Сизов

Данил

Евгеньевич

Информатика

или

Введение в специальность

**#1** **Гибкие методологии проектирования**

Кабан и скрап гибкие методологи проектирования в управлении проектами, которая активно используется в разработке ПО и не только.

Это методология визуального управления рабочим процессом, которая помогает увеличить эффективность и гибкость работы команды. Основная идея заключается в визуализации всех задач на доске (физической или цифровой), где каждая задача представлена в виде карточки, проходящей через несколько этапов.

**Пример:**

Ожидает => В процессе => Готово

Kanban помогает выявлять узкие места и оптимизировать поток задач. Главные элементы Kanban включают:

* Визуализация работы
* Лимитирование незавершенных задач
* Постоянное измерение и совершенствование процесса

Scrum

Это гибкая методология, которая фокусируется на **медитативной** и **инкрементальной** методологии.

Основные компоненты Scrum:

1. Роли:

* Владелец продукта (PO) - отвечает за приватизацию задач и общение с клиентами
* Scrum мастер (SM) - помогает команде работать эффективно и решает её проблемы
* Команда разработки (DT) - работает над задачами и создаёт программный продукт

1. События:

* Планирование Sprint
* Ежедневные Scrum встречи
* Обзор Sprint
* Ретроспектива

1. Артефакты:

* BackLog продукта
* BackLog Sprint
* Инкремент

**Определения:**

**BackLog** - список задач, которые надо выполнить для создания продукта. Он содержит все изменения, требования и функции, которые надо реализовать. BackLog является основным артефактом Scrum и управляется владельцем продукта.

Так же BackLog имеет некоторые характеристики:

* Он является приоре тезированным
* Он является изменяемым, задачи могут быть изменены, добавлены или обновлены в любое время
* Он доступен для всей команды, чтобы обеспечить прозрачность и понимание задач

**Sprint** — это короткий временной интервал, в течение которого Scrum-команда выполняет заданный объем работы.

* Фиксированная длинна. Команда знает сколько времени у неё есть на выполнение задач
* Он является итеративным. Команда работает в цикле спринтов
* Он сосредоточен на доставке продукта заказчику

**#****2 Колмогоровская сложность**

**Колмогоровская сложность** — это мера сложности алгоритма или компьютерной программы, предложенная Андреем Колмогоровым в 1960-х годах. Она измеряет количество информации, необходимой для описания объекта или алгоритма.

**Колмогоровская сложность объекта x**, обозначаемая как K(x), — это длина самого короткого бинарного кода, который может быть использован для описания объекта x.

Колмогоровская сложность может быть рассчитана по следующей формуле:

**x** - объект, для которого мы хотим рассчитать сложность

**p** - бинарный код, описывающий объект x

**U** - универсальная машина Тьюринга, которая может интерпретировать бинарный код p и выдавать объект x

**|p|** - длина бинарного кода p

**Пример**

Допустим, мы хотим рассчитать Колмогоровскую сложность строки "Hello, world!". Самый короткий бинарный код, который может быть использован для описания этой строки, — это сама строка, закодированная в бинарном формате. Длина этой строки составляет 96 бит (12 символов x 8 бит на символ). Поэтому Колмогоровская сложность строки "Hello, world!" составляет 96 бит.

**Важные свойства**

Колмогоровская сложность имеет следующие важные свойства:

* Асимптотическая неопределенность: Колмогоровская сложность не может быть точно рассчитана для большинства объектов, потому что это проблема остановки для машины Тьюринга.
* Аддитивность: Колмогоровская сложность объекта, состоящего из двух частей, равна сумме Колмогоровских сложностей этих частей.
* Непереводимость: Колмогоровская сложность не зависит от используемого языка программирования или машины Тьюринга.

**#3** **Списки (массивы) в Python**

Списки — это тип данных в Python, который позволяет хранить коллекцию элементов одного типа или разных типов. Списки являются изменяемыми, то есть их содержимое может быть изменено после создания.

**Создание списка**

Список создается с помощью квадратных скобок и элементов, разделенных запятыми. Например:

**Индексация списка**

Каждый элемент списка имеет свой индекс, который начинается с 0. Индекс используется для доступа к элементу списка. Например:

**Методы списка**

Списки имеют несколько методов, которые могут быть использованы для манипуляции элементами списка:

добавляет элемент в конец списка

добавляет все элементы из iterable в конец списка

вставляет элемент на указанный индекс

удаляет первый элемент, равный указанному

удаляет элемент на указанном индексе и возвращает его

сортирует список в порядке возрастания

сортирует список в порядке убывания

**Примеры использования методов списка**

Содержание

1.Гибкие методологии проектирования2

2.Колмогоровская сложность 4

3.Списки (массивы) в Python 6