

# Лекция 5: Моделирование поведения

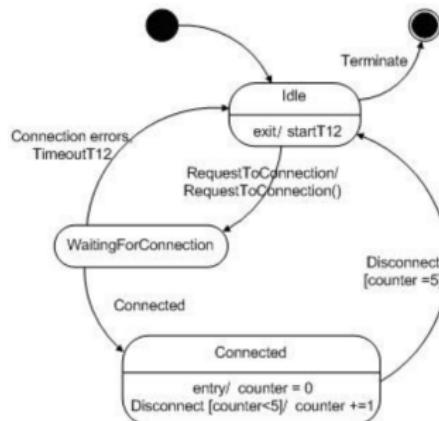
Юрий Литвинов  
y.litvinov@spbu.ru

30.09.2022

# Диаграммы конечных автоматов

## Диаграммы состояний

- ▶ Состояния объекта как часть жизненного цикла
- ▶ Моделирование реактивных объектов
  - ▶ Например, сетевое соединение
  - ▶ Или знакомый пример с торговым автоматом
- ▶ Имеют исполнимую семантику
- ▶ Д. Харел, 1987



# Диаграммы конечных автоматов, синтаксис

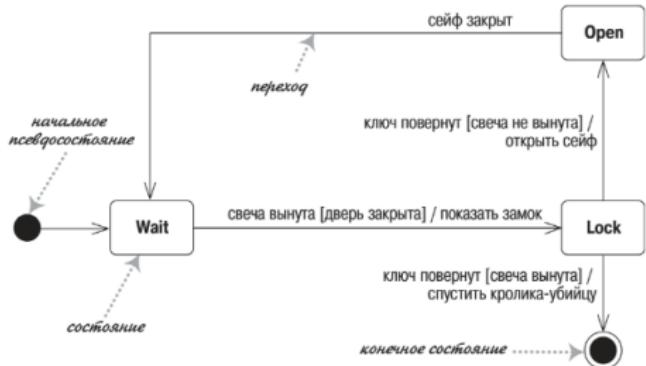
## ▶ Состояние

- ▶ entry activity
- ▶ exit activity
- ▶ do activity
- ▶ внутренний переход

## ▶ Событие

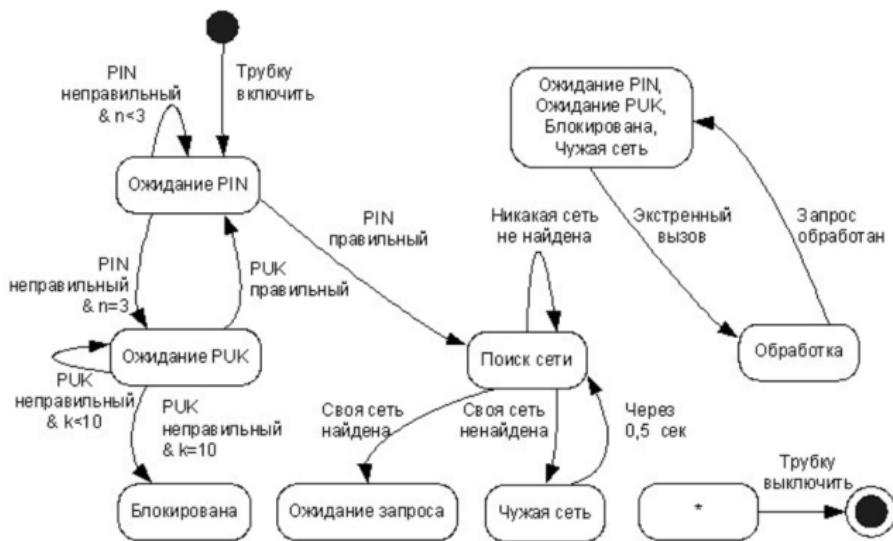
## ▶ Переход

- ▶ [<trigger> [, <trigger>]\* '[' <guard>']] [/<behavior-expression>]]



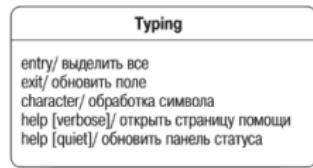
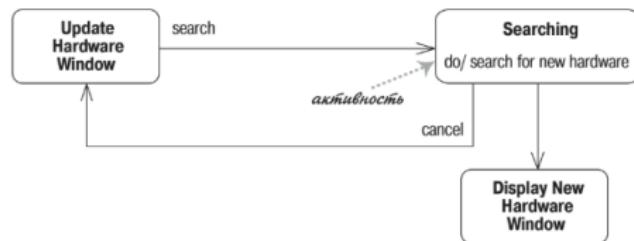
© М. Фаулер, UML. Основы

# Пример, мобильный телефон

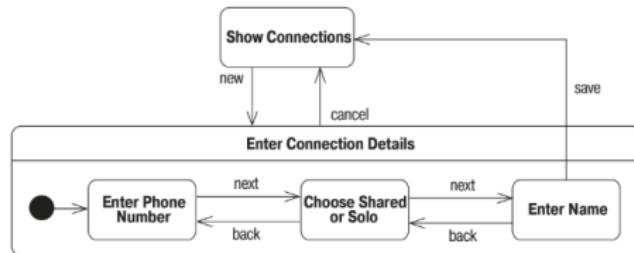


# Диаграммы конечных автоматов, прочие вещи

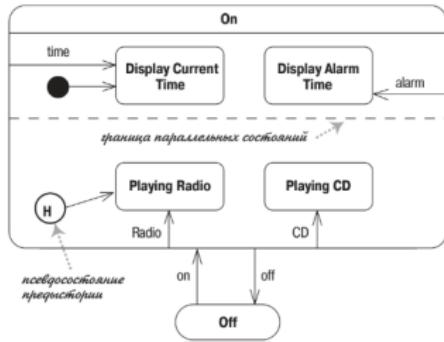
## Активности:



