Data Context Interactions

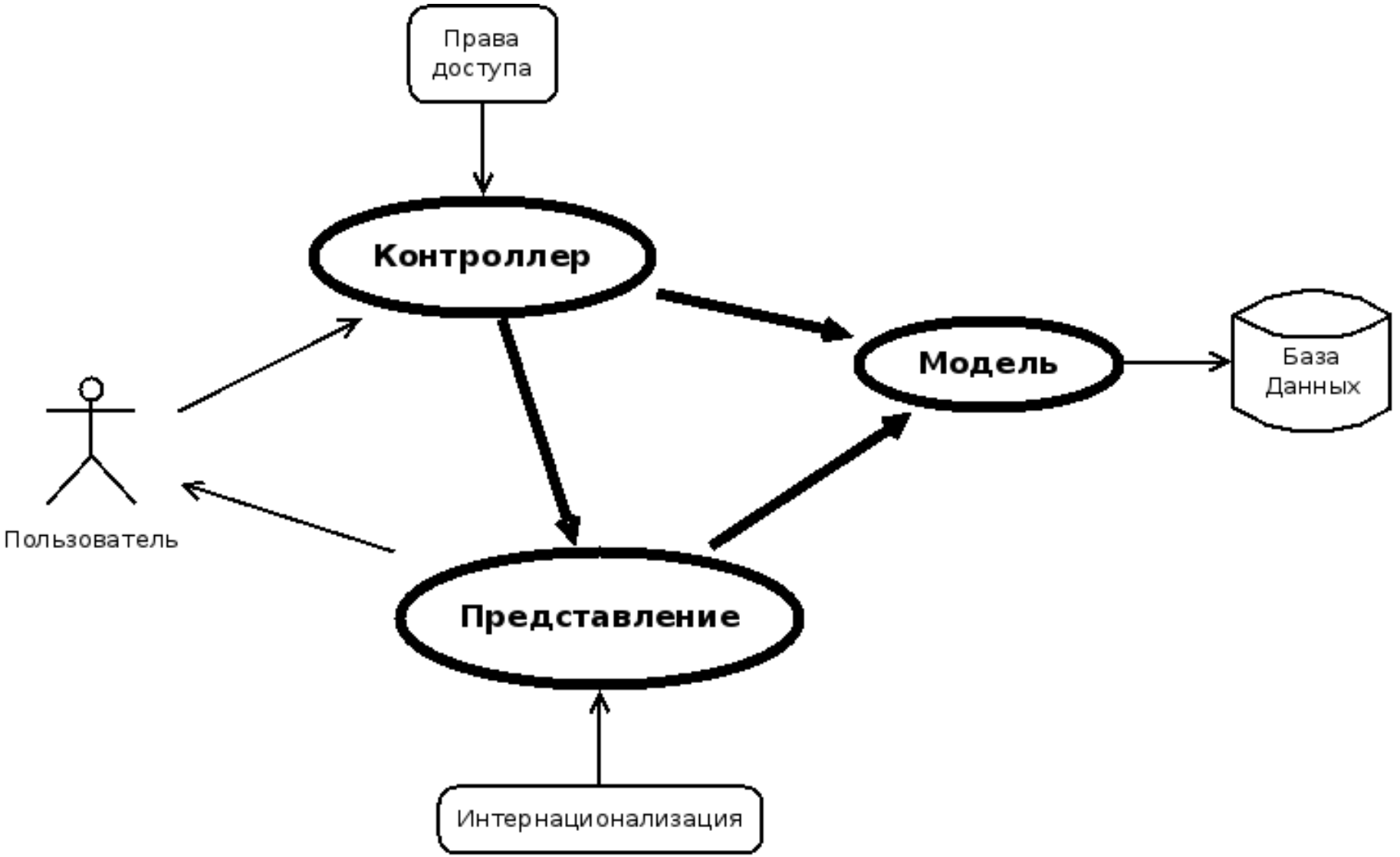
РХТУ им. Д.И. Менделеева
каф. ИКТ

Глеб Аверчук
7.06.2012

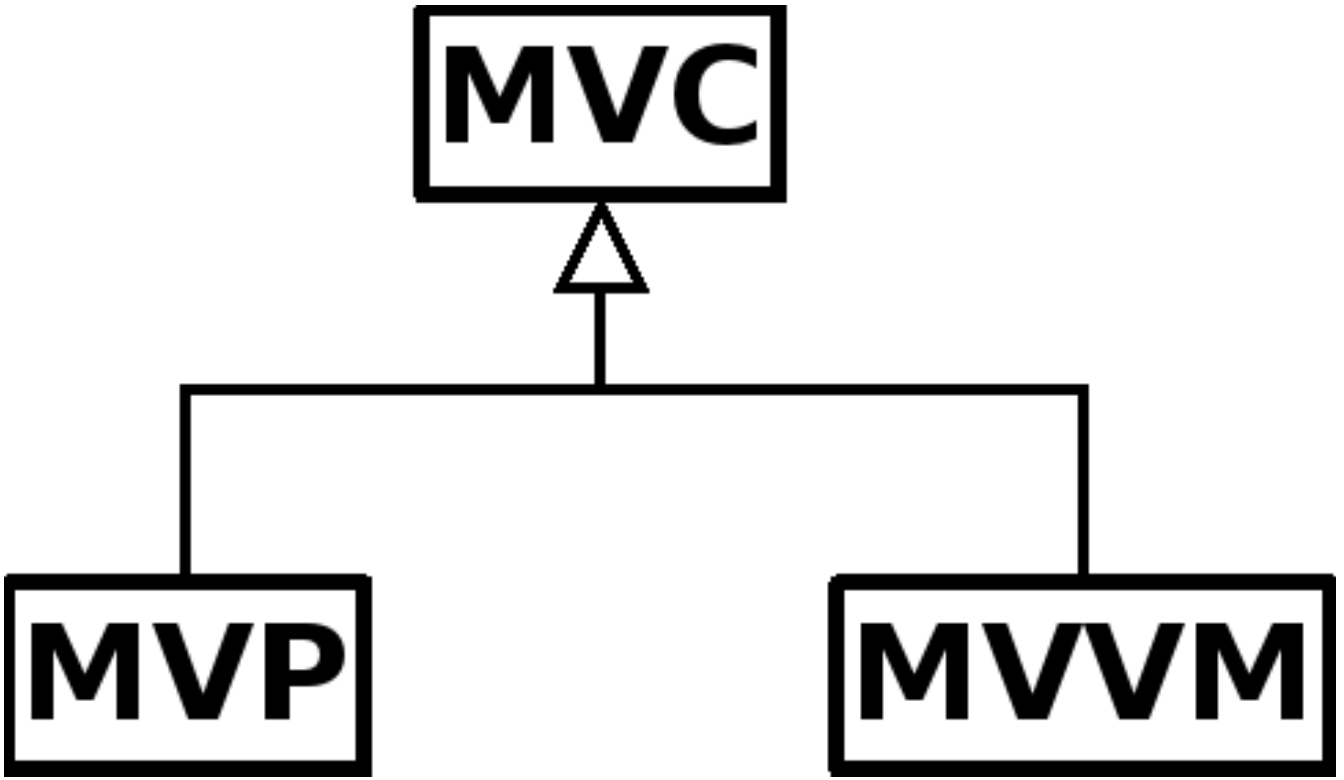
Трюгве Реенскауг

Профессор компьютерных наук университета Осло

Model-View-Controller



Родственники **MVC**



Model
View
Presenter

Model
View
View**M**odel

Плюсы и минусы **MVC**

+ «Мухи отдельно от котлет»
+ Можно писать отдельные тесты для каждой компоненты
+ Этим просто управлять посвящённому человеку
– *Что* система делает и то, *как* она это делает —разные вещи
– Логика работы приложения «размазана» по всему коду

