

# **Математические методы в объектно-ориентированном проектировании**

## **Лекция 12**

### **Понятия исполнительного устройства и очереди в системе массового обслуживания**

**Овчинников П.Е.**

**МГТУ «СТАНКИН»,**

**ст.преподаватель кафедры ИС**

# Информационная система как СМО

Система массового обслуживания (СМО) — система, которая производит обслуживание **поступающих** в неё **требований**

Обслуживание требований в СМО осуществляется **обслуживающими приборами**

В зависимости от наличия **возможности ожидания** поступающими требованиями начала обслуживания СМО подразделяются на:

- **системы с потерями**, в которых требования, не нашедшие в момент поступления ни одного свободного прибора, теряются
- **системы с ожиданием**, в которых имеется накопитель бесконечной ёмкости для буферизации поступивших требований, при этом ожидающие требования образуют **очередь**
- **системы с накопителем** конечной ёмкости (ожиданием и ограничениями), в которых длина очереди не может превышать ёмкости накопителя; при этом требование, поступающее в переполненную СМО (отсутствуют свободные места для ожидания), теряются

# Информационная система как СМО

**Выбор** требования **из очереди** на обслуживание производится с помощью так называемой дисциплины обслуживания, например:

- FCFS/FIFO (пришедший первым обслуживается первым)
- LCFS/LIFO (пришедший последним обслуживается первым)
- random (случайный выбор)

Основные понятия СМО:

- **требование** (заявка)  
запрос на обслуживание.
- **входящий поток требований**  
совокупность требований, поступающих в СМО
- **время обслуживания**  
период времени, в течение которого обслуживается требование
- **математическая модель**  
совокупность математических выражений, описывающих:
  - входящий поток требований,
  - процесс обслуживания и
  - их взаимосвязь

# Очередь (информатика)

Очередь сообщений (или почтовый ящик) — в информатике — программно-инженерный компонент, используемый для межпроцессного или межпотокowego взаимодействия внутри одного процесса

Для обмена сообщениями используется очередь

Парадигма очереди сообщений сродни шаблону издатель-подписчик и обычно является частью более крупной системы промежуточного программного обеспечения, ориентированной на обработку сообщений

Большинство систем обмена сообщениями в своих API поддерживают модели как очереди сообщений, так и «издатель-подписчик»

Очередь — абстрактный тип данных с дисциплиной доступа к элементам «первый пришёл — первый вышел» (FIFO, англ. *first in, first out*)

Добавление элемента (принято обозначать словом **enqueue** — поставить в очередь) возможно лишь в конец очереди, выборка — только из начала очереди (что принято называть словом **dequeue** — убрать из очереди), при этом выбранный элемент из очереди удаляется

# Очередь и стек (программирование)

Очередь с приоритетом (англ. *priority queue*) — абстрактный тип данных в программировании, поддерживающий две обязательные операции — добавить элемент и извлечь максимум (минимум)

Предполагается, что для каждого элемента можно вычислить его **приоритет** — действительное число или в общем случае элемент линейно упорядоченного множества

В качестве примера очереди с приоритетом можно рассмотреть **список задач работника**. Когда он **заканчивает** одну задачу, он **переходит** к очередной — самой приоритетной (ключ будет величиной, обратной приоритету) — то есть выполняет операцию извлечения максимума

Начальник добавляет **задачи в список**, указывая их приоритет, то есть выполняет операцию добавления элемента

Стек (англ. *stack* — стопка; читается *стэк*) — абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. *last in — first out*, «последним пришёл — первым вышел»)

В цифровом вычислительном комплексе стек называется магазином — по аналогии с магазином в огнестрельном оружии (стрельба начнётся с патрона, заряженного последним)

# Куча (программирование)

Куча (англ. *heap*) — это специализированная структура данных типа дерево, которая удовлетворяет *свойству кучи*:

если  $B$  является узлом-потомком узла  $A$ , то  $\text{ключ}(A) \geq \text{ключ}(B)$ .

Из этого следует, что элемент с наибольшим ключом всегда является корневым узлом кучи, поэтому иногда такие кучи называют *max-кучами* (в качестве альтернативы, если сравнение перевернуть, то наименьший элемент будет всегда корневым узлом, такие кучи называют *min-кучами*).

Не существует никаких ограничений относительно того, сколько узлов-потомков имеет каждый узел кучи, хотя на практике их число обычно не более двух. Куча является максимально эффективной реализацией абстрактного типа данных, который называется очередью с приоритетом.

