Программирование

Осипов Никита Алексеевич
naosipov@itmo.ru

Все материалы доступны по ссылке:
https://drive.google.com/drive/folders/12GicINdAM_iMaAf5PCk1r3xmI87ZXkg7?usp=sharing

Учебный план

Направление подготовки: 11.03.02 Инфокоммуникационные технологии и системы связи	
Наименование образовательной программы "Программирование в инфокоммуникационных системах" по ОГНП 6: "Трансляционные информационные технологии"	
Очная форма обучения, срок получения образования - 4 года, год начала подготовки - 2021	
Специализация 1: Прикладное программирование в инфокоммуникационных системах	Русский
Специализация 2: Сетевые и облачные технологии	Русский

Распределение учебного времени

Трудоемкость		Семестр	Вид контроля (экз./диф.зач./зач.)	Занятий лекц. типа, час.	Практич. занятий, час.	СРО, час.
зач. ед.	час.					
4	144	1	Экзамен	16	32	96

8 лекций по 2 часа

8 занятий по 4 часа

Домашнее задание

Распределение учебного времени

Трудоемкость		Семестр	Вид контроля (экз./диф.зач./зач.	Занятий лекц. типа, час.	Лаборат. занятий, час.	Практич. занятий, час.	СРО, час.
зач. ед.	час.						
4	144	2	Экзамен	16	16	16	91,2

8 лекций по 2 часа

4 занятия по 4 часа

4 занятия по 4 часа

Включает выполнение домашнего задания

Лекция 1

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ

- *Вигерс К.* Разработка требований к программному обеспечению/Пер. с англ. – М.: Русская Редакция, 2014. – 736 с.

Руководство по разработке качественных требований к программному обеспечению.

Описаны приемы выявления, формулирования, разработки, проверки, утверждения и тестирования требований, которые помогут создать эффективное ПО.

- *Соммервилл Иан*. Инженерия программного обеспечения/Пер. с англ. – М.: Вильямс. – 736 с.

Введение в инженерию программного обеспечения, описаны все этапы и технологии разработки программных систем.

Является ли выбор языка проблемой?

- Выбирать какой популярнее или тот который более эффективен в решении конкретных задач?
  - Как определить популярность языка?
  - Как определить эффективность применения языка?

Технологии развиваются и меняются очень быстрыми темпами. Поэтому знание того, что будет востребовано сегодня или завтра, является небольшим преимуществом

Является ли выбор языка проблемой?

- Выбирать какой популярнее или тот который более эффективен в решении конкретных задач?
  - Как определить популярность языка?
    - Опросы разработчиков
    - Анализ объявлений о вакансиях, чтобы увидеть, какие навыки ищут работодатели
    - Подсчет количества поисковых запросов
    - Анализ практического применения

Язык программирования

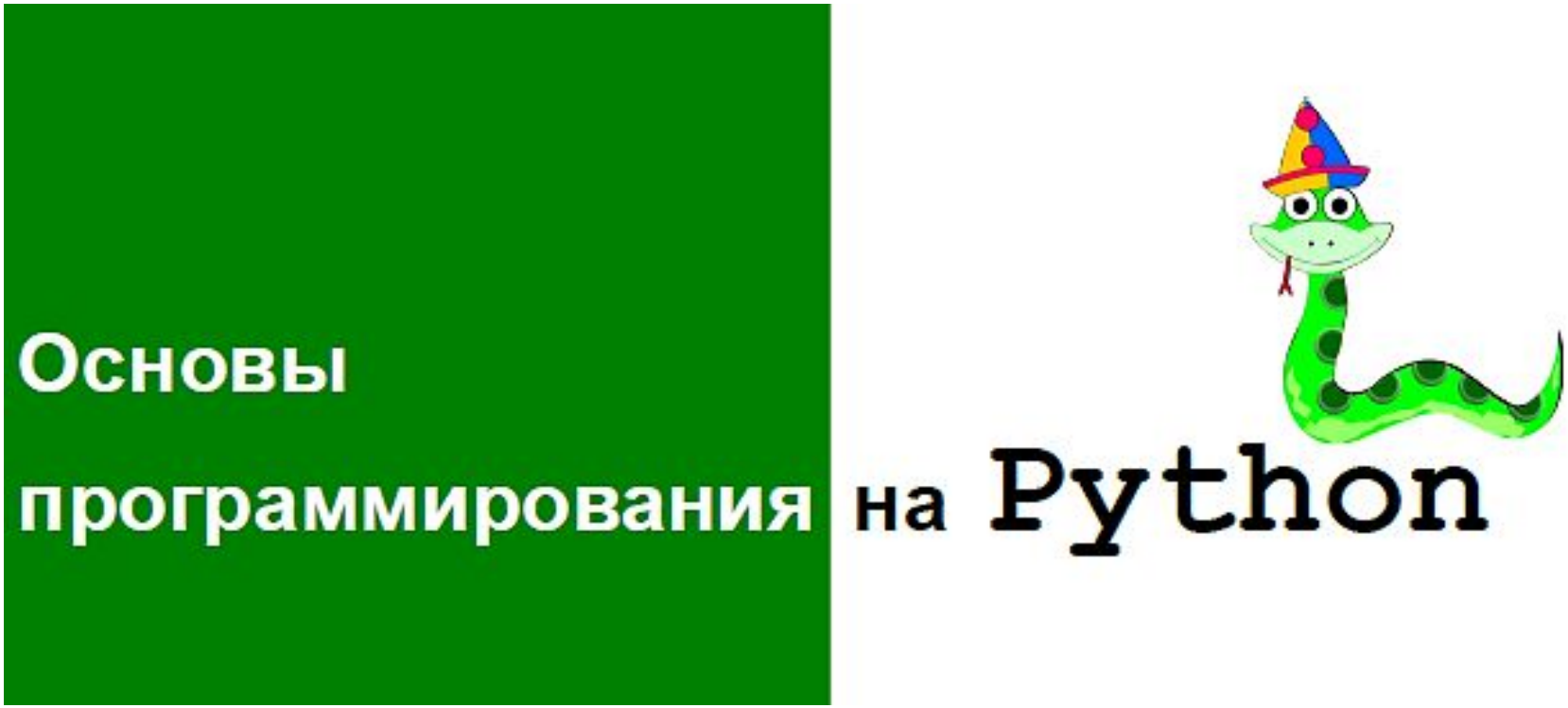
- Какой язык программирования изучать?
- Что значит легкий или трудный язык программирования?
- В каком случае запись кода выглядит проще?

```
a = 5
$a = 5;
var a = 5;
int a = 5;
var a:Integer = 5;
```

- В чем смысл операции?
  - что значит "переменная"?
  - что значит "присваивается"?
  - что такое "значение", "тип"?
  - почему оно 5?
- Эти вопросы от выбора языка не зависят

Наша цель – долгосрочное погружение в программирование

Основная литература



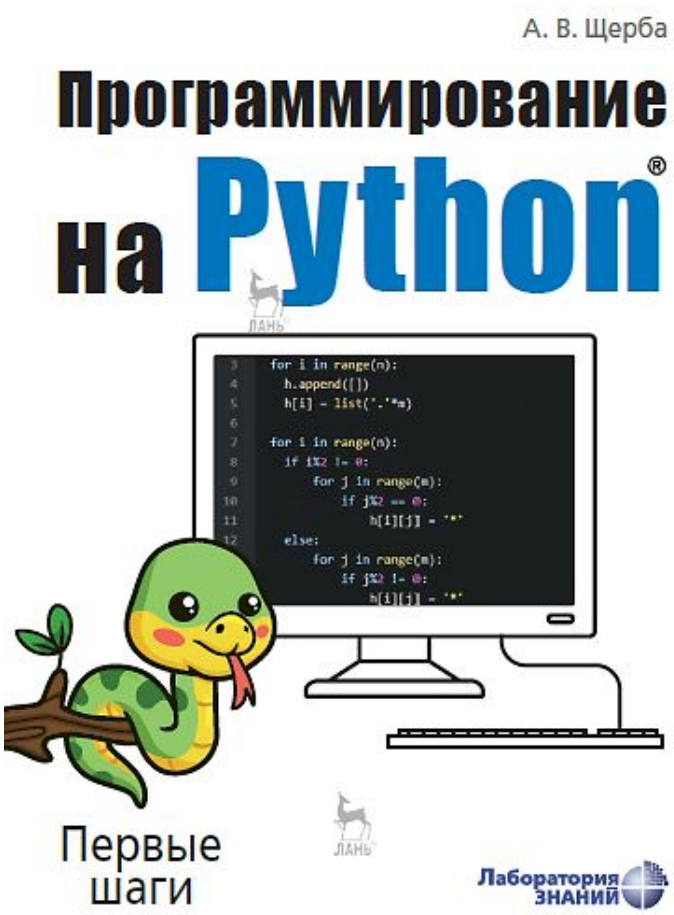
- Основы программирования на Python. Учебник. [Электронный ресурс]. Режим доступа: python_structured_programming.pdf..

Основная литература

- *Златопольский Д. М*. Основы программирования на языке Python. – М.: ДМК Пресс, 2017. – 284 с.: ил. [Электронный ресурс]. Режим доступа: FundamentalsofProgramminginPythonZlatopolsky.pdf..

Основная литература

- *Жуков Р.А.* Язык программирования Python: практикум : учеб, пособие / Р.А. Жуков. — М.: ИНФРА-М, 2019. — 216с. [Электронный ресурс]. Режим доступа: yazyk_programmirovaniya_python_praktikum.pdf

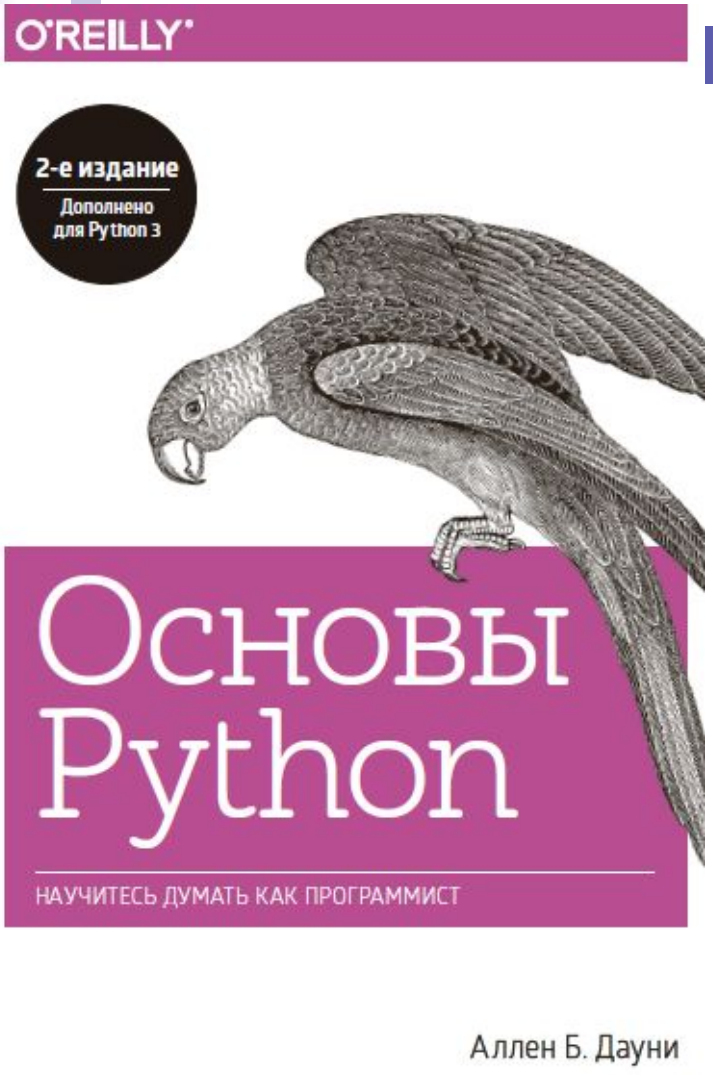


Описаны базовые конструкции программирования на языке Python: от именования переменных до многострочных программ с несколькими вложенными циклами и условными конструкциями.

В каждой главе разбор возможных ошибок и задания с ответами в конце книги.

- *Щерба А.В*. Программирование на Python: Первые шаги / М. : Лаборатория знаний, 2022.—253 с. [Электронный ресурс]. Режим доступа: Щерба_Первые шаги. Python.pdf

Закрепление теории



Это практическое руководство последовательно раскрывает основы программирования на языке Python.

Вы будете продвигаться от самых простых тем к сложным и получите полное представление об одном из самых популярных языков программирования.