## Вложенные состояния:



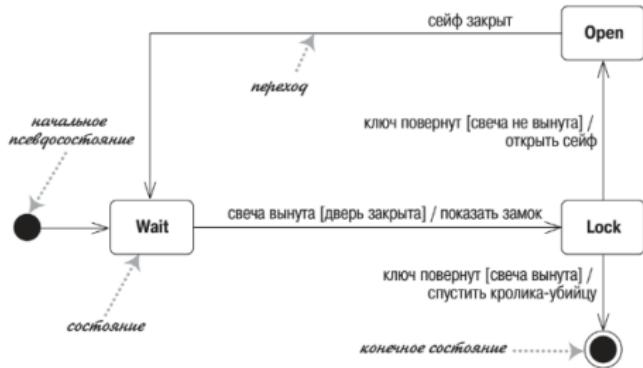
## Параллельные состояния, псевдосостояние истории:



© M. Фаулер, UML. Основы

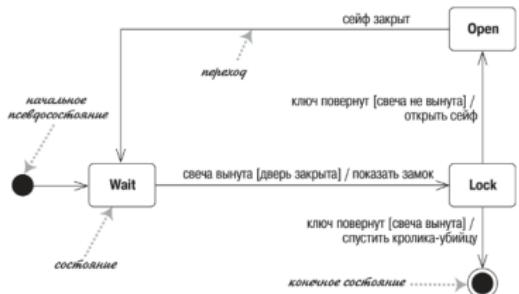
# Генерация кода

```
public void handleEvent(PanelEvent anEvent) {
    switch (currentState) {
        case PanelState.Open:
            switch (anEvent) {
                case PanelEvent.SafeClosed:
                    currentState = PanelState.Wait;
            }
            break;
        case PanelState.Wait:
            switch (anEvent) {
                case PanelEvent.CandleRemoved:
                    if (isDoorOpen) {
                        revealLock();
                        currentState = PanelState.Lock;
                    }
            }
            break;
        case PanelState.Lock:
            switch (anEvent) {
                case PanelEvent.KeyTurned:
                    if (isCandleIn) {
                        openSafe();
                        currentState = PanelState.Open;
                    } else {
                        releaseKillerRabbit();
                        currentState = PanelState.Final;
                    }
            }
            break;
    }
}
```



© M. Фаулер, UML. Основы

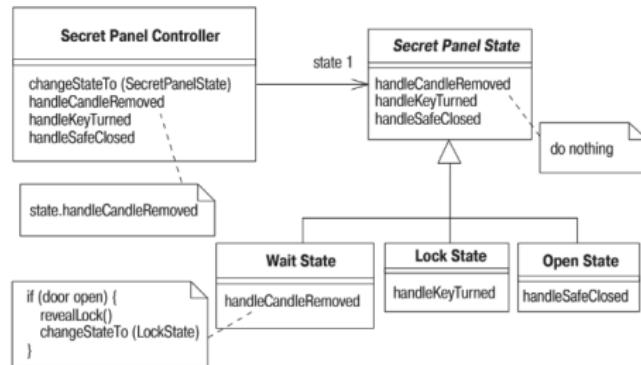
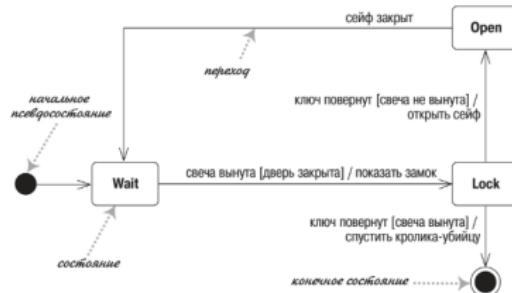
# Таблица состояний



Исходное состояние	Целевое состояние	Событие	Защита	Процедура
Wait	Lock	Candle removed (свеча удалена)	Door open (дверца открыта)	Reveal lock (показать замок)
Lock	Open	Key turned (ключ повернут)	Candle in (свеча на месте)	Open safe (открыть сейф)
Lock	Final	Key turned (ключ повернут)	Candle out (свеча удалена)	Release killer rabbit (освободить убийцу-кролика)
Open	Wait	Safe closed (сейф закрыт)		

© М. Фаулер, UML. Основы

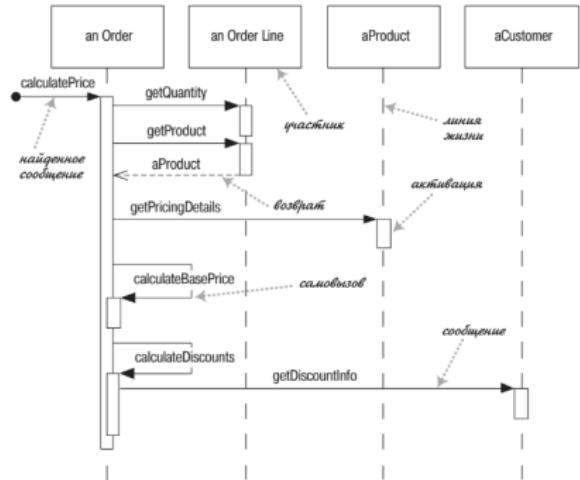
# Паттерн “Состояние”



© М. Фаулер, UML. Основы

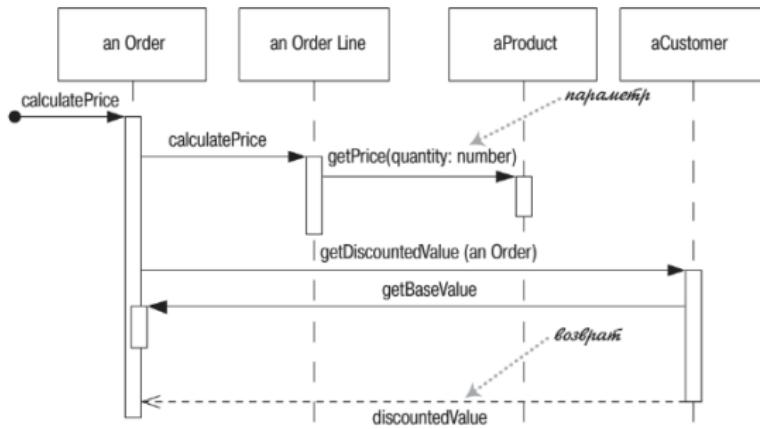
# Диаграммы последовательностей

- ▶ Применяются для визуализации взаимодействия между объектами
  - ▶ Особо удобно для асинхронных вызовов
  - ▶ Телекоммуникационные протоколы
- ▶ Могут применяться на этапе анализа предметной области
- ▶ Могут применяться для составления плана тестирования
- ▶ И даже для визуализации логов работающей системы