Кучи имеют решающее значение в некоторых эффективных алгоритмах на графах, таких, как алгоритм Дейкстры на d-кучах и сортировка методом пирамиды.

Структуру данных *куча* не следует путать с понятием куча в динамическом распределении памяти

# Очередь и стек (алгоритмы)

Транспортная задача (задача Монжа — Канторовича) — математическая задача линейного программирования специального вида. Её можно рассматривать как задачу об оптимальном плане перевозок грузов из пунктов отправления в пункты потребления, с минимальными затратами на перевозки. Транспортная задача по теории сложности вычислений входит в класс сложности  $P$ . Когда суммарный объём предложений (грузов, имеющихся в пунктах отправления) не равен общему объёму спроса на товары (грузы), запрашиваемые пунктами потребления, транспортная задача называется **несбалансированной (открытой)**.

Задача о максимальном потоке в сети изучается уже более 60 лет. Интерес к ней подогревается огромной практической значимостью этой проблемы.

Методы решения задачи применяются на транспортных, коммуникационных, электрических сетях, при моделировании различных процессов физики и химии, в некоторых операциях над матрицами, для решения родственных задач теории графов.

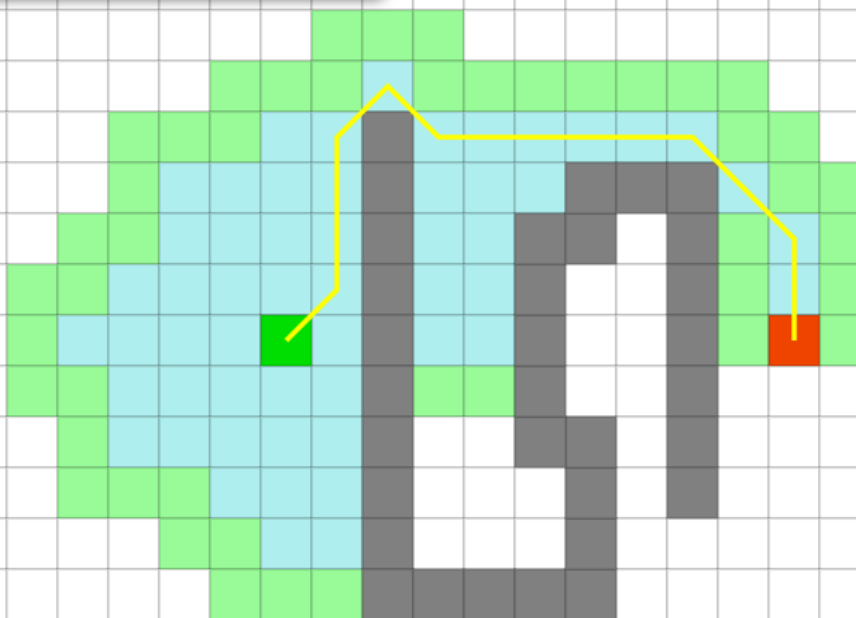
Решение транспортной задачи при помощи генетического алгоритма как часть SOA.

# Критический путь: алгоритмы

## Instructions

hide

Click within the white grid and drag your mouse to draw obstacles.  
Drag the **green** node to set the start position.  
Drag the **red** node to set the end position.  
Choose an algorithm from the right-hand panel.  
Click Start Search in the lower-right corner to start the animation.



length: 17.07  
time: 14.7500ms  
operations: 157



# Модно: Agile

Гибкая методология разработки ([англ. Agile software development, agile-методы](#)) — серия подходов к [разработке программного обеспечения](#), ориентированных на использование [итеративной](#) разработки, **динамическое формирование требований** и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля

Существует несколько методик, относящихся к классу гибких методологий разработки, в частности [экстремальное программирование](#), [DSDM](#), [Scrum](#), [FDD](#).

- ☐ **Люди и взаимодействия** важнее чем процессы и инструменты
- ☐ **Работающий код** важнее совершенной документации
- ☐ **Сотрудничество с заказчиком** важнее контрактных обязательств
- ☐ **Реакция на изменения** важнее следования плану

# Методология ХР

**Экстремальное программирование** ([англ. Extreme Programming, XP](#)) — одна из [гибких методологий разработки программного обеспечения](#).

