Министерство науки и образования Украины

Харьковский национальный университет им. В.Н. Каразина

Факультет компьютерных наук

Индивидуальное задание

по курсу «Управление IT – проектами»

на тему:

**«Жизненный цикл проекта и надежность программного обеспечения»**

Выполнил:

Лысяк Владислав

студент группы КС-41

Харьков, 2017

**Жизненный цикл проекта**

Жизненный цикл проекта – это набор, как правило, последовательных и иногда перекрывающихся фаз проекта, названия и количество которых определяются потребностями в управлении и контроле организации или организаций, вовлеченных в проект, характером самого проекта и его прикладной областью. Жизненный цикл может документироваться с помощью методологии. Жизненный цикл проекта может определяться или формироваться уникальными аспектами организации, отрасли промышленности или используемой технологии. Поскольку каждый проект имеет определенное начало и конец, конкретные результаты и действия, имеющие место в этом промежутке, широко варьируются для каждого проекта. Жизненный цикл обеспечивает базовую структуру для управления проектом, независимо от включенных в него конкретных работ.

Проекты различаются по размеру и сложности. Независимо от размеров и степени сложности, все проекты могут быть представлены в виде жизненного цикла со следующей структурой (см. рисунок 1):

* начало проекта;
* организация и подготовка;
* выполнение работ проекта;
* завершение проекта.

Обобщенная структура жизненного цикла, как правило, отображает следующие характеристики:

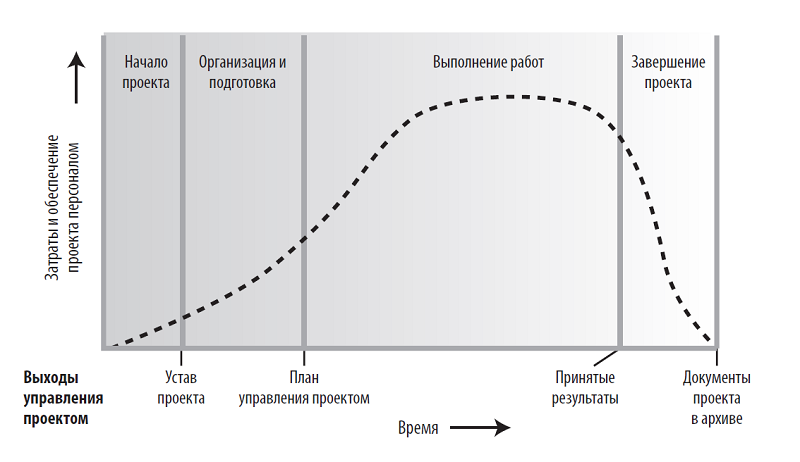
Стоимость и вовлечение персонала в проект невелики в начале, достигают пикового значения по мере выполнения работ и стремительно падают на этапе завершения проекта. Пунктирная линия на рис. 1 отображает данный типовой пример.   


Рисунок 1. Типовые уровни стоимости и обеспечения проекта персоналом на протяжении жизненного цикла проекта

Влияние заинтересованных сторон проекта, риск и неопределенность (как показано на рис. 2) имеют наибольшие значения в начале проекта. Эти факторы уменьшаются по ходу проекта.

Способность влиять на конечные характеристики продукта проекта без существенного влияния на стоимость имеет наивысшее значение в начале проекта и уменьшается по мере продвижения проекта к завершению. На рис. 2 отражена идея, что стоимость изменений и коррекции ошибок, как правило, существенно возрастает по мере приближения к завершению проекта.



Рисунок 2. Влияние переменной, основанной на сроках проекта

Фазы проекта – это отдельные части в рамках проекта, требующие дополнительного контроля для эффективного получения основного результата проекта. Фазы проекта обычно выполняются последовательно, но в некоторых проектных ситуациях могут перекрываться. Высокоуровневый характер фаз проекта превращает их в элемент жизненного цикла проекта. Фаза проекта не является группой процессов управления проектом.

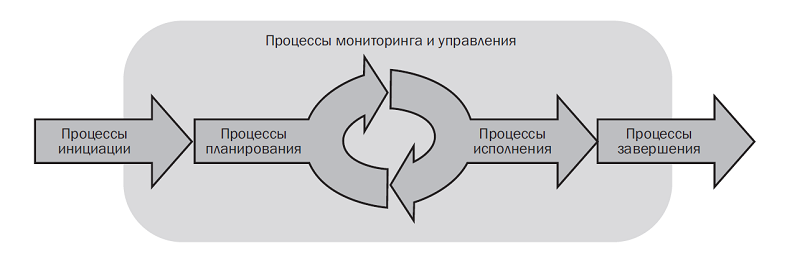
Структура фаз позволяет разделить проект на логические подгруппы для более легкого управления, планирования и контроля. Количество фаз, необходимость в них и степень налагаемого контроля зависят от размера фаз, сложности и потенциального влияния на проект.