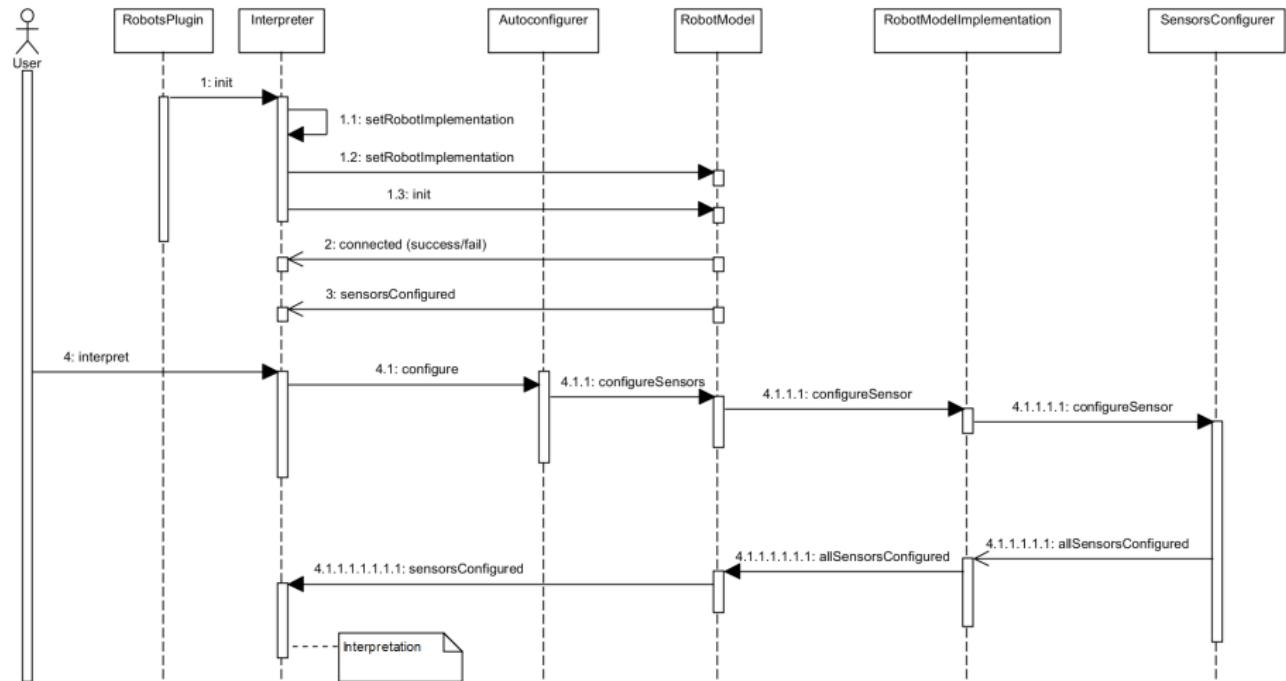
© М. Фаулер, UML. Основы

# Ещё немного о синтаксисе

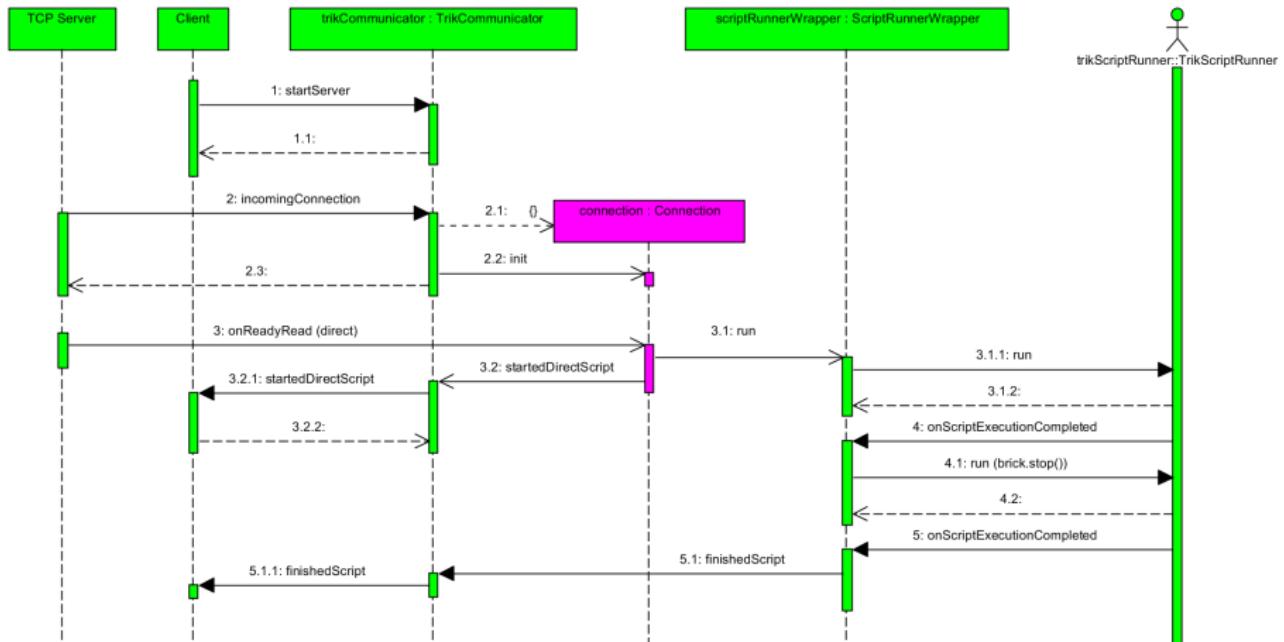


© М. Фаулер, UML. Основы

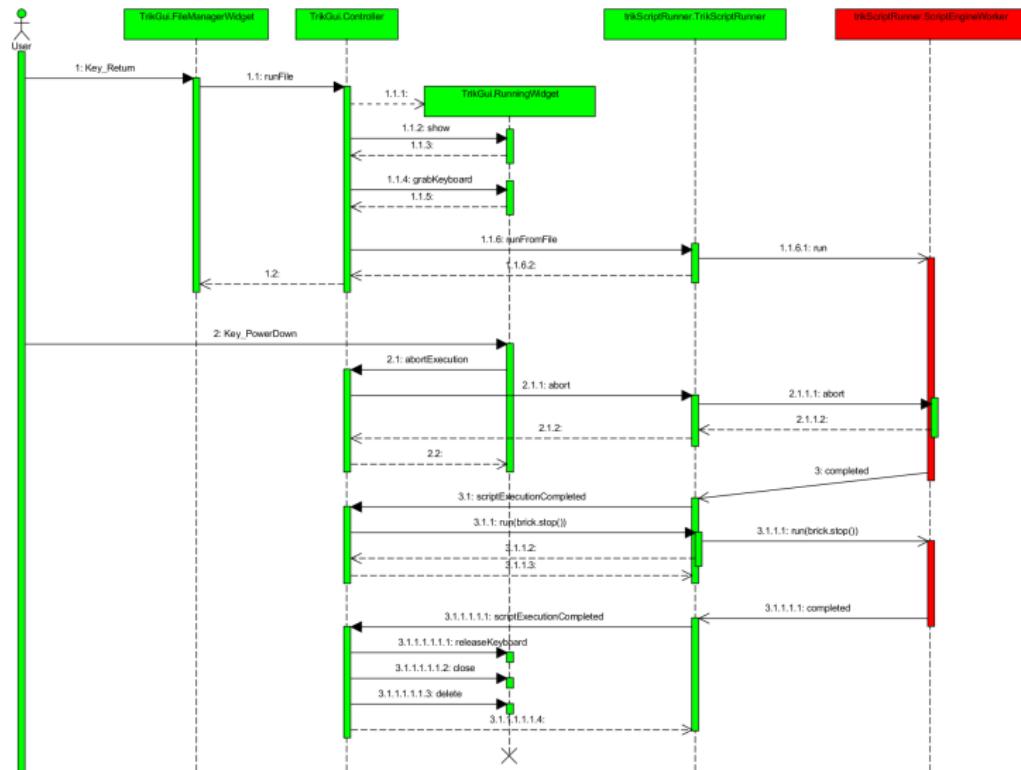
# Пример



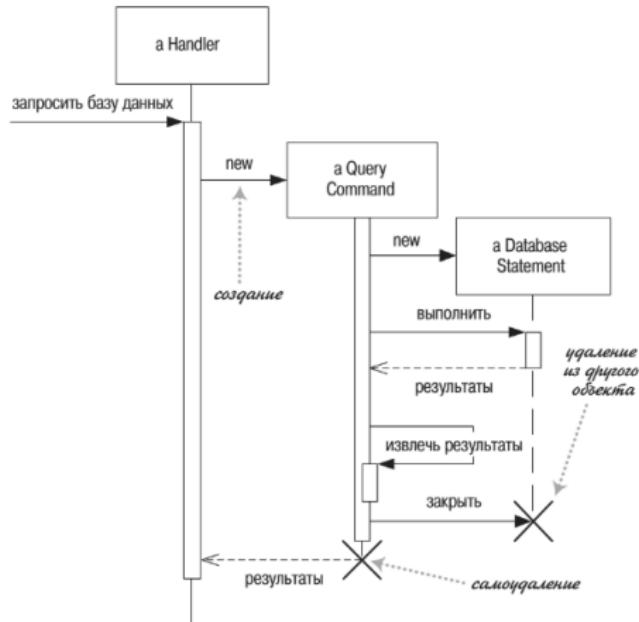
# Ещё пример



# И ещё пример



# Создание и удаление