А еще вы поймете, как думают программисты, и сможете применять этот подход к решению даже повседневных задач.

- *Дауни Аллен* Основы Python. Научитесь думать как программист / — М.:Манн, Иванов и Фербер, 2021. — 304 с. [Электронный ресурс]. Режим доступа: Основы python_Дауни_Ален.pdf

Закрепление теории

Это руководство поможет шаг за шагом прокачать навыки разработки.
В книге 150 задач от изучения основ языка к решению более сложных задач.

- *Лейси Никола* Python, например. — СПб.: Питер, 2021. — 208 с.:. [Электронный ресурс]. Режим доступа: Лейси_Никола Python_например.pdf

Что в этом коде не так?

```
first_value = 4
second_value = 5
operator = "+"
if operator == "+":
    print(first_value + second_value)
elif operator == "-":
    print(first_value - second_value)
elif operator == "/":
    print(first_value / second_value)
elif operator == "*":
    print(first_value * second_value)
else:
    print("Operator is wrong. Choose from given: + - / *")
```

Иерархия мастерства

Программа и программирование

- **Цель программирования** — разработка **программ** (программного обеспечения) для управления компьютером с целью решения различных информационных задач
  - решение задач управления и планирования

Как воспринимать понятие «программное обеспечение»?
Можно ли его увидеть, пощупать или почувствовать другими органами?

Как воспринимать понятие «программирование»?

- Реальность программного обеспечения не встраивается естественным образом в пространство
- У программного обеспечения нет готового геометрического представления подобно тому, как местность представляется картой, компьютеры – схемами соединений

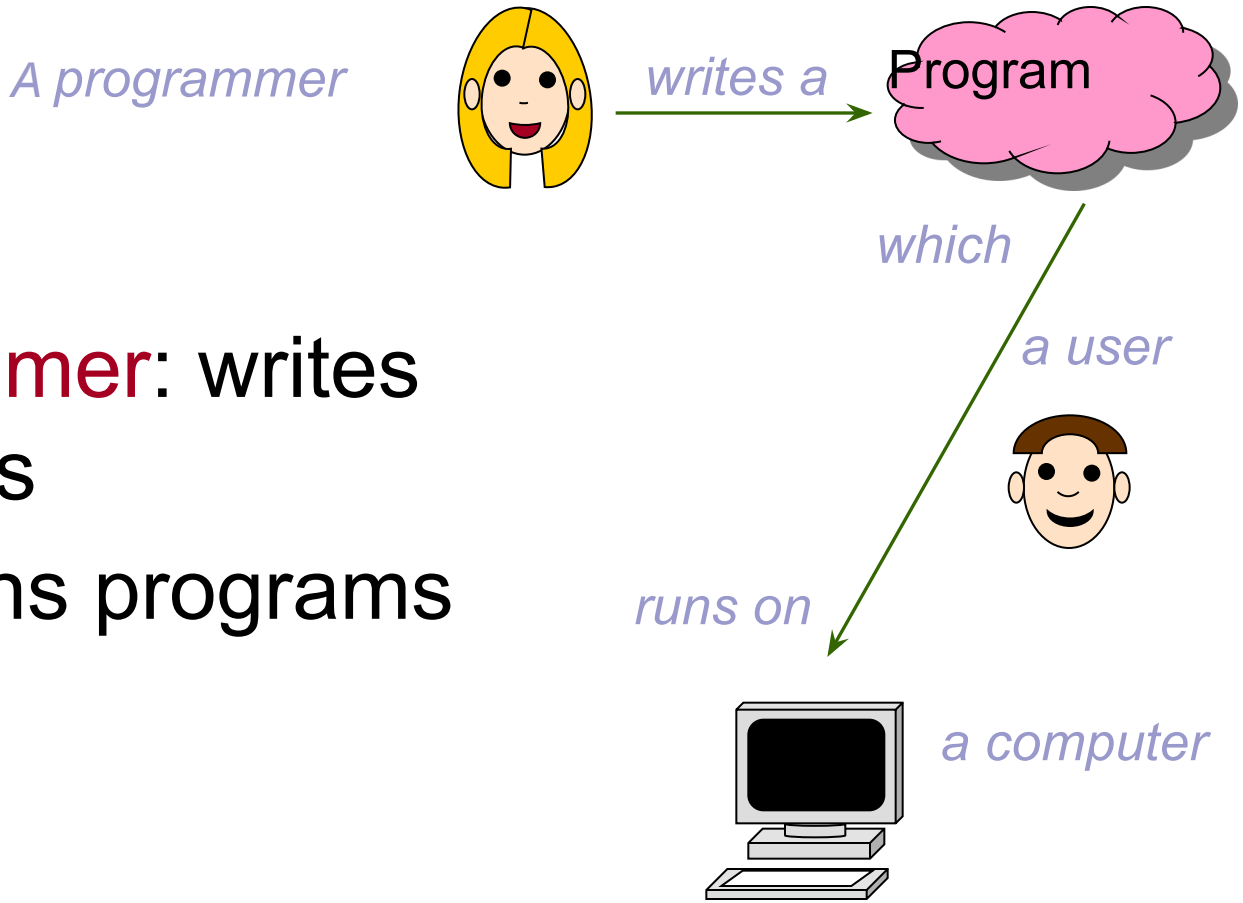
Есть некая абстрактная сущность, которая работает - производя видимые результаты, которые отделимы от самой конструкции:

- печатает значения данных,
- рисует картинки, производит звуки,
- приводит в движение рычаги

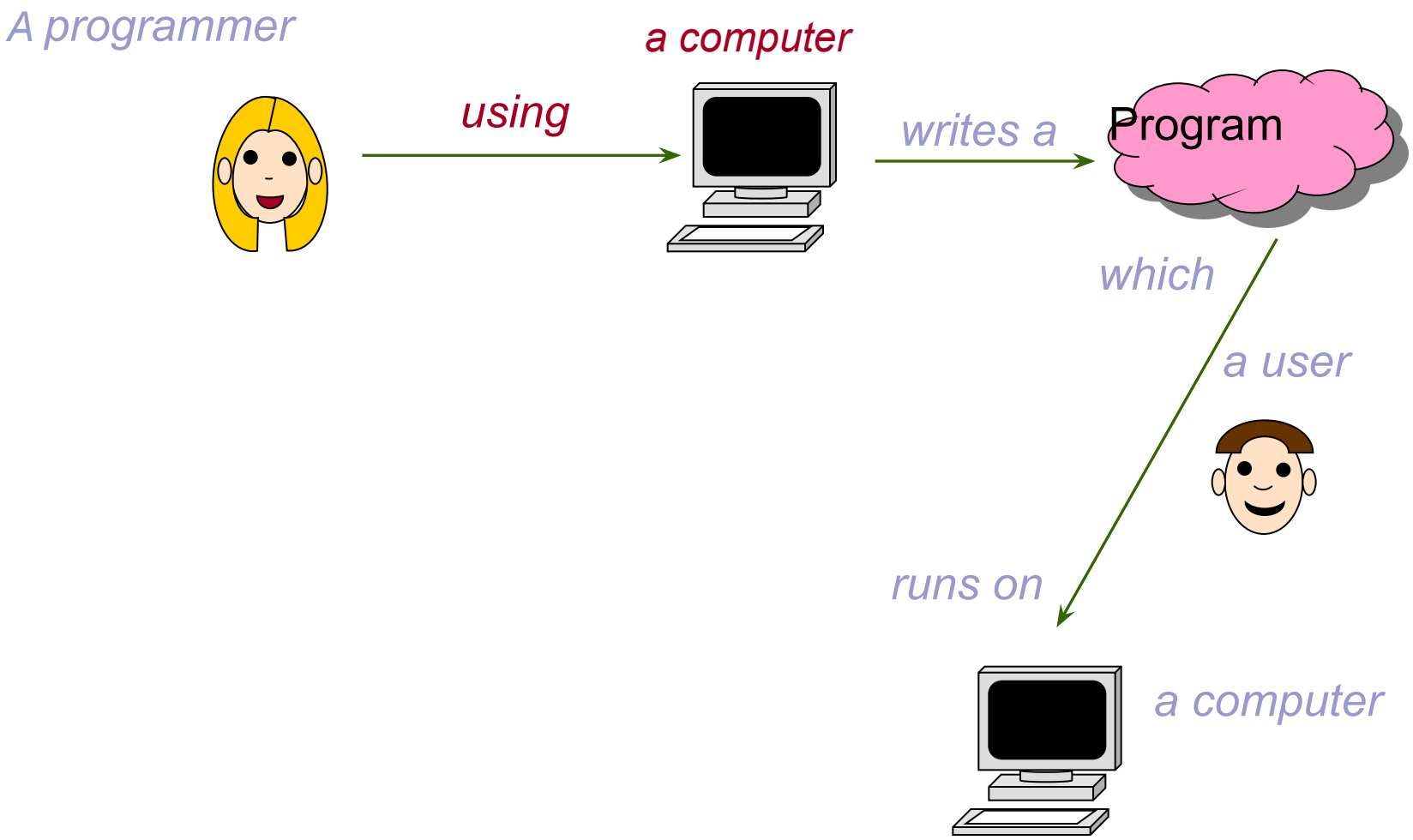
То есть она реальна

Создание и запуск программ

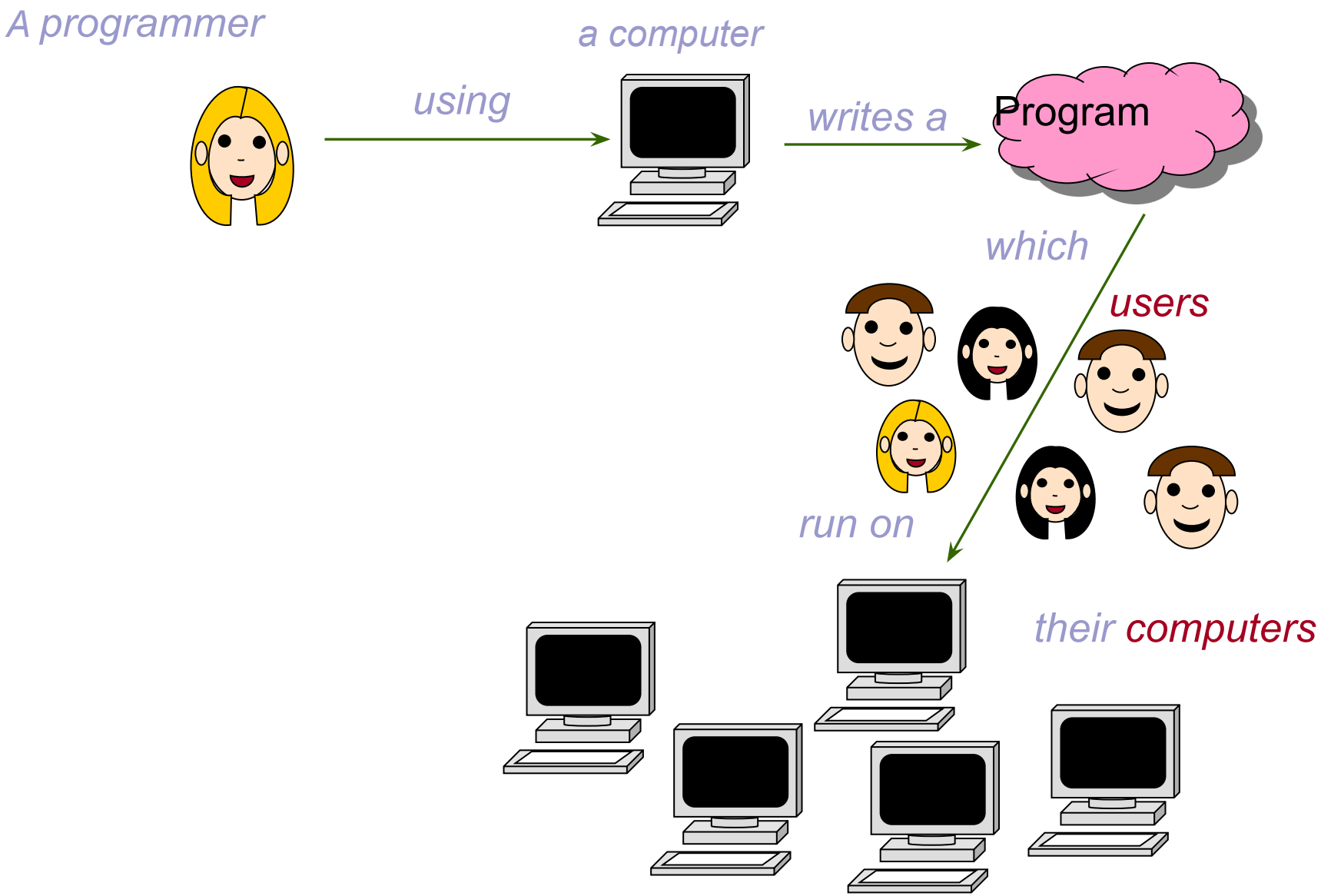
- Programmer: writes programs
- User: runs programs