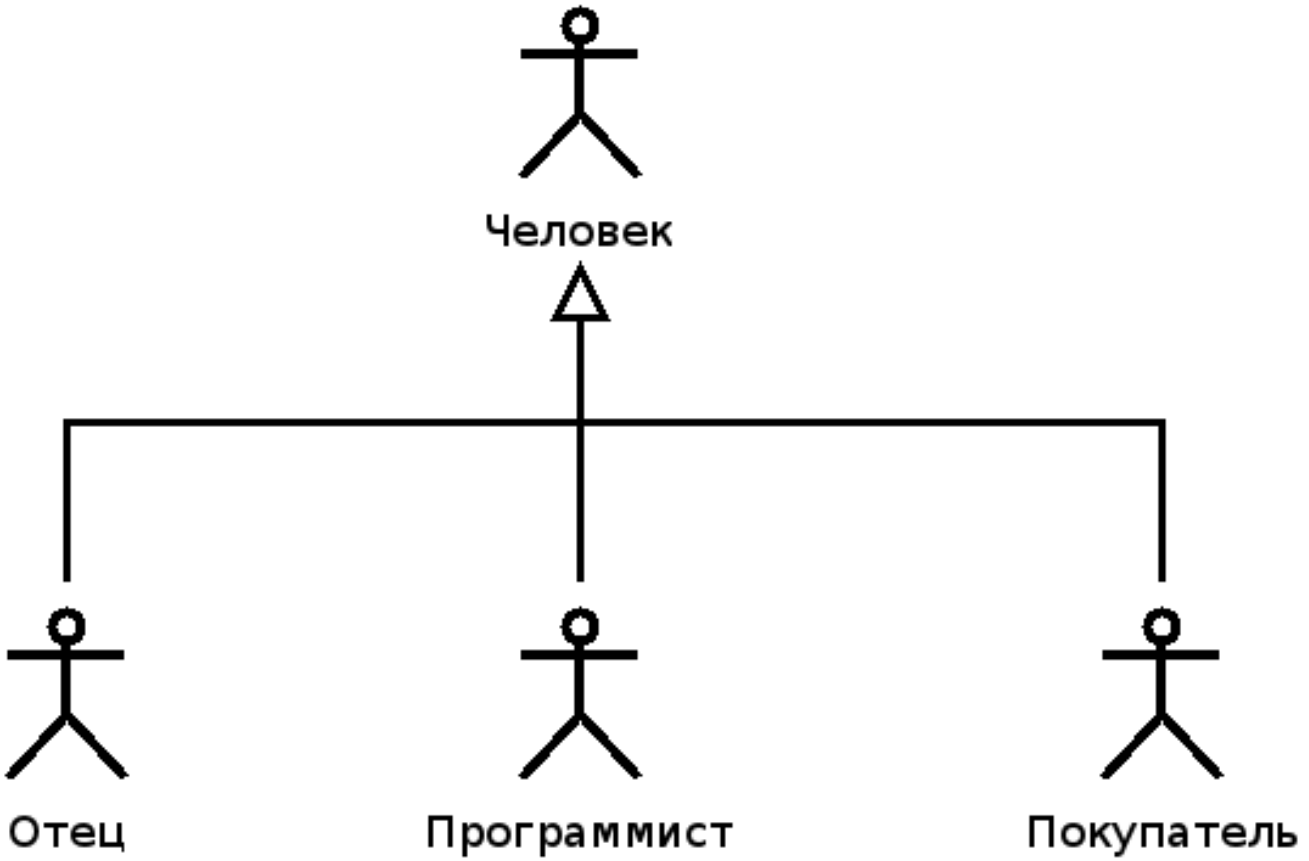
«Существует два способа построения дизайна программного обеспечения:

Один из способов сделать его настолько простым, что будет очевидно: недостатков нет,

а другой — сделать его настолько сложным, что не будет никаких очевидных недостатков.»