объектов

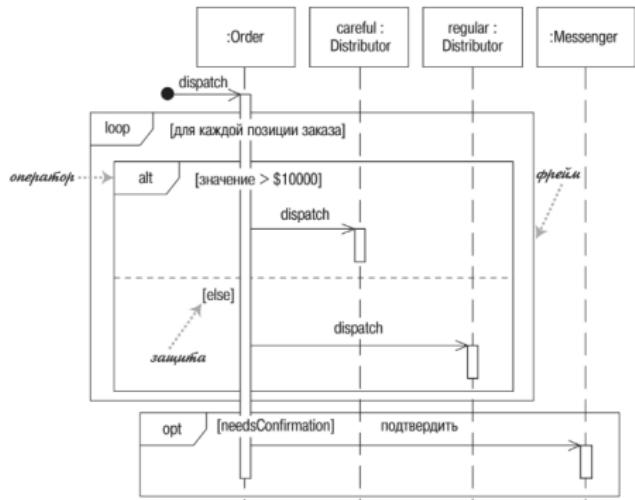


© M. Фаулер, UML. Основы

# Фреймы

```

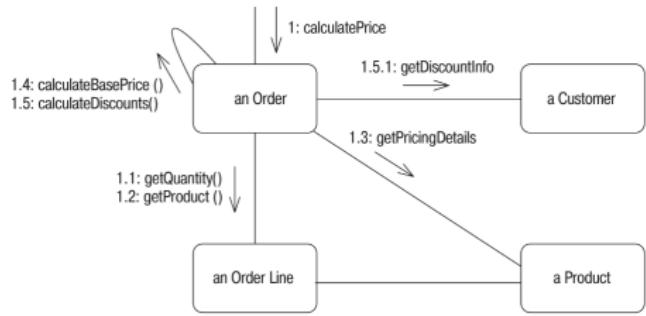
foreach (lineitem)
    if (product.value > $10K)
        careful.dispatch
    else
        regular.dispatch
    end if
end for
if (needsConfirmation)
    messenger.confirm
  
