1 Первая лаба

Программа — набор команд и данных, которые позволяют аппаратному обеспечению выполнять вычисления или функции управления.

Программное обеспечение (ПО, software) — программа или множество программ, которые используются для управления компьютером.

Системное ПО (system software) — ПО, которое управляет компонентами компьютера (процессор, сетевое оборудование, устройства ввода-вывода) и предоставляет сервисную поддержку прикладному ПО.

Инструментальное ПО (programming software) — ПО для разработки программ (генераторы документации, текстовые редакторы, компиляторы, отладчики и т.д.).

Прикладное ПО (приложение, application software) — ПО, предназначенное для выполнения задач, которые напрямую не касаются работы компьютера. Обычно рассчитано на непосредственное взаимодействие с пользователем.

Программный продукт (ПП) — ПО для широкого распространения или продажи. **Коробочный** ПП предоставляется с одним набором функций для всех пользователей. В **заказном** ПП набор функций определяется требованиями конкретного заказчика.

Жизненный цикл ПП — процесс, который начинается в момент принятия решения о создании ПП и заканчивается в момент полного изъятия из эксплуатации этого ПП.

Основные этапы процесса разработки ПО:

- планирование,
- сбор и анализ требований,
- разработка архитектуры,
- кодирование,
- тестирование,
- документрирование,
- внедрение,
- сопровождение.

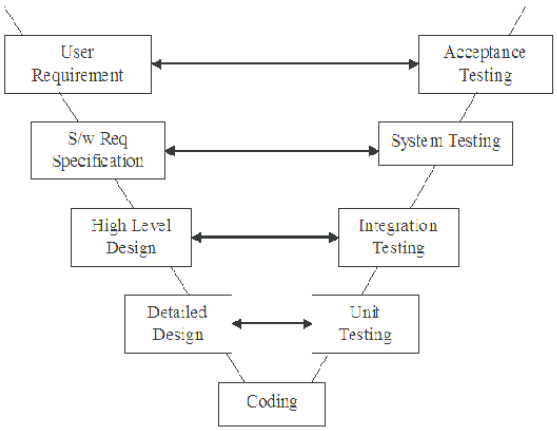
Модели жизненного цикла ПО:

- Каскадная (водопадная). Этапы идут строго друг за другом. Список требований строго фиксирован, необходима высокая квалификация исполнителей, сроки и затраты хорошо планируются.

- Итеративная или инкрементная. В первом случае следующая версия создается путем улучшения предыдущей, во втором — дополнения предыдущей.

- Спиральная. Сочетает каскадную и итеративную модель. Каждый виток спирали соответствует созданию новой версии, скорость работы определяется длиной витка. Акцент на анализе рисков, который выполняется на каждом витке.

- V-модель. Идея заключается в параллельном тестировании каждого этапа разработки.



Проект — уникальный процесс, в ходе выполнения которого получают уникальный продукт. Имеются требования, ограничения по ресурсам, времени, контролируется качество.

Команда разработки ПП (роли):

- руководитель проекта,
- аналитик,
- архитектор,
- инженер по тестированию,
- технический писатель.

Тест-кейс (ТК, test case) — единица тестирования, целью которой является нахождение одного дефекта (несоответствия поведения системы требованию). Последовательность тест-кейсов называется **тестовым сценарием**.

Методология «черный ящик» заключается в разработке тест-кейсов без рассмотрения исходного кода.

Техники тестирования черного ящика:

- Эквивалентное разбиение. **Классом эквивалетности** (КЭ, equivalence class) называется множество данных, которые в данном контексте считаются неразличимыми. Один ТК должен покрывать один КЭ. Например, если мы тестируем функцию $f : Z \to Z$, то достаточно использовать одно целое число в качестве аргумента, чтобы понять, что функция работает.

  КЭ формируются из имеющихся требований и группируются в правильные и неправильные классы. Правильный класс содержит данные, которые не нарушают условия, неправильный — данные, которые нарушают **одно** условие (описывают строго один недочет).

  Может возникнуть желание описать несколько правильных классов одним регулярным выражением, но это не информативно. Если условия допускают несколько наборов данных, отличных друг от друга по какому-нибудь признаку, то следует проверить каждый из этих наборов.

Обычно классы определяются при помощи текстового описания. Стандарта описания нет, поэтому следует избегать формальных обозначений и общих фраз, чтобы не допустить неоднозначного понимания. Описание класса следует приводить подробно, чтобы не было необходимости смотреть в требования, так как это отнимает время. В описании не следует использовать понятие бесконечности, так как это затруднит тестирование.