**Короткий цикл обратной связи** (Fine-scale feedback)

- [Разработка через тестирование](#) (Test-driven development)
- **Игра в планирование (Planning game)**
- ~~Заказчик всегда рядом (Whole team, Onsite customer)~~
- [Парное программирование](#) (Pair programming)

**Непрерывный, а не пакетный процесс**

- [Непрерывная интеграция](#) (Continuous integration)
- [Рефакторинг](#) (Design improvement, Refactoring)
- Частые небольшие релизы (Small releases)

**Понимание, разделяемое всеми**

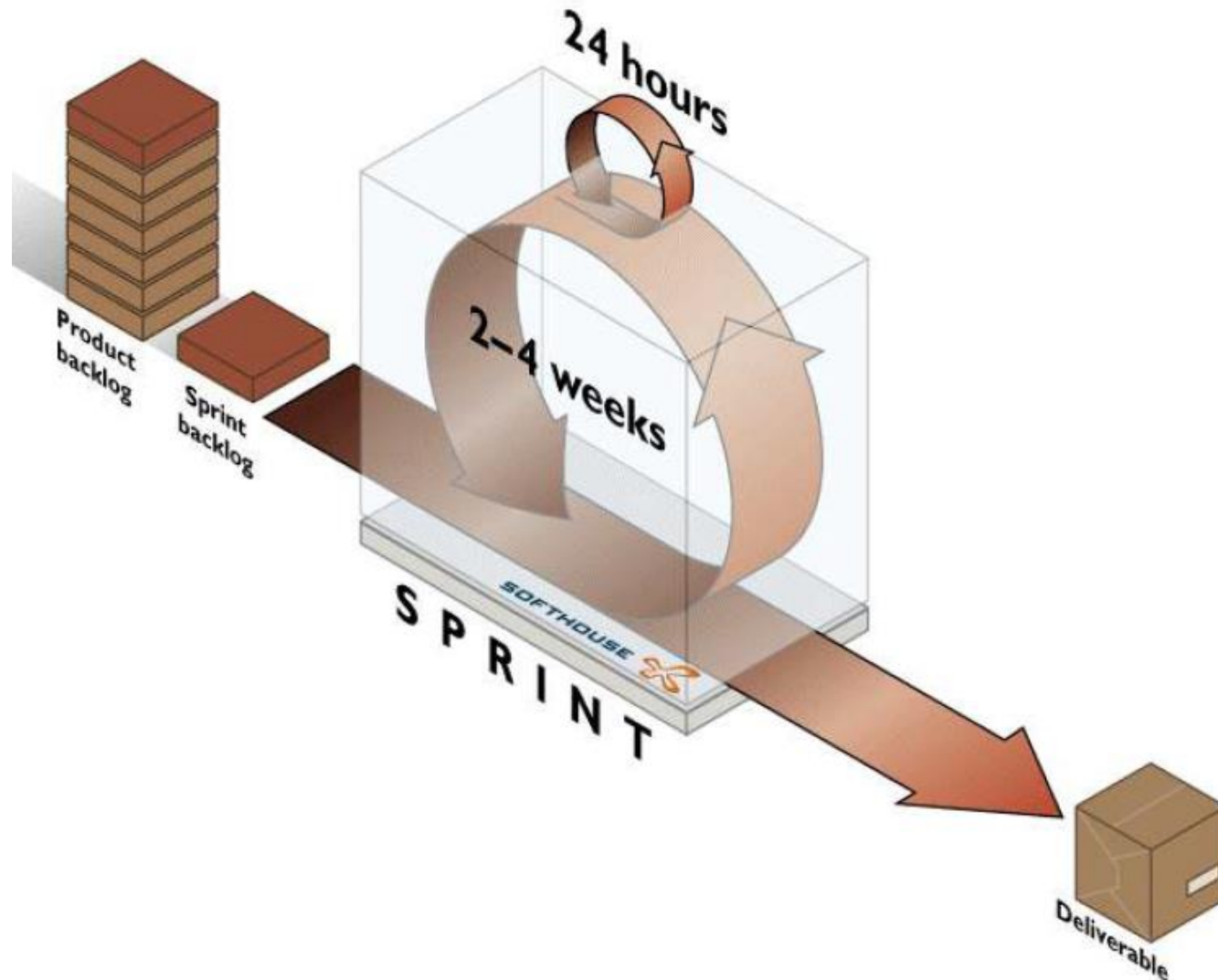
- Простота проектирования (Simple design)
- Метафора системы
- Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership)
- [Стандарт оформления кода](#) (Coding standard or Coding conventions)

**Социальная защищённость программиста** (Programmer welfare):

- 40-часовая рабочая неделя (Sustainable pace, Forty-hour week)

# Модно: Scrum

**Scrum** ([/skrʌm/](#); [англ.](#) *scrum* «схватка») — методология [гибкой разработки](#) ПО. Методология делает акцент на **качественном контроле** процесса разработки.



