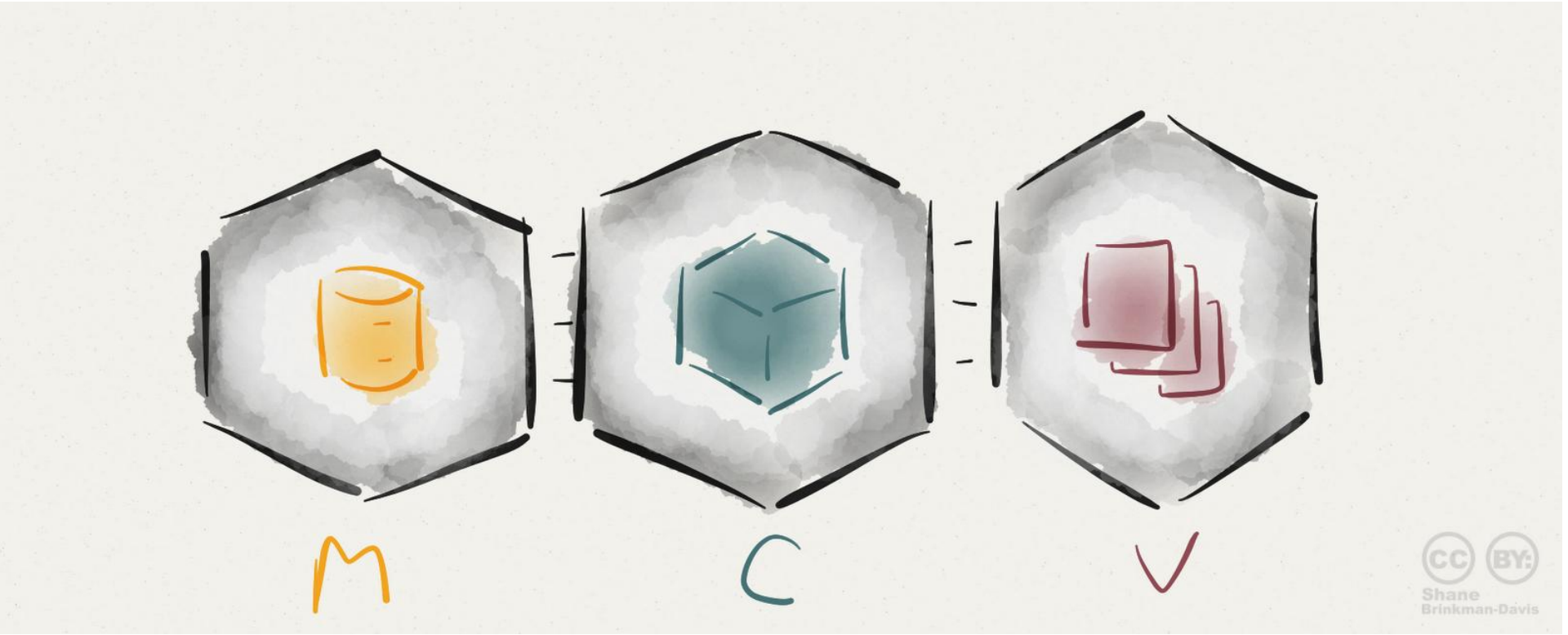


http://www.essenceandartifact.com/2012/12/the-essence-of-mvc.html

Курс: «Математические модели комплексов программ»

Модуль: **«Архитектура программных систем»**

Лекция 6 (Князьков К.В.)

Model-View-Controller

MVC

- **Model-view-controller** (MVC, «модель-представление-поведение», «модель-представление-контроллер», «модель-вид-контроллер») — **схема** использования нескольких шаблонов проектирования, с помощью которых модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные.