Рисунок 3. Пример однофазового проекта

В рамках методологии [Института управления проектами](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D0%BF%D0%BE_%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8E_%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Project Management Institute*) жизненный цикл проекта имеет 5 фаз:

1. Инициализация ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Initialization*);
2. Планирование ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Planning*);
3. Выполнение ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Executing*);
4. Контроль и мониторинг ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Controlling and Monitoring*);
5. Завершение ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Closing*).

**Надежность программного обеспечения**

Надежность программного обеспечения -способность программного продукта безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью.

Степень надежности характеризуется вероятностью работы программного продукта без отказа в течение определенного периода времени.

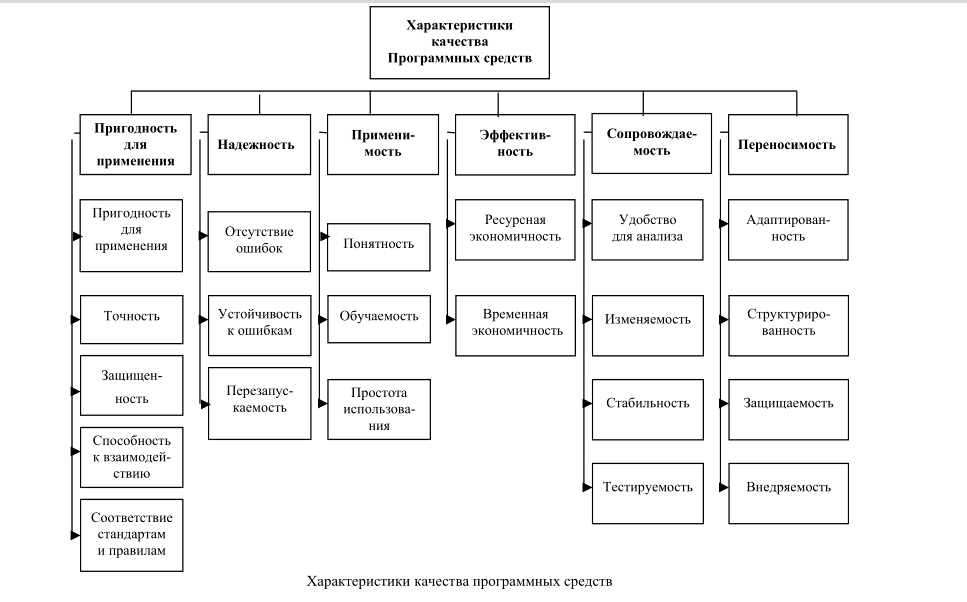
Основные показатели качества и надежности программных средств представлены на рисунке.

Рисунок 4. Характеристики качества программных средств

**Показатели надежности**

*Вероятность безотказной работы* – вероятность того, что в пределах заданной наработки (времени или объема работы) не возникнет отказ

*Средняя наработка до отказа* (Mean Time To Failure) – мат. ожидание наработки объекта до первого отказа

*Средняя наработка на отказ* (Mean Time Between Failures) – отношение суммарной наработки к мат. ожиданию числа отказов в течении этой наработки

*Интенсивность отказов* – плотность вероятности возникновения отказов

*Коэффициент готовности* – вероятность того, что объект окажется в работоспособном состоянии в произвольный момент времени

*Средний срок службы*

*Вероятность восстановления*

**Ошибки** в ПО - все возможные несоответствия между демонстрируемыми характеристиками его качества и сформулированными или подразумеваемыми требованиями и ожиданиями пользователей.