# Модно: Scrum

**Спринт** — **итерация** в скраме, в ходе которой создается инкремент бизнес-продукта. **Жестко фиксирован по времени**. Длительность одного спринта от 1 до 4 недель

Возможности к реализации в очередном спринте определяются в начале спринта на совещании **Sprint Planning Meeting** планирования методом **Planning Poker** и **не могут изменяться** на всем его протяжении. При этом строго фиксированная небольшая длительность спринта придает процессу разработки предсказуемость и гибкость

Чем короче спринт, тем более гибким является процесс разработки, **релизы** выходят чаще, быстрее поступают отзывы от потребителя, меньше времени тратится на работу в неправильном направлении

С другой стороны, при более длительных спринтах скрам-команда уменьшает издержки на совещания, демонстрации продукта и т. п.

Для оценки объема работ в спринте можно использовать предварительную оценку, измеряемую в **очках истории**. Предварительная **оценка длины** спринта фиксируется в бэклоге проекта.

# Модно: Scrum

**Журнал пожеланий проекта** ([англ. Project backlog](#))

это **список требований** к функциональности, упорядоченный по их степени важности, подлежащих реализации

Элементы этого списка называются [пользовательскими историями](#) (*user story*) или элементами беклога (*backlog items*).

Журнал пожеланий проекта **открыт** для редактирования **для всех участников** скрам-процесса. Project backlog ведется SCRUM Product Owner

**Журнал пожеланий спринта** ([англ. Sprint backlog](#))

содержит функциональность, выбранную владельцем продукта из журнала пожеланий проекта

Все **функции разбиты по задачам**, каждая из которых **оценивается** скрам-командой.

На **Sprint Planning Meeting** команда оценивает объем работы, который нужно сделать для завершения спринта методом Planning Poker

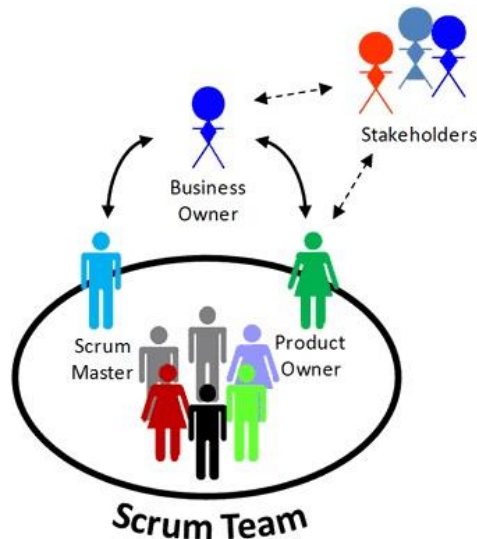
# Модно: Scrum

## Задачи истории спринта (Sprint Story Tasks)

Добавляются к историям спринта. Выполнение каждой задачи оценивается в часах. Каждая задача не должна превышать 12 часов (зачастую команда настаивает, чтобы максимальная продолжительность задачи равнялась одному рабочему дню)

## Ежедневное стоячее SCRUM-совещание (Daily SCRUM)

- начинается в одно и то же время в одном месте
- все могут наблюдать, но только «свиньи» говорят
- в митинге участвуют SCRUM Master, SCRUM Product Owner и SCRUM Team
- **длится ровно 15 минут**
- все участники во время Daily SCRUM стоят (митинг в формате Daily Standup)



# Модно: Scrum

## Ежедневное стоячее SCRUM-совещание (Daily SCRUM)

SCRUM-мастер задает каждому члену SCRUM-команды три вопроса:

- что я **сделал** с момента прошлой встречи для того, чтобы помочь команде разработки достигнуть цели спринта?
- что я **сделаю** сегодня для того, чтобы помочь команде разработки достичь цели спринта?
- вижу ли я **препятствия** для себя или команды разработки, которые могли бы затруднить достижение цели спринта?