```



© М. Фаулер, UML. Основы

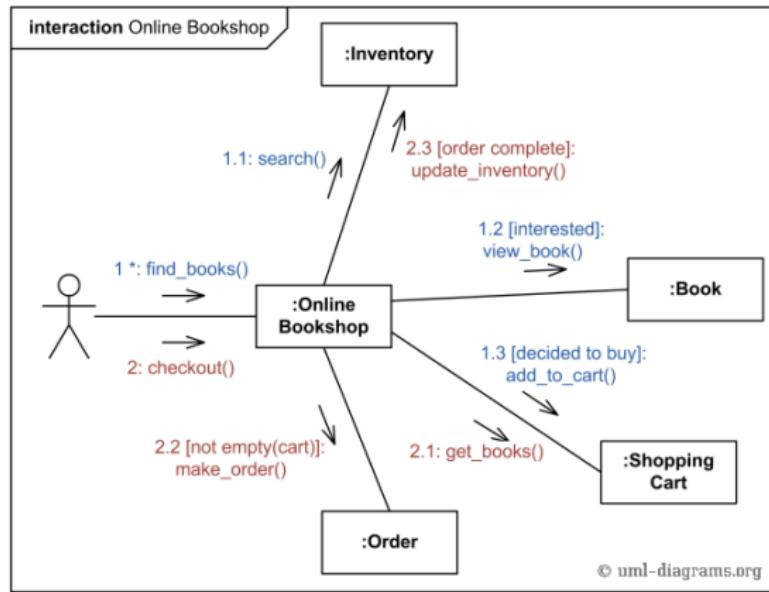
# Коммуникационные диаграммы

- ▶ Применяются для визуализации взаимодействия между объектами
  - ▶ Более легковесный аналог диаграмм последовательностей
  - ▶ Тоже отображают один сценарий взаимодействия



© М. Фаулер, UML. Основы

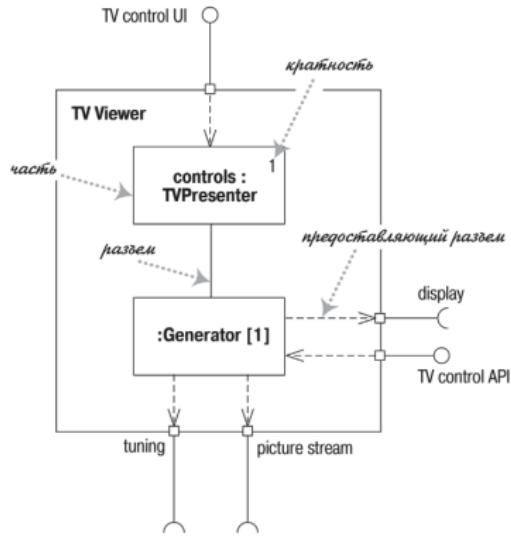
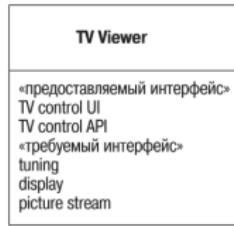
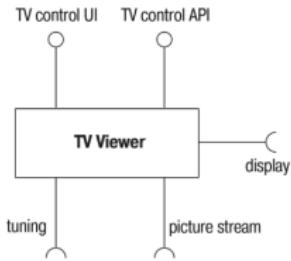
# Коммуникационные диаграммы, пример



© <http://www.uml-diagrams.org/>

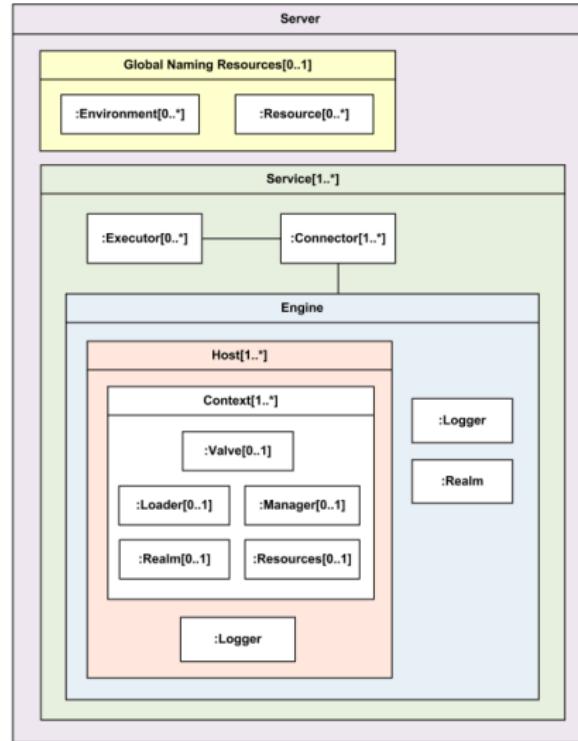
# Диаграммы составных структур

- ▶ По сути, продвинутые диаграммы компонентов
- ▶ Внутри компоненты не другие компоненты, а части (роли)



© M. Фаулер, UML. Основы

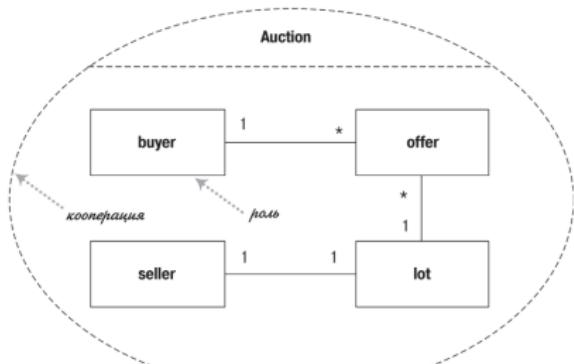
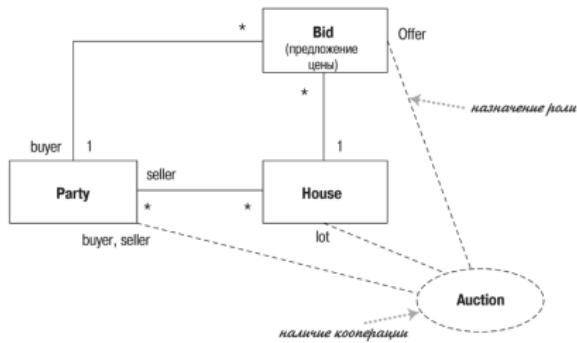
# Диаграммы составных структур, пример



