

**CASE-средства  
проектирования  
Лекция 8  
Проектирование  
программного продукта**

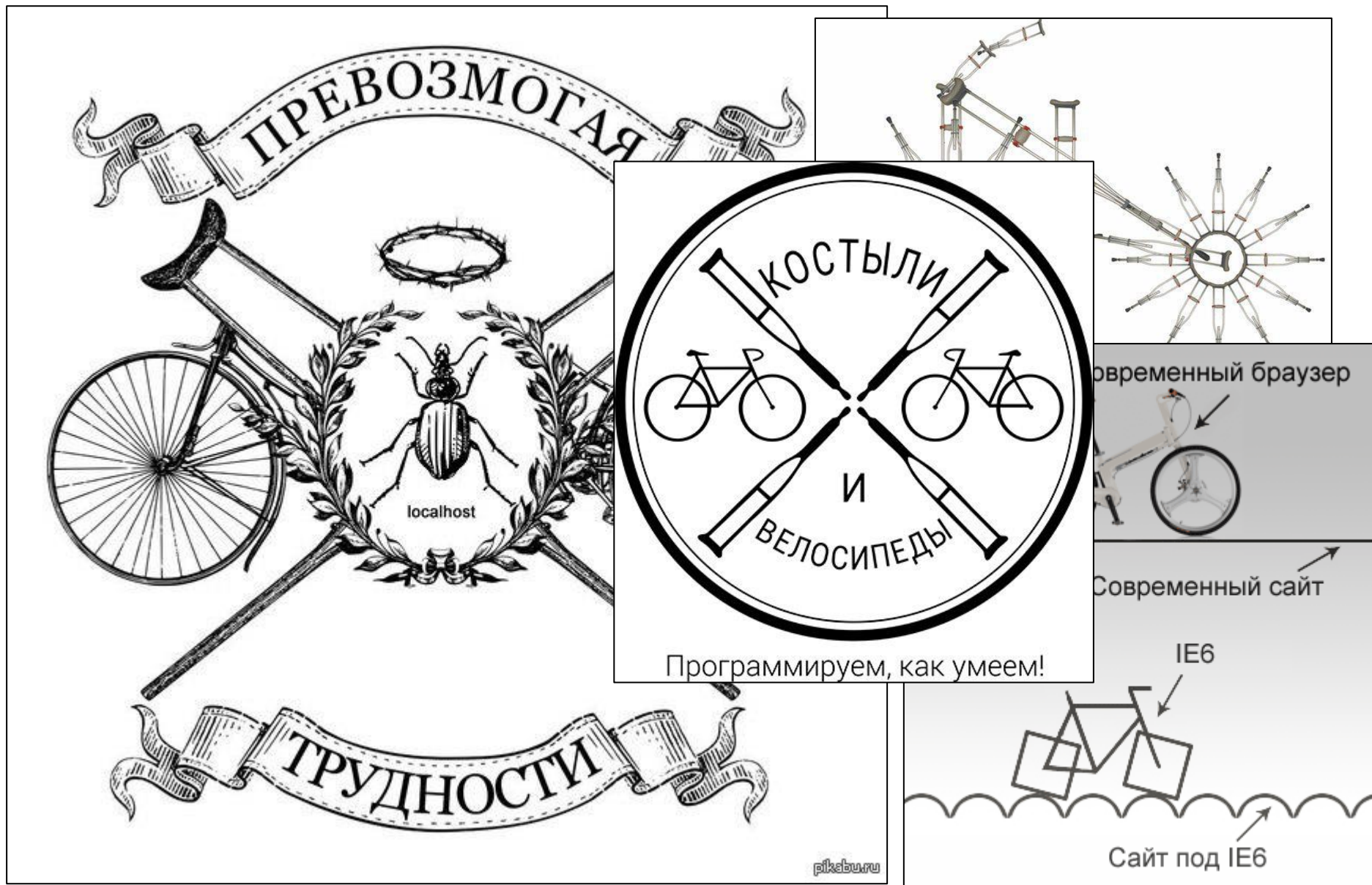
Овчинников П.Е.  
МГТУ «СТАНКИН»,  
ст.преподаватель кафедры ИС

# Шаблоны (паттерны)

## КАК ПОЯВЛЯЮТСЯ СТАНДАРТЫ:



# Анти-шаблоны (анти-паттерны)



# Серебряной пули нет

Крайне упрощая, сформулируем Закон Брукса:

***Если проект не укладывается в сроки, то добавление рабочей силы задержит его еще больше.***

Это развенчивает миф о человеко-месяце. Продолжительность осуществления проекта зависит от ограничений, накладываемых последовательностью работ. Максимальное количество разработчиков зависит от числа независимых подзадач. Эти две величины позволяют получить график работ, в котором будет меньше занятых разработчиков и больше месяцев. (Единственная опасность заключается в возможном устаревании продукта.) Нельзя, однако, составить работающие графики, в которых занято больше людей и требуется меньше времени. Программные проекты чаще проваливаются из-за нехватки календарного времени, чем по всем остальным причинам вместе взятым.

## *Глава 16 Серебряной пули нет – сущность и акциденция в программной инженерии*

*Нет ни одного открытия ни в технологии, ни в методах управления, одно только использование которого обещало бы в течение ближайшего десятилетия на порядок повысить производительность, надежность, простоту разработки программного обеспечения.*

# Терминология: программный продукт

## **ГОСТ 28806-90 Качество программных средств. Термины и определения**

### **Программное средство; ПС (software)**

Объект, состоящий из программ, процедур, правил, а также, если предусмотрено, сопутствующих им документации и данных, относящихся к функционированию системы обработки информации

## **ГОСТ 34.003-90 Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения**

### **Программное изделие в автоматизированной системе (program product in AS)**

Программное средство, изготовленное, прошедшее испытания установленного вида и поставляемое как продукция производственно-технического назначения для применения в АС

## **ГОСТ 19.004-80 Единая система программной документации. Термины и определения**

### **Программное изделие (Program product)**

Программа на носителе данных, являющаяся продуктом промышленного производства

# Терминология: программы и данные

## **ГОСТ 34.321-96 Информационные технологии (ИТ). Система стандартов по базам данных. Эталонная модель управления данными**

### **данные (data)**

Информация, представленная в формализованном виде, пригодном для передачи, интерпретации или обработки с участием человека или автоматическими средствами

## **ГОСТ 19781-90 Обеспечение систем обработки информации программное. Термины и определения**

### **Программа (Program)**

Данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма

### **Программное обеспечение**

Совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ

### **Программирование (Programming)**

Научная и практическая деятельность по созданию программ



# Объект проектирования: программный продукт



**Internal Logical Files (ILF)** – понятная для пользователя группа логически-связанных данных, которые находятся полностью внутри приложения и обслуживаются через внешние входы системы.

**External Interface Files (EIF)** – понятная для пользователя группа логически-связанных данных, которые используются только для целей ссылки. Сюда относятся так же данные, которые использует наше приложение, но которые управляются другим приложением.

**External Inputs (EI)** - Элементарный процесс, в котором данные вводятся в систему с наружи. Эти данные могут поступать от экранов ввода данных, электронных устройств или от другого приложения.

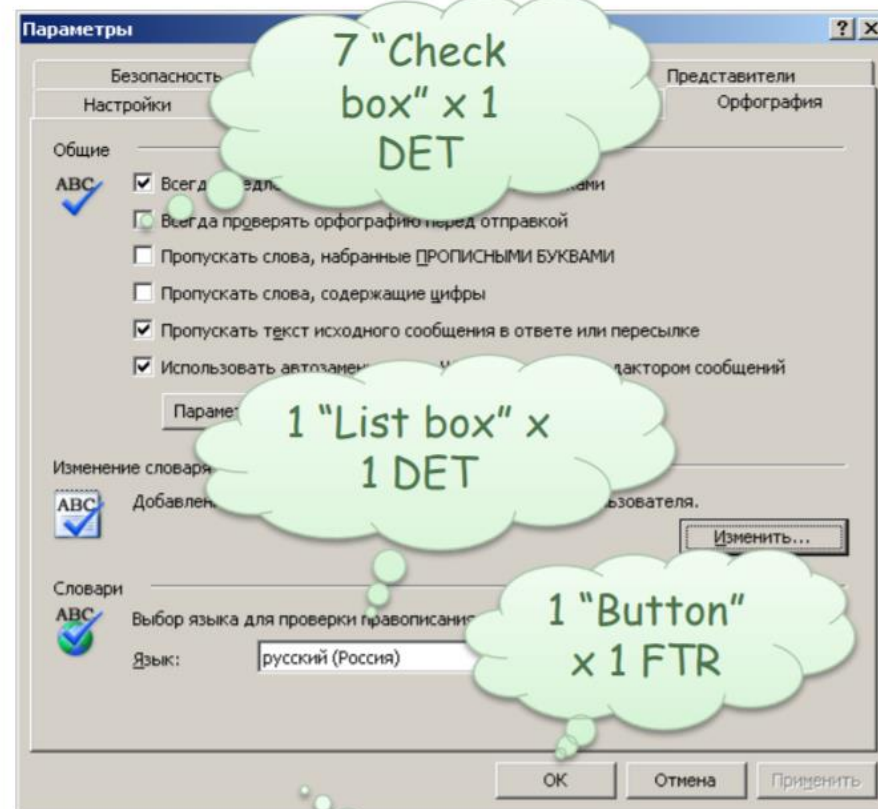
**External Outputs (EO)** - Элементарный процесс, в котором данные выводятся из системы. Данные создают отчеты или внешние файлы, которые пересылаются другим приложениям. Они могут создаваться одним или несколькими внутренними логическими файлами (ILF) и внешним файлом интерфейса (EIF).

**External Inquiry (EQ)** - Элементарный процесс, в котором данные выводятся из системы в результате выполнения комплексного запроса и процесса обработки внутренних логических файлов (ILF) и внешних интерфейсных файлов (EIF).

# Объект проектирования: интерфейс

Сложность транзакций EI & EQ	Количество UFP
Low	3
Average	4
High	6

Сложность транзакций EO	Количество UFP
Low	4
Average	5
High	7



(1 FTR, 8 DET) -> Low -> 3 UFP



# Объект проектирования: интерфейс

**Интерфейс** (англ. interface — сопряжение, поверхность раздела, перегородка) — совокупность возможностей, способов и методов взаимодействия двух систем, устройств или программ для обмена информацией между ними, определенная их характеристиками, характеристиками соединения сигналов обмена и т.п.

## **ГОСТ Р ИСО 9241-210-2016 Эргономика взаимодействия человек-система. Часть 210. Человеко-ориентированное проектирование интерактивных систем**

**человеко-ориентированное проектирование** (human-centred design): Способ проектирования и разработки систем с применением при проектировании принципов эргономики для повышения пригодности использования интерактивных систем

**интерактивная система** (interactive system): Система компонентов аппаратного и программного обеспечения, которая получает информацию, вводимую пользователем, и передает ему свой ответ, помогая в работе или выполнении задачи.

**пользовательский интерфейс** (интерфейс пользователя) (user interface): Все компоненты интерактивной системы (программное обеспечение или аппаратное обеспечение), которые предоставляют пользователю информацию и являются инструментами управления для выполнения определенных задач

**пригодность использования** (usability): Свойство системы, продукции или услуги, при наличии которого установленный пользователь может применить продукцию в определенных условиях использования для достижения установленных целей с необходимой результативностью, эффективностью и удовлетворенностью.