Над решением этих проблем методом фасилитации работает скрам-мастер. Обычно это решение проходит за рамками ежедневного совещания и в составе лиц, непосредственно затронутых данным препятствием



# Модно: Scrum

## Обзор итогов спринта (Sprint review meeting)

Проводится **в конце спринта:**

- Команда демонстрирует прирост инкремента продукта всем заинтересованным лицам.
- Все члены команды участвуют в демонстрации (один человек на демонстрацию или каждый показывает, что сделал за спринт).
- Нельзя демонстрировать незавершенную функциональность.
- Ограничена четырьмя часами в зависимости от продолжительности итерации и прироста функциональности продукта.

## Грумминг беклога (Grooming)

Беклог отправляется в парикмахерскую для того, чтобы скрам-команда и владелец продукта могли:

- Добавить, убрать или разбить элементы беклога продукта (PBI).
- Уточнить или дать новые оценки.
- Изменить порядок следования элементов беклога продукта.
- Обсудить и прояснить требования.



# Модно: Scrum

## Ретроспективное совещание (Retrospective meeting)

Проводится в конце спринта.

Члены скрам-команды, скрам-мастер и продукт-оунер высказывают свое мнение о прошедшем спринте.

Скрам-мастер задает два вопроса всем членам команды:

**Что было сделано хорошо в прошедшем спринте?**

**Что надо улучшить в следующем?**

Выполняют улучшение процесса разработки (обсуждают варианты решения проблем, фиксируют удачные решения и вызвавшего владельца продукта).

Ограничена четырьмя часами для спринта любой длины.

# Модно: CI

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

В обычном проекте, где над разными частями системы разработчики трудятся независимо, стадия интеграции является заключительной. Она может непредсказуемо задержать окончание работ

Переход к непрерывной интеграции позволяет снизить трудоёмкость интеграции и сделать её более предсказуемой за счёт наиболее раннего обнаружения и устранения ошибок и противоречий, но основным преимуществом является сокращение **стоимости исправления дефекта**, за счёт **раннего его выявления**

# Модно: CI

**Сборки по расписанию** ([англ. daily build](#) — ежедневная сборка), как правило, проводятся в нерабочее время, ночью ([англ. nightly build](#)), планируются таким образом, чтобы **к началу очередного рабочего дня** были готовы **результаты тестирования**

Для различия дополнительно вводится система нумерации сборок — обычно, **каждая сборка нумеруется** натуральным числом, которое **увеличивается с каждой новой сборкой**

Исходные тексты и другие исходные данные при взятии их из репозитория (хранилища) системы контроля версий помечаются **номером сборки**

Благодаря этому, точно такая же сборка может быть **точно воспроизведена в будущем** — достаточно взять исходные данные по нужной метке и запустить процесс снова. Это даёт возможность повторно выпускать даже очень старые версии программы с небольшими исправлениями.

О программе Microsoft Word



Microsoft® Word 2010 (14.0.7228.5000) SP2 MSO (14.0.7229.5000)

Включено в Microsoft Office стандартный 2010

© Корпорация Майкрософт (Microsoft Corporation), 2010. Все права защищены.

# Модно: инструменты DevOps

**DevOps** — это командная работа сотрудников, занимающимися **разработкой**, **операциями** и **тестированием**, а также необходимый набор 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