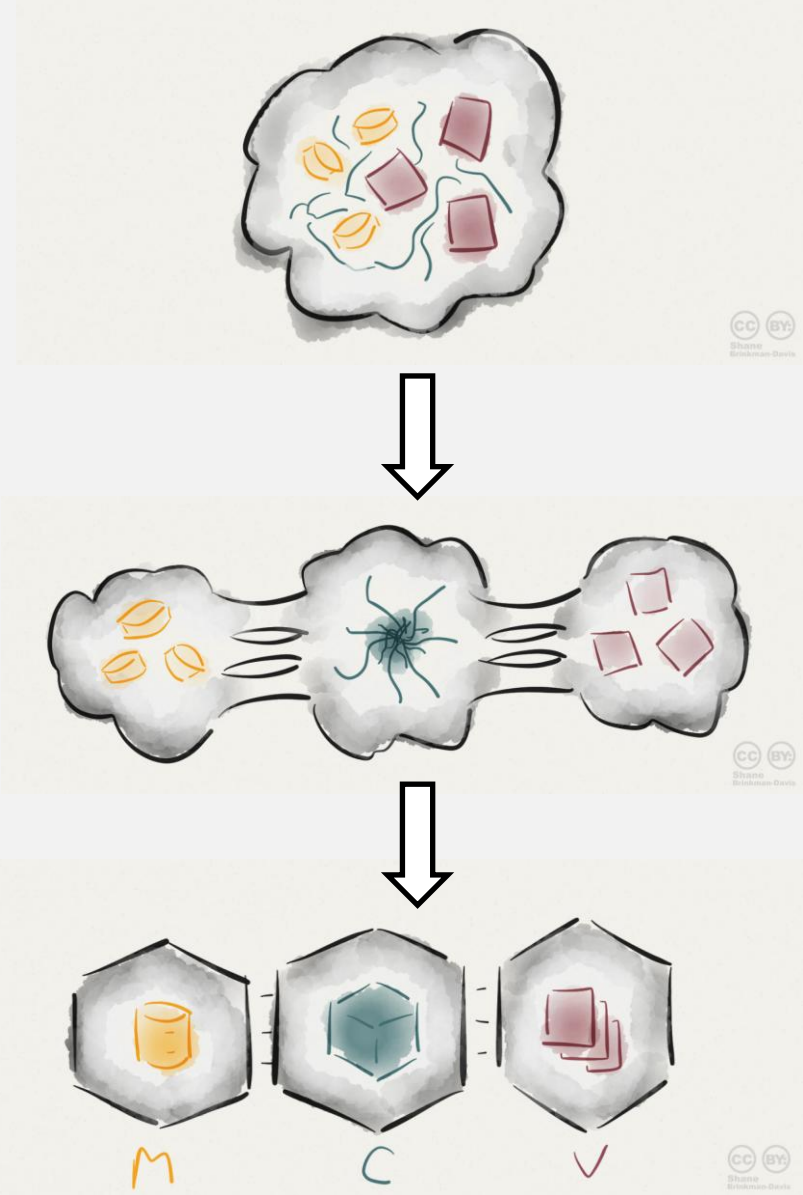
- Схема описана Трюгве Реенскаугом в 1979 г. для Smalltalk
- На текущий момент существует множество модификаций модели

Для чего?

- **Отделение бизнес-логики от представления**
  - возможность изменения одного компонента без другого
  - повторное использование

- К одной *модели* можно присоединить несколько *видов*
- Не затрагивая реализацию *видов*, можно изменить реакции на действия пользователя
- Разные специализации разработчиков

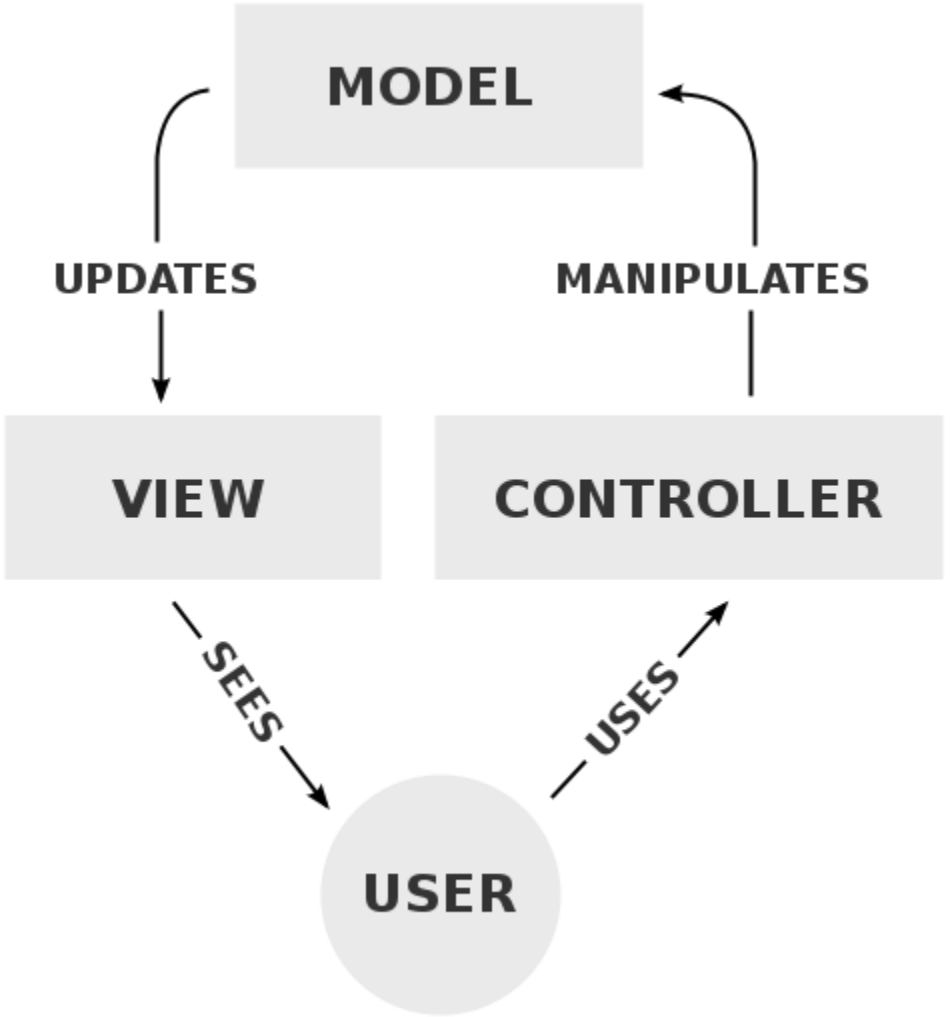
- **Модель** (англ. Model).
  - Автономна, не знает о других компонентах
  - Модель описывает бизнес-логику, данные и методы работы с этими данными
  - Не содержит информации, как эти знания можно визуализировать.
- **Представление**, вид (англ. View).
  - Отвечает за отображение информации (визуализацию).
- **Контроллер** (англ. Controller).
  - Обеспечивает связь между пользователем и системой
  - Отвечает за взаимодействие модели и представления