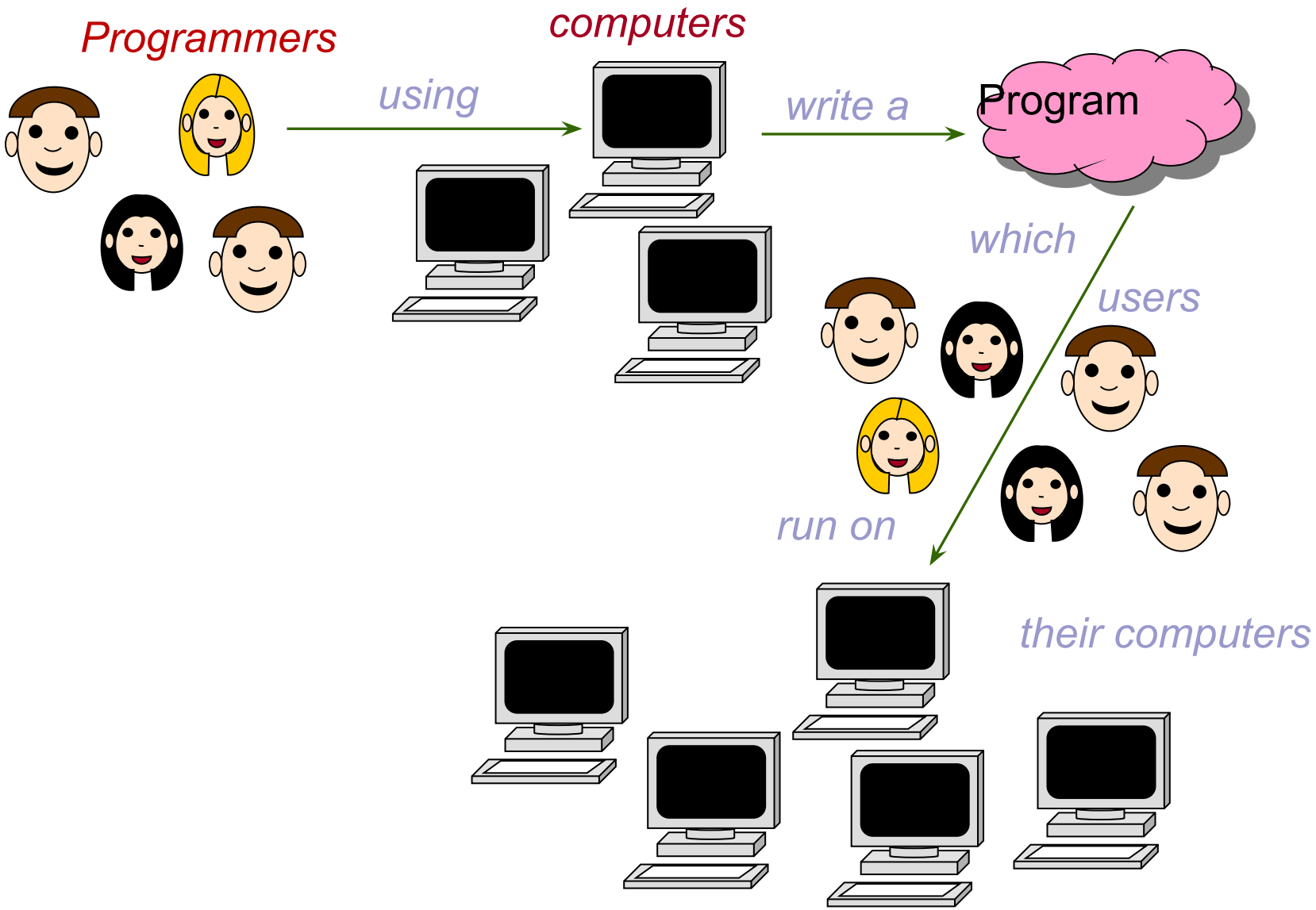
Создание и запуск программ



Создание и запуск программ



Создание и запуск программ



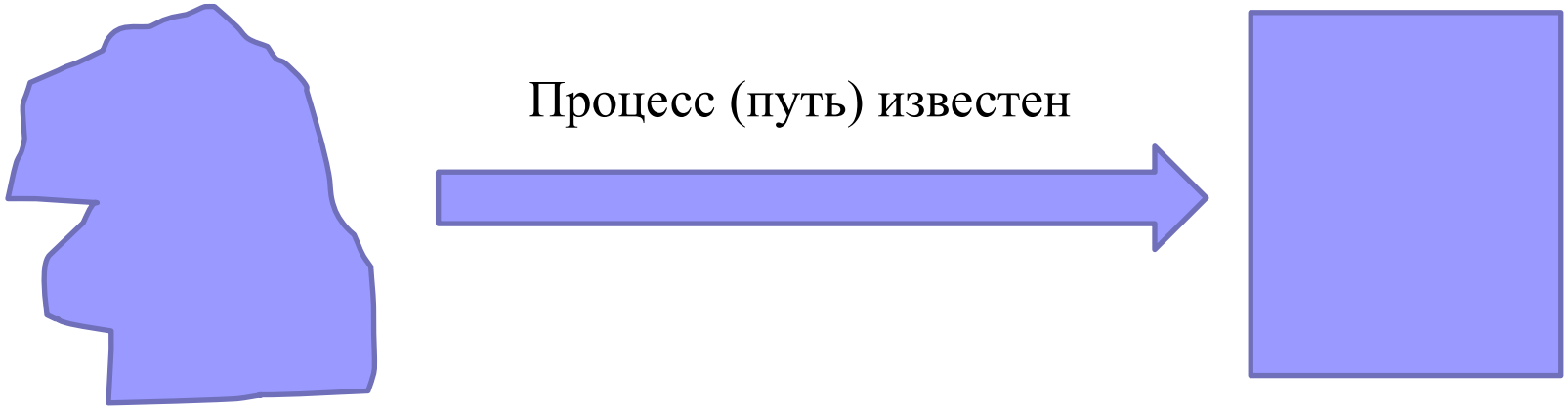
Программа и программирование

- Решение задач управления и планирования
- Пример. Уборка.
  - Что такое уборка?

Как должен выглядеть порядок известно

Беспорядок

Порядок

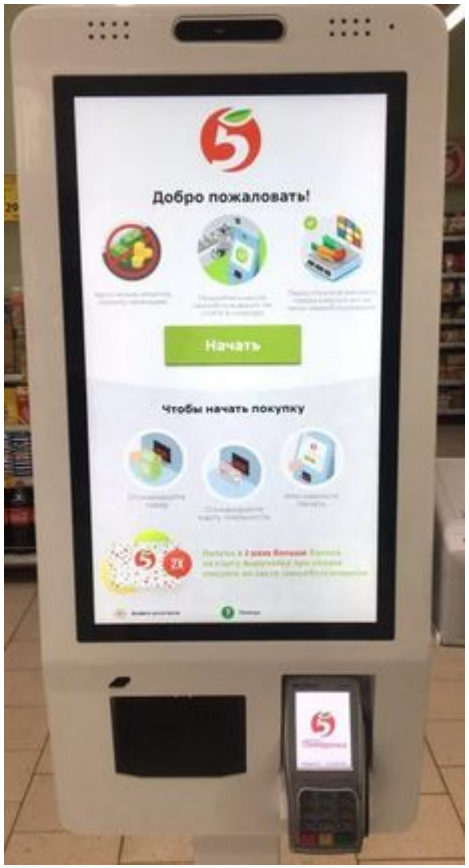


Программирование – это два состояния и план, т.е. задание пути от одного состояния к другому

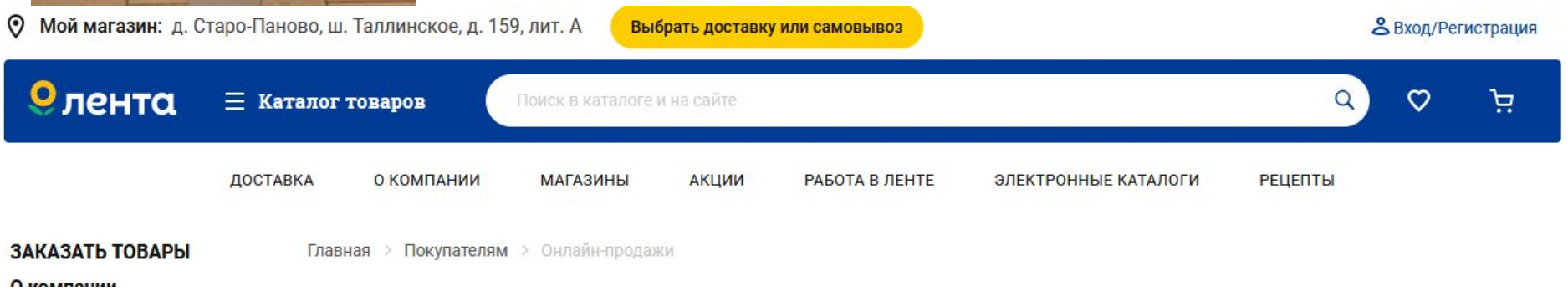
Какие должны быть программы?

- Программные тексты должны быть понятными и расширяемыми
  - простыми для внесения изменений
- Программные элементы должны быть повторно используемыми:
  - чтобы при повторной встрече с похожей задачей не пришлось бы повторно изобретать ее решение.
- Программы должны быть устойчивыми:
  - защищать себя от ошибочно введенных данных.
- Программы должны быть корректными:
  - выдавать ожидаемые результаты.

Примеры программных систем



[Как внедряли электронную кассу в X5 Retail Group]



Как работает компьютер?

Хранение и извлечение

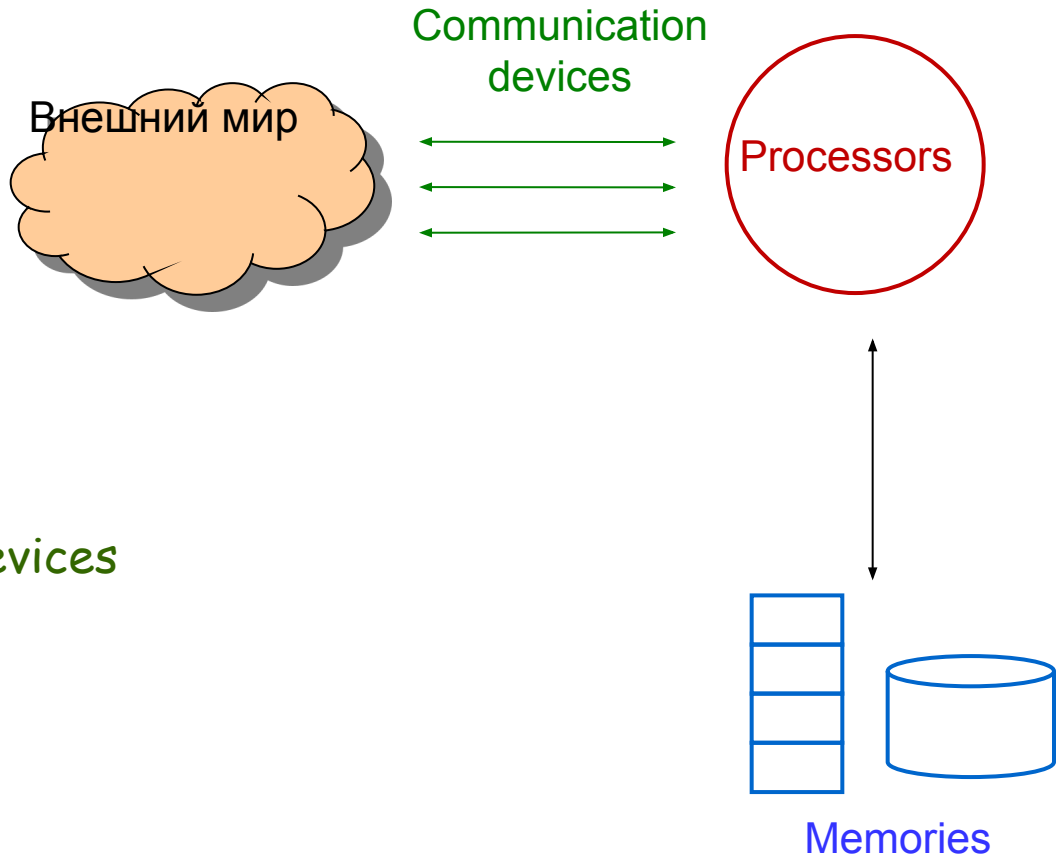
- Memories

Операции

- Processors

Коммуникация

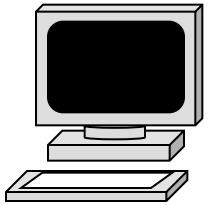
- Communication devices



Память, процессоры и устройства коммуникации составляют hardware

Как работает компьютер?