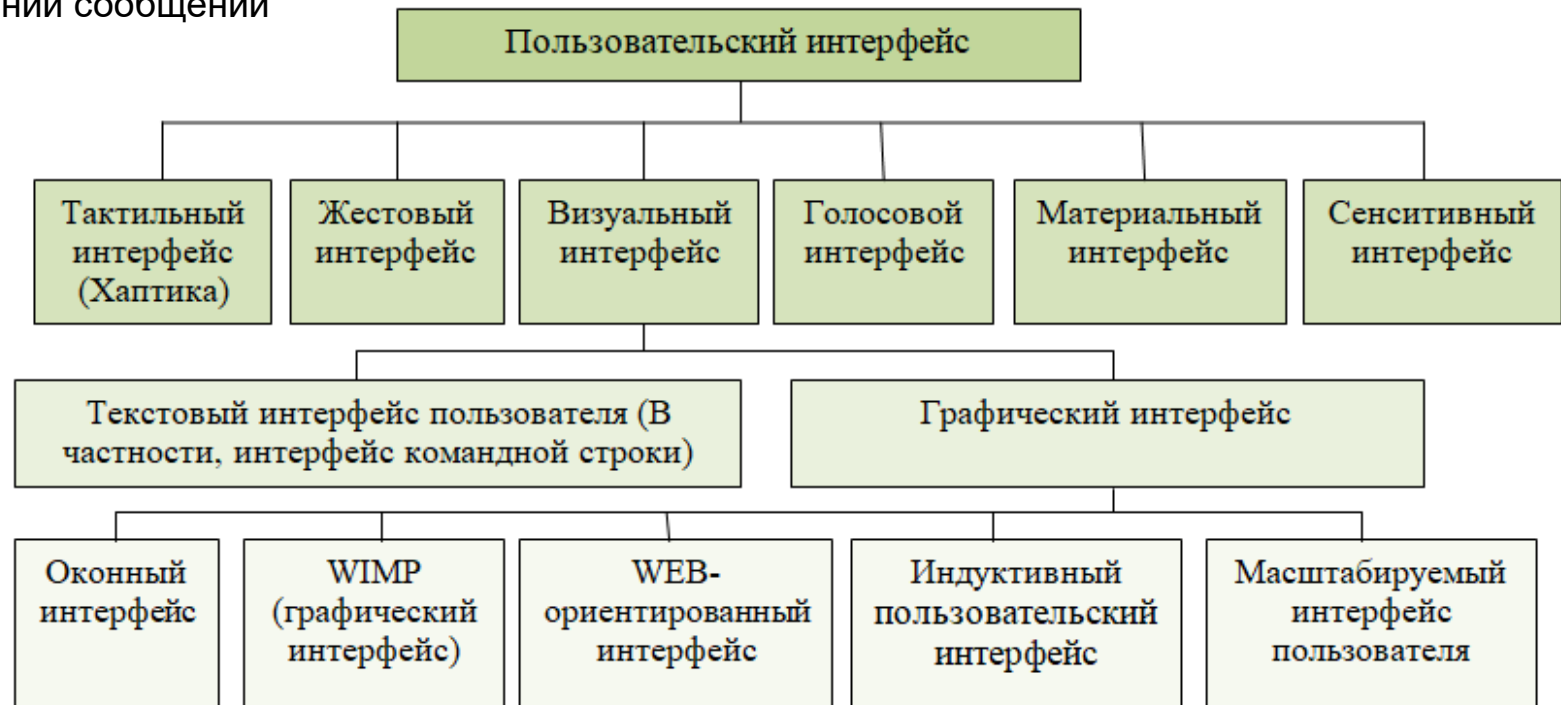
# Объект проектирования: интерфейс

## ГОСТ Р 43.0.2-2006 Информационное обеспечение техники и операторской деятельности. Термины и определения

**оператор:** Человек, занимающийся какой-либо деятельностью с использованием технических(ого) устройств(а)

## ГОСТ Р 43.0.3-2009 Информационное обеспечение техники и операторской деятельности. Ноон-технология в технической деятельности. Общие положения

**ноон-технология:** Технология создания информации в виде, соответствующем психофизиологии человека (с использованием результатов исследований, полученных в ноонике), для реализации оптимизированных информационно-обменных процессов в СЧИ при создании, хранении, передаче, применении сообщений



# Объект проектирования: интерфейс

## Тактильный интерфейс

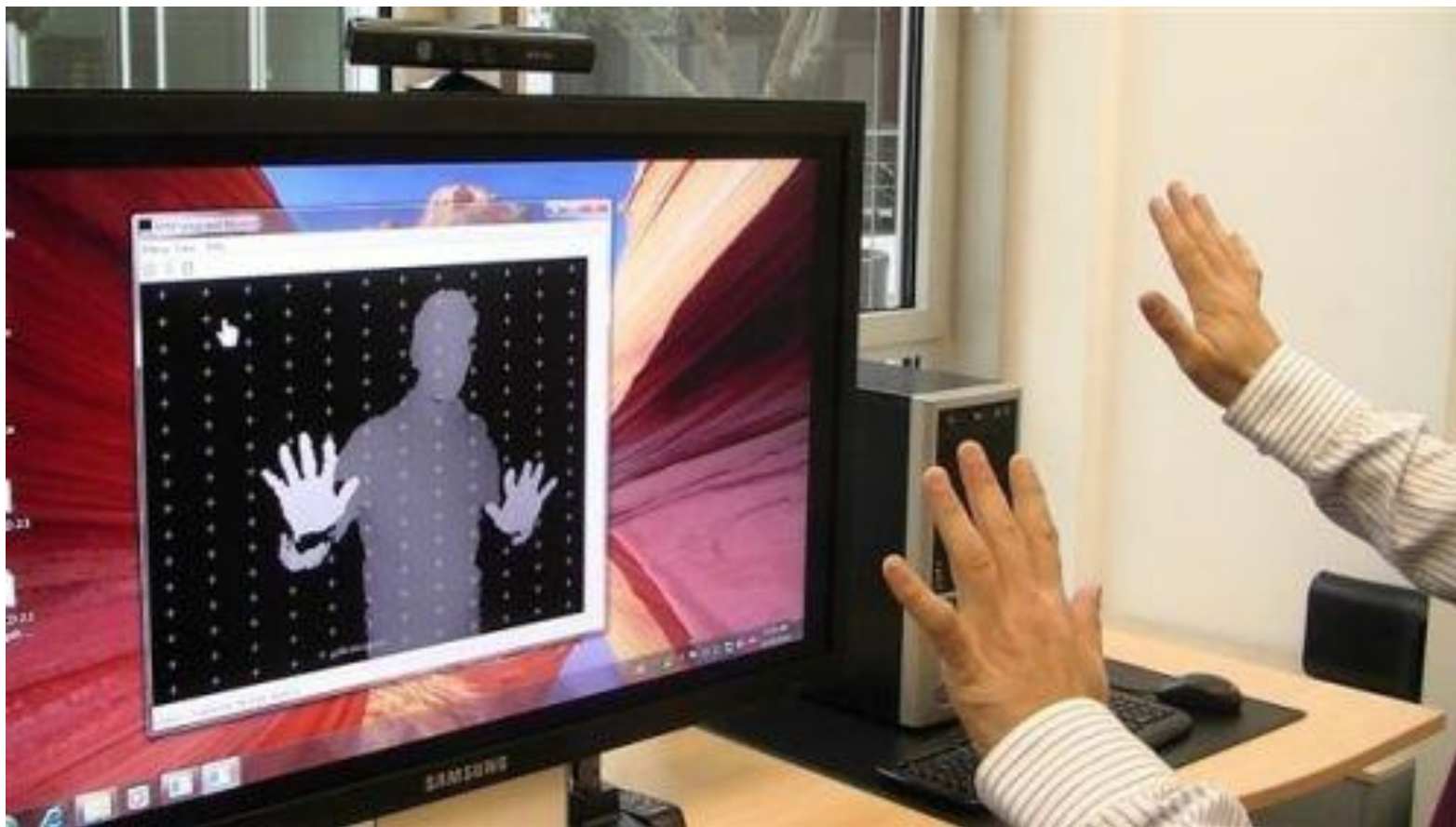
Средства отображения и позиционирования функционально объединены и пространственно совмещены. Тактильный интерфейс описывает управление посредством касания сенсорного экрана с изображением стилусом, пальцами, другими частями тела.



# Объект проектирования: интерфейс

## Жестовый интерфейс

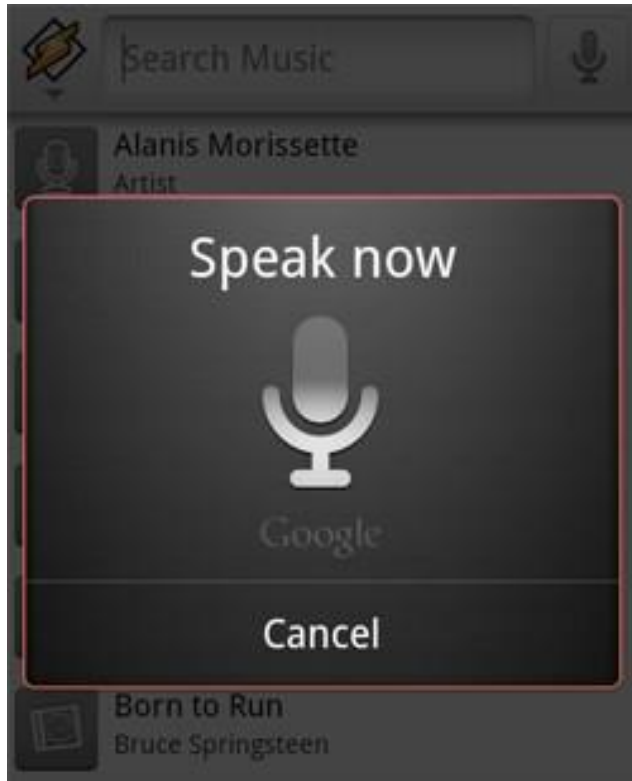
Позволяет эмулировать клавиатурные команды (либо сочетания клавиш, касания) при помощи жестов.



# Объект проектирования: интерфейс

## Голосовой интерфейс

Вместо меню используются слышимые инструкции, которые были ранее сохранены или созданы синтезатором голоса в реальном времени.





# Объект проектирования: интерфейс

## Материальный интерфейс

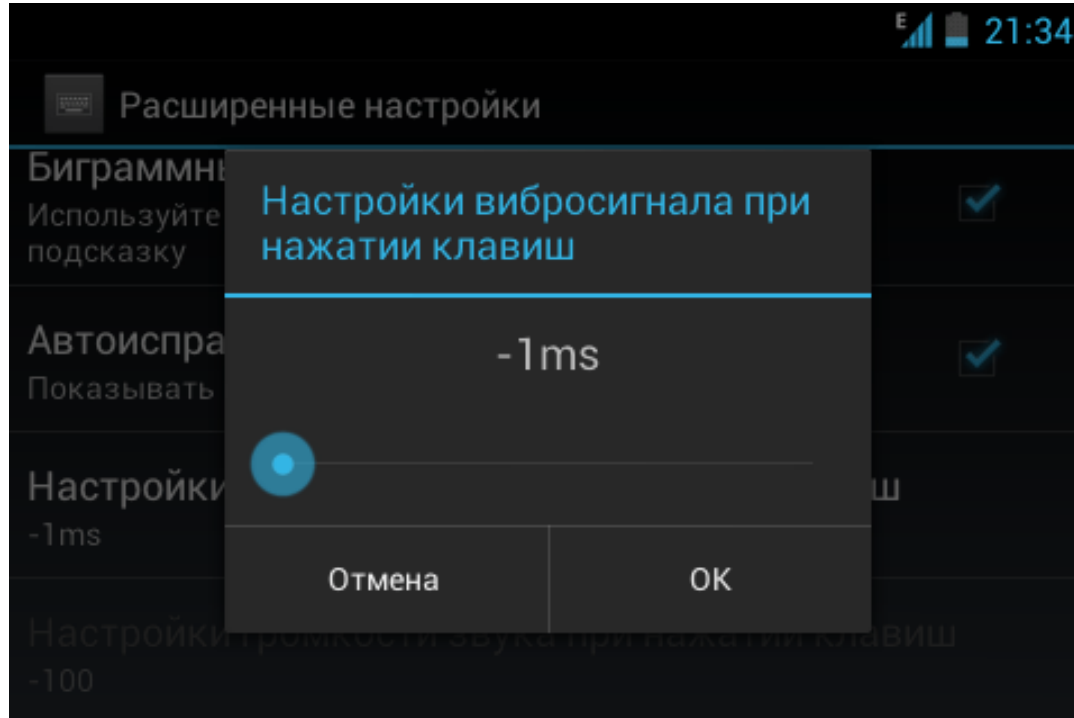
Взаимодействие человека с электронными устройствами происходит при помощи материальных предметов и конструкций.