© <http://www.uml-diagrams.org/>

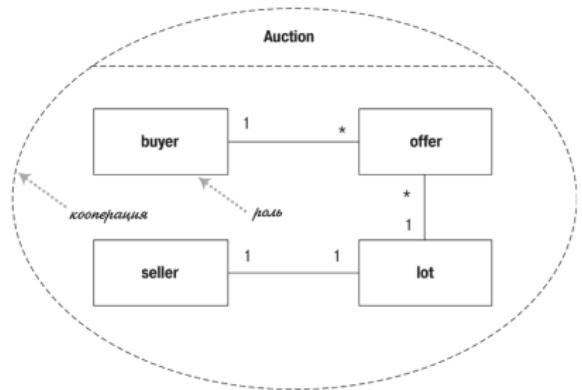
# Диаграммы коопераций

- ▶ Показывают взаимодействие между объектами (ролями) в рамках одного сценария использования



© M. Фаулер, UML. Основы

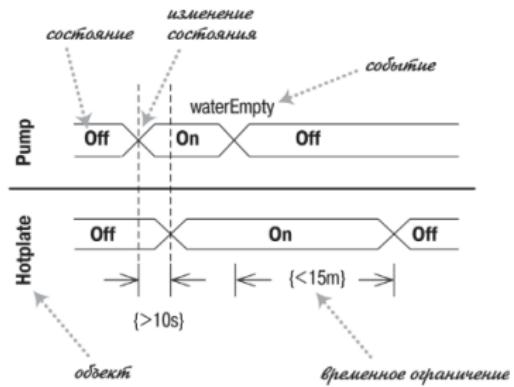
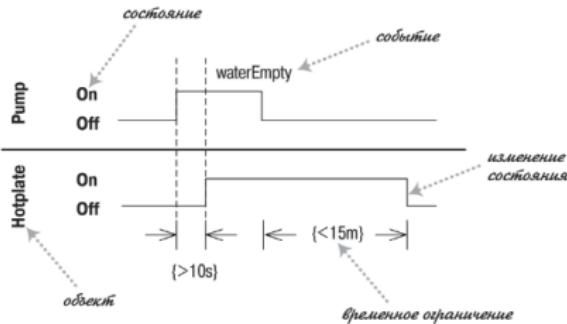
# Диаграммы коопераций, последовательности