Тони Хоар,
1991 год

Проблема 1



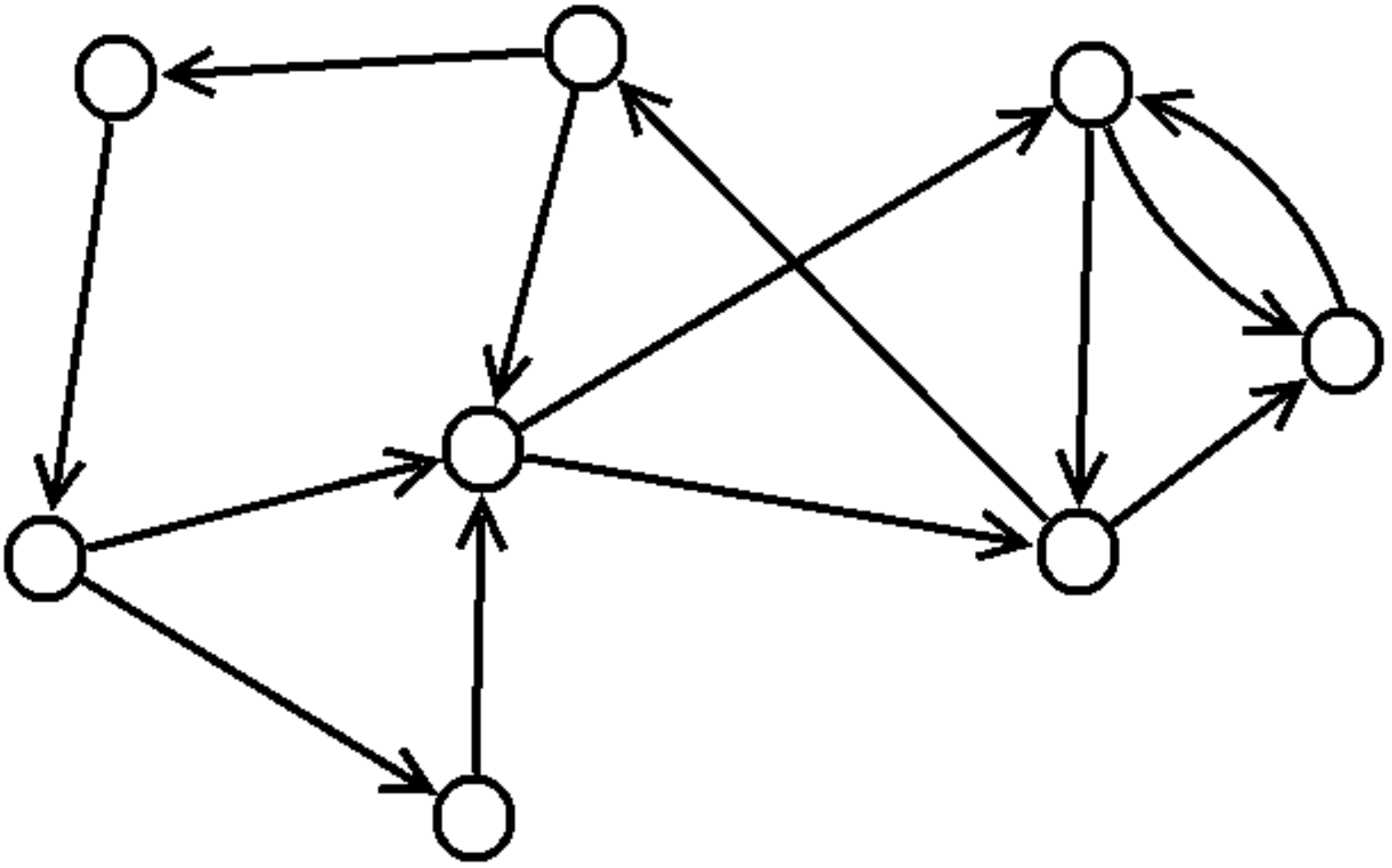
Наследование?