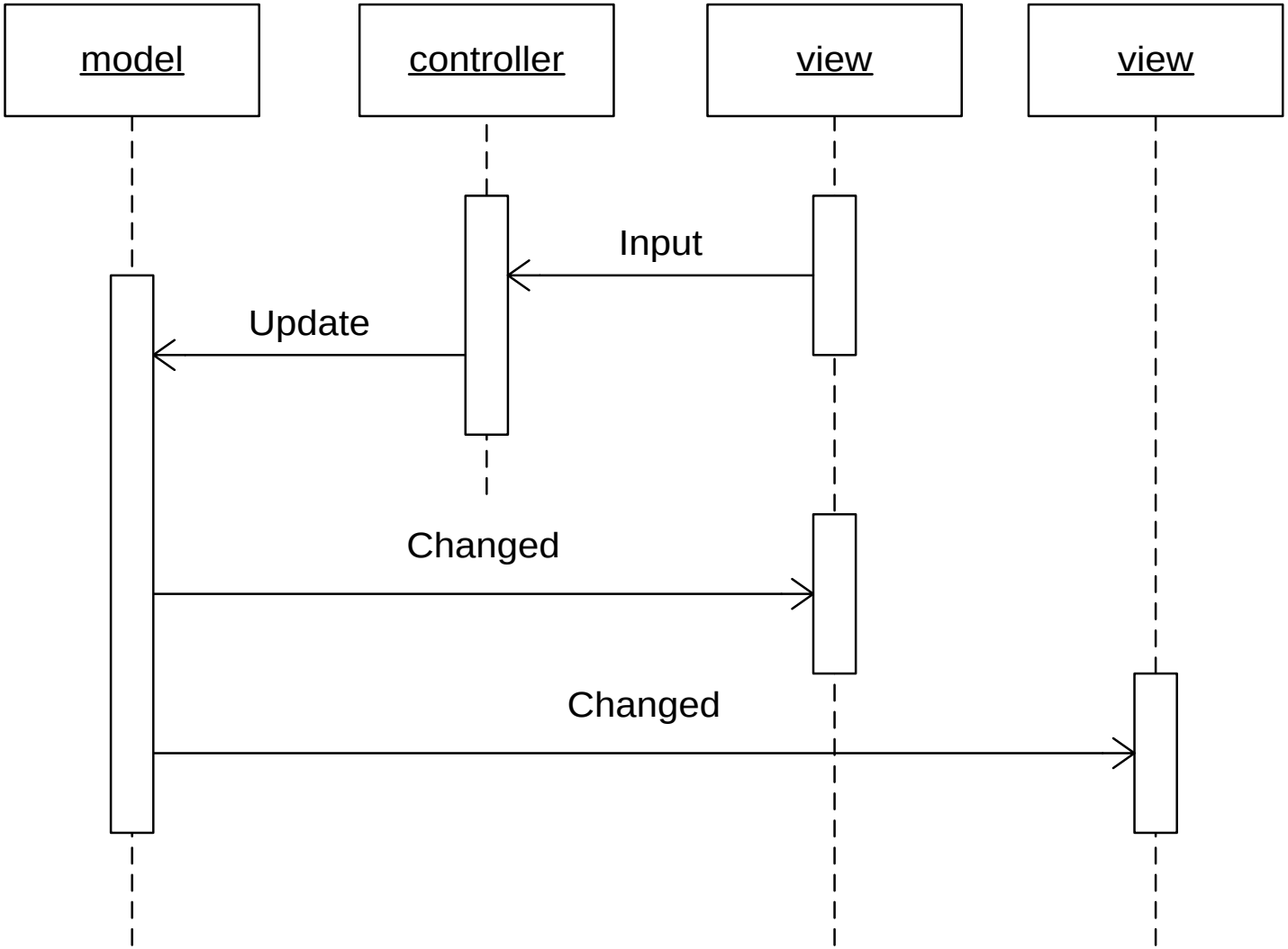
http://www.essenceandartifact.com/2012/12/the-essence-of-mvc.html

Стандартный «активный» вариант MVC

- Контроллер реагирует на действия пользователя и меняет модель
- Модель оповещает представление



