объединенных в соответствии с тем или иным **методом интеграции**, см. [раздел 5.2](http://citforum.ru/SE/project/pattern/p_4.shtml#5.2). Далее, интеграция корпоративных информационных систем подразумевает тем или иным способом организованный обмен данными между системами. Для организации обмена информацией между отдельными системами, включенными в интегрированную метасистему, служит раздел 5.3. Следует отметить, что в отличие от паттернов проектирования классов/обьектов и архитектурных системных паттернов, отнесение отдельного паттерна интеграции к тому или иному виду является менее условным.

* + [5.1 Структурные паттерныны интеграции](http://citforum.ru/SE/project/pattern/p_4.shtml#5.1)
    - [5.1.1 Взаимодействие "точка - точка"](http://citforum.ru/SE/project/pattern/p_4.shtml#5.1.1)
    - [5.1.2 Взаимодействие "звезда" (интегрирующая среда)](http://citforum.ru/SE/project/pattern/p_4.shtml#5.1.2)
    - [5.1.3 Смешанный способ взаимодействия](http://citforum.ru/SE/project/pattern/p_4.shtml#5.1.3)
  + [5.2 Паттерны по методу интеграции](http://citforum.ru/SE/project/pattern/p_4.shtml#5.2)
    - [5.2.1 Интеграция систем по данным (data-centric).](http://citforum.ru/SE/project/pattern/p_4.shtml#5.2.1)
    - [5.2.2 Функционально-центрический (function-centric) подход.](http://citforum.ru/SE/project/pattern/p_4.shtml#5.2.2)
    - [5.2.3 Объектно-центрический (object-centric).](http://citforum.ru/SE/project/pattern/p_4.shtml#5.2.3)
    - [5.2.4 Интеграция на основе единой понятийной модели предметной области (concept-centric).](http://citforum.ru/SE/project/pattern/p_4.shtml#5.2.4)
  + [5.3 Паттерны интеграции по типу обмена данными](http://citforum.ru/SE/project/pattern/p_4.shtml#5.3)
    - [5.3.1 Файловый обмен](http://citforum.ru/SE/project/pattern/p_4.shtml#5.3.1)
    - [5.3.2 Общая база данных](http://citforum.ru/SE/project/pattern/p_4.shtml#5.3.2)
    - [5.3.3 Удаленный вызов процедур](http://citforum.ru/SE/project/pattern/p_4.shtml#5.3.3)
    - [5.3.4 Обмен сообщениями](http://citforum.ru/SE/project/pattern/p_4.shtml#5.3.4)