Проблема 2

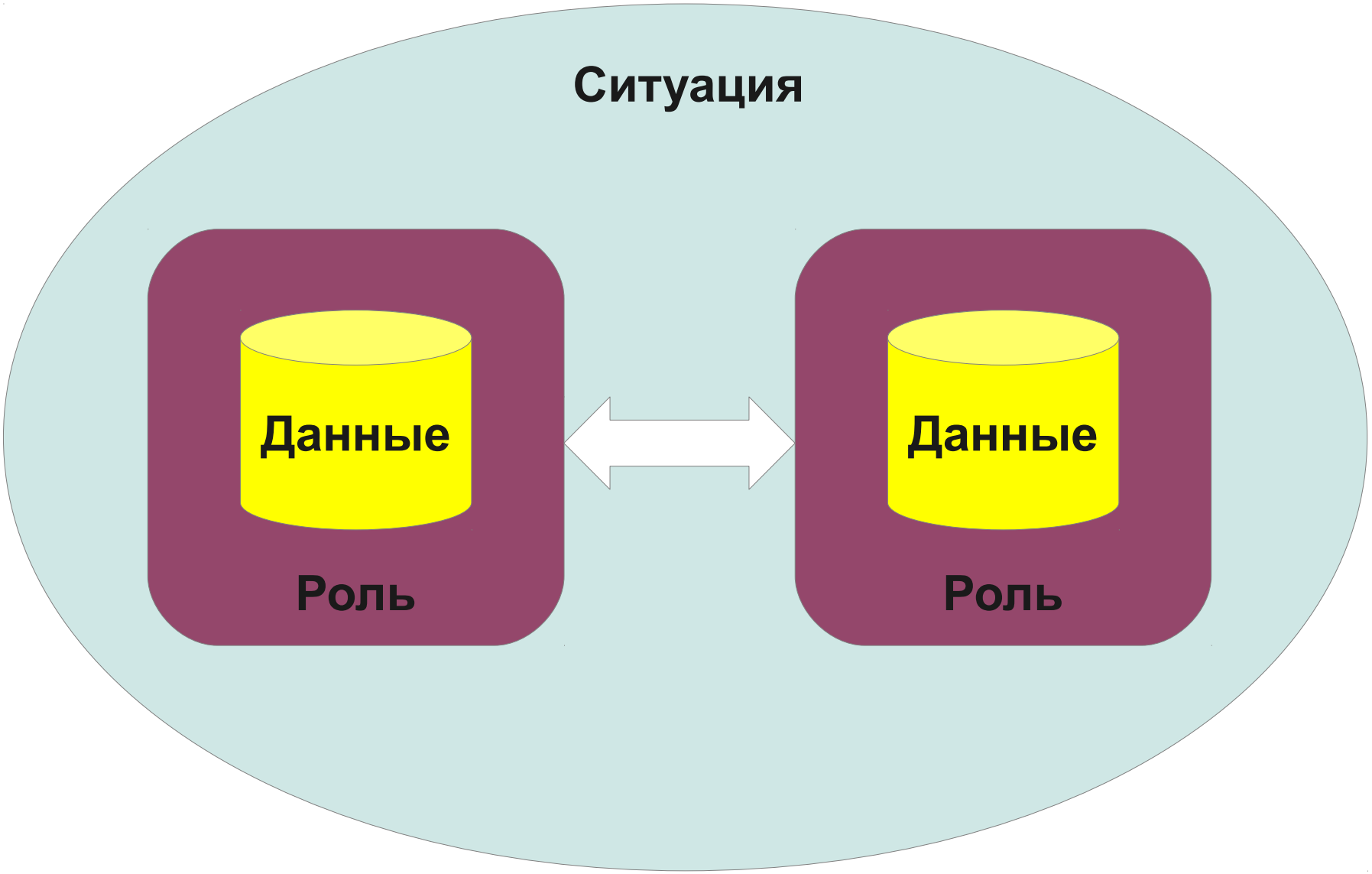
Добавить товар?

Добавить в корзину?