# Объект проектирования: интерфейс

## Сенситивный интерфейс

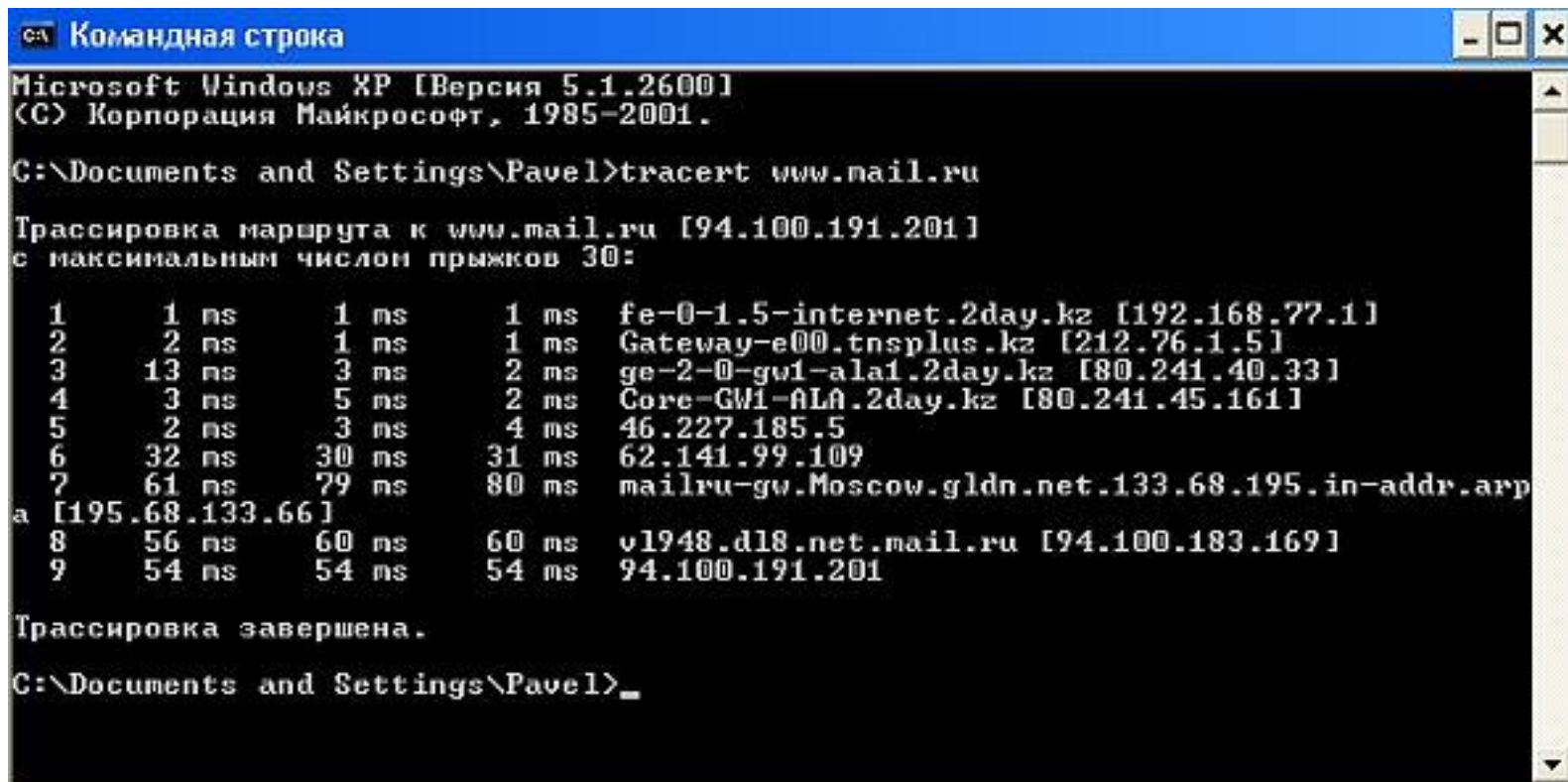
В ответ на взаимодействие человека с машиной происходит воздействие на органы чувств.



# Объект проектирования: интерфейс

## Текстовый интерфейс пользователя

Инструкции компьютеру даются путём ввода с клавиатуры текстовых строк (команд). Система как ответ на действия оператора тоже выдаёт или сообщения, или результат выполнения введенной команды, опять же в текстовом виде. Данный интерфейс также используется для взаимодействия с другими прикладными программами, включая удаленно расположенные.



```
Командная строка
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

C:\Documents and Settings\Pavel>tracert www.mail.ru

Трассировка маршрута к www.mail.ru [94.100.191.201]
с максимальным числом прыжков 30:

  1      1 ns      1 ms      1 ms  fe-0-1.5-internet.2day.kz [192.168.77.1]
  2      2 ns      1 ms      1 ms  Gateway-e00.tnsplus.kz [212.76.1.5]
  3     13 ns      3 ms      2 ms  ge-2-0-gw1-ala1.2day.kz [80.241.40.33]
  4      3 ns      5 ms      2 ms  Core-GW1-ALA.2day.kz [80.241.45.161]
  5      2 ns      3 ms      4 ms  46.227.185.5
  6     32 ns     30 ms     31 ms  62.141.99.109
  7     61 ns     79 ms     80 ms  mailru-gw.Moscow.gldn.net.133.68.195.in-addr.arp
a [195.68.133.66]
  8     56 ns     60 ms     60 ms  v1948.d18.net.mail.ru [94.100.183.169]
  9     54 ns     54 ms     54 ms  94.100.191.201

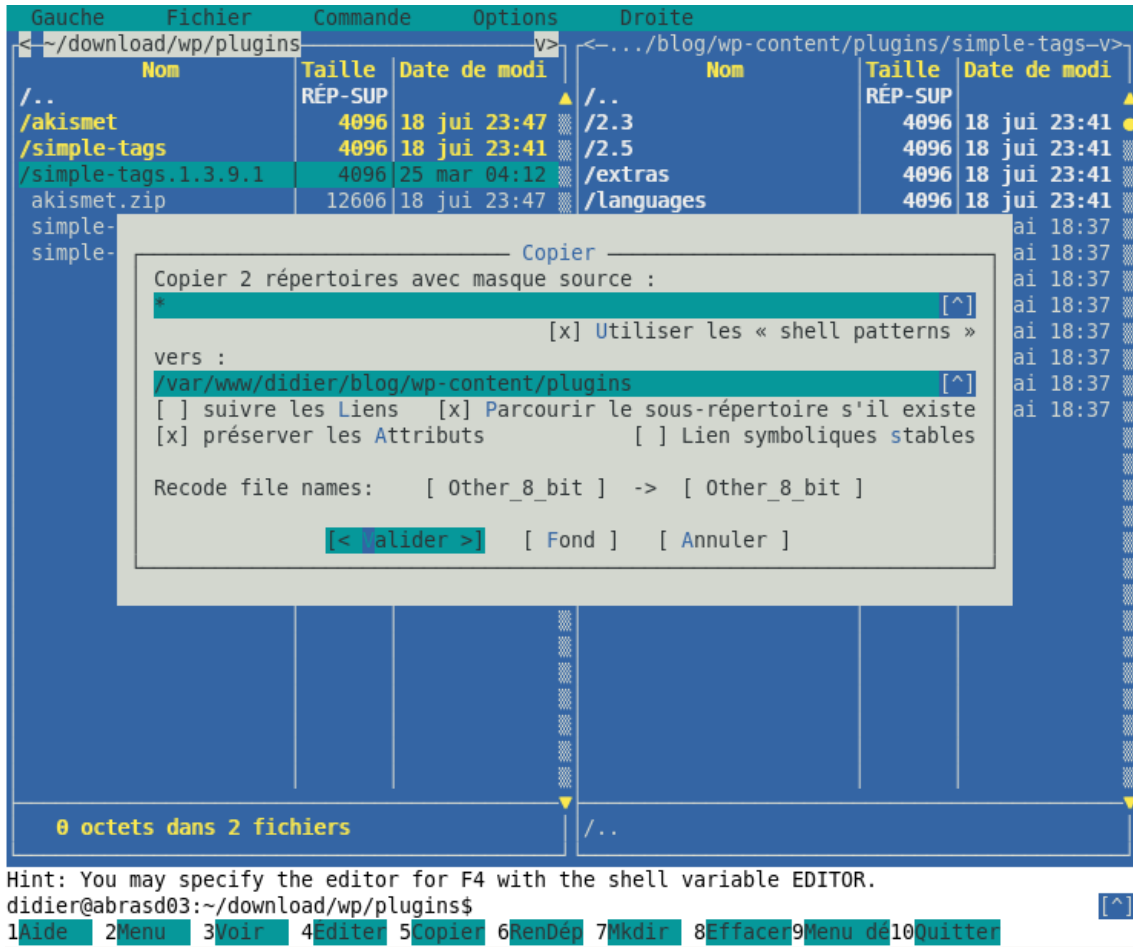
Трассировка завершена.

C:\Documents and Settings\Pavel>
```

# Объект проектирования: интерфейс

## Оконный интерфейс

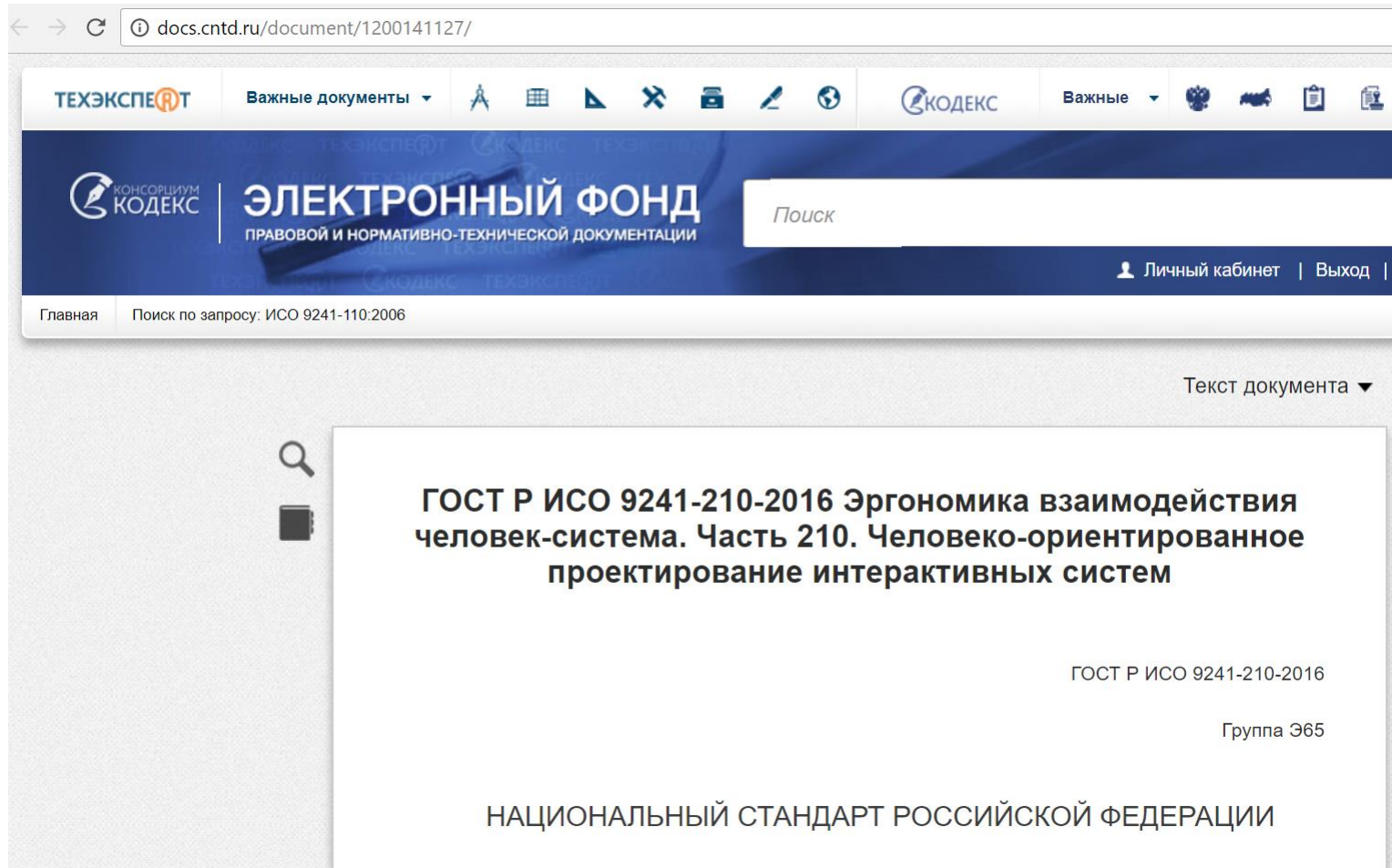
Каждая интегральная часть располагается в окне — собственном субэкранном пространстве, находящемся в произвольном месте «над» основным экраном.