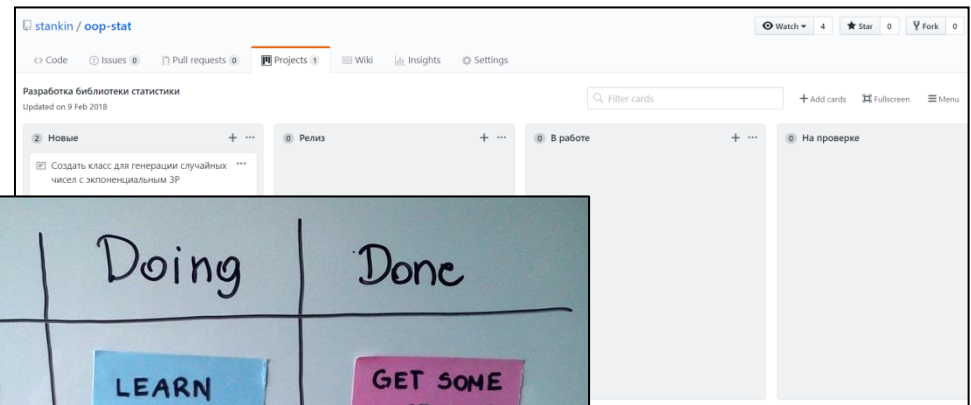
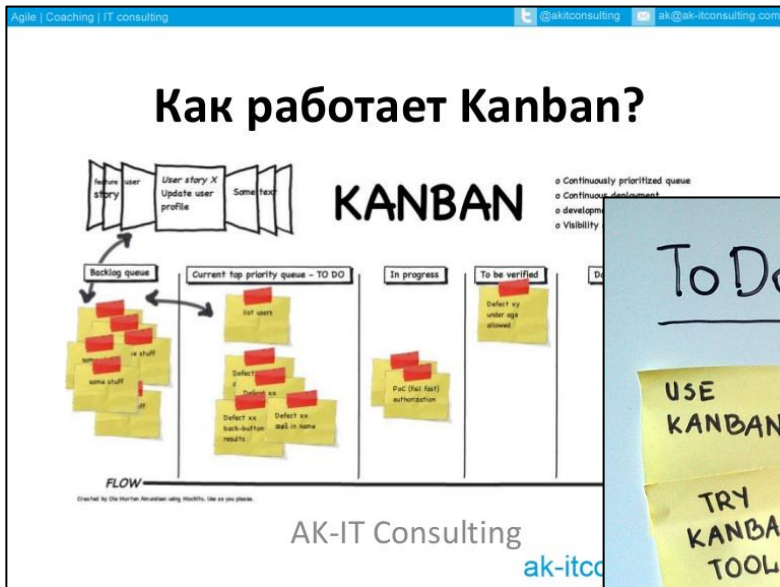
**Monitor** — мониторинг производительности приложений, опыт работы с конечным пользователем

# Модно: Канбан

**Канбан** (от [яп.](#) 看板 «рекламный щит, вывеска») — метод управления [разработкой](#), реализующий принцип «[точно в срок](#)» и способствующий **равномерному распределению нагрузки** между работниками.

При данном подходе весь процесс разработки прозрачен для всех членов команды. Задачи по мере поступления **заносятся в отдельный список**, откуда каждый разработчик может **извлечь требуемую задачу**.

Канбан — наглядная система разработки, показывающая, что необходимо производить, когда и сколько. Метод основан на [одноименном методе](#) в [производственной системе «Тойоты»](#) и [бережливом производстве](#).



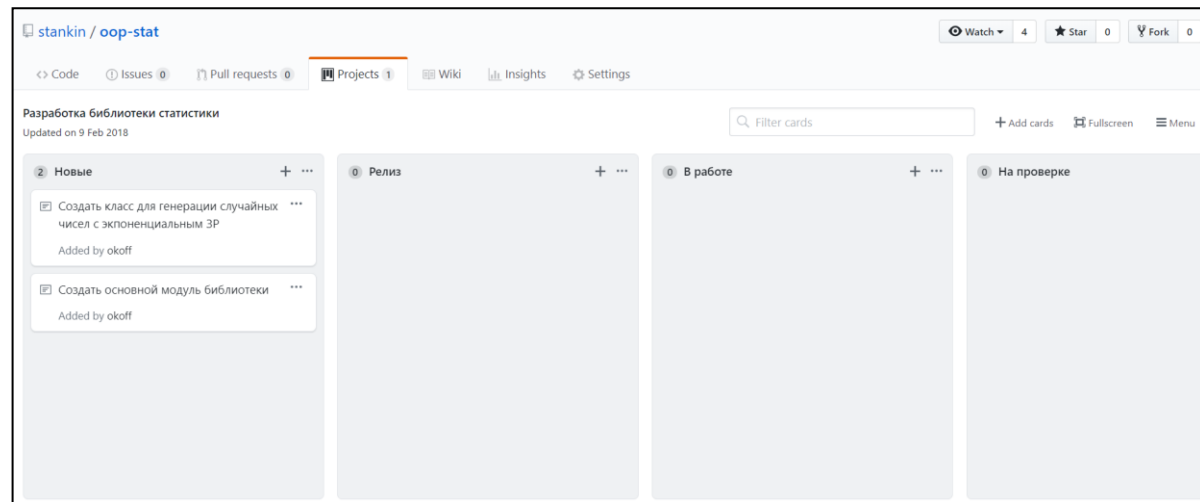
[Канбан \(2005-2007\)](#)  
[Что такое канбан](#)