© М. Фаулер, UML. Основы

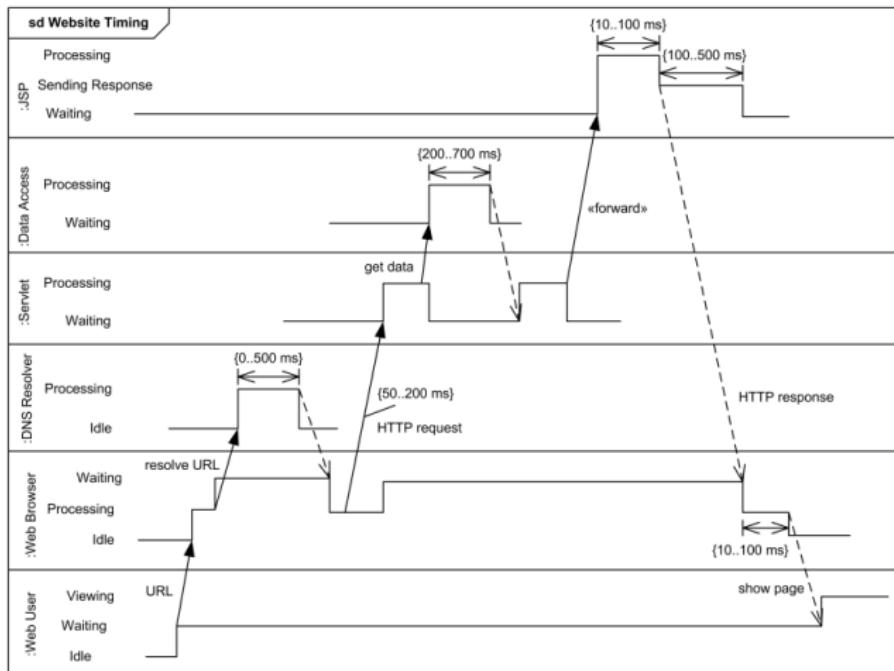
# Временные диаграммы

- ▶ Для моделирования временных ограничений в системах реального времени



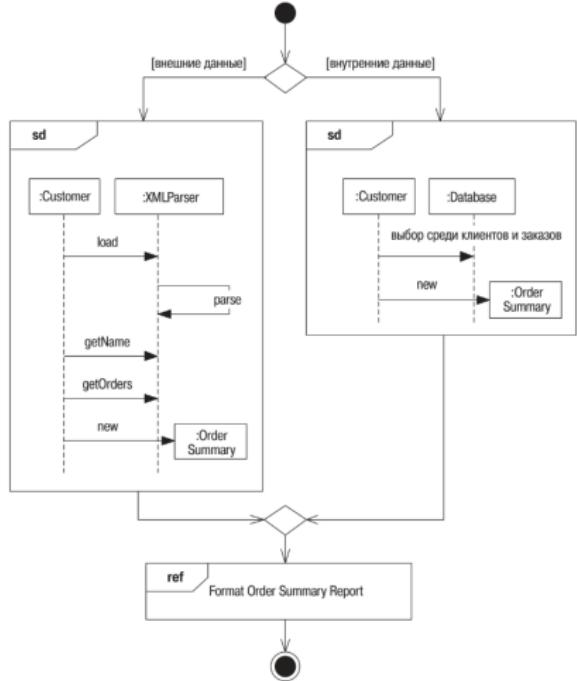
© M. Фаулер, UML. Основы

# Временная диаграмма, пример



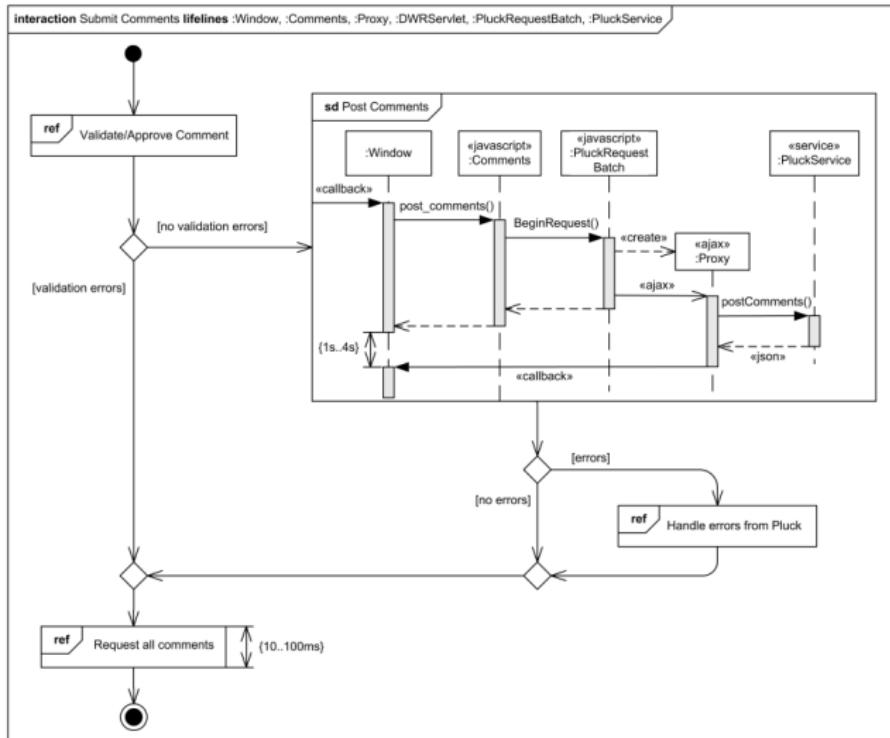
# Диаграммы обзора взаимодействия

- ▶ Диаграммы активностей + диаграммы последовательностей
- ▶ Применяются при наличии взаимодействия со сложной логикой, когда фреймы неудобны



© М. Фаулер, UML. Основы

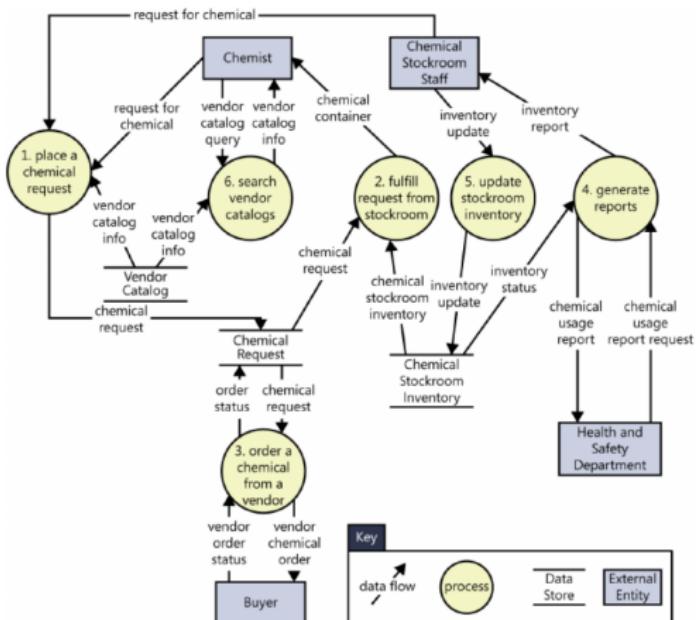
# Диаграмма обзора взаимодействия, пример



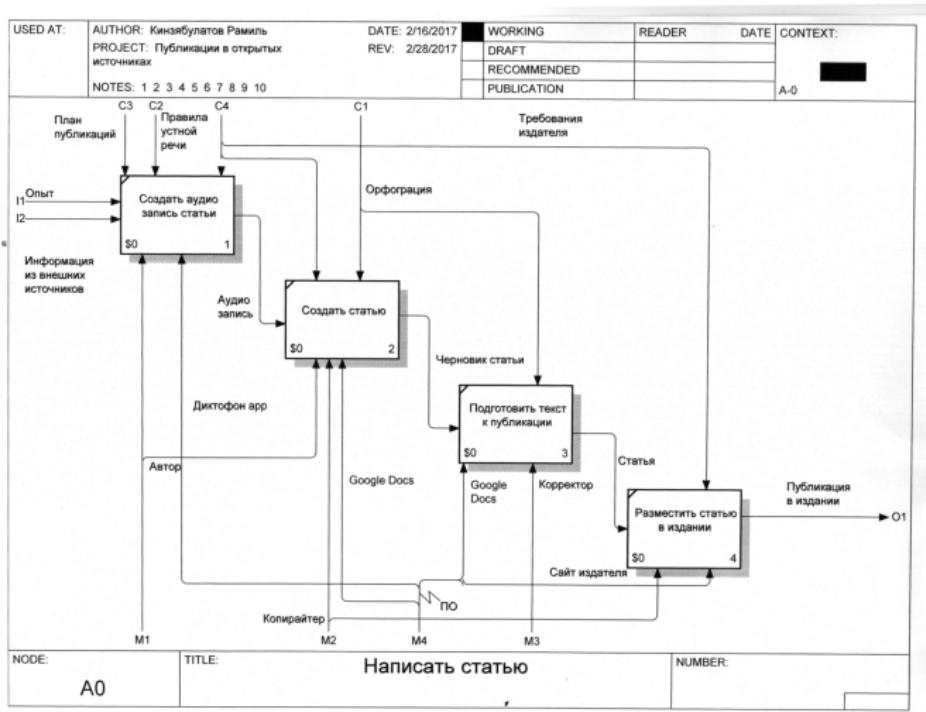
# Диаграммы потоков данных

DFD

- ▶ Показывают обмен данными в системе
- ▶ Внешние сущности, процессы внутри системы, потоки данных