Стандартный «активный» вариант MVC



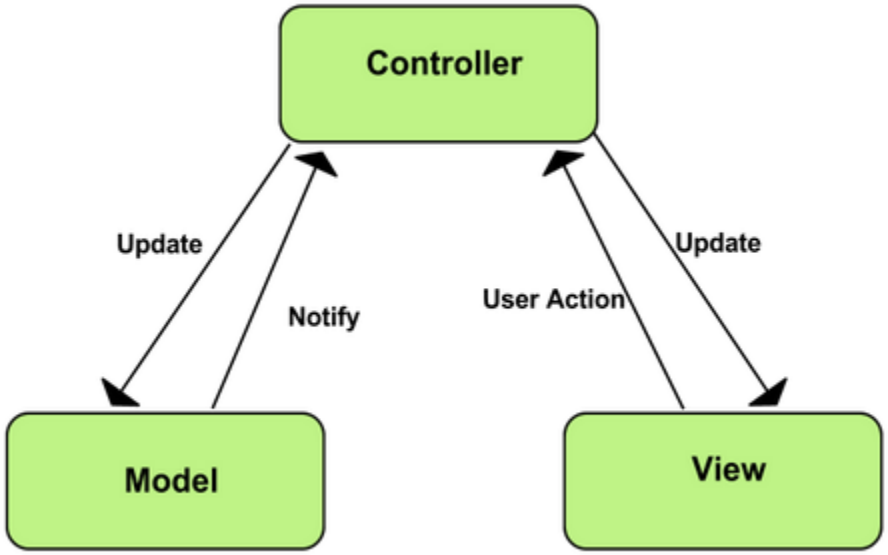
https://sites.google.com/site/abreslav2/softwaredesign2

Шаблоны проектирования в MVC

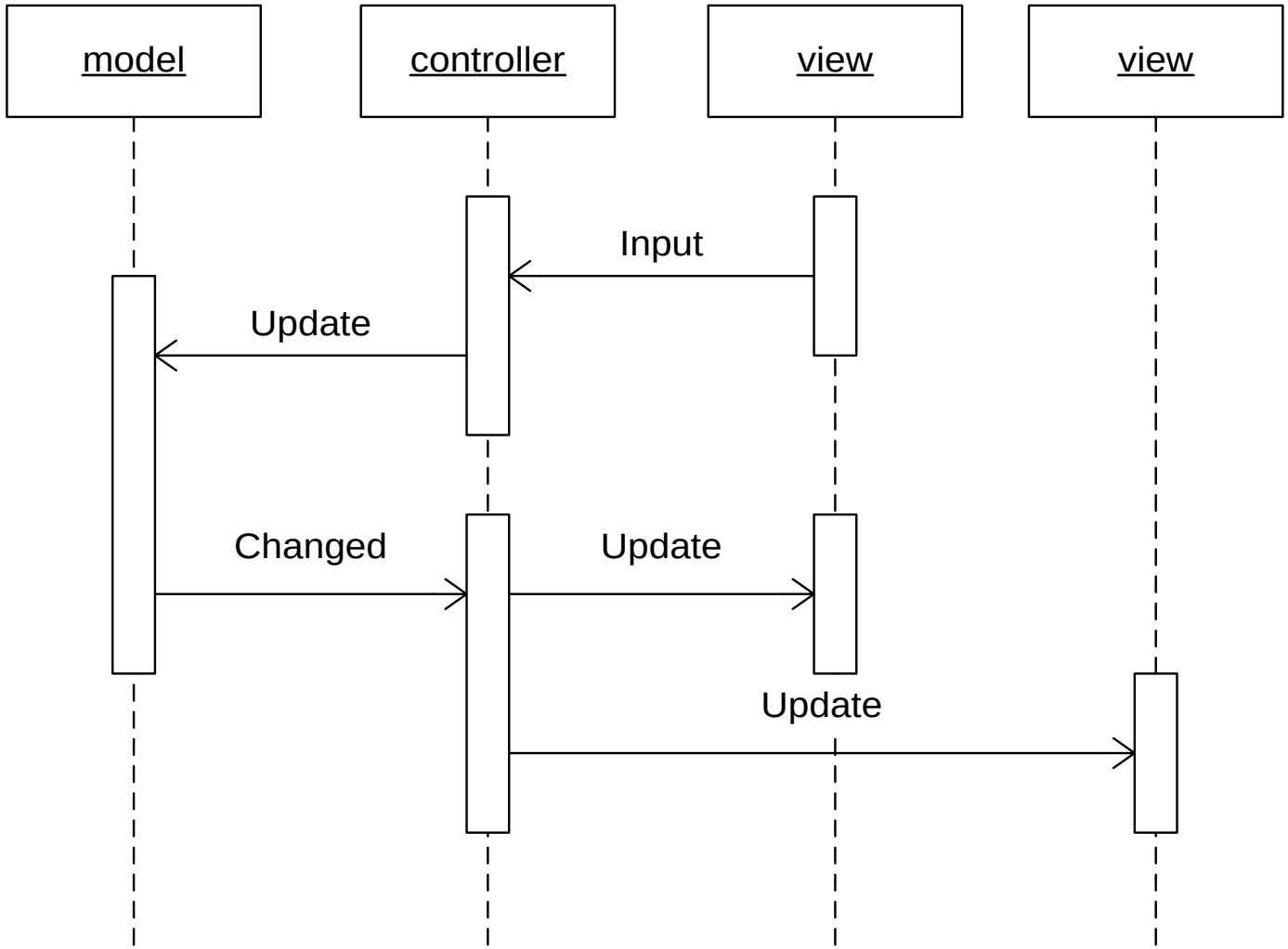
- **«наблюдатель»** используется для отделения вида от модели на основе протокола событий;
- **«стратегия»** используется при обработке реакции пользователя контроллером;
- **«компоновщик»** используется для представления сложного вида;
- **«фабричный метод»** может быть использован для задания типа контроллера для вида