# Объект проектирования: интерфейс

## WEB-ориентированный интерфейс

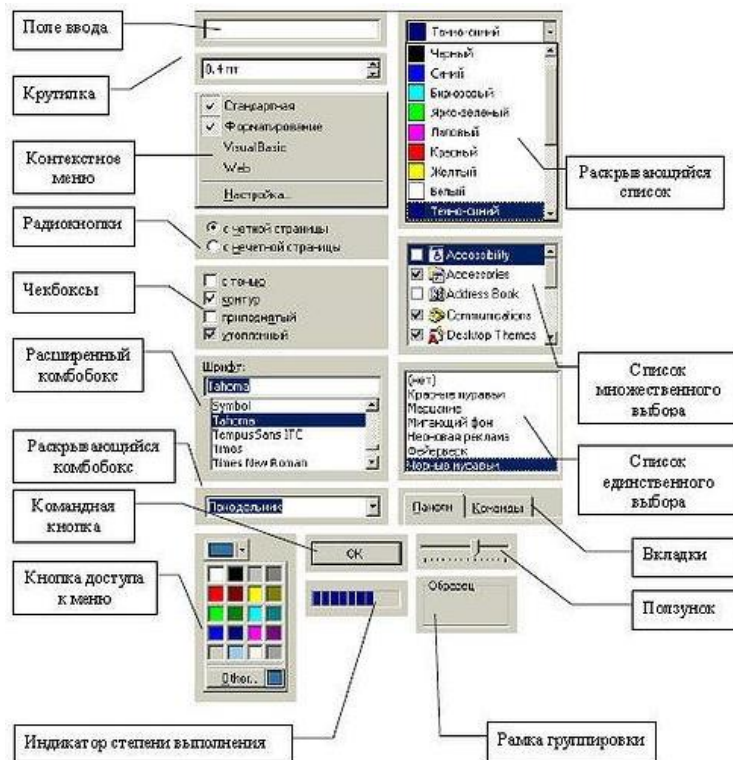
Совокупность средств, при помощи которых пользователь взаимодействует с [веб-сайтом](#) или любым другим [приложением](#) через [браузер](#).





# Объект проектирования: интерфейс

**Окулография** (отслеживание [глаз](#), [трекинг](#) глаз; **айтрекинг**) — определение [координат взгляда](#) («точки пересечения [оптической оси](#) [глазного яблока](#) и [плоскости](#) наблюдаемого объекта или экрана, на котором предъявляется некоторый зрительный раздражитель»)



# Объект проектирования: интерфейс

**Дизайн взаимодействия с пользователем** (*UX-дизайн* ← [англ. User Experience](#), или *UXD*, или *UED*, или *XD* ← [англ. User Experience Design](#))<sup>[1]</sup> включает в себя традиционное [взаимодействие человека с компьютером](#) (HCI), в том числе все аспекты продукта, как они воспринимаются пользователями

UX-дизайн — многомерное определение, которое охватывает широкую область знаний и характеризует ощущения пользователя, взаимодействующего с онлайн-ресурсом и включает множество составляющих: интерактивный дизайн, информационную архитектуру, визуальный дизайн, юзабилити и взаимодействие между человеком и компьютером.

Цель UX-дизайна — улучшить степень удовлетворенности пользователя и его увеличить лояльность благодаря пользе, простоте использования и удовольствию, которое он получает в процессе взаимодействия с интернет-ресурсом. Процесс создания дизайна взаимодействия с пользователем включает конкурентный анализ, разработку шаблонов, которые будут ценными для определенной целевой аудитории. UX-дизайн отвечает за функциональное восприятие и взаимодействие пользователя с ресурсом.

**[ГОСТ Р 56645.1-2015](#) Системы дизайн-менеджмента. Руководство по управлению дизайном промышленной продукции (Переиздание)**

Передовой опыт в области дизайна является важным дифференцирующим фактором между конкурирующей продукцией и может быть ключевым фактором выживания компании в условиях растущей конкуренции на мировых рынках.

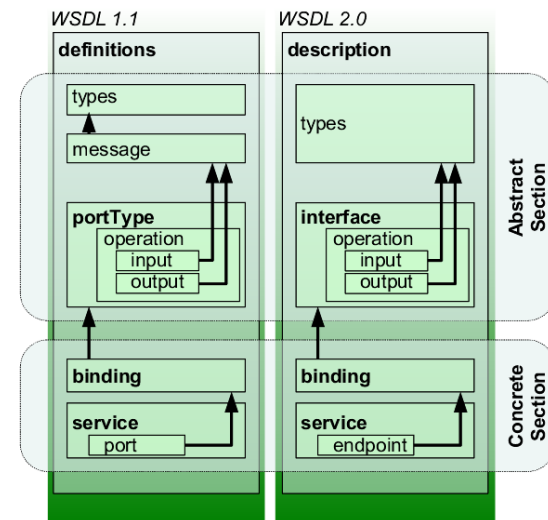
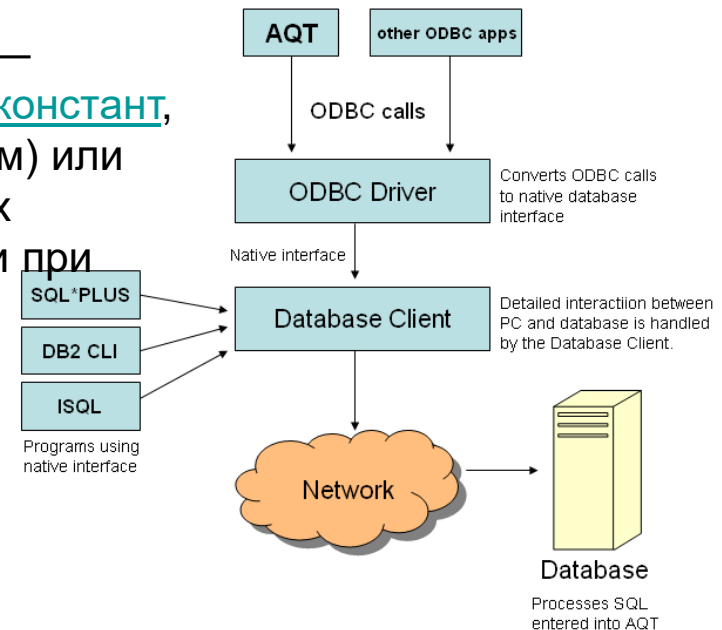
# Объект проектирования: интерфейс

**API** (программный интерфейс приложения, интерфейс прикладного программирования)

([англ.](#) *application programming interface*, *API* [эй-пи-ай]) — набор готовых [классов](#), [процедур](#), [функций](#), [структур](#) и [констант](#), предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах. Используется программистами при написании всевозможных приложений.

**ODBC** ([англ.](#) *Open Database Connectivity*) — это программный интерфейс ([API](#)) доступа к [базам данных](#), разработанный компанией [Microsoft](#) в сотрудничестве с [Simba Technologies](#) на основе спецификаций [Call Level Interface](#) (CLI), который разрабатывался организациями [SQL Access Group](#), [X/Open](#) и [Microsoft](#). Стандарт CLI призван унифицировать программное взаимодействие с [СУБД](#), сделать его независимым от поставщика СУБД и программно-аппаратной платформы

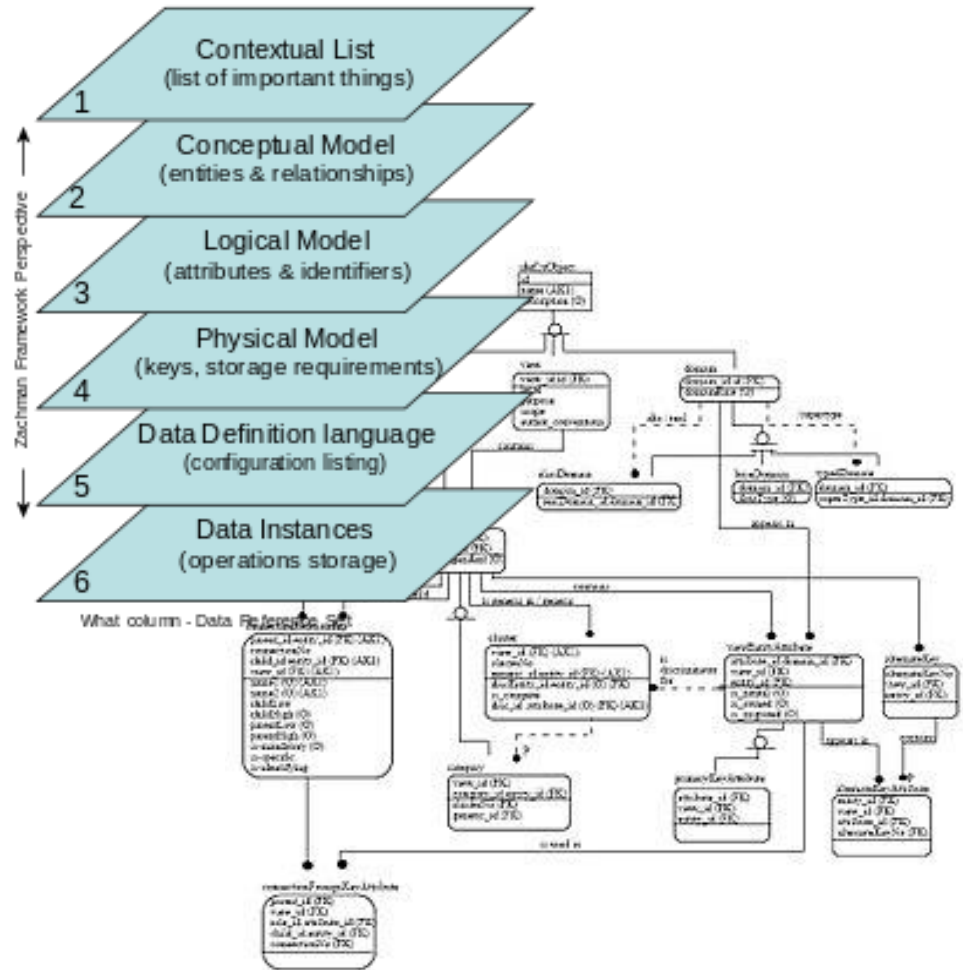
**WSDL** ([англ.](#) *Web Services Description Language*) — язык описания [веб-сервисов](#) и доступа к ним, основанный на языке [XML](#).



# Объект проектирования: данные

**IDEF1** (*integration definition for information modeling*) — одна из методологий семейства [IDEF](#). Применяется для построения информационной модели, которая представляет структуру информации, необходимой для поддержки функций производственной системы или среды.