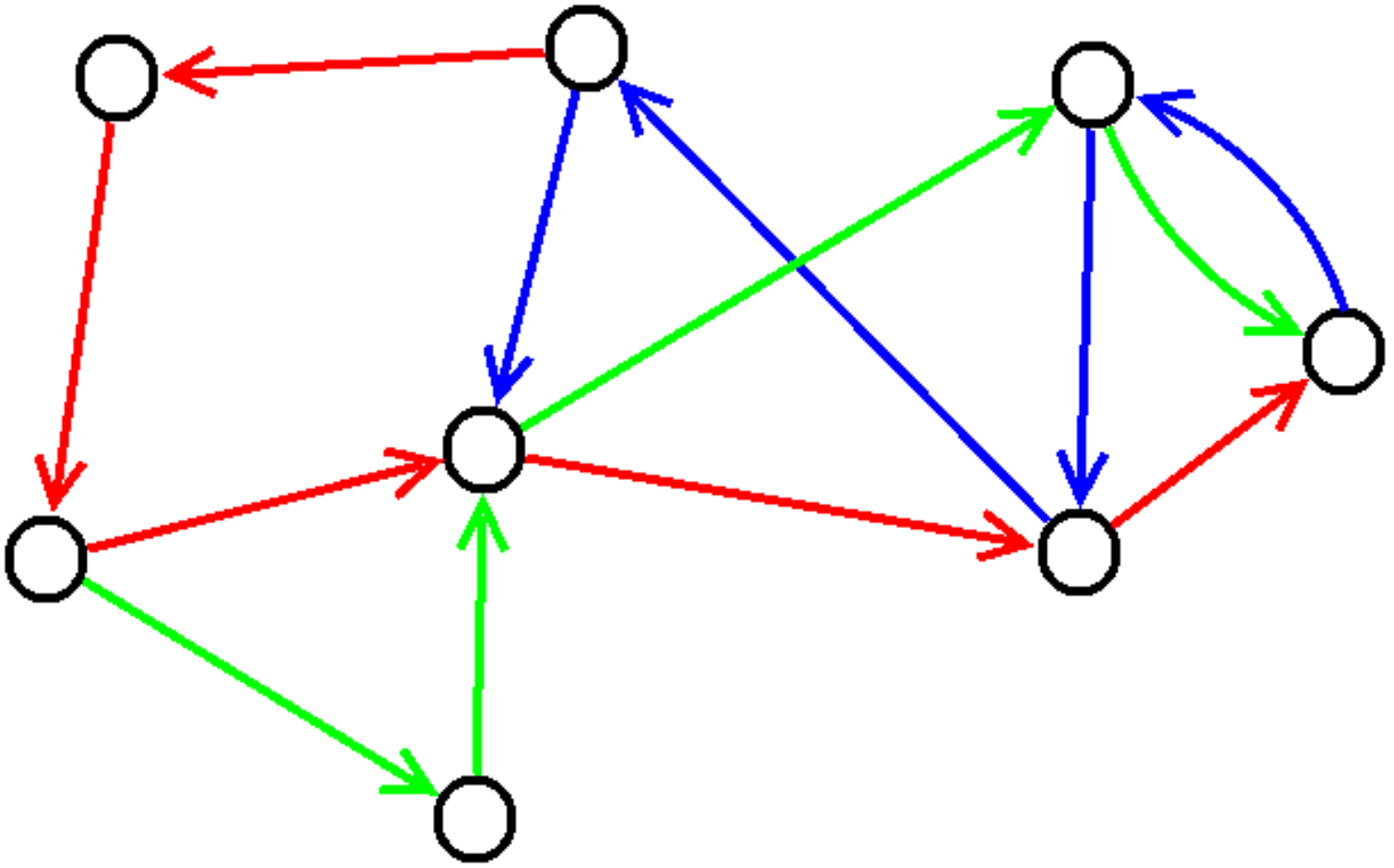
Проблема 3



Data Context Interactions



Упрощение понимания работы системы



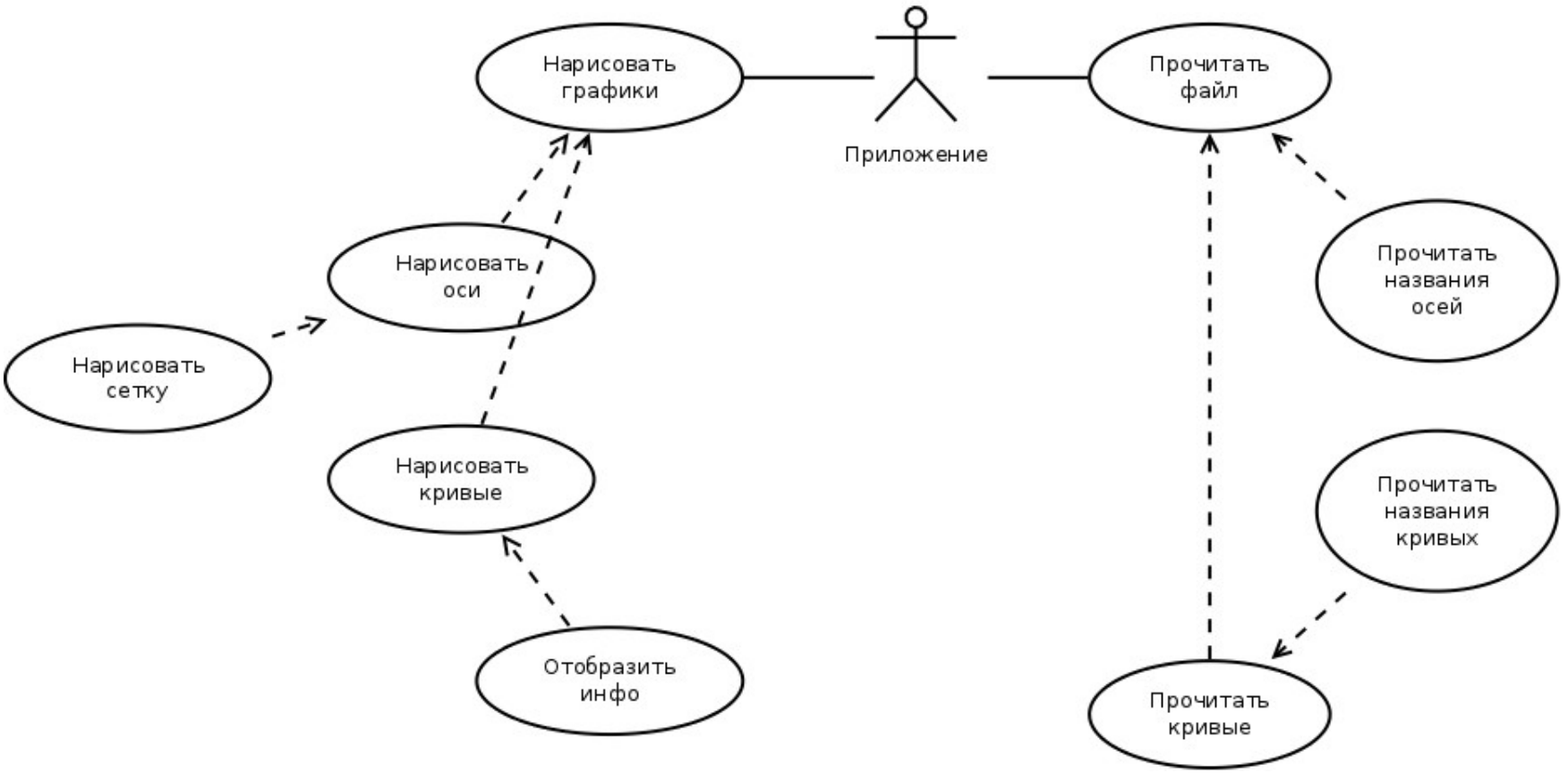
Идея парадигмы **DCI**

«Разделить код, который описывает состояние системы, от кода, который описывает поведение системы.»