Пример описания: «значение содержит меньше 7 или больше 12 символов, имеется хотя бы одна заглавная буква русского алфавита, хотя бы одна цифра, хотя бы один символ из списка: ?, %, *, возможно наличие строчных букв русского алфавита, других символов значение не содержит».

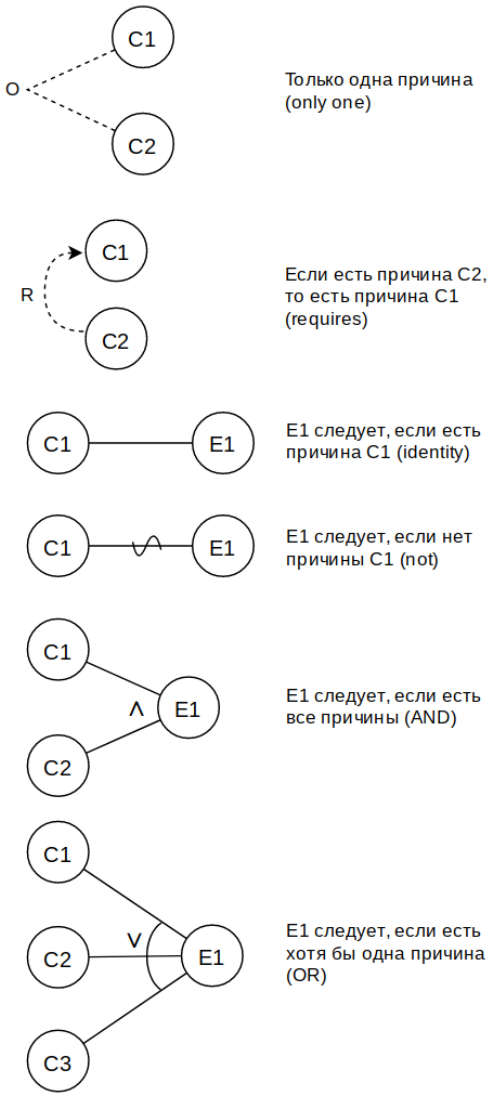
- Анализ граничных значений (boundary value testing). Такие значения возникают на верхних и нижних границах входных КЭ. Если граница одна (например, слово состоит из b символов), то граничными значениями будут: $b - \epsilon$, b, $b + \epsilon$. Если границы две (например, номер должен принадлежать отрезку от a до b), то граничные значения: $a - \epsilon$, a, b, $b + \epsilon$.

- Анализ причинно-следственных связей (decision table testing). Тестировщик формирует набор причин и следствий. Под причиной понимается входной КЭ, под следствием — результат системы на данном КЭ. Результатом анализа является таблица решений, которая отражает все возможные комбинации причин и следствий. Если имеются невозможные комбинации, то это поясняется в примечании к таблице. Пример упрощенной небинарной таблицы:

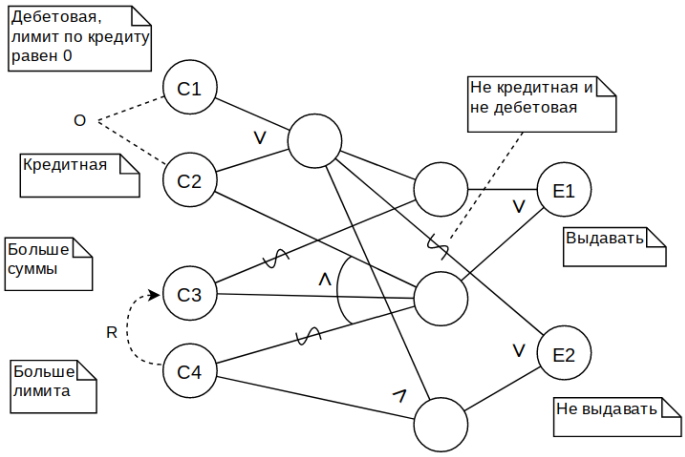
Кейс	1	2	3
Условие			
Пол	М	М	Ж
Возраст	Мол.	Зрел.	Пож.
Действие			
Предложить напиток	Сок	Кофе	Чай
Дать кредит	0	1	0

ТК создается для каждого столбца.

Функциональная диаграмма (cause-effect graph) позволяет упростить или автоматизировать построение сложной таблицы решений. Нотация диаграммы:



Пример диаграммы для функции выдачи денег у банкомата:



- Предположение об ошибке. Тестировщик с большим опытом формирует список возможных ошибок без использования конкретных методов (эвристиче-

ский подход).