**Модель Захмана** ([англ. Zachman Framework](#), некорректная передача фамилии — «Захман») — [онтология](#) предприятия, представляющая собой подход к описанию [архитектуры предприятия](#). Онтология является двумерной классификационной схемой, клетки которой — пересечения элементов двух исторических классификаций. Первая классификация включает набор простейших вопросов: «что», «как», «когда», «кто», «где» и «почему». Вторая классификация проистекает из философской концепции овеществления, то есть претворения абстрактных идей в жизнь. В модели Захмана использованы следующие овеществляющие трансформации: «идентификация», «определение», «представление», «спецификация», «конфигурация» и «конкретизация»



# Объект проектирования: данные

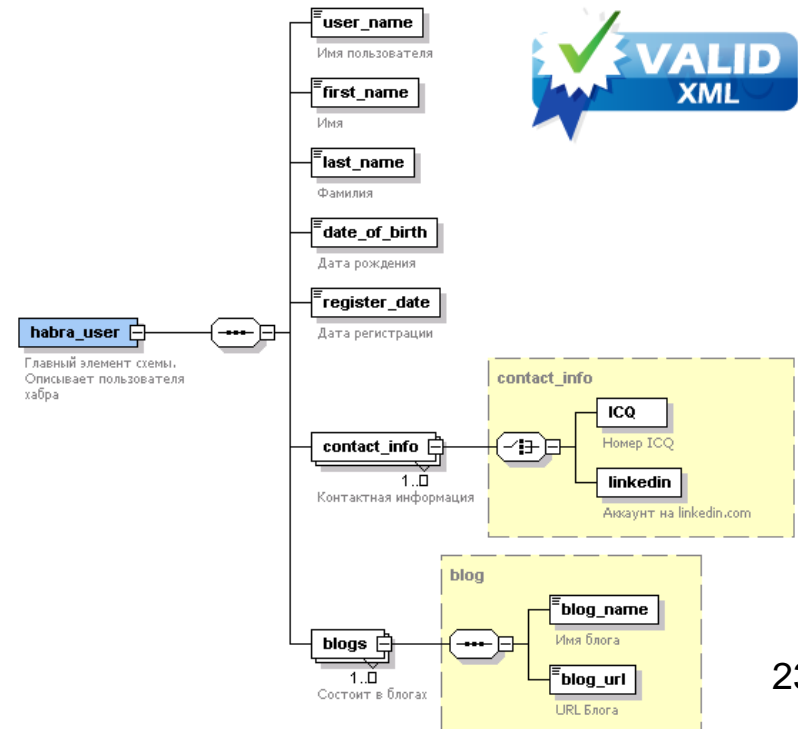
**XML Schema** — язык описания структуры [XML](#)-документа. Спецификация XML Schema является рекомендацией [W3C](#).

Как большинство языков описания XML, XML Schema была задумана для определения правил, которым должен подчиняться документ. Но, в отличие от других языков, XML Schema была разработана так, чтобы её можно было использовать в создании программного обеспечения для обработки документов XML.

После проверки документа на соответствие XML Schema читающая программа может создать модель данных документа, которая включает:

- словарь (названия элементов и атрибутов);
- модель содержания (отношения между элементами и атрибутами и их структура);
- типы данных.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="country">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="country_name" type="xs:string"/>
        <xs:element name="population" type="xs:decimal"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```





# Объект проектирования: метамодель

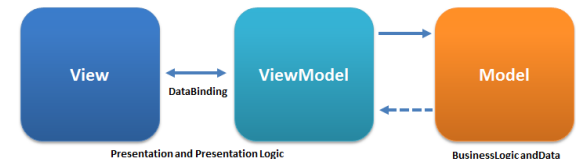
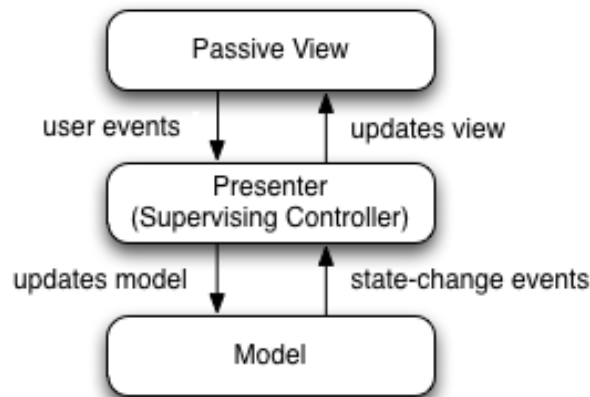
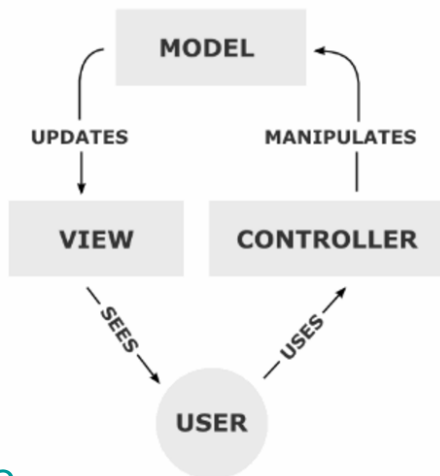
**Фрэймворк** (иногда *фреймвóрк*; [англицизм](#), [неологизм](#) от *framework* — каркас, структура) — [программная платформа](#), определяющая структуру программной системы; [программное обеспечение](#), облегчающее разработку и объединение разных компонентов большого программного проекта.

**Model-View-Controller (MVC)**, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, [пользовательского интерфейса](#) и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо

**Модель** (*Model*) предоставляет данные и реагирует на команды контроллера, изменяя свое состояние<sup>[1]</sup>.

**Представление** (*View*) отвечает за отображение данных модели пользователю, реагируя на изменения модели

**Контроллер** (*Controller*) интерпретирует действия пользователя, оповещая модель о необходимости изменений



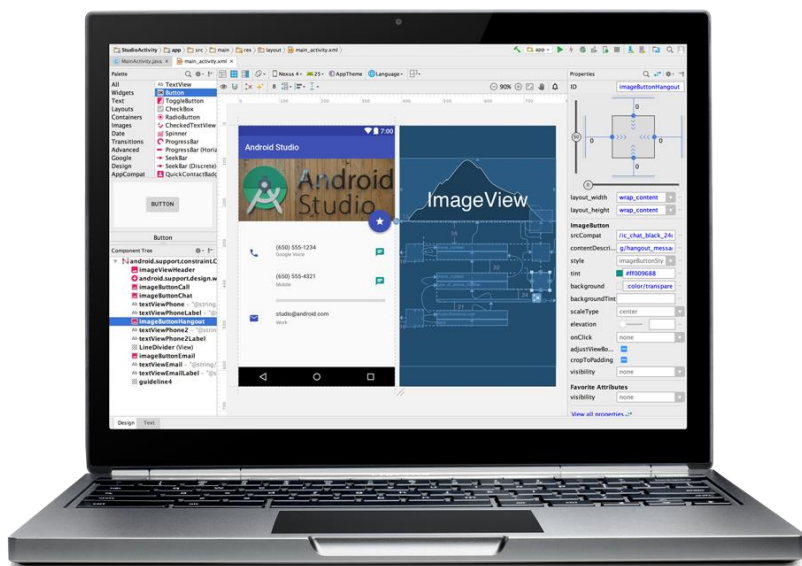
# Объект проектирования: метамодель



# Инструменты: симуляторы и эмуляторы

**Симулятор** — имитатор (обычно механический или компьютерный), задача которого состоит в имитации управления каким-либо процессом, аппаратом или транспортным средством.

**Эмуляция** ([англ. emulation](#)) в [вычислительной технике](#) — комплекс программных, аппаратных средств или их сочетание, предназначенное для копирования (или *эмулирования*) функций одной вычислительной системы (*гостя*) на другой, отличной от первой, вычислительной системе (*хосте*) таким образом, чтобы эмулированное поведение как можно ближе соответствовало поведению оригинальной системы (*гостя*).



# Инструменты: DevOps и CI

**DevOps** — это командная работа (между сотрудниками, занимающимися разработкой, операциями и тестированием), нет единого инструмента «DevOps»: это скорее набор (или «инструментальная цепочка DevOps»), состоящий из нескольких инструментов. Как правило, инструменты DevOps вписываются в одну или несколько из этих категорий, что отражает ключевые аспекты разработки и доставки программного обеспечения:

**Code** — разработка и анализ кода, инструменты контроля версий, слияние кода

**Build** — инструменты непрерывной интеграции, статус сборки

**Test** — инструменты непрерывного тестирования, которые обеспечивают обратную связь по бизнес-рискам

**Package** — репозиторий артефактов, предварительная установка приложения

**Release** — управление изменениями, официальное утверждение выпуска, автоматизация выпуска

**Configure** — Конфигурация и управление инфраструктурой, Инфраструктура как инструменты кода

**Monitor** — мониторинг производительности приложений, опыт работы с конечным пользователем

**Непрерывная интеграция** (CI, [англ. Continuous Integration](#)) — это практика [разработки программного обеспечения](#), которая заключается в слиянии рабочих копий в общую основную ветвь разработки несколько раз в день и выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем.

# Инструменты: регрессионное тестирование

**Модульное тестирование**, иногда **блочное тестирование** или **юнит-тестирование** ([англ. unit testing](#)) — процесс в [программировании](#), позволяющий проверить на корректность отдельные модули [исходного кода](#) программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к [регрессии](#), то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

[ГОСТ Р 56920-2016/ISO/IEC/IEEE 29119-1:2013](#) Системная и программная инженерия. Тестирование программного обеспечения. Часть 1. Понятия и определения

**регрессионное тестирование** (regression testing):

Тестирование после изменений элемента тестирования или его рабочей среды для определения того, происходят ли регрессивные отказы.

**критерий успешного/неуспешного прохождения** (pass/fail criteria):

Правила решения, используемые для определения того, прошли ли тестирование элемент тестирования или функция элемента тестирования или перестали работать после тестирования.

[ГОСТ Р 56921-2016](#) Системная и программная инженерия. Тестирование программного обеспечения. Часть 2. Процессы тестирования



# Терминология: пользовательские истории

**Пользовательские истории** ([англ. User Story](#)) — способ описания требований к разрабатываемой системе, сформулированных как одно или более предложений на **повседневном** или **деловом** языке **пользователя**

Пользовательские истории используются [гибкими методологиями разработки](#) программного обеспечения для спецификации требований вместе с [приёмочными испытаниями](#)

Каждая пользовательская история ограничена в **размере** и **сложности**. Часто история пишется на маленькой бумажной карточке. Это гарантирует, что она не станет слишком большой

В [Экстремальном программировании](#) пользовательские истории **пишутся пользователями (заказчиками)** системы.

В методологии [SCRUM](#) — **пишутся** либо **одобряются** ролью **владельца продукта** ([англ. Product Owner](#))

Для заказчиков (пользователей) пользовательские истории являются основным инструментом влияния на разработку программного обеспечения

# Терминология: пользовательские истории

Пользовательская история – это легковесный инструмент для документации пользовательских требований к разработке продукта. История описывает функциональность системы с точки зрения пользователя с определенной ролью и целью этой системы.