#### Шаблоны параллельного программирования ([Concurrency](https://en.wikipedia.org/wiki/Concurrency_pattern))

|  |  |  |
| --- | --- | --- |
| [Active Object](https://ru.wikipedia.org/wiki/Active_Object) | [Active object](https://en.wikipedia.org/wiki/Active_object) | Служит для отделения потока выполнения метода от потока, в котором он был вызван. Использует шаблоны асинхронный вызов методов и планировщик. |
| [Balking](https://ru.wikipedia.org/w/index.php?title=Balking&action=edit&redlink=1) | [Balking](https://en.wikipedia.org/wiki/Balking_pattern) | Служит для выполнения действия над объектом только тогда, когда тот находится в корректном состоянии. |
|  | Binding Properties | Комбинирует несколько наблюдателей для обеспечения синхронизации свойств в различных объектах[[7]](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F#cite_note-7) |
| [Обмен сообщениями](https://ru.wikipedia.org/w/index.php?title=Messaging_design_pattern_(MDP)&action=edit&redlink=1) | [Messaging design pattern (MDP)](https://en.wikipedia.org/wiki/Messaging_pattern) | Позволяет компонентам и приложениям обмениваться информацией (сообщениями). |
| [Блокировка с двойной проверкой](https://ru.wikipedia.org/wiki/Double_checked_locking) | [Double-checked locking](https://en.wikipedia.org/wiki/Double_checked_locking_pattern) | Предназначен для уменьшения накладных расходов, связанных с получением блокировки. |
|  | [Event-based asynchronous](https://en.wikipedia.org/wiki/Event-Based_Asynchronous_Pattern) | Адресные проблемы с Асинхронным паттерном, которые возникают в программах с несколькими потоками.[[8]](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F#cite_note-PC.232008-8) |
| [Охраняемая приостановка](https://ru.wikipedia.org/wiki/%D0%9E%D1%85%D1%80%D0%B0%D0%BD%D1%8F%D0%B5%D0%BC%D0%B0%D1%8F_%D0%BF%D1%80%D0%B8%D0%BE%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0_(%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)) | [Guarded suspension](https://en.wikipedia.org/wiki/Guarded_suspension) | Используется для блокировки выполнения действия над объектом только тогда, когда тот находится в корректном состоянии. |
| [Half-Sync/Half-Async](https://ru.wikipedia.org/w/index.php?title=Half-Sync/Half-Async&action=edit&redlink=1) |  |  |
| [Leaders/followers](https://ru.wikipedia.org/w/index.php?title=Leaders/followers&action=edit&redlink=1) |  |  |
|  | [Lock](https://en.wikipedia.org/wiki/Lock_(computer_science)) | Один поток блокирует ресурс для предотвращения доступа или изменения его другими потоками.[[9]](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F#cite_note-9) |
| [Монитор](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D0%B8%D1%82%D0%BE%D1%80_(%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)) | [Monitor object](https://en.wikipedia.org/wiki/Monitor_(synchronization)) | Объект, предназначенный для безопасного использования более чем одним потоком. |
| [Reactor](https://ru.wikipedia.org/w/index.php?title=Reactor&action=edit&redlink=1) | [Reactor](https://en.wikipedia.org/wiki/Reactor_pattern) | Предназначен для синхронной передачи запросов сервису от одного или нескольких источников. |
| [Read write lock](https://ru.wikipedia.org/w/index.php?title=Read_write_lock&action=edit&redlink=1) | [Read-write lock](https://en.wikipedia.org/wiki/Read/write_lock_pattern) | Позволяет нескольким потокам одновременно считывать информацию из общего хранилища, но позволяя только одному потоку в текущий момент времени её изменять. |
| [Планировщик](https://ru.wikipedia.org/wiki/Scheduler) | [Scheduler](https://en.wikipedia.org/wiki/Scheduler_pattern) | Обеспечивает механизм реализации политики планирования, но при этом не зависящих ни от одной конкретной политики. |
| [Thread pool](https://ru.wikipedia.org/w/index.php?title=Thread_pool&action=edit&redlink=1) | [Thread pool](https://en.wikipedia.org/wiki/Thread_pool_pattern) | Предоставляет пул потоков для обработки заданий, представленных обычно в виде очереди. |
| [Thread-Specific Storage](https://ru.wikipedia.org/w/index.php?title=Thread-Specific_Storage&action=edit&redlink=1) | [Thread-specific storage](https://en.wikipedia.org/wiki/Thread-Specific_Storage) | Служит для предоставления различных глобальных переменных для разных потоков. |
| [Однопоточное выполнение](https://ru.wikipedia.org/wiki/Single_Thread_Execution) | Single Thread Execution | Препятствует конкурентному вызову метода, тем самым запрещая параллельное выполнение этого метода. |
| [Кооперативный паттерн](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD&action=edit&redlink=1) | Cooperative pattern | Обеспечивает механизм безопасной остановки потоков исполнения, используя общий флаг для сигнализирования прекращения работы потоков. |

**Достоинства**

В сравнении с полностью самостоятельным проектированием, шаблоны обладают рядом преимуществ. Основная польза от использования шаблонов состоит в снижении сложности разработки за счёт готовых абстракций для решения целого класса проблем. Шаблон даёт решению свое имя, что облегчает коммуникацию между [разработчиками](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82), позволяя ссылаться на известные шаблоны. Таким образом, за счёт шаблонов производится унификация деталей решений: модулей, элементов проекта, — снижается количество ошибок. Применение шаблонов концептуально сродни использованию готовых библиотек кода. Правильно сформулированный шаблон проектирования позволяет, отыскав удачное решение, пользоваться им снова и снова. Набор шаблонов помогает разработчику выбрать возможный, наиболее подходящий вариант проектирования.

**Недостатки**

Хотя легкое изменение кода под известный шаблон может упростить понимание кода, по мнению [Стива Макконнелла](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D0%BA%D0%BE%D0%BD%D0%BD%D0%B5%D0%BB%D0%BB,_%D0%A1%D1%82%D0%B8%D0%B2){Сти́вен Макко́ннелл — американский программист, автор книг по разработке программного обеспечения. Журнал «Software Development» дважды удостоил его книги премии Jolt Excellence как лучшие книги года о разработке программного обеспечения.}, с применением шаблонов могут быть связаны две сложности. Во-первых, слепое следование некоторому выбранному шаблону может привести к усложнению программы. Во-вторых, у разработчика может возникнуть желание попробовать некоторый шаблон в деле без особых оснований.