- Компьютеры – это универсальные машины – они выполняют программу, которую им передают



+

Program

=

Specialized machine

Software

Введение в программирование

Что будем делать?

- Простые «учебные» программы
- Новые системы, создаваемые с чистого листа
- Расширения существующих программ
- Сопровождение старой базы кода

Программирование – это искусство?

- Чем руководствуетесь – инстинктом или планом?
- Как понять, что хочет заказчик?

МАГИЯ ПРОГРАМИРОВАНИЯ

не понятна 95% населения

Проблема понимания ПО

- Так думает разработчик

Проблема понимания ПО

- А так думают пользователи

Проблема восприятия ПО

- Для пользователя любая система – это набор функций, которые он может пощупать и увидеть сложность реализации функции оценивается по ее “визуальному богатству”

Секрет айсберга

Чем больше форм и кнопок, тем интуитивно задача кажется более сложной

Программное обеспечение

- Программное обеспечение – совокупность всех программ, хранящихся на всех устройствах долговременной памяти компьютера, а также сопутствующая документация и конфигурационные данные

Прикладное программное обеспечение

Виды программ (приложений)

- **Консольное приложение** - это программа, которая работает с командной строкой.
  - это обычное окно, где пользователь может ввести какую-то команду и получить результат – нет графического интерфейса пользователя
- **Оконное приложение** - это программа с графическим интерфейсом (GUI – Graphical User Interface)
  - Настольное приложение (например, калькулятор)
  - Web-приложение (web-сайт)

Мобильные приложения
Клиент-серверные приложения
Библиотеки
Утилиты и драйверы
API (Application Program Interface)

Информация и данные

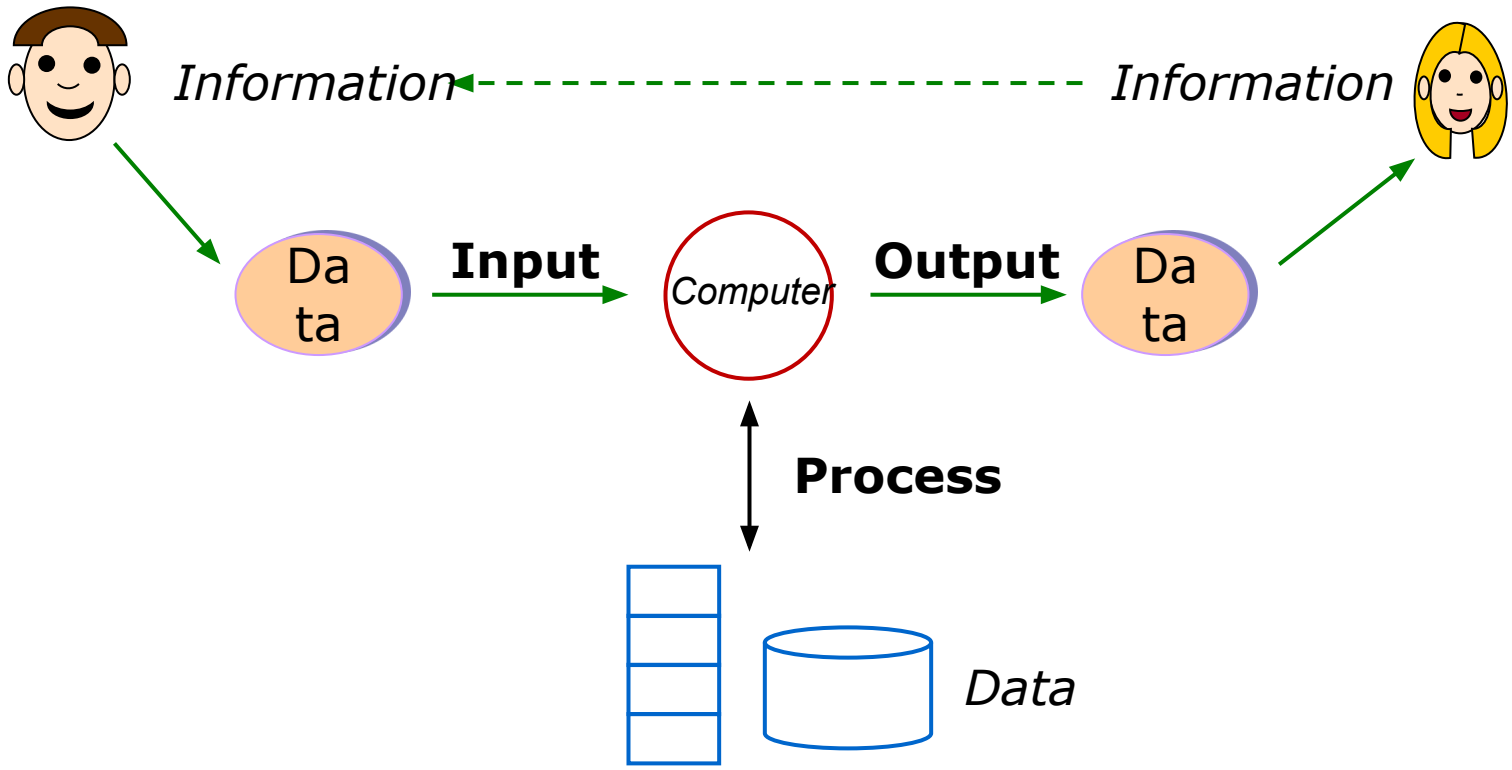
- Информация – это то, что человек может воспринять,
  - Пример: текст или музыка
- Данные – это то, как информация кодируются для компьютера
  - Пример: аудиоформат MP3

Данные: наборы символов, хранящиеся в компьютере

Информация: интерпретация данных для человеческих целей

Информация и обработка данных

- Данные хранятся в памяти
- Устройства ввода вырабатывают данные из информации
- Устройства вывода получают информацию из данных



Где находится программа?

- Компьютер с сохраненной программой: программа находится в памяти
  - “Executable data” – исполняемые данные
- Программа может быть в памяти в различных формах:
  - Источник (Source): удобочитаемая форма (язык программирования)
  - Целевая форма (machine code, object form): форма, выполняемая компьютером
- Компилятор (compiler) преобразует исходный текст в машинный код
- Компьютер (платформа + операционная система) находит программу в памяти для ее выполнения

Программирование

- Цель программирования – описание процессов обработки данных
- Данные – это представление фактов и идей в формализованном виде, пригодном для передачи и переработке в некоем процессе.
- Информация – это смысл, который придается данным при их представлении.

Программирование

- Обработка данных – выполнение систематической последовательности действий с данными.
  - Данные представляются и хранятся на носителях данных.
- Совокупность носителей данных, используемых при какой-либо обработке данных – *информационная среда*
- Набор данных, содержащихся в какой-либо момент в информационной среде – состояние этой информационной среды

Программирование

- Процесс – последовательность сменяющих друг друга состояний некоторой информационной среды
- Описать процесс, значит, определить последовательность состояний заданной информационной среды

Чтобы по заданному описанию требуемый процесс порождался автоматически на каком-либо компьютере, необходимо, чтобы это описание было формализованным – такое описание называется **программой**

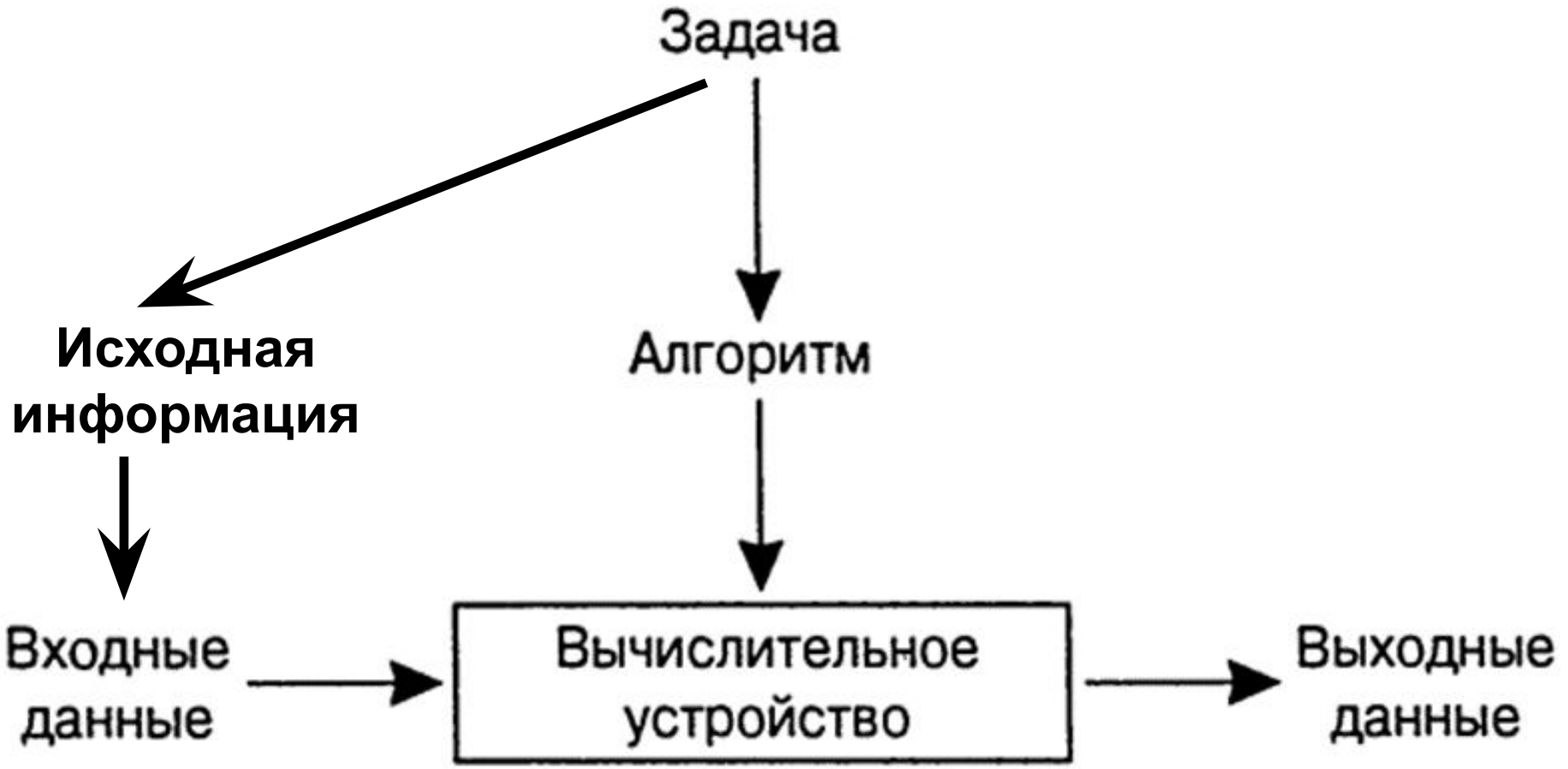
Программирование

- Программировать – это *исключительно точно*, шаг за шагом, описать последовательность преобразования информации с такой строгостью, которая требуется для *автоматического* выполнения этого преобразования
- Результат программирования – компьютерная программа – это некое послание человека машине

✔ Сложность программирования связана не столько с изучением какого-нибудь языка программирования, сколько с освоением нового способа мышления

Программирование

- Процесс обработки данных



Пример.

Точное определение некого явления (понятия)

Високосный год

- Все хорошо знают что такое високосный год (но лучше уточните у гугла ☺).
- В соответствии с этим правилом можно определить является ли некий конкретный год високосным.

Цель упражнения: научиться давать точные определения интуитивно ясных понятий – не операционные знания при конструировании ПО

Требуется:

- Определить (сформулировать) условия, при которых конкретный год определяется как високосный.
- Другими словами, требуется дать определение високосного года. Это понятие в дальнейшем будет использовано при программировании.
- Убедитесь, что ваше определение покрывает все возможные случаи и соответствует правилам високосного года.