**Как <роль/персона юзера> я <что-то хочу получить> <с такой-то целью>.**

# Приемочные испытания (acceptance testing)

**Аттестация или Валидация (Validation)** —

- подтверждение на основе представления объективных свидетельств того, что элемент (элемент системы, система, документ, услуга, задача, требование и т.д.) соответствует его назначению и функциям, описанным в требованиях к нему ([SEBoK](#))
- подтверждение на основе представления объективных свидетельств того, что требования, предназначенные для конкретного использования или применения, выполнены ([ISO 9000](#))

**Пользовательская история** остается **неофициальным** определением **требований**, пока отсутствует процедура **приемочного тестирования**

**Прежде** чем **реализовывать** пользовательскую историю, клиент должен определить соответствующую **приемочную процедуру**, чтобы гарантировать, что **цели** пользовательской истории были **достигнуты**

# Терминология: пользовательские истории

**Эпик**—это **большая история**

Это требование, слишком большое чтобы реализовать его за один спринт

Эпики нужно разбивать на более мелкие работы (истории)



# Прототипирование в программировании

## Классификация прототипов

по назначению:

- **горизонтальные** или **поведенческие** (horizontal, behavioral)
- **вертикальные** или **структурные** (vertical, structural)

по времени жизни:

- **одноразовые** или **исследовательские** (throwaway, exploratory)
- **эволюционные**

	Одноразовые	Эволюционные
Горизонтальные	<ul style="list-style-type: none"><li>○ Прояснение и уточнение примеров использования и функциональных требований</li><li>○ Выявление пропущенных требований</li><li>○ Исследование возможных вариантов интерфейса пользователя</li></ul>	<ul style="list-style-type: none"><li>○ Реализация базовых вариантов использования</li><li>○ Реализация дополнительных вариантов использования по приоритетам</li><li>○ Реализация и доработка web-сайтов</li><li>○ Адаптация системы к быстро меняющимся требованиям бизнеса</li></ul>
Вертикальные	<ul style="list-style-type: none"><li>○ Демонстрация технической осуществимости</li></ul>	<ul style="list-style-type: none"><li>○ Реализация и наращивание ключевой клиент-серверной функциональности и уровней коммуникации</li><li>○ Реализация и оптимизация основных алгоритмов</li><li>○ Тестирование и настройка производительности</li></ul>



# Прототипирование в программировании

**Одноразовый** или **исследовательский** прототип (*throwaway prototype*, *exploratory prototype*) создается, когда нужно быстро **промакетировать** те или иные аспекты и компоненты системы

Целям создания исследовательских прототипов служит технология RAD (rapid application development) - быстрая разработка приложений, см. ["Выявление требований"](#)

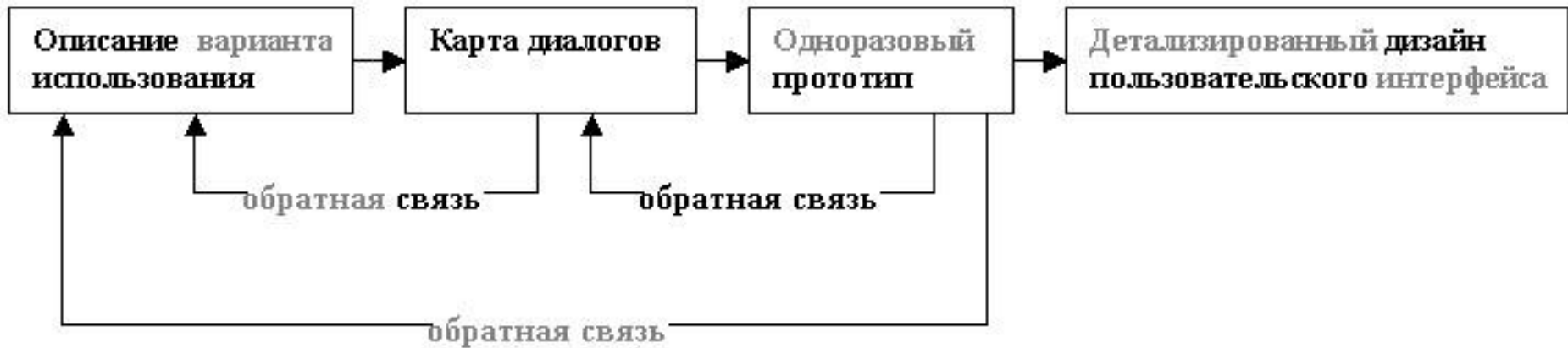
Одноразовый прототип **должен создаваться быстро**

При его разработке **не следует** уделять внимание вопросам:

- **повторного использования кода**
- **качества**
- **быстродействия**
- **технологичности** и т.п.

В результате получается "сырой" код, который может содержать **значительное количество дефектов**. Необходимо **принять меры** к тому, чтобы фрагменты кода, реализующие такого рода прототипы, не стали частью целевой системы.

# Прототипирование в программировании



На рисунке присутствует новое, не раскрытое ранее понятие: "**карта диалога**", говорят также "схема диалога". **Прежде** чем создавать одноразовый прототип, необходимо определиться:

- какие **основные экраны** будут присутствовать
- какие **окна** будут открываться
- какие **правила перехода** между ними будут поддерживаться.

Информация такого рода хорошо ложится на модель **диаграммы состояний**, см. "[Расширенный анализ требований. Моделирование](#)", где разным экранам (окнам) сопоставляются состояния, а активным элементам управления, вызывающим закрытие одних интерфейсных элементов и открытие других - переходы.

# Прототипирование в программировании

В языке UML под **состоянием** понимается абстрактный **метакласс**, используемый для моделирования отдельной ситуации, в течение которой имеет место выполнение **некоторого условия**

Состояние может быть задано в виде набора конкретных **значений атрибутов** класса или объекта, при этом изменение их отдельных значений будет отражать изменение состояния моделируемого класса или объекта

**Переход** системы из состояния в состояние осуществляется при **наступлении событий**, При этом говорится, что переход срабатывает

Переход может быть безальтернативным, либо содержать альтернативы. Во втором случае переход обусловлен наступлением **сторожевых условий**

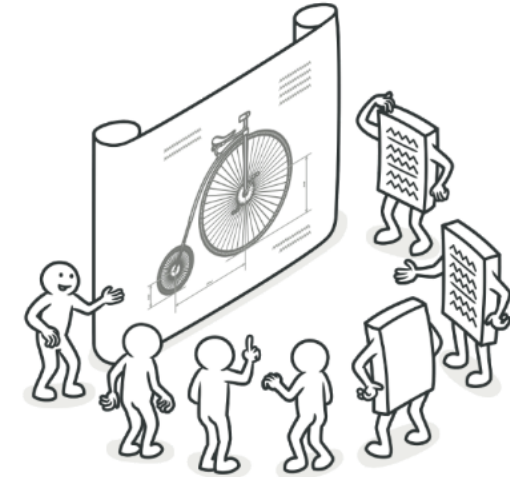


# Шаблон: рефакторинг и паттерны ООП

## ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ

Паттерны (или шаблоны) проектирования описывают типичные способы решения часто встречающихся проблем при проектировании программ.

Что такое паттерн?



### Польза паттернов

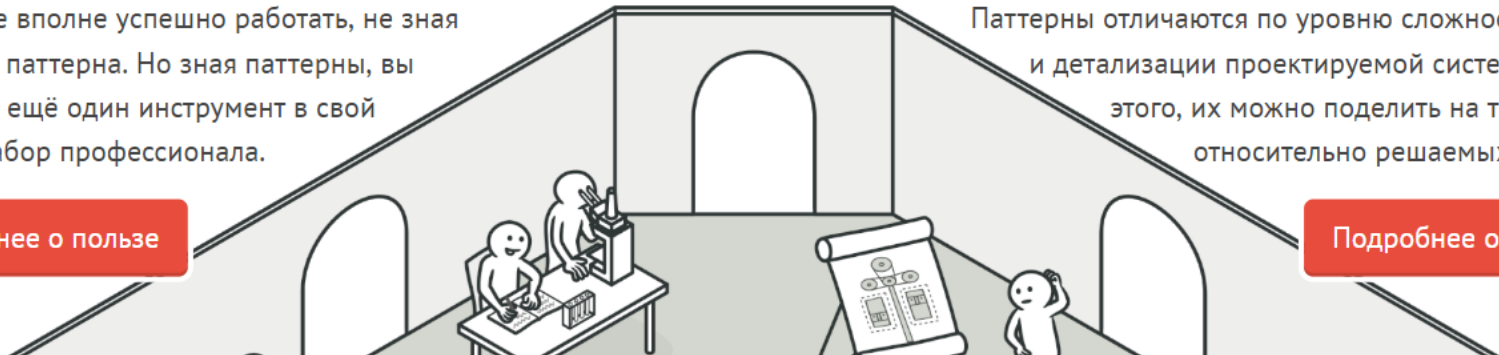
Вы можете вполне успешно работать, не зная ни одного паттерна. Но зная паттерны, вы получаете ещё один инструмент в свой личный набор профессионала.

Подробнее о пользе

### Классификация

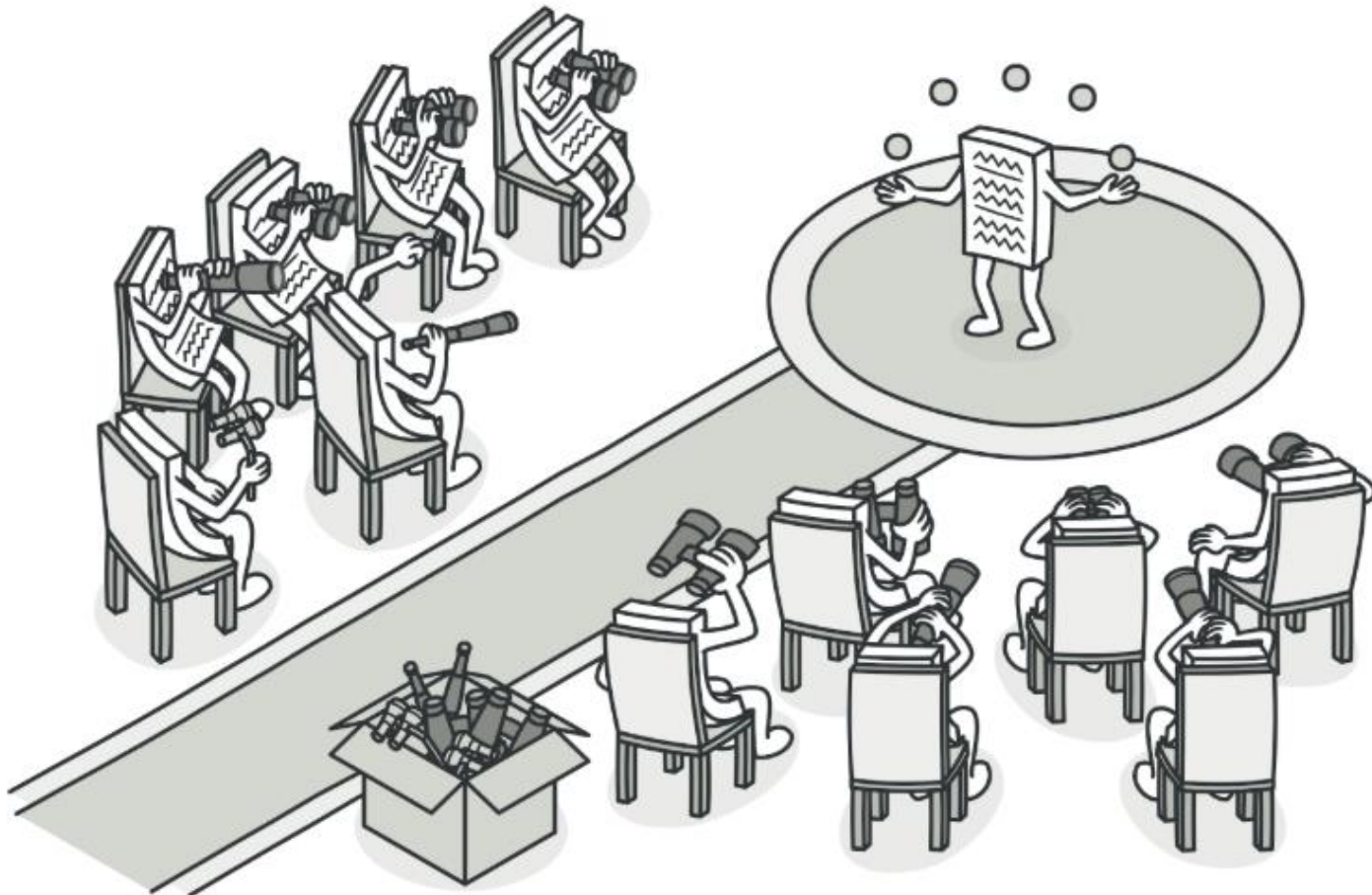
Паттерны отличаются по уровню сложности, охвата и детализации проектируемой системы. Кроме этого, их можно поделить на три группы, относительно решаемых проблем.