**Недостаток** (**Fault**) **/ Дефект** (**Defect**)– некорректное поведение системы, вызывающее сбой

**Сбой** (**Failure**) – наблюдаемый нежелательный эффект, вызванный дефектами в системе

**Ошибка** (**Error**) – человеческое действие, которое вызывает некорректный результат работы системы

**Классификация ошибок**

*Функциональные ошибки* – нарушения программной спецификации (несоответствие функциональным или нефункциональным требованиям). Приводят к ухудшению функциональности ПО (пригодность, точность)

*Нефункциональные ошибки* – нарушения правил языка программирования, неправильное использование библиотечных функций и т.п. Приводят к снижению надежности (зрелости) и ухудшению функциональности (защищенности)

**Методы измерения и определения количественных показателей надежности**

**Аналитичесике.** Базируются на данных о поведении программы в процессе тестирования.

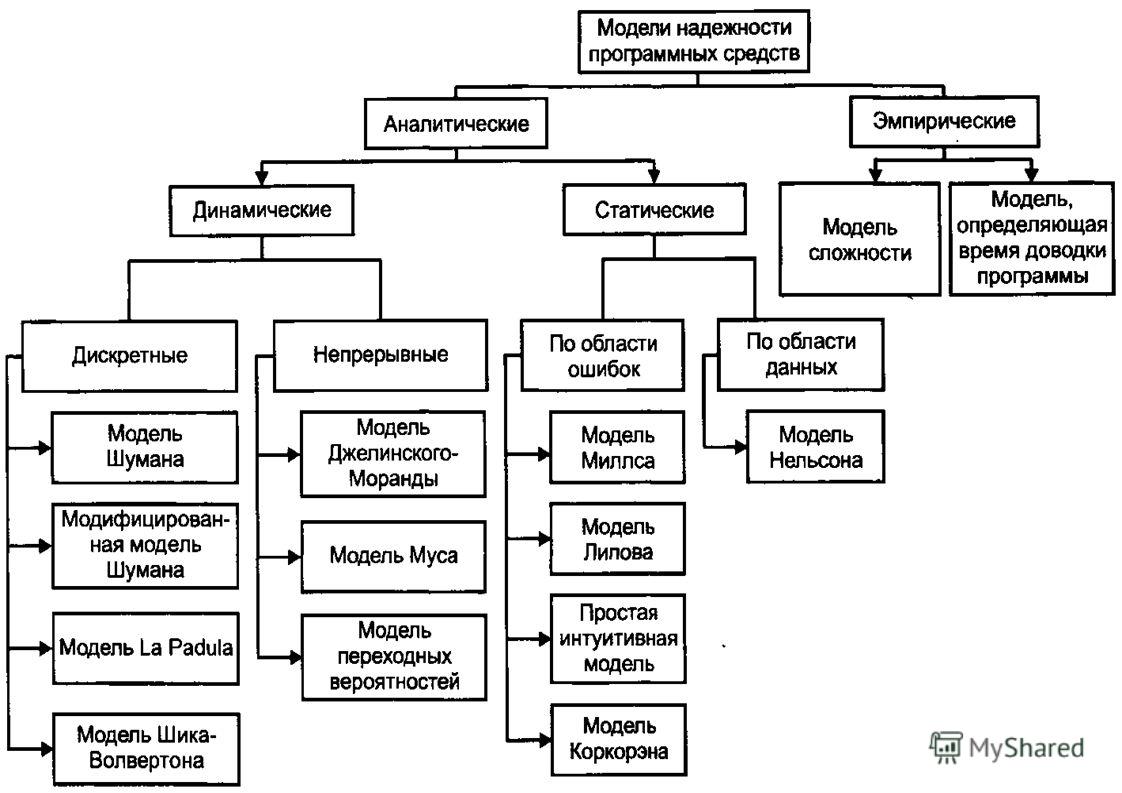
**Эмпирические.** Базируются на анализе структурных особенностей программы (линейные разветвления программы)

**Динамические.** Появление отказов программных средств рассматривается во времени

**Статистические.** Учитывает зависимость количества ошибок от числа тестовых прогонов или от характеристики входных данных

**Непрерывные.** Фиксируют число отказов за произвольный интервал времени

**Дискретные.** Фиксируют интервалы каждого отказа



**Аналитические модели**

*Статистическая модель надежности Миллса*

Использование этой модели предполагает необходимость перед началом тестирования искусственно вносить в программу (засорять) некоторое количество известных ошибок. Ошибки вносятся случайным образом и фиксируются в протоколе искусственных ошибок. Специалист, проводящий тестирование, не знает ни количества ошибок, и характера внесенных ошибок до момента оценки показателей надежности по модели Миллса. Предполагается, что все ошибки (как естественные, так и искусственно внесенные) имеют равную вероятность быть найденными в процессе тестирования. Тестируя программу в течение некоторого времени, собирается статистика об ошибках. В момент оценки надежности по протоколу искусственных ошибок все ошибки делятся на собственные и искусственные.

Модель надежности Миллса образуется двумя связанными между собой по смыслу соотношениями.