Программирование

- Программное средство – программа или логически связанная совокупность программ на носителях данных, снабженная программной документацией

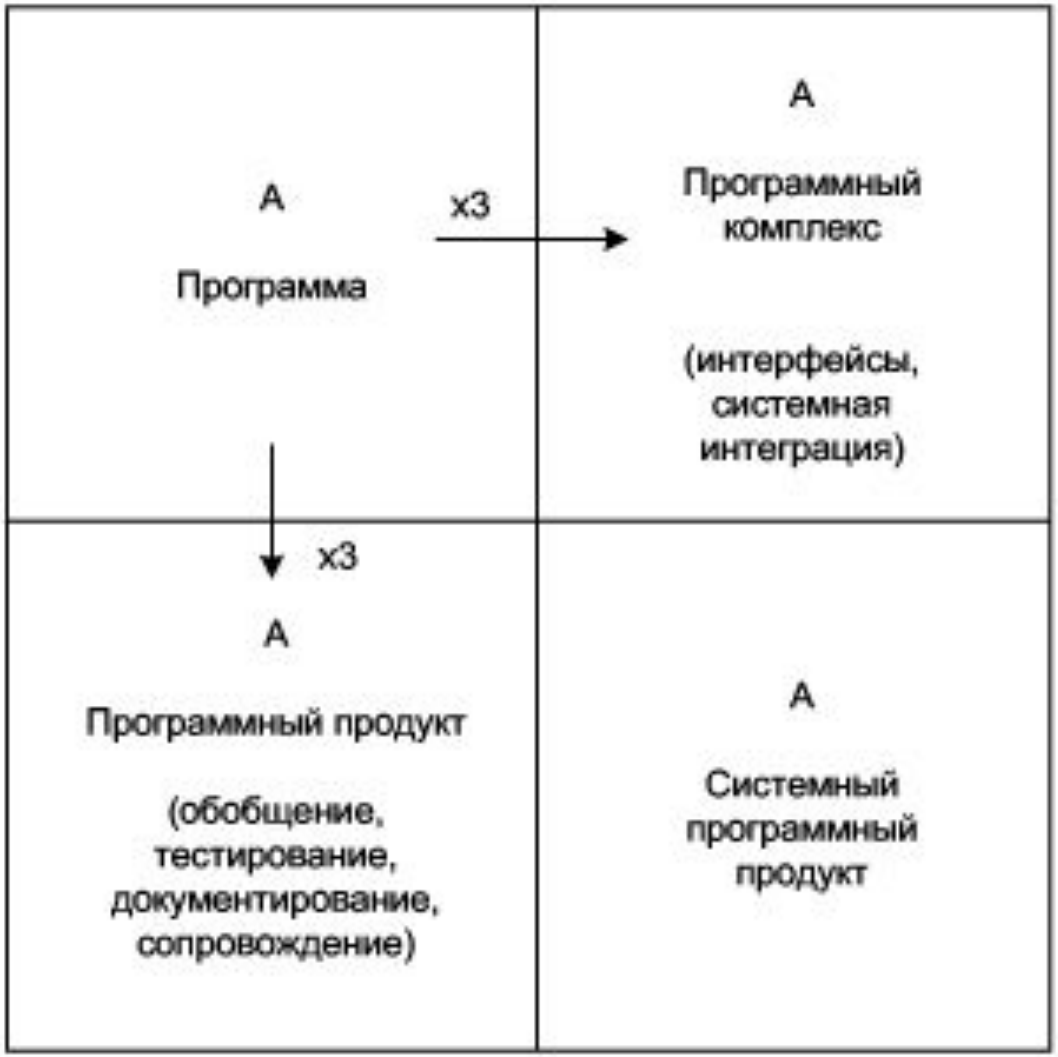
Программный продукт

- Общие программные продукты – автономные программные системы, которые созданы для продажи на открытом рынке программных продуктов *<u>любому потребителю</u>*
  - Примеры: системы управления базами данных, текстовые процессоры, графические пакеты, средства управления проектами, планировщики задач и т.п.
  - Распространяются как обычный товар «в коробке» или по Интернету
  - Клиент – массовый потребитель
  - Стандартные операции установки на машину клиента

Программный продукт

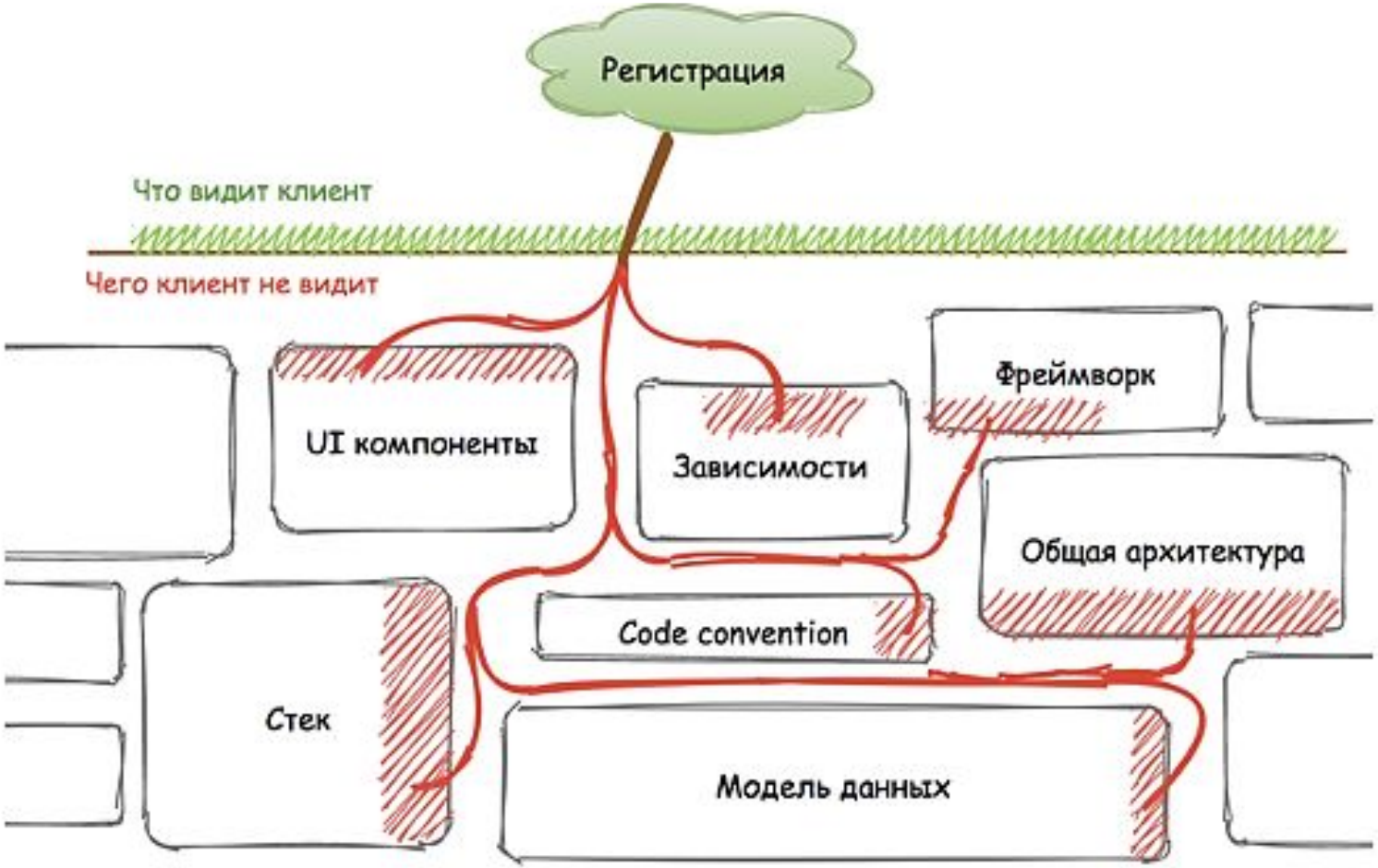
- Программные продукты, *созданные на заказ* - программные системы, которые создаются по заказу определенного потребителя
  - Примеры: системы поддержки определенных производственных или бизнес-процессов и т.п.
  - Разрабатываются специально для данного потребителя согласно заключенному контракту
  - Распространяются как внедрение в аппаратно-программной системе клиента – «решение»
  - Клиент – «заказчик» - организация (в широком смысле) со своими особыми требованиями
  - Развертывание и установка с учетом особенностей клиента

Системный программный продукт



Компонентная разработка

- Разработчики пользуются готовыми компонентами, но это именно **программные** компоненты – “сухие смеси”, а не готовые изделия.



Разработка программного обеспечения

- Хаотическая деятельность –"code and fix" ("пишем и правим")
  - единого плана не существует
  - общий проект представляет собой просто смесь краткосрочных решений

Для сложных систем нужна некая организация работы по созданию программного обеспечения

Технология разработки программного обеспечения – комплекс организационных мер, операций и приемов, направленных на разработку программных продуктов высокого качества в рамках отведенного бюджета и в срок

Методики разработки ПО различаются используемой моделью жизненного цикла ПО и уровнем формализма при его создании

Разработка программного обеспечения

- Хаотическая деятельность –"code and fix" ("пишем и правим")
  - единого плана не существует,
  - общий проект представляет собой просто смесь краткосрочных решений
- Технология превращает создание программного продукта в упорядоченный процесс
  - работа программиста более прогнозируемая и эффективна
  - создается детальное описание процесса создания системы, особое место в котором занимает планирование (аналогично другим инженерным дисциплинам)
- Облегчённые (lightweight) или гибкие (agile) технологии
  - меньшая ориентация на документацию
  - ориентированность на код

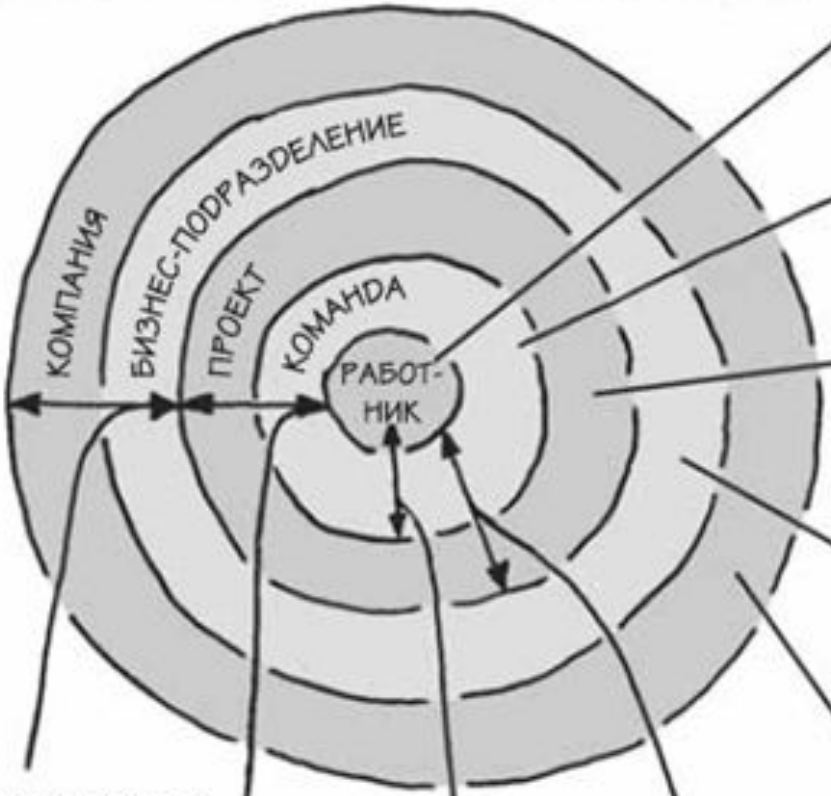
Командная разработка

- Работа в команде является необходимым условием успешности проекта
- Умение работать в команде – важное качество высококвалифицированного разработчика программного обеспечения

Уровни команд

- Отдельный программный компонент, входящий в более крупный проект;
- Компонент должен войти в более общий продукт.
- Разработка нескольких проектов одновременно

УРОВНИ ГРУППОВОЙ РАБОТЫ



Динамика организации (верхний уровень групповой работы)

Динамика группы (нижний уровень групповой работы)