Подробнее о группах



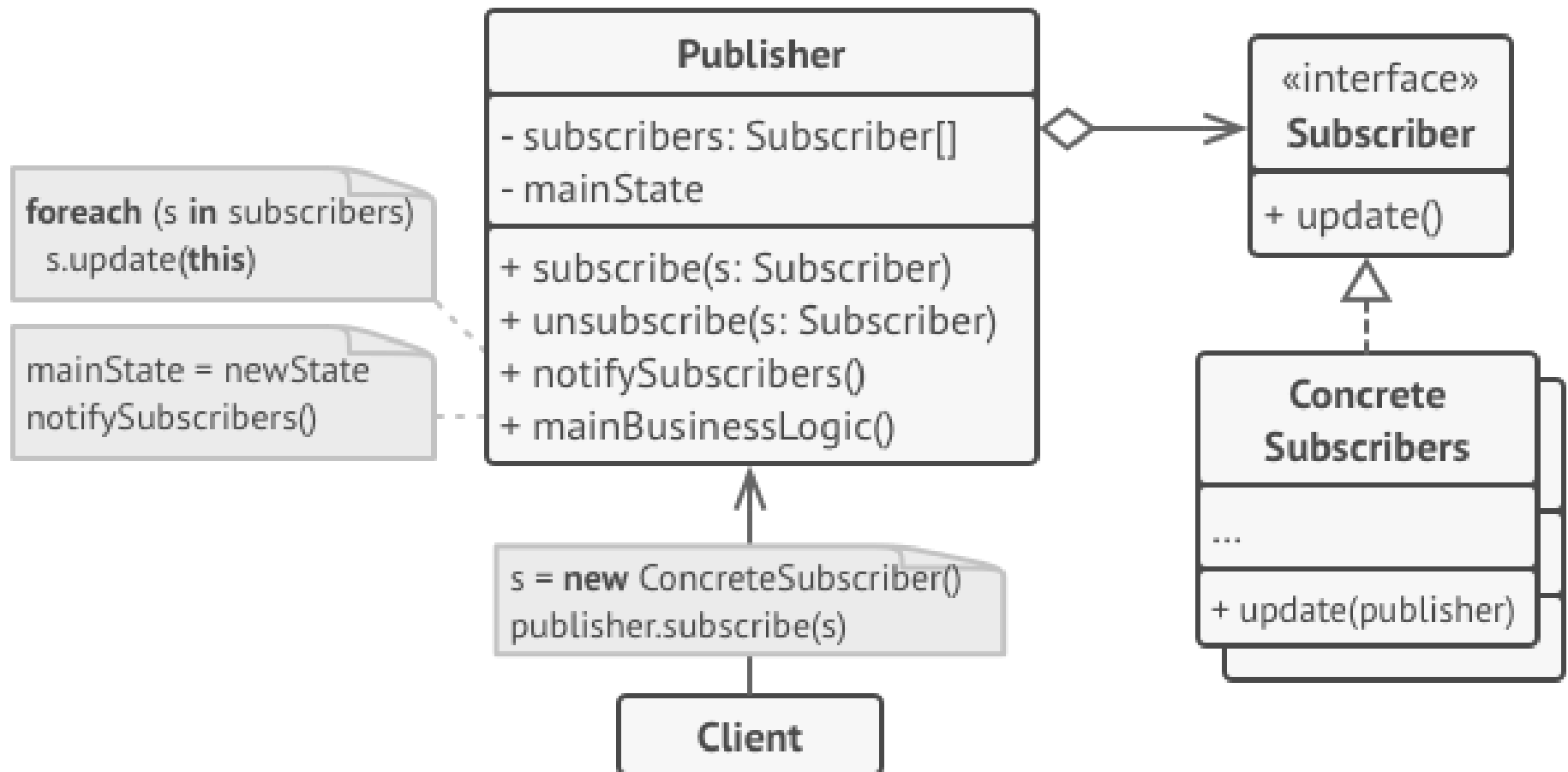
# Шаблон: наблюдатель (observer)

**Наблюдатель** — это поведенческий паттерн проектирования, который создаёт механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах.