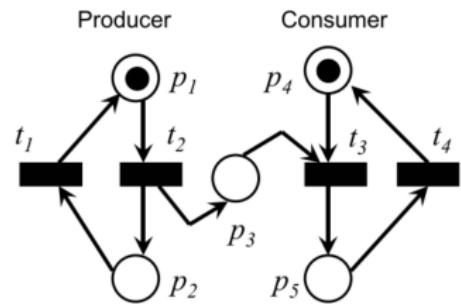
## Диаграммы IDEF0



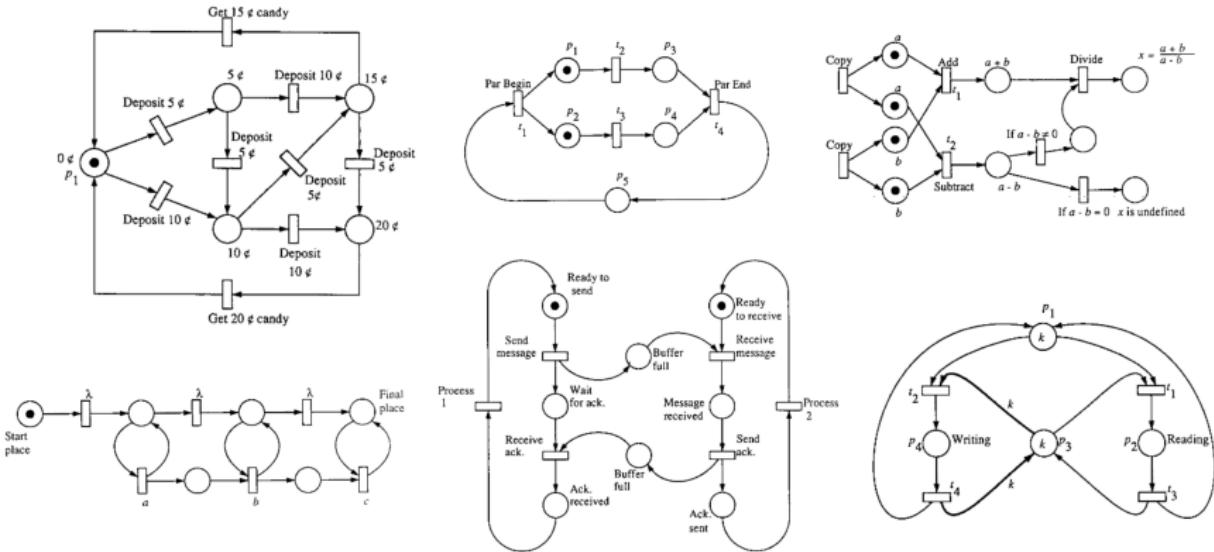
© <https://habrahabr.ru/post/322832/>

# Сети Петри

- ▶ Тройка  $(P, T, \phi)$ 
  - ▶ Множество мест
  - ▶ Множество переходов
  - ▶ Функция потока  
 $\phi : (P \times T) \cup (T \times P) \rightarrow N$
- ▶ Маркировка:  $\mu : P \rightarrow N$
- ▶ Срабатывание (firing):  $\mu \xrightarrow{t} \mu' :$   
 $\mu'(p) = \mu(p) - \phi(p, t) + \phi(t, p), \forall p \in P$



# Зачем это



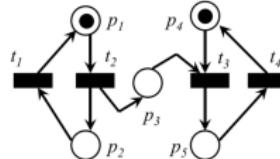
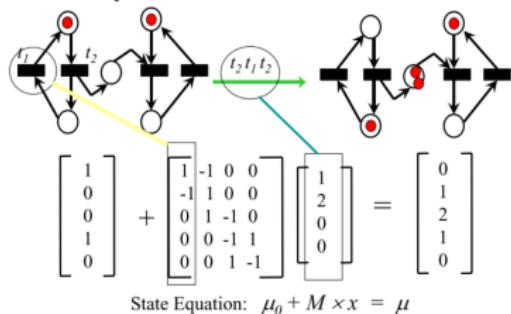
© Murata Tadao. Petri nets: Properties, analysis and applications

# Свойства, которые можно проверить

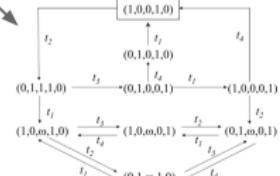
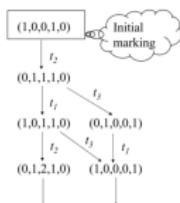
- ▶ Поведенческие свойства:
  - ▶ Достигимость
  - ▶ Ограничность (безопасность)
  - ▶ Живость (L0 - L4)
  - ▶ “Реверсабельность” и “домашнее состояние”
  - ▶ ...
- ▶ Структурные свойства
  - ▶ Структурная живость
  - ▶ Полная контролируемость
  - ▶ Структурная ограниченность
  - ▶ ...

# Способы анализа

## ► Алгебраический

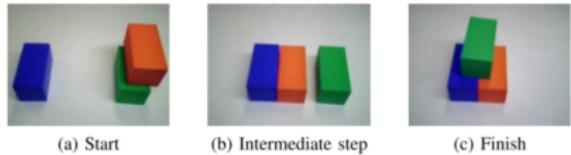


- Структурный
- Редукцией
- Пространства состояний

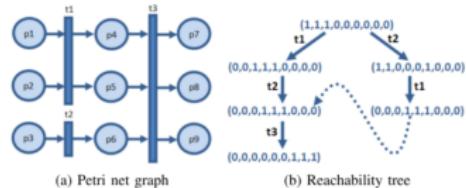


## Пример использования

- ▶ Сеть Петри как форма представления знаний, генерируется автоматически по демонстрации
  - ▶ Поиск по дереву достижимости для определения пути решения задачи или подзадачи
    - ▶ “как программисту вскипятить чайник, если в нём уже есть вода?”



(c) Finish



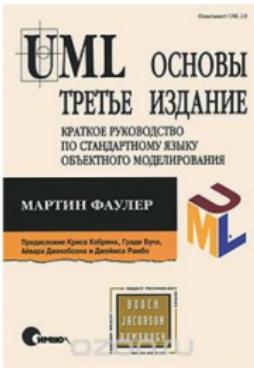
(a) Petri net graph

(b) Reachability tree

© Robot Task Learning from Demonstration

## Using Petri Nets

# Книжка



М. Фаулер, UML. Основы. Краткое руководство по стандартному языку объектного моделирования. СПб., Символ-Плюс, 2011. 192 С.