Взаимодействие внутри команды

Взаимодействие между командами

Требует:
- умения разрабатывать ПО
- умения мыслить
- обучения
- мотивации

Вводит:
- социальные навыки
- динамику внутри команды

Объединяет несколько маленьких команд, вводя динамику между ними; большие усилия по координации.
Особого внимания требуют:
- обмен данными
- планирование
- управление ресурсами

Каждый проект должен решать задачи бизнес-подразделения.
На этом уровне на работу влияют:
- политика корпорации
- культура
- процедуры

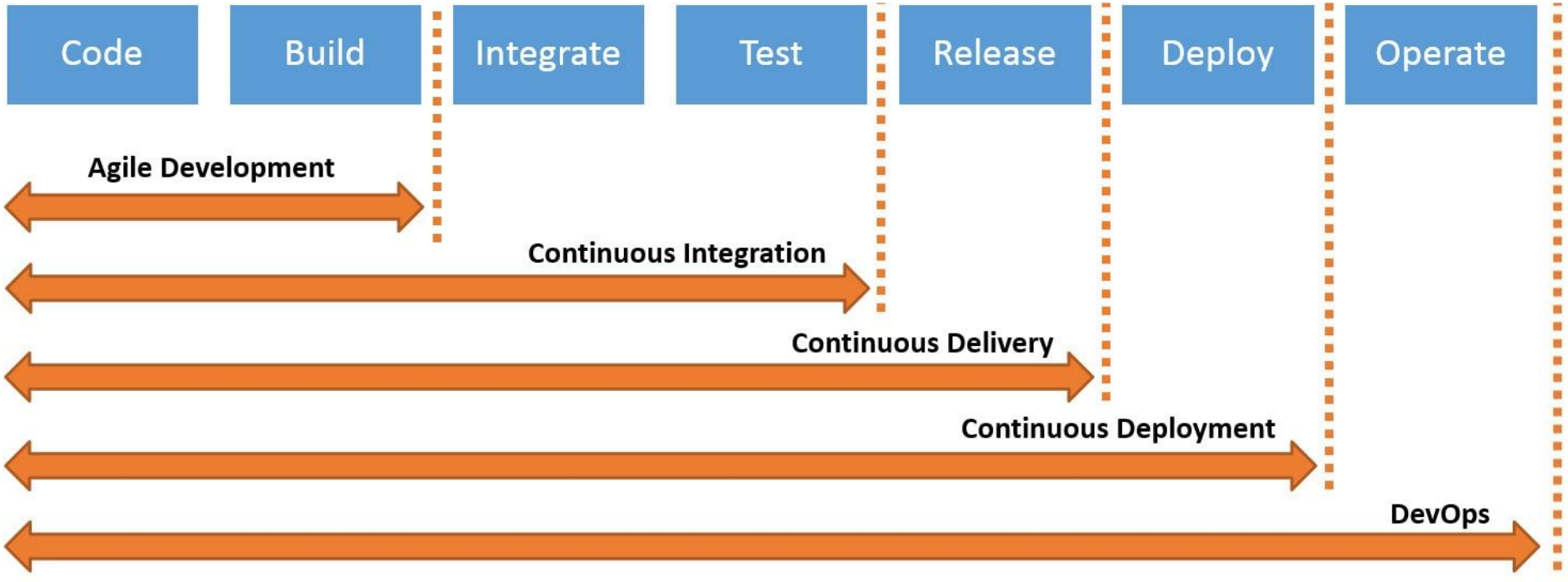
Корпоративный контекст:
- взаимодействие с другими компаниями (клиентами, поставщиками)
- бизнес-стратегия
- схемы стимулирования

Навыки и умения

- Документирование
- Тестирование
- Управлениями версиями
- Непрерывная интеграция и непрерывная доставка (CI/CD)

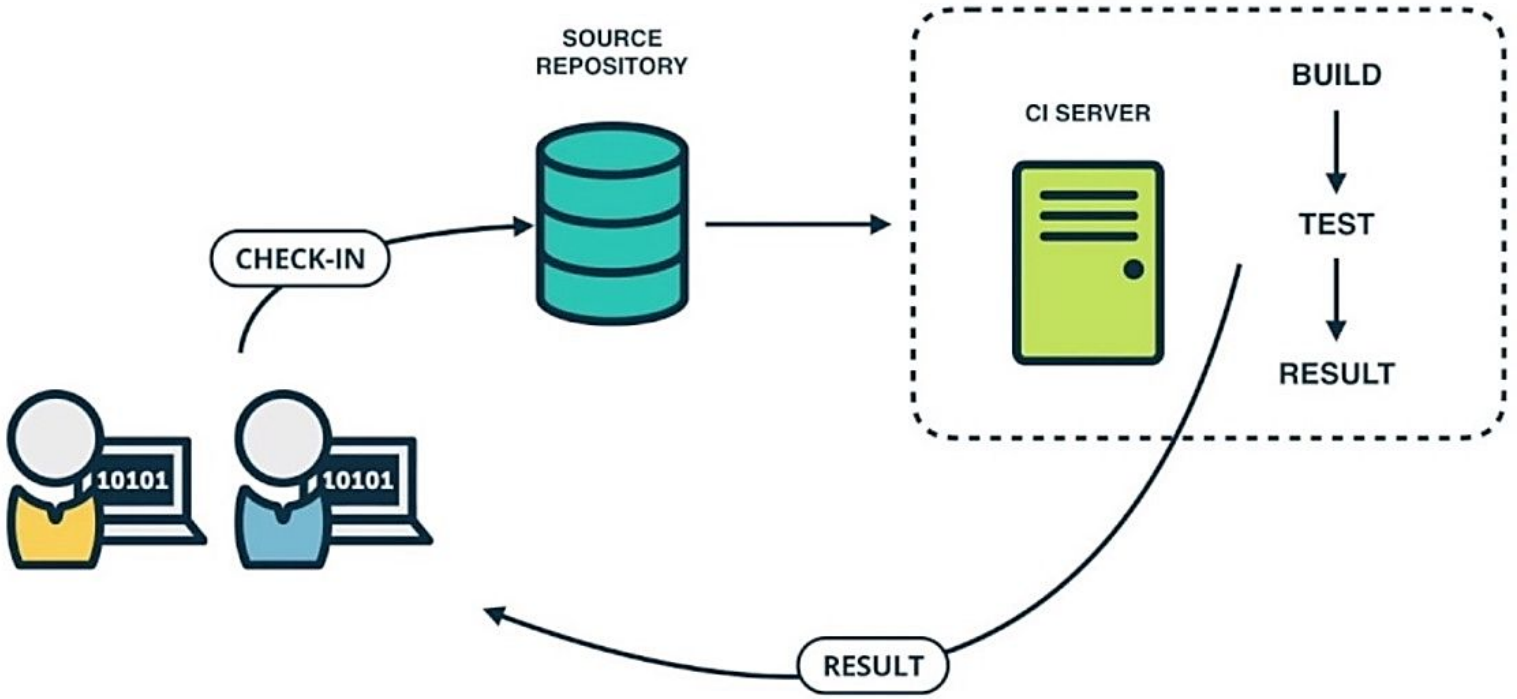
Непрерывный жизненный цикл доставки приложений

- Continuous Integration (Непрерывная интеграция)
- Continuous Delivery (Непрерывная доставка)
- Continuous Deployment (Непрерывное развертывание)



Continuous Integration (CI)

- Процесс непрерывной интеграции (Continuous Integration) нацелен на автоматизированную проверку интеграции между изменениями разработчика и остальным кодом
  - заключается в постоянном слиянии рабочих копий софта в одну основную общую ветвь разработки

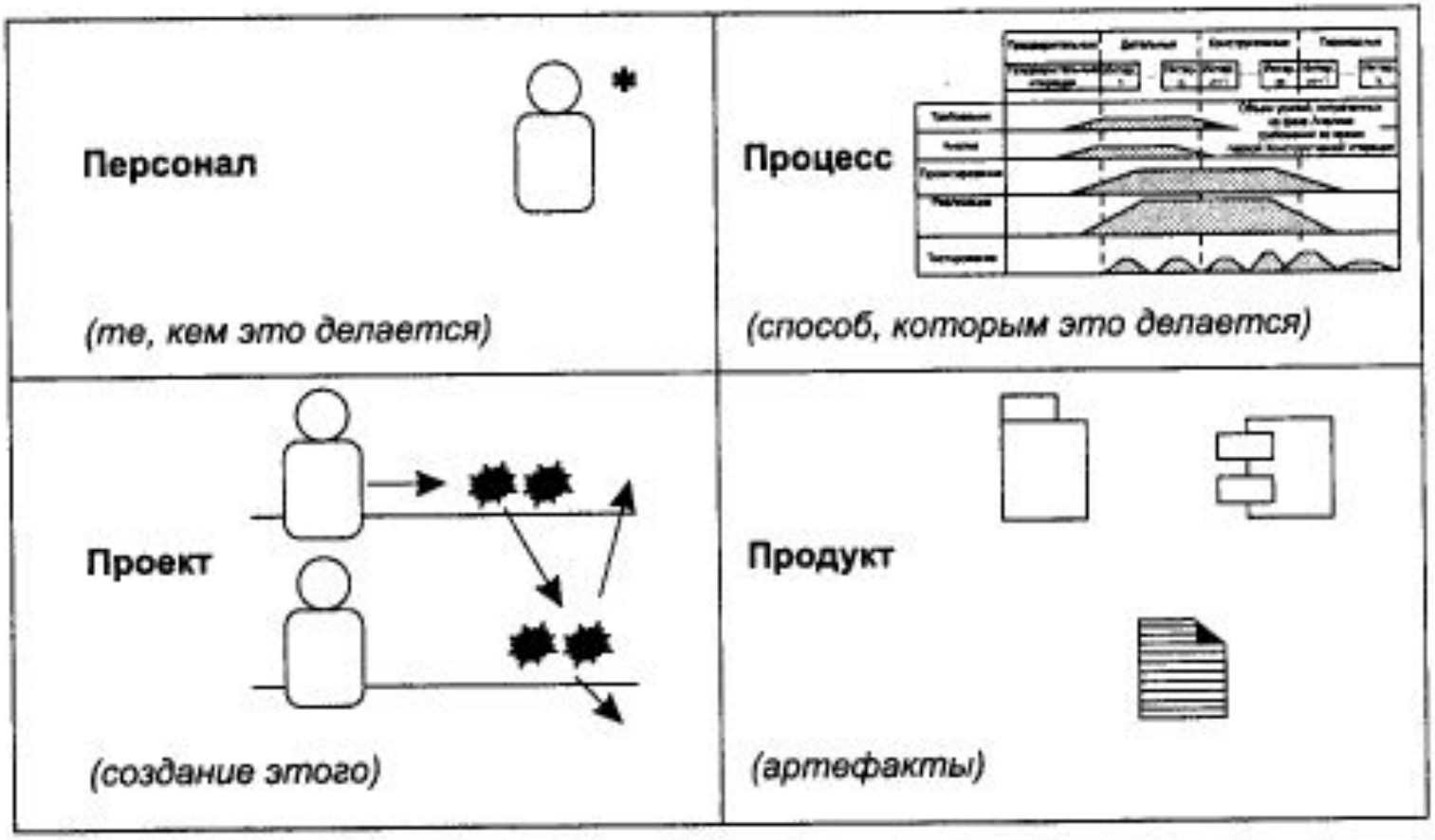


Непрерывная доставка + Непрерывное развертывание

- Continuous Delivery (CD)
  - Целью этого этапа является доставка измененной версии приложения в эксплуатацию
  - Автоматизация процессов CD позволяет ускорить процесс доставки, исключить влияние человеческого фактора, а также сделать доставку более доступной для остальных членов команды
- Continuous Deployment (CD)
  - Процесс нацелен на развертывание новой версии приложения в окружение эксплуатации или тестирования
  - Обычно этот этап не выделяют как отдельный, развертывание включают в процесс доставки

Система разработки ПО

- Система разработки программного обеспечения включает в себя *персонал, процесс, проект* и *продукт*



Процесс и стадии создания ПО

Процесс создания ПО – совокупность мероприятий, целью которых является создание или модернизация ПО

- Анализ предметной области (постановка задачи)
- Разработка проекта системы
  - Создание модели, отражающей основные функциональные требования, предъявляемые к программе
  - Выбор метода решения (построение математической модели)
  - Разработка алгоритма – последовательности действий по решению задачи
- Реализация программы на языке программирования (кодирование)
- Анализ полученных результатов (тестирование)
- Внедрение и сопровождение

Проблема желаний

При разработке ПО

Необходимость удовлетворить множество различных, иногда взаимоисключающих желаний (требований)

- Механизмы функционирования современных систем сами по себе довольно сложны для понимания
- Дополнительные требования (часто неявные и трудно формализуемые)
  - например, удобство, производительность, стоимость, устойчивость и надежность

- Пользователям очень трудно выразить свои потребности в форме, понятной разработчикам
  - каждая из этих групп испытывает недостаток знаний в предметной области другой группы

Требования к ПО

IEEE Standard Glossary of Software Engineering Terminology (1990):

- Условия или возможности, необходимые пользователю для решения проблем или достижения целей
- Условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам

- Основная задача этапа определения требований
  - выяснить, обсудить и зафиксировать, что действительно требуется от системы в форме, понятной и клиентам и членам команды разработчиков.