«Чистый» вариант MVC

- Модель не знает о контроллере
- Модель не имеет непосредственной связи с представлением
- Представление не знает о контроллере
- Представление не имеет непосредственной связи с моделью
- Контроллер полностью реализует связь между моделью и представлением



«Чистый» вариант MVC

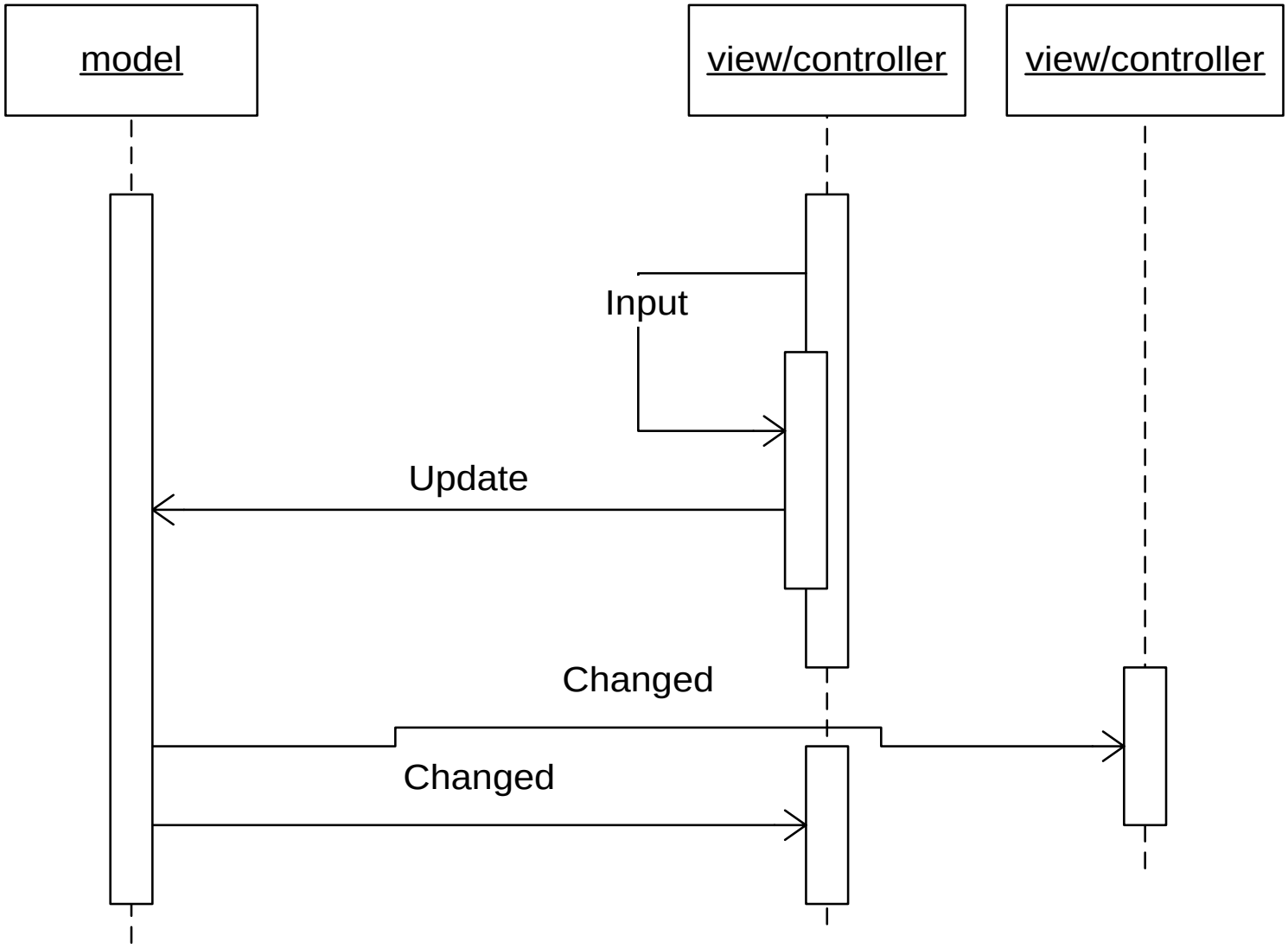


https://sites.google.com/site/abreslav2/softwaredesign2

Вариант MVC с вырожденным контроллером

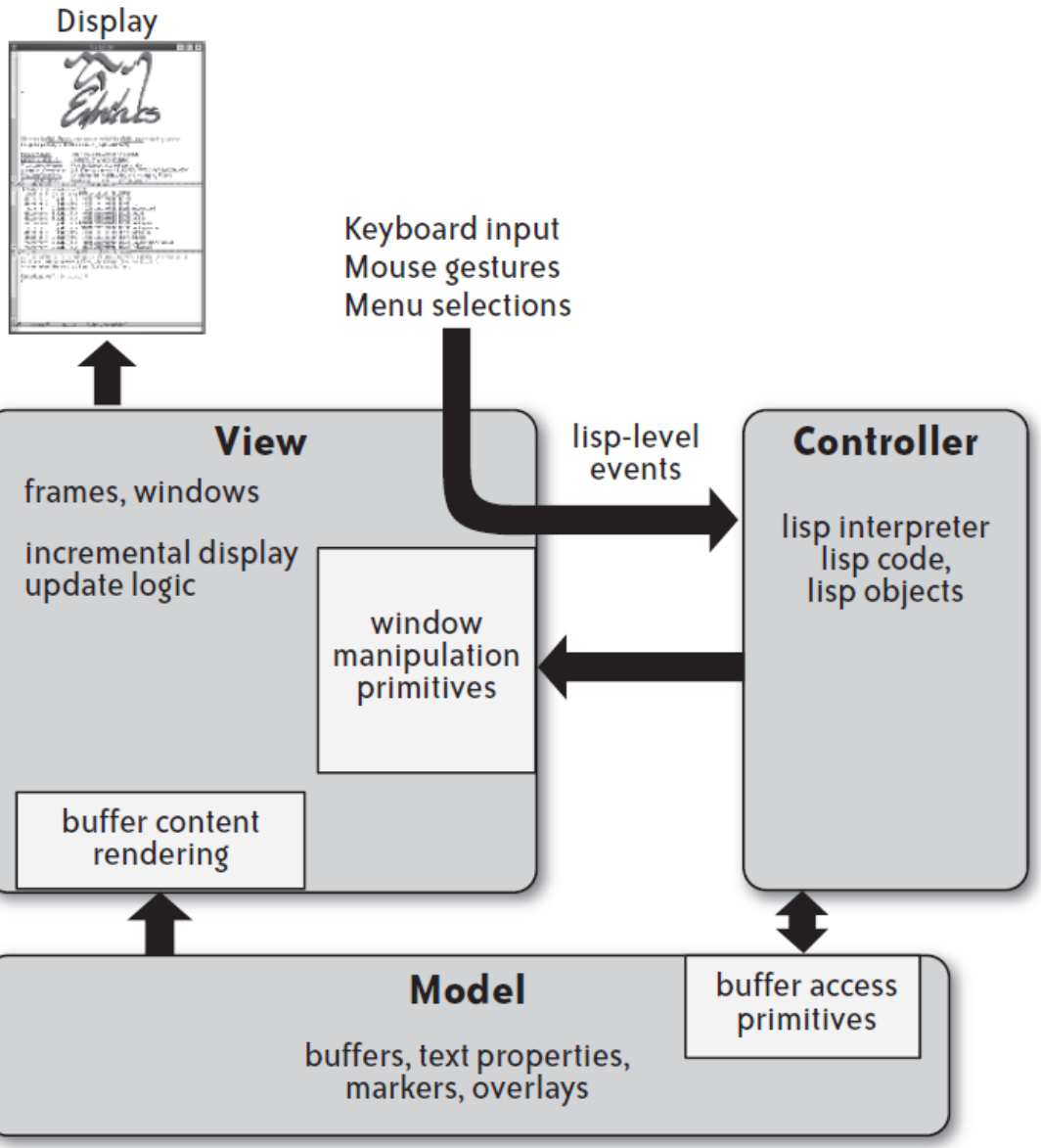
- Представление и контроллер реализуются одним классом
  - Знают друг о друге
  - Знают о модели
- Модель оповещает представление-контроллер