# Шаблон: издатель (publisher)



# Шаблон: паттерны в MVC

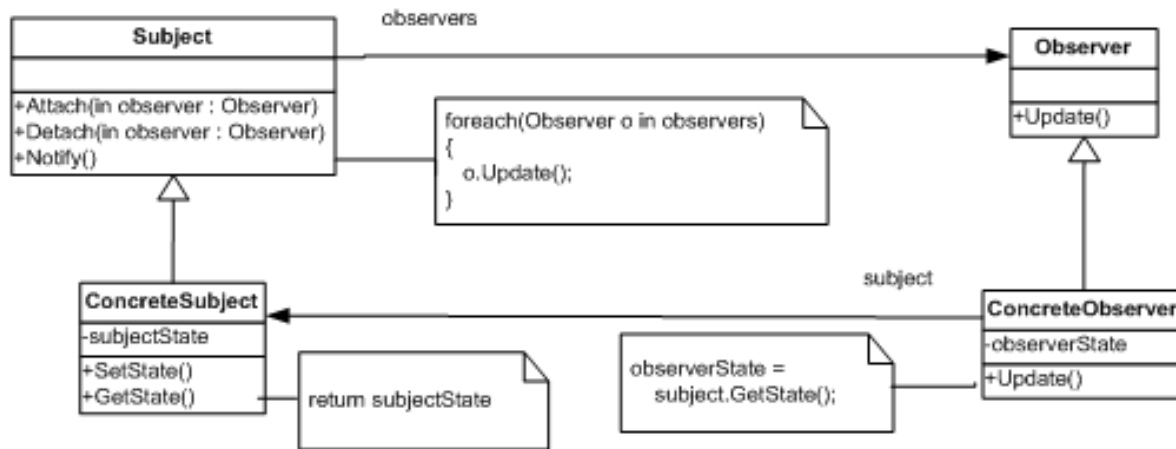


Figure 1: Observer

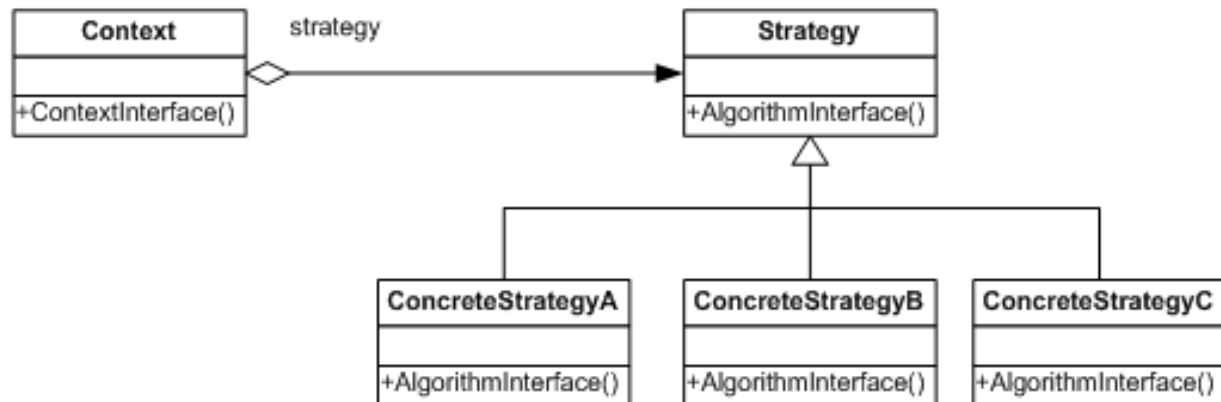


Figure 2: Strategy

# Шаблон: MVC

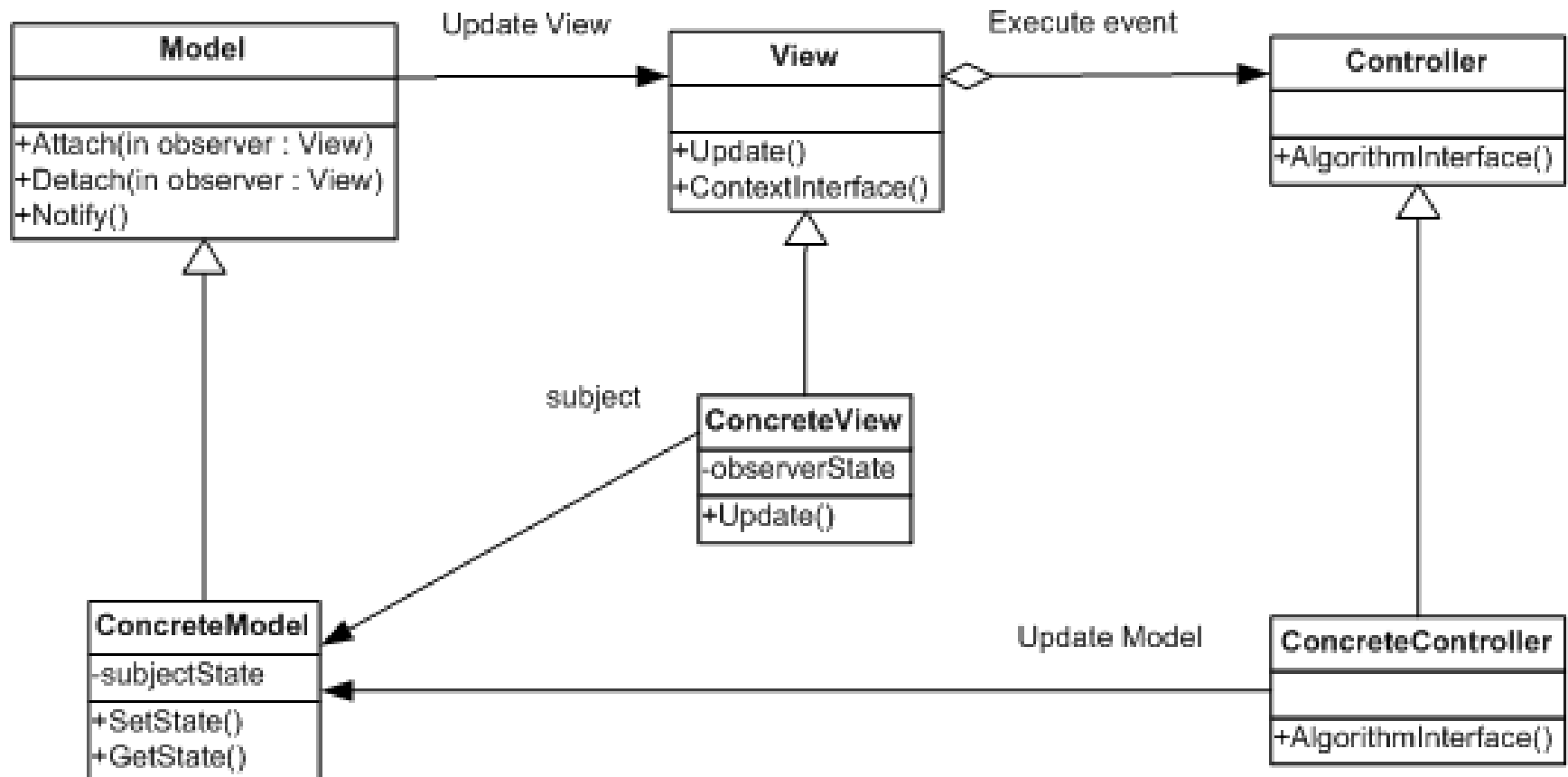


Figure 3: MVC

# Шаблон: MVP

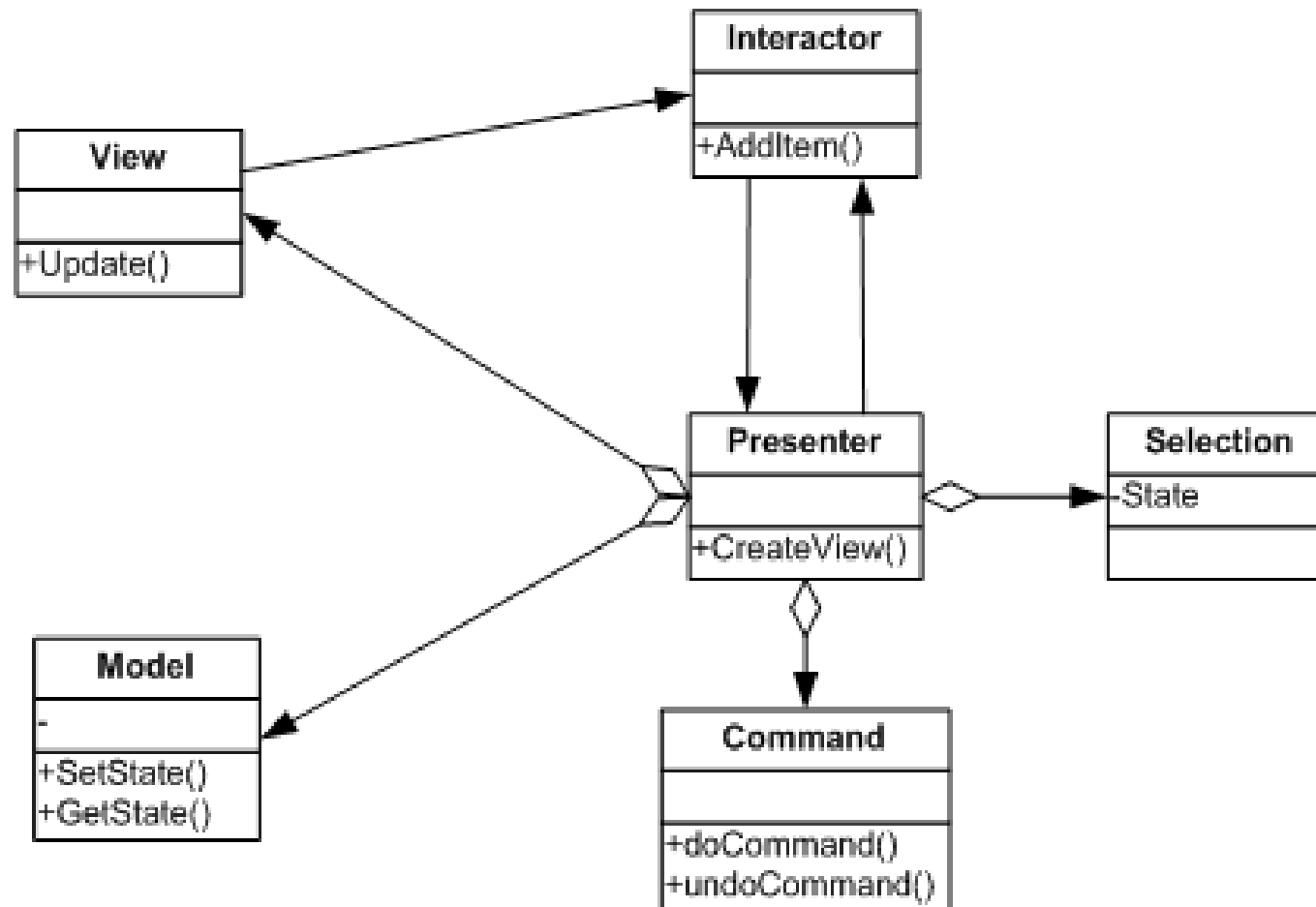


Figure 6: MVP

# Шаблон: MVP, MVVM

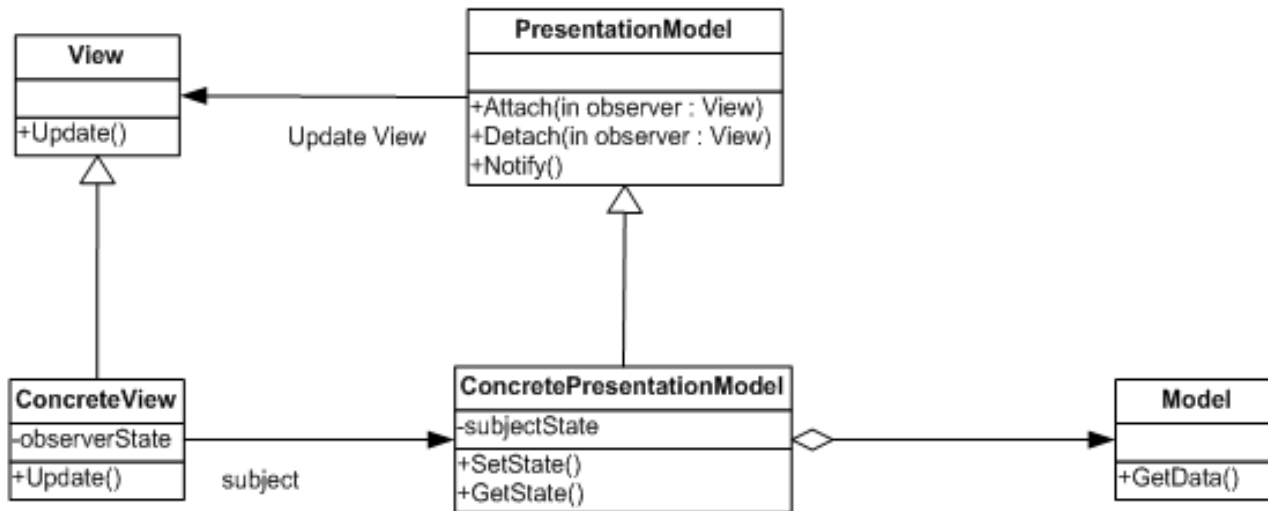


Figure 7: Presentation Model

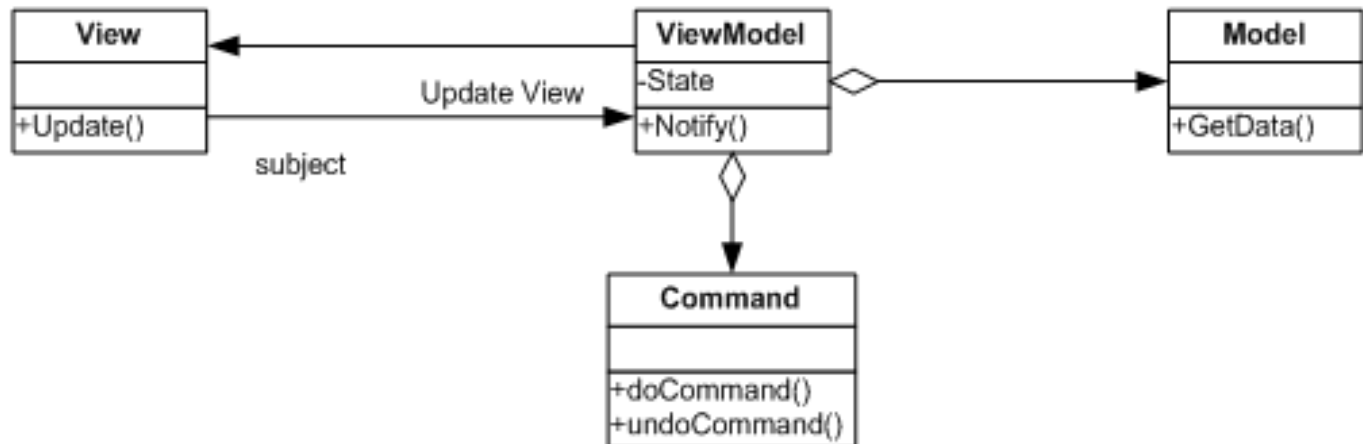
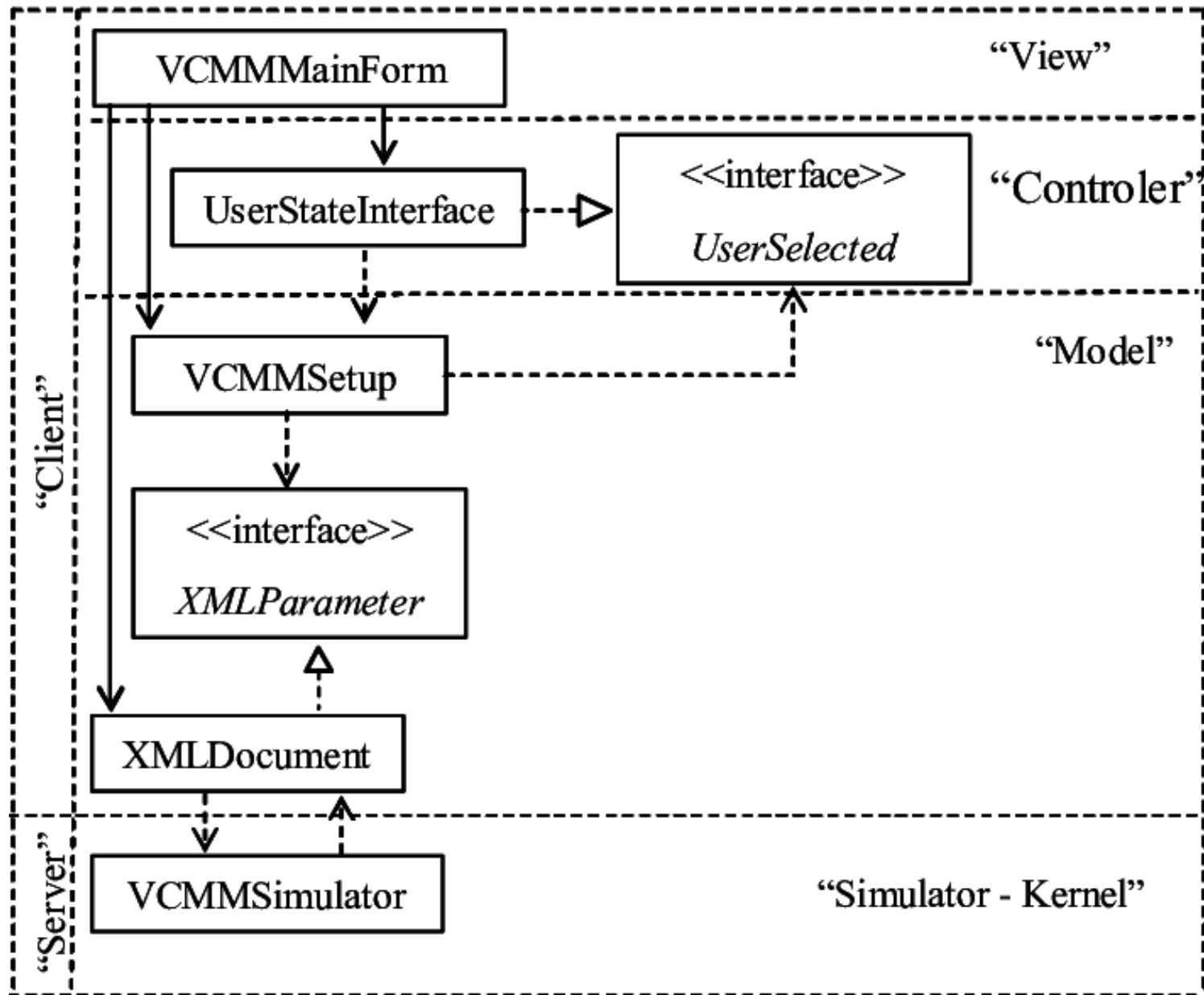


Figure 8: MVVM



# Шаблон: MVC



# Шаблон: стереотипы UML

**Стереотипы** являются одним из трех типов механизмов расширяемости в унифицированном языке моделирования ([UML](#))

Они позволяют проектировщикам расширять словарь UML для создания новых элементов [моделирования](#), получаемых из существующих, но имеющих определенные свойства, которые подходят для конкретной проблемы предметной области или для другого специализированного использования