Анализ и проектирование

- Этап анализа (analysis) состоит в исследовании требований и решаемой проблемы

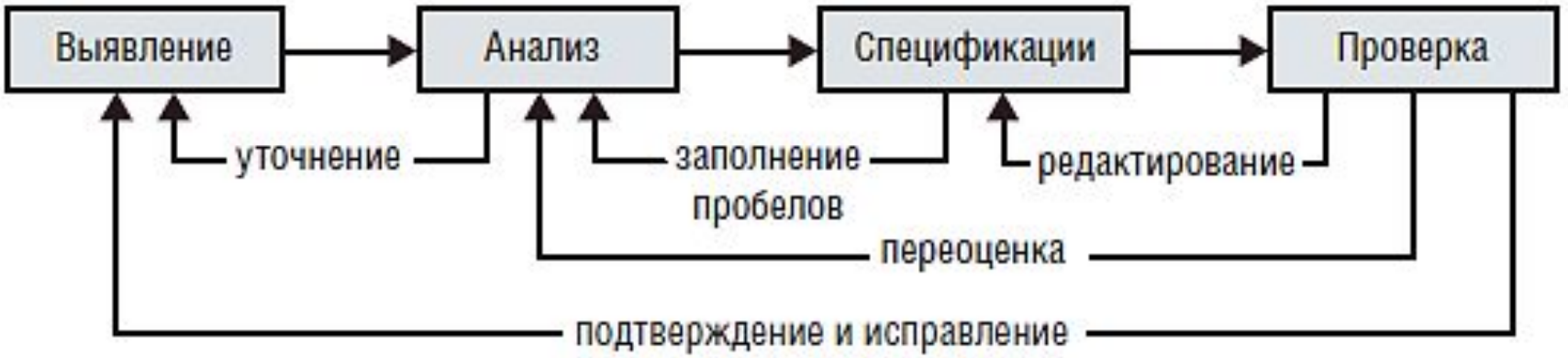
Различают:

- Анализ требований (requirements analysis) – исследование требований к системе
- Объектный анализ (object analysis) – исследование объектов предметной области

- В процессе проектирования (design) основное внимание уделяется концептуальному решению, обеспечивающему выполнение основных требований
  - Например, на этапе проектирования описываются программные объекты или схема базы данных

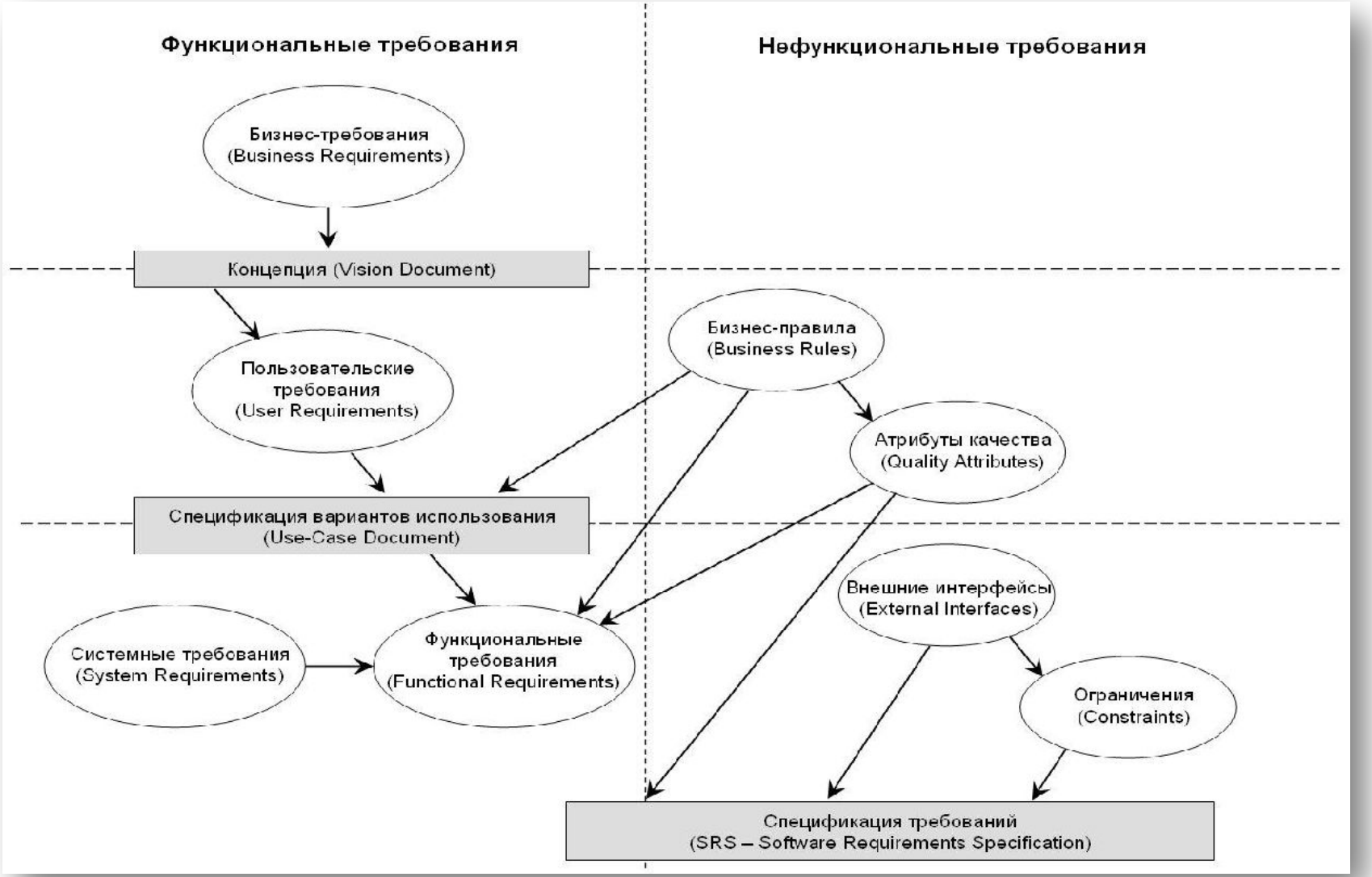
Итеративный процесс формулирования требований

- Разработка требований состоит из выявления, анализа, документирования и проверки



Выявление: задавать клиентам вопросы, слушать их ответы и наблюдать, что они делают
Анализ: классифицировать информацию по категориям, связать потребности клиентов с возможными программными требованиями
Спецификация: структурировать информацию
от пользователей и выведенные требования в виде письменных требований-утверждений и диаграмм
Проверка: подтверждение, что «представленное» точно и полно отражает потребности, исправление ошибок

Функциональные и нефункциональные требования к программному средству



- *Бизнес-требования* (business requirements) содержат высокоуровневые цели организации или заказчиков системы
- *Требования пользователей* (user requirements) описывают цели и задачи, которые пользователям позволит решить система
- *Функциональные требования* (functional requirements) определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований
- *Системные требования* (system requirements) определяют высокоуровневые требования к продукту, которые содержат многие подсистемы
- *Бизнес-правила* (business rules) включают корпоративные политики, правительственные постановления, промышленные стандарты и вычислительные алгоритмы
- *Атрибуты качества* (quality attributes) представляют собой дополнительное описание функций продукта, выраженное через описание его характеристик, важных для пользователей или разработчиков

Пользовательские истории (User story)

- Пользовательские истории составляются в свободной форме, в виде историй или некоторых сценариев использования системы
  - Каждая история имеет условного рассказчика (автора) истории, повествующего о наиболее значимых для исполнения требований к проектируемой программной системе
  - Эти истории являются основой для формулирования функциональных требований
  - Могут быть использованы в дальнейшем для создания приемочных тестов (критериев) при оценивании качества ПО

Как писать User Story

Как писать качественные пользовательские истории

techtarget.com/ user-story

Пользовательские истории - примеры и шаблоны

Пользовательские истории (User story)

Пользовательские истории включают:

- Контекст
  - “находясь в окне … приложения…”
- Событие
  - “при выполнении ….. когда делаю…..”
- Результат
  - “получаю ответ (уведомление) системы …..”

Пример. Книжный интернет-магазин

Пользовательская история	Функция (как часть функциональности) системы
Как клиент я хочу обновить свой профиль, чтобы оплачивать покупки новой кредитной картой	Обновление профиля клиента
Как клиент я хочу просматривать и выбирать товары из каталога	Поиск товара
Как клиент я хочу оплачивать товар кредитной картой	Покупка товара
Как клиент я хочу отменять заказ в любой момент времени	Отмена заказа

Представление требований

- **Документация**, в которой используется четко структурированный и аккуратно используемый естественный язык
- **Графические модели**, иллюстрирующие процессы преобразования, состояния системы и их изменения, отношения данных, логические потоки и т. п.
- **Формальные спецификации**, где требования определены с помощью математически точных, формальных логических языков

Рамки решения и рамки проекта

- Требования к системе («рамки решения») и соответствующие им задачи разработчика («рамки проекта») требования сводят в таблицу

URS ID	Описание «рамки решения»	FRS ID	Описание «рамки проекта»
Функция: ***Название функции***			
1	Система обеспечивает возможность пользователям……	101	Создание меню……
		102	Реализация просмотра результатов….

Подробные функциональные и нефункциональные требования к продукту записываются в **спецификации к требованиям** к ПО, которая предоставляется тем, кто должен проектировать, разрабатывать и проверять решение.

Пользовательские интерфейсы и спецификация требований

- **Вопрос**: включать описание элементов пользовательского интерфейса в спецификацию?
- Однозначного ответа нет!
  - Изображения и архитектура пользовательского интерфейса отображают дизайн, а не требования.
  - Если у пользователей продукта есть ожидания насчет того, как должны выглядеть и вести себя те или иные части продукта, то можно включить описание UI в спецификацию

Как оценить качество требований?

- **Полнота.** Каждое требование должно содержать всю информацию, необходимую разработчику, чтобы реализовать его
- **Корректность**. Каждое требование должно точно описывать возможность, которая будет удовлетворять какую-то потребность заинтересованного лица и четко определять функциональность, которую надо построить
- **Осуществимость**. Необходима возможность реализовать каждое требование при известных возможностях и ограничениях системы и рабочей среды, а также в рамках временных, бюджетных и ресурсных ограничений проекта.

Как оценить качество требований?

- **Необходимость**. Каждое требование должно отражать возможность, которая действительно предоставит заинтересованным лицам ожидаемую бизнес-пользу, выделит продукт на рынке или необходима для соблюдения внешних стандартов, политик или правил.
- **Назначение приоритетов.** Определяйте приоритеты требований на основании важности для получения требуемой пользы.
- **Недвусмысленность**. Читатели должны понимать, о чем идет речь в требовании.

Общий шаблон формулировки требований

- С точки зрения системы:

[*необязательное предварительное условие*] [*необязательный триггер события*] **система должна** [*ожидаемая реакция системы*]

- Пример (без триггера).

«Если запрошенный материал есть на складе, система должна отобразить список всех хранимых на складе контейнеров с указанным материалом»

- Можно фразу «система должна» не указывать, если удаление фразы не изменит смысла требования

Общий шаблон формулировки требований

- С точки зрения пользователя:

[*класс пользователя или имя действующего лица*] **должен иметь возможность** *[выполнить что-то] [с каким-то объектом] [условия выполнения, время отклика или декларация качества*]

- Пример.