Вариант MVC с вырожденным контроллером



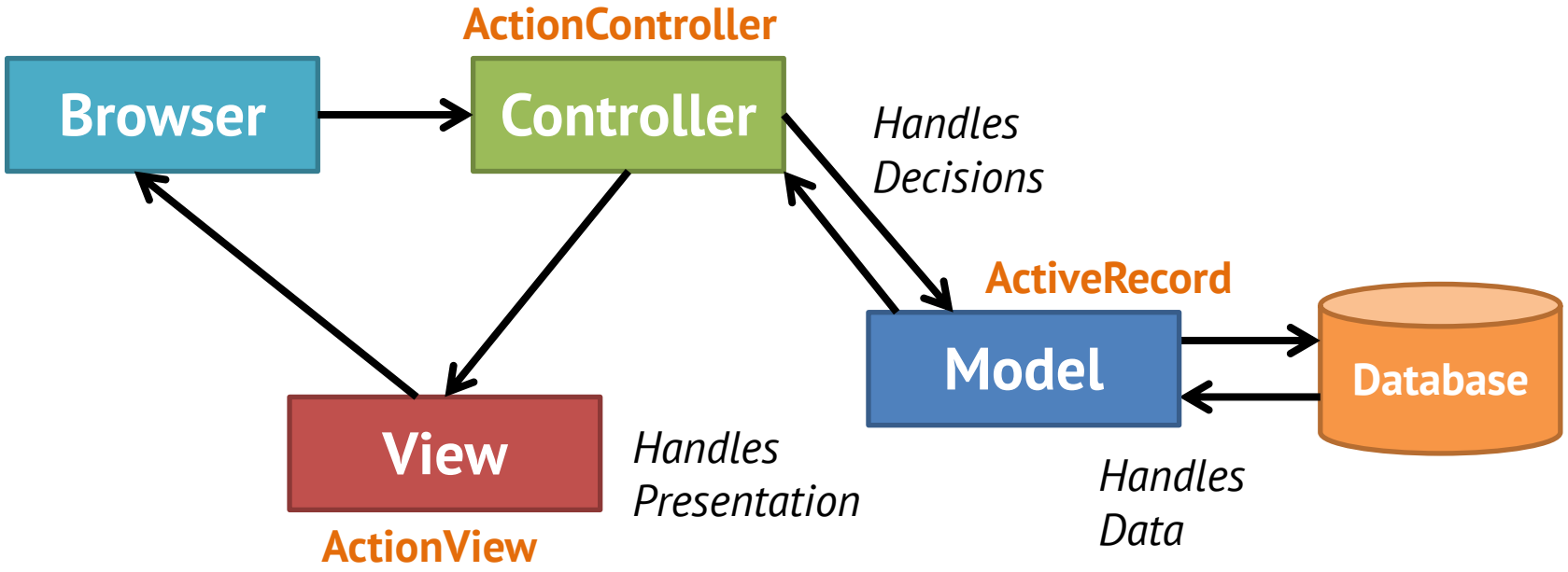
https://sites.google.com/site/abreslav2/softwaredesign2

EMACS



Beautiful Architecture. Leading Thinkers Reveal the Hidden Beauty in Software Design, O'Reilly Media, 2009

Ruby on Rails



```ruby
class Bookmark < ActiveRecord::Base
  belongs_to :user
  # ensure that a user_id is present
  validates :user_id, presence: true
  # ensure that title is present and at least 10 chars long
  validates :title, length: { minimum: 10 }, presence: true
  # ensure the url is present, and respects the URL format for http/https
  validates :url, format: {with: Regexp.new(URI::regexp(%w(http https)))}, presence: true
end
```

```ruby
class SiteController < ApplicationController
  def index
    # retrieve all Bookmarks ordered by descending creation timestamp
    @bookmarks = Bookmark.order('created_at desc')
  end
end
```

```erb
<h2>Latest Bookmarks</h2>
<table style="width: 100%">
  <thead><tr><th>Url</th></tr></thead>
  <tbody>
  <% @bookmarks.each do |bookmark| %>
    <tr><td><%= link_to bookmark.title, bookmark.url %></td></tr>
  <% end %>
  </tbody>
</table>
```

BACKBONE.JS

- MVC-framework для создания одностраничных web-приложения на основе JavaScript и RESTful API
- Используется: foursquare, digg, pinterest, airbnb…

- Controller вырожден и включен в View

Beautiful Architecture. Leading Thinkers Reveal the Hidden Beauty in Software Design, O'Reilly Media, 2009

```
//Define Model
var Todo = Backbone.Model.extend({
    defaults: function() {
        return {
            title: "no title...",
            order: Todos.nextOrder(),
            done: false
        };
    },
    toggle: function() {
      this.save({done: !this.get("done")});
    }
});
```

```
//Model Collection
var TodoList = Backbone.Collection.extend({
    model: Todo,
    localStorage: new Backbone.LocalStorage("todos-backbone"),
    done: function() {
        return this.where({done: true});
    },
    remaining: function() {
        return this.without.apply(this, this.done());
    },
    nextOrder: function() {
        if (!this.length) return 1;
        return this.last().get("order") + 1;
    },
    comparator: 'order'
});
```

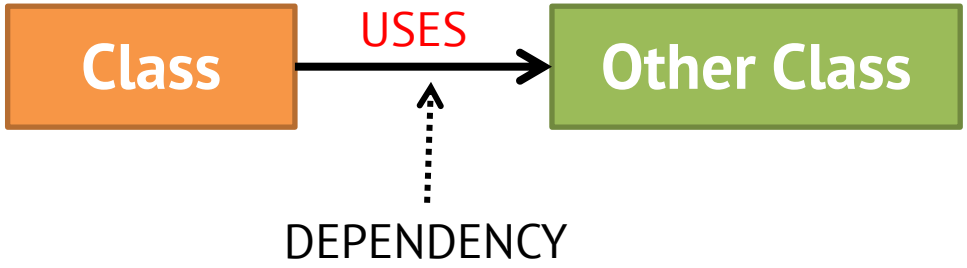
```
//Model View & event action
var TodoView = Backbone.View.extend({
    tagName:  "li",
    template: _.template($("#item-template").html()),
    events: {
        "click .toggle"   : "toggleDone",
        "dblclick .view"  : "edit",
        "click a.destroy" : "clear",
        "keypress .edit"  : "updateOnEnter",
        "blur .edit"      : "close"
    },
    initialize: function() {
        this.listenTo(this.model, "change", this.render);
        this.listenTo(this.model, "destroy", this.remove);
    },
    render: function() {
        this.$el.html(this.template(this.model.toJSON()));
        this.$el.toggleClass("done", this.model.get("done"));
        this.input = this.$(".edit");
        return this;
    },
    toggleDone: function() {
        this.model.toggle();
    },
    edit: function() {
        this.$el.addClass("editing");
        this.input.focus();
    },
    close: function() {
        var value = this.input.val();
        if (!value) {
            this.clear();
        } else {
            this.model.save({title: value});
            this.$el.removeClass("editing");
        }
    },
    updateOnEnter: function(e) {
        if (e.keyCode == 13) this.close();
    },
    clear: function() {
        this.model.destroy();
    }

});
```