# Полезно: репозиторий

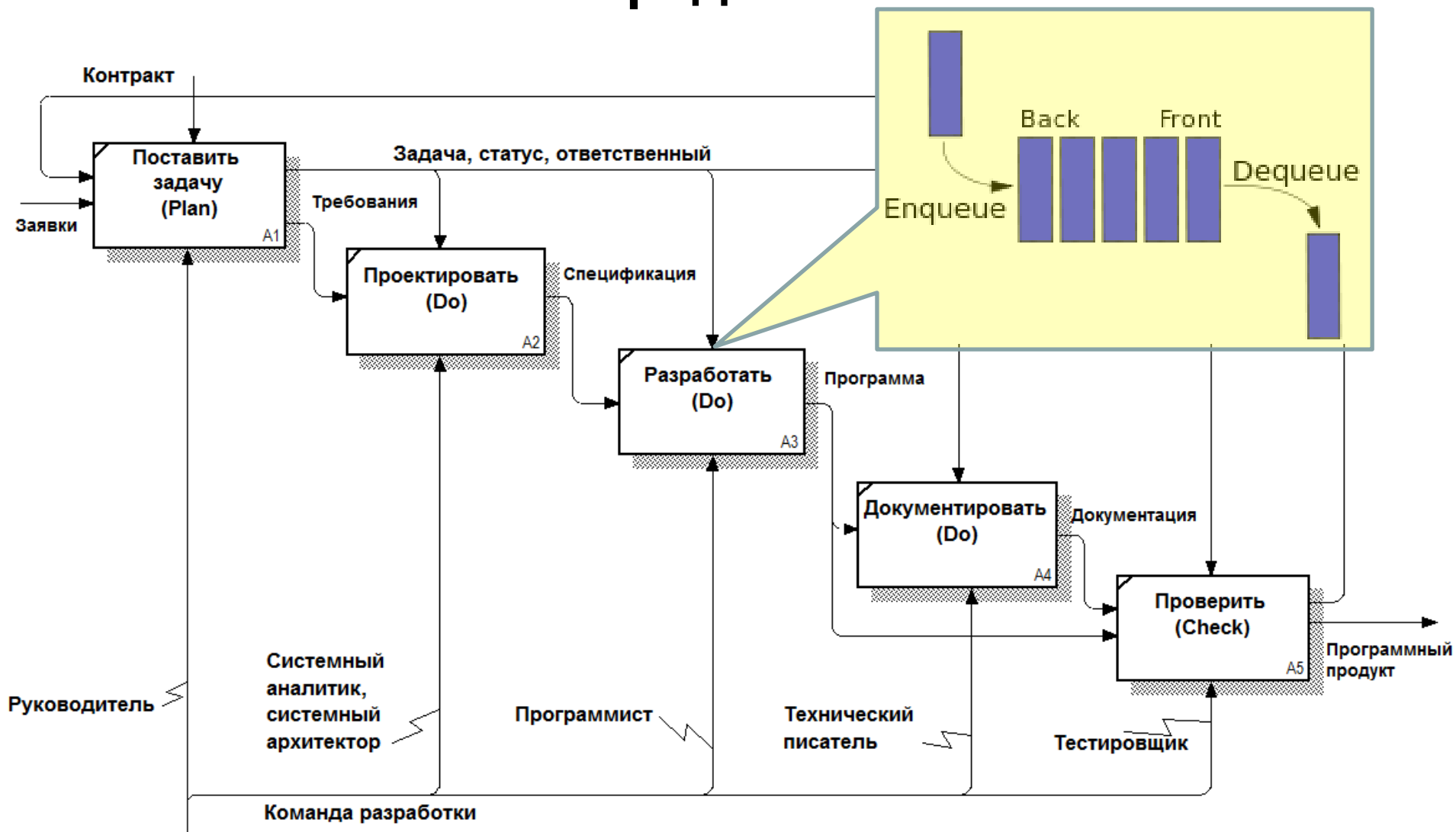
**Репозиторий, хранилище** — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по [сети](#)

Репозитории используются в [системах управления версиями](#), в них хранятся все документы вместе с историей их изменения и другой служебной информацией

**Система управления версиями** (от [англ.](#) *Version Control System*, *VCS* или *Revision Control System*) — [программное обеспечение](#) для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое



# Очереди в FDD



# Очереди в веб-разработке: AJAX

**AJAX**, Ajax (['eɪdʒæks](#), от [англ.](#) *Asynchronous Javascript and XML* — «асинхронный [JavaScript](#) и [XML](#)») — подход к построению интерактивных [пользовательских интерфейсов веб-приложений](#), заключающийся в «фоновом» [обмене данными браузера](#) с [веб-сервером](#)

В результате, при обновлении данных [веб-страница](#) не перезагружается полностью, и веб-приложения становятся быстрее и удобнее. При использовании AJAX:

- Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент
- [Скрипт](#) (на языке [JavaScript](#)) определяет, какая информация необходима для обновления страницы
- [Браузер](#) отправляет соответствующий запрос на [сервер](#)
- Сервер возвращает только ту часть документа, на которую пришёл запрос
- [Скрипт](#) вносит изменения с учётом полученной информации (без полной перезагрузки страницы)



# Очереди в веб-разработке: AJAX

AJAX базируется на использовании технологий:

- динамического обращения к [серверу](#) «на лету», без перезагрузки всей страницы полностью, например с использованием [XMLHttpRequest](#) и
- использования [DHTML](#) для динамического изменения содержания страницы

**XMLHttpRequest** (XMLHTTP, XHR) — [API](#), доступный в [скриптовых языках браузеров](#), таких как [JavaScript](#)

Использует запросы [HTTP](#) или [HTTPS](#) напрямую к [веб-серверу](#) и загружает данные ответа сервера напрямую в вызывающий скрипт

Информация может передаваться в любом [текстовом формате](#), например, в [XML](#), [HTML](#) или [JSON](#). Позволяет осуществлять HTTP-запросы к серверу без перезагрузки страницы

# Очереди в веб-разработке: AJAX

Метод	Описание
<code>abort()</code>	Отменяет текущий запрос, удаляет все заголовки, ставит текст ответа сервера в null.
<code>getAllResponseHeaders()</code>	Возвращает полный список HTTP-заголовков в виде строки. Заголовки разделяются знаками переноса (CR+LF). Если флаг ошибки равен true, возвращает пустую строку. Если статус 0 или 1, вызывает ошибку <code>INVALID_STATE_ERR</code> .
<code>getResponseHeader(headerName)</code>	Возвращает значение указанного заголовка. Если флаг ошибки равен true, возвращает null. Если заголовок не найден, возвращает null. Если статус 0 или 1, вызывает ошибку <code>INVALID_STATE_ERR</code> .
<code>open(method, URL, async, userName, password)</code>	Определяет метод, URL и другие опциональные параметры запроса; параметр <code>async</code> определяет, происходит ли работа в асинхронном режиме. Последние два параметра необязательны.
<code>send(content)</code>	Отправляет запрос на сервер.
<code>setRequestHeader(label, value)</code>	Добавляет HTTP-заголовок к запросу.
<code>overrideMimeType(mimeType)</code>	Позволяет указать mime-type документа, если сервер его не передал или передал неправильно.

# Очереди в веб-разработке: AJAX

Свойство	Тип	Описание
onreadystatechange	EventListener	Обработчик события, которое происходит при каждой смене состояния объекта. Имя должно быть записано в нижнем регистре.
readyState	unsigned short	Текущее состояние объекта (0 — не инициализирован, 1 — открыт, 2 — отправка данных, 3 — получение данных и 4 — данные загружены)
responseText	DOMString	Текст ответа на запрос. Если состояние не 3 или 4, возвращает пустую строку.
responseXML	Document	Текст ответа на запрос в виде XML, который затем может быть обработан посредством <a href="#">DOM</a> . Если состояние не 4, возвращает null.
status	unsigned short	HTTP-статус в виде числа ( <a href="#">404</a> — «Not Found», <a href="#">200</a> — «OK» и т. д.)
statusText	DOMString	Статус в виде строки («Not Found», «OK» и т. д.). Если статус не распознан, браузер пользователя должен вызвать ошибку INVALID_STATE_ERR.

# Очереди в веб-разработке: AJAX и JQuery

## jQuery ajax() Method

◀ jQuery AJAX Methods

### Example

Change the text of a <div> element using an AJAX request:

```
$("#button").click(function(){
    $.ajax({url: "demo_test.txt", success: function(result){
        $("#div1").html(result);
    }});
});
```

Try it Yourself »