«Менеджер должен иметь возможность повторно заказать любой материал, который он ранее заказывал, путем извлечения и редактирования параметров ранее введенного заказа»

- В требовании рекомендуется использовать название класса пользователя – Менеджер, а не общий термин «пользователь» (явная формулировка максимально снижает вероятность неверного истолкования)

Общие рекомендации по формулировке требований

- Избегайте смешения активного и пассивного залогов в попытке сделать материал более интересным для чтения.
- Не обозначайте одно понятие разными терминами, чтобы разнообразить свой текст
- Пишите требования полными предложениями, с правильной грамматикой, правописанием и пунктуацией, предложения и абзацы должны быть краткими и ясными.
- Язык должен быть простым и прямолинейным, характерным для соответствующей предметной области, но не используйте профессиональный жаргон. Определения используемых терминов размещайте в словаре терминов
- Избегайте длинных повествовательных абзацев, которые содержат несколько требований

Стандартизация проектирования ПО

- ГОСТ 19.xxx-xx Единая система программной документации
- ГОСТ 34.601-90 - распространяется на автоматизированные системы и устанавливает стадии и этапы их создания
- ISO/IEC 12207 - стандарт на процессы и организацию *жизненного цикла*. Распространяется на все виды заказного ПО
- Custom Development Method (методика Oracle) по разработке прикладных информационных систем
- Rational Unified Process (RUP)
- Microsoft Solution Framework (MSF)
- Программная инженерия Google

Жизненный цикл программного обеспечения

- *Жизненный цикл* - это непрерывный процесс, который начинается
  - с момента принятия решения о необходимости его создания
  - и заканчивается в момент его полного изъятия из эксплуатации.

Структура ЖЦ ПО

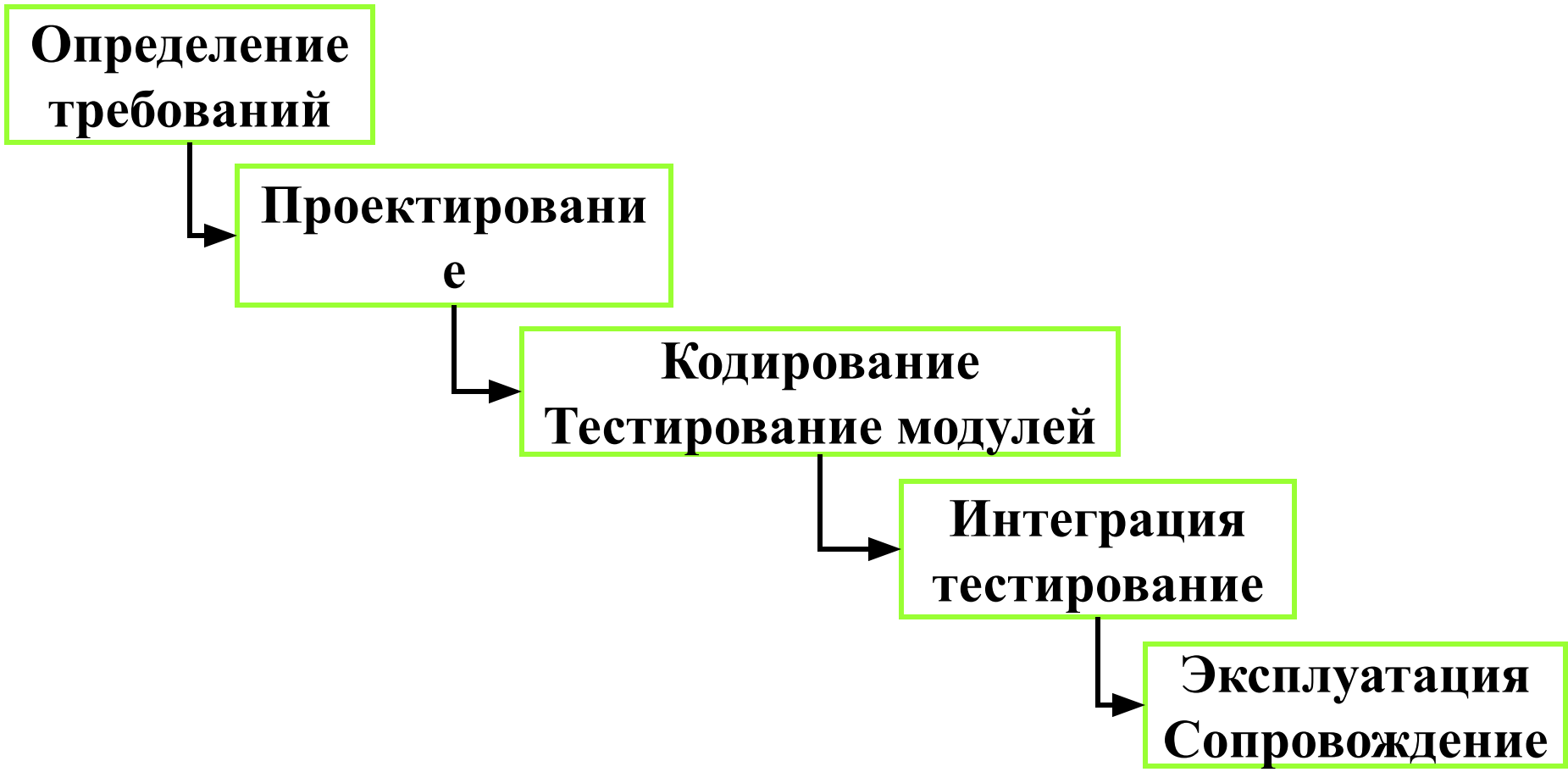
Жизненный цикл ПО базируется на трех группах процессов:

- **основные** процессы
  - реализуются под управлением основных сторон (заказчик, поставщик, разработчик, оператор и персонал сопровождения), вовлеченных в жизненный цикл программных средств
- **вспомогательные** процессы
  - обеспечивают выполнение основных процессов;
- **организационные** процессы
  - применяются для создания, реализации и постоянного совершенствования основной структуры, охватывающей взаимосвязанные процессы жизненного цикла и персонал.

Модель жизненного цикла

- ГОСТ Р ИСО/МЭК 15288 — 2005 Информационная технология. Системная инженерия. Процессы жизненного цикла систем
- **Модель жизненного цикла** (life cycle model): Структурная основа процессов и действий, относящихся к жизненному циклу, которая также служит в качестве общей ссылки для установления связей и взаимопонимания сторон.
- **Процесс** (process) – совокупность взаимосвязанных и взаимодействующих видов деятельности, преобразующих входы в выходы

Каскадная модель



- **Каскадная модель**:
  - Фиксированный набор стадий
  - Каждая стадия -> законченный результат
  - Стадия начинается, когда закончилась предыдущая.
- **Недостатки**: **негибкость**
  - фаза д.б. закончена, прежде чем приступить к следующей
  - Набор фаз фиксирован
  - Тяжело реагировать на изменения требований
- **Использование**: там, где требования хорошо понятны и стабильны.

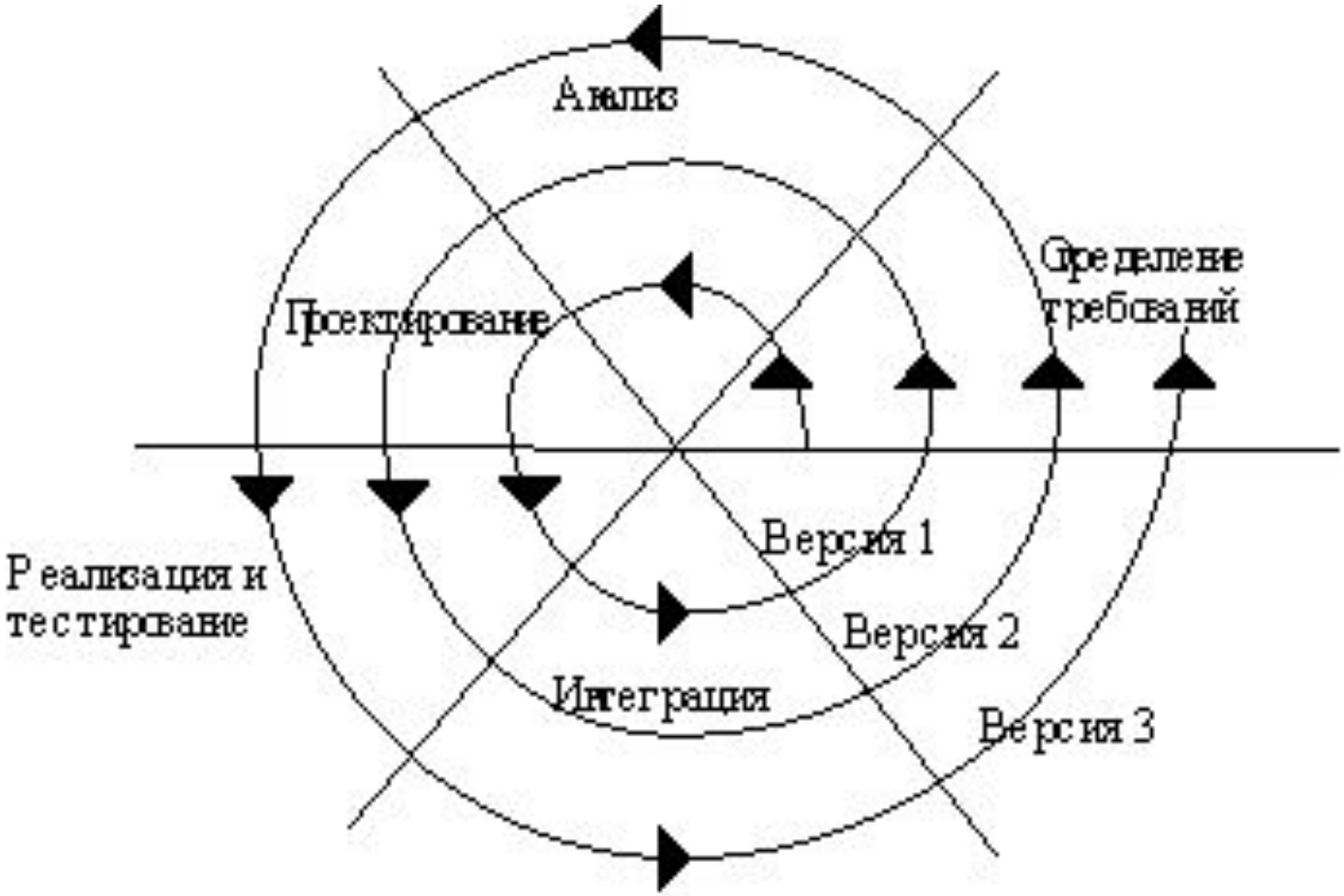
Итерационный подход

- Часто подходы, перечисленные ранее, используется **в совокупности**.
- **Требования** всегда меняются в ходе разработки.
- К каждой из предыдущих моделей можно применить **итерации**.
- Следовательно, важна возможность выполнения итераций, результатом которых является прототип продукта с частичной функциональностью.
- Это достигается в итерационных моделях.
  - **Модель пошаговой разработки**
  - **Спиральная модель разработки**

Спиральная модель

- Вместо действий с обратной связью – спираль.
- Каждый виток спирали соответствует 1 итерации.
- Нет заранее фиксированных фаз. В зависимости от потребностей.
- Каждый виток разбит на 4 сектора:
  - Определение целей
  - Оценка и разрешение рисков
  - Разработка и тестирование
  - Планирование
- **Главное отличие: акцент на анализ и преодоление рисков.**
- На каждом витке могут применяться разные модели процесса разработки ПО.

Спиральная модель



Единая система программной документации

- **Единая система программной документации** - комплекс государственных стандартов, устанавливающих взаимоувязанные правила разработки, оформление и обращения программ и программной документации

Назначение ЕСПД

- В стандартах ЕСПД устанавливают требования, регламентирующие разработку, сопровождение, изготовление и эксплуатацию программ, что обеспечивает возможность:
  - унификации программных изделий для взаимного обмена программами и применение ранее разработанных программ в новых разработках;
  - снижение трудоёмкости и повышение эффективности разработки, сопровождения, изготовления и эксплуатации программных изделий;
  - автоматизации изготовления и хранения программной документации

Состав ЕСПД

В состав ЕСПД входят:

- основополагающие и организационно-методические стандарты
- стандарты, определяющие формы и содержание программных документов, применяемых при обработке данных
- стандарты, обеспечивающие автоматизацию разработки программных документов.

Состав ЕСПД

- ГОСТ 19.101-87 Виды программ и программных документов
- ГОСТ 19.102-77 Стадии разработки
- ГОСТ 19.201-78. Техническое задание, требования к содержанию и оформлению
- ГОСТ 19.202-78. Спецификация, требования к содержанию и оформлению
- ГОСТ 19.401-78. Текст программы, требования к содержанию и оформлению
- ГОСТ 19.402-78. Описание программы
- ГОСТ 19.501-78. Формуляр, требования к содержанию и оформлению

Вопросы?