Многие шаблоны проектирования в [объектно-ориентированном проектировании](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) можно рассматривать как [идиоматическое](https://ru.wikipedia.org/wiki/%D0%98%D0%B4%D0%B8%D0%BE%D0%BC%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) воспроизведение *элементов* [функциональных языков](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). [Пол Грэхэм](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D1%8D%D0%BC,_%D0%9F%D0%BE%D0%BB){Пол Грэм — американский предприниматель, программист, известный сторонник и пропагандист использования языка программирования Lisp. В числе прочего, создал диалект лиспа, названный им Arc. Автор ряда книг по программированию.} считает саму идею шаблонов проектирования — [анти-паттерном](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D1%82%D0%B8-%D0%BF%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD), сигналом о том, что система не обладает достаточным уровнем [абстракции](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), и необходима её тщательная переработка. Нетрудно видеть, что само определение шаблона как «*готового решения, но не прямого обращения к*[*библиотеке*](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))» по сути означает отказ от [повторного использования](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0) в пользу [дублирования](https://ru.wikipedia.org/wiki/%D0%94%D1%83%D0%B1%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0). Это, очевидно, может быть *неизбежным* для сложных систем при использовании языков, не поддерживающих [комбинаторы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80) и [полиморфизм типов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2), и это в принципе может быть *исключено* в языках, обладающих этими свойствами (хотя и не обязательно эффективно), так как любой шаблон может быть реализован в виде исполнимого кода.

**Другие типы шаблонов**

**Анти-паттерны**

 это классы наиболее часто внедряемых плохих решений проблем. Они изучаются, как категория, в случае когда их хотят избежать в будущем, и некоторые отдельные случаи их могут быть распознаны при изучении неработающих систем.

Термин происходит паттернов.

* [Дым и зеркала](https://ru.wikipedia.org/w/index.php?title=%D0%94%D1%8B%D0%BC_%D0%B8_%D0%B7%D0%B5%D1%80%D0%BA%D0%B0%D0%BB%D0%B0_(%D0%B0%D0%BD%D1%82%D0%B8-%D0%BF%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD)&action=edit&redlink=1) (Smoke and mirrors): Демонстрация того, как будут выглядеть ненаписанные функции (название происходит от двух излюбленных способов, которыми[фокусники](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D0%BA%D1%83%D1%81%D0%BD%D0%B8%D0%BA) скрывают свои секреты)
* [Раздувание ПО](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%B4%D1%83%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%9F%D0%9E) (Software bloat): Разрешение последующим версиям системы требовать всё больше и больше ресурсов
* [Функции для галочки](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D0%B4%D0%BB%D1%8F_%D0%B3%D0%B0%D0%BB%D0%BE%D1%87%D0%BA%D0%B8): Превращение программы в конгломерат плохо реализованных и не связанных между собой функций (как правило, для того, чтобы заявить в[рекламе](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D0%BB%D0%B0%D0%BC%D0%B0), что функция есть)
* [Полтергейст](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%BE%D0%BB%D1%82%D0%B5%D1%80%D0%B3%D0%B5%D0%B9%D1%81%D1%82_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)&action=edit&redlink=1) (Poltergeist): Объекты, чьё единственное предназначение — передавать информацию другим объектам
* [Слепая вера](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%BB%D0%B5%D0%BF%D0%B0%D1%8F_%D0%B2%D0%B5%D1%80%D0%B0&action=edit&redlink=1) (Blind faith): Недостаточная проверка корректности исправления ошибки или результата работы подпрограммы
* [Воняющий подгузник](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%BE%D0%BD%D1%8F%D1%8E%D1%89%D0%B8%D0%B9_%D0%BF%D0%BE%D0%B4%D0%B3%D1%83%D0%B7%D0%BD%D0%B8%D0%BA&action=edit&redlink=1) (The Diaper Pattern Stinks): Сброс флага ошибки без её обработки или передачи вышестоящему обработчику
* [Мыльный пузырь](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D1%8B%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%83%D0%B7%D1%8B%D1%80%D1%8C_(%D0%BF%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD)&action=edit&redlink=1) (Soap bubble): Объект, инициализированый мусором, максимально долго притворяется, что содержит какие-то данные
* [Бензиновая фабрика](https://ru.wikipedia.org/w/index.php?title=%D0%91%D0%B5%D0%BD%D0%B7%D0%B8%D0%BD%D0%BE%D0%B2%D0%B0%D1%8F_%D1%84%D0%B0%D0%B1%D1%80%D0%B8%D0%BA%D0%B0&action=edit&redlink=1) (Gas factory): Необязательная сложность дизайна
* [Золушкина туфелька](https://ru.wikipedia.org/w/index.php?title=%D0%97%D0%BE%D0%BB%D1%83%D1%88%D0%BA%D0%B8%D0%BD%D0%B0_%D1%82%D1%83%D1%84%D0%B5%D0%BB%D1%8C%D0%BA%D0%B0&action=edit&redlink=1): Попытка "натянуть" на объект уже имеющийся малоподходящий по смыслу интерфейс, вместо создания нового.
* [Коммит-убийца](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BC%D0%BC%D0%B8%D1%82-%D1%83%D0%B1%D0%B8%D0%B9%D1%86%D0%B0&action=edit&redlink=1) (Commit assasin): Внесение отдельных изменений в систему контроля версий без проверки влияния их на другие части системы. Как правило, после подобных коммитов работа коллектива парализуется на время исправления проблем в местах, которые ранее работали безошибочно.