Трюгве Реенскауг, 2008 год

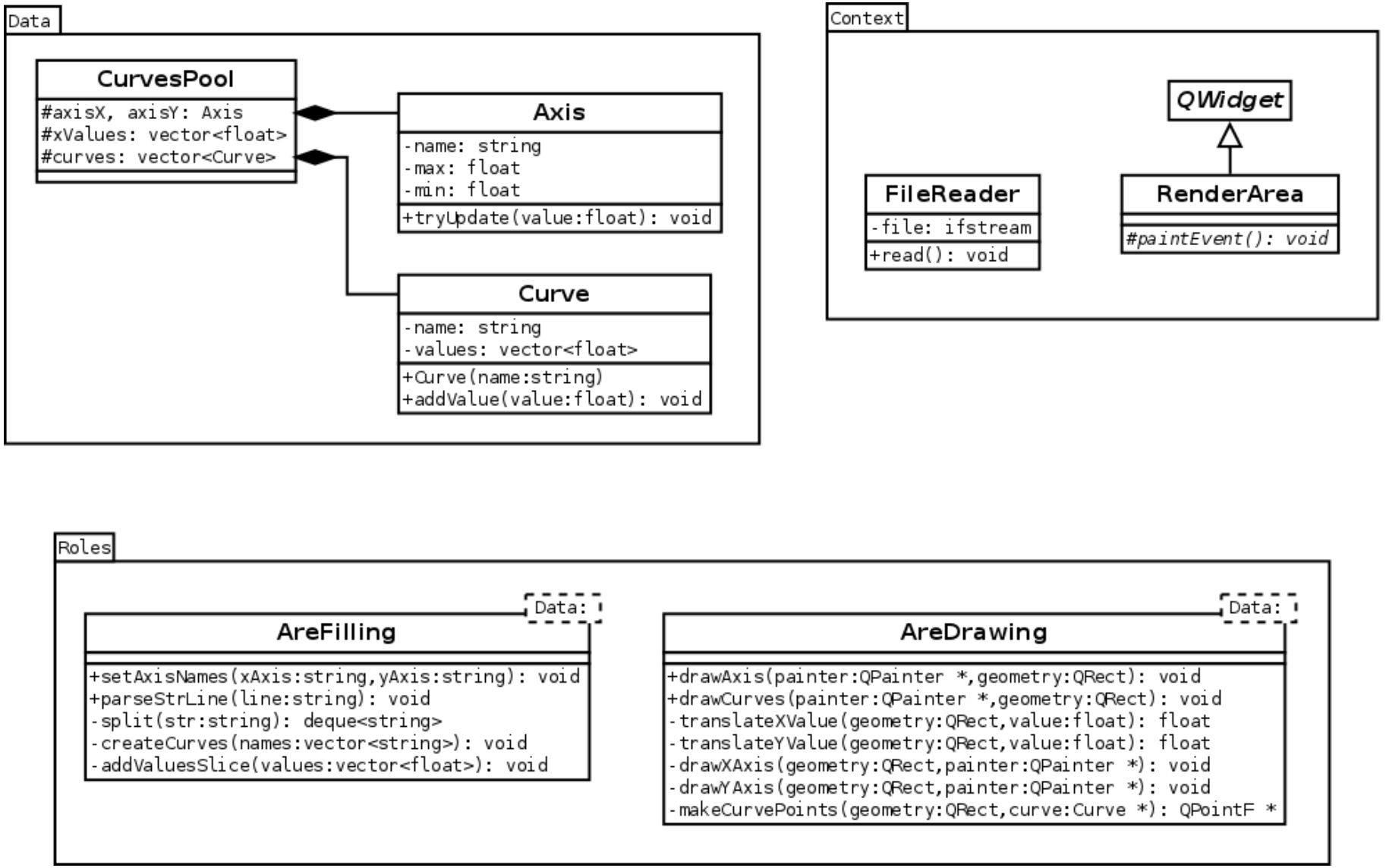
Проектирование

(use-cases)



Проектирование

(classes)



Реализация на C++

(Data)

```
class CurvesPool
{
public:
    CurvesPool();

protected:
    Axis _axisX, _axisY;
    std::vector<Curve> _curves;
    std::vector<float> _xValues;
};
```

Реализация на C++

(Role)

```
template <class Data>
class AreFilling : public Data
{
public:
    AreFilling() {}

    void parseStrLine(const std::string &line);

private:
    std::deque<std::string> split(const std::string &str) const;

    void setAxisNames(const std::string &xAxisName,
                      const std::string &yAxisName);
    void createCurves(const std::deque<std::string> &names);
    void addValuesSlice(const std::deque<float> &values);
    void addValuesSlice(const std::string &line);

    void pushXValue(float value);
    void pushYValue(Curve *curve, float value);
};
```

Реализация на C++

(Context)

```
FileReader::FileReader(const char *filename) : _file(filename) {
    if (!_file) {
        std::cerr << "Ошибка чтения файла "
                  << filename << std::endl;
    }
}

void FileReader::read(CurvesPool *curvesPool) {
    AreFilling<CurvesPool> *fillingPool =
                static_cast<AreFilling<CurvesPool> *>(curvesPool);

    std::string line;
    while (getline(_file, line)) {
        fillingPool->parseStrLine(line);
    }
}
```

Реализация на Ruby

(Data, Role)

```
class CurvesPool
  def initialize
    @axis_x, @axis_y = Axis.new, Axis.new
    @curves, @x_values = [], []
  end

  protected
  attr_accessor :axis_x, :axis_y
  attr_accessor :curves, :x_values
end

module AreFilling
  def parse_str_line(line)
    # ...
  end

  private

  # ...
end
```

Реализация на Ruby

(Context)

```
class FileReader
  def initialize(file_name)
    @file = File.open(file_name, 'r')
  end

  def read(curves_pool)
    curves_pool.extend(AreFilling)

    @file.readlines.each do |line|
      curves_pool.parse_str_line(line)
    end
  end
end
```

Языки реализации

- C++
- Ruby
- JavaScript
- C#
- Python
- Java (Qi4J)
- PHP
- Scala

Материалы

- «Основная» статья:
  http://www.artima.com/articles/dci_vision.html

- Мои проекты на github.com про DCI:
  https://github.com/newmen/qt_plots,
  https://github.com/newmen/monte-carlo_techs

- Эта презентация:

  http://newmen.pro/dci.pdf