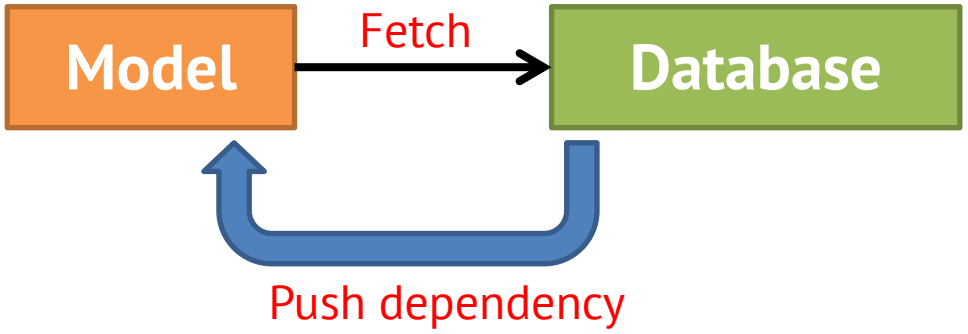
Dependency Injection

Dependency injection

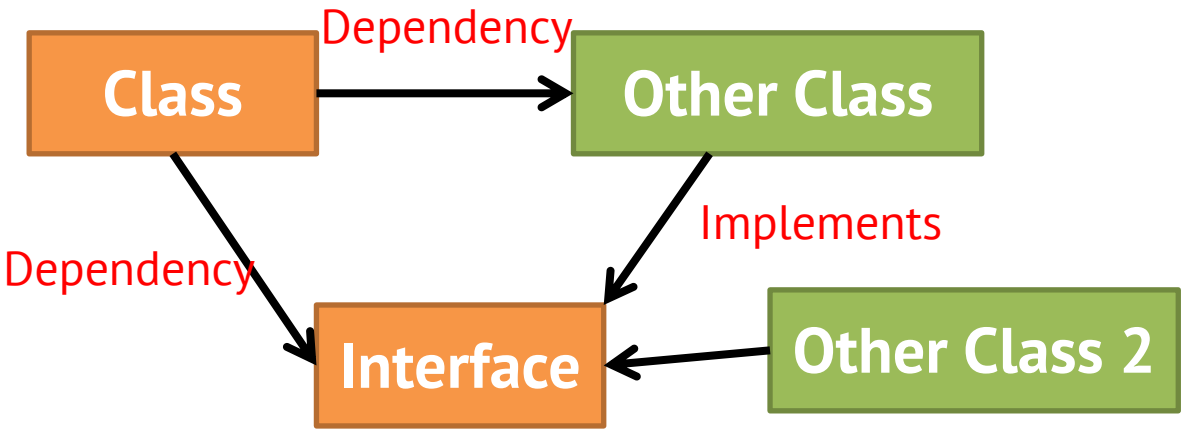
- Внедрение зависимости (англ. **Dependency injection**, DI) — процесс предоставления внешней зависимости программному компоненту.
- Является специфичной формой «инверсии управления» (англ. **Inversion of control**, IoC), где изменение порядка связи осуществляется путём получения необходимой зависимости.

Beautiful Architecture. Leading Thinkers Reveal the Hidden Beauty in Software Design, O'Reilly Media, 2009





public Model(Database db)



Способы реализации

- Construction injection

```
Client(Service service) {
    this.service = service;
}
```

- Setter injection

```
public void setService(Service service) {
    this.service = service;
}
```

- Interface injection

```
public interface ServiceSetter {
    public void setService(Service service);
}
public class Client implements ServiceSetter {
    private Service service;
    @Override
    public void setService(Service service) {
        this.service = service;
    }
}
```

Ninject

```
public class Samurai {
    public IWeapon Weapon { get; private set; }
    public Samurai(IWeapon weapon)
    {
        this.Weapon = weapon;
    }
}
```


```
public class WarriorModule : NinjectModule
{
    public override void Load()
    {
        this.Bind<IWeapon>().To<Sword>();
    }
}
```

```
IKernel kernel = new StandardKernel();
var samurai = kernel.Get<Samurai>();
```

Источники

- **Книги**
  - Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. «Приемы объектно-ориентированного проектирования. Паттерны проектирования» Питер-ДМК, 2001.
  - Эрик Фримен, Элизабет Фримен, Кэтти Сьерра, Берт Бейтс. «Паттерны проектирования», Питер, 2011.

- **Статьи**
  - http://www.martinfowler.com/articles/injection.html

- **Курсы**
  - Курс Андрея Бреслава «Software Design» https://sites.google.com/site/abreslav2/softwaredesign2