Первое соотношение предсказывает, N - первоначальное количество ошибок в программе. В данном соотношении, которое называется формулой Миллса, S - количество искусственно внесенных ошибок, n - число найденных собственных ошибок, V - число обнаруженных к моменту оценки искусственных ошибок.

Предположим, что в программе имеется K собственных ошибок, и внесем в нее еще S ошибок. Если в процессе тестирования были обнаружены все S внесенных ошибок и n собственных ошибок, то по формуле Миллса мы предполагаем, что первоначально в программе было N = n ошибок.

Второе соотношение используется для установления доверительного уровня прогноза С (вероятности, с которой можно высказать предположение об N). Величина С является мерой доверия к модели.

Если обнаружены все искусственно рассеянные ошибки (V = N) вероятность того, что значение N найдено правильно, можно рассчитать по следующему соотношению:

Достоинством модели является простота применения математического аппарата, наглядность и возможность использования в процессе тестирования. Однако, модель не лишена и ряда недостатков, самые существенные из которых - это необходимость внесения искусственных ошибок (этот процесс плохо формализуется) и достаточно вольное допущение величины K, которое основывается исключительно на интуиции и опыте человека, проводящего оценку, т.е. допускается большое влияние субъективного фактора.

*Динамическая модель Шумана*

**Модель Шумана** относится к динамическим моделям дискретного времени. Исходные данные для модели Шумана собираются в процессе тестирования ПС в течение фиксированных или случайных временных интервалов. Каждый интервал – это стадия, на которой выполняется последовательность тестов и фиксируется некоторое число ошибок.

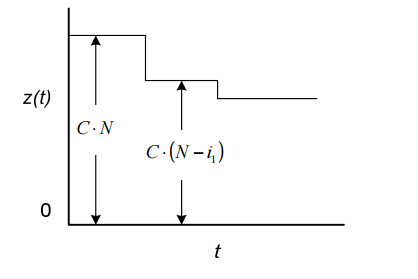
Модель Шумана может быть использована при определенным образом организационной процедуре тестирования. Использование модели Шумана предполагает, что тестирование проводится в несколько этапов. Каждый этап представляет собой выполнение программы на полном комплексе разработанных тестовых данных. Выявленные ошибки регистрируются, т.е. собирается статистика об ошибках. Но они не исправляются. По завершении этапа на основе собранных данных о поведении ПС на очередном этапе тестирования может быть использована модель Шумана для расчета количественных показателей надежности. После этого исправляются ошибки, обнаруженные на предыдущем этапе. При необходимости корректируются тестовые наборы и проводится новый этап тестирования. При использовании модели Шумана предполагается, что исходное количество ошибок в программе постоянно и в процессе тестирования может уменьшаться по мере того, как ошибки выявляются и исправляются. Новые ошибки при корректировке не вносятся. Скорость обнаружения ошибок пропорциональна числу оставшихся ошибок. Общее число машинных инструкций в рамках одного этапа тестирования постоянно.

Предполагается что в начальный момент в программе N ошибок.

Выполняется несколько этапов тестирования, на которых обнаруживаются ошибки, исправление ошибок выполняется в конце этапов.

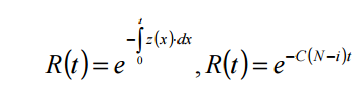
Функция риска z(t) – условная вероятность появления ошибки на интервале , при условии, что до момента t ошибок не было.

Cчитается, что z(t) не изменяется между этапами тестирования и пропорциональна числу оставшихся ошибок

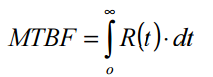


Вероятность отсутствия отказа на интервале времени

При R(0) = 1 решение этого уравнения



Среднее время между отказами



Неизвестные значения C и N определяются по результатам нескольких экспериментов с помощью функции максимального правдоподобия.

**Эмпирические модели**

*Модель сложности*

Сложность ПО характеризуется его размером (количеством программных модулей), количеством и сложностью межмодульных интерфейсов. Под программным модулем в данном случае следует понимать программную единицу, выполняющую определенную функцию и взаимосвязанную с другими модулями ПО. Существует несколько разновидностей модели сложности. В каждой из них дается некая оценка сложности программы, которая считается пропорциональной ее надежности.

*Модель, определяющая время доводки программ*

Анализ модульных связей ПО строится на том, что каждая пара модулей имеет конечную (возможно, нулевую) вероятность того, что изменения в одном модуле вызовут изменения в другом. Данная модель позволяет на этапе тестирования, а точнее при тестовой сборке системы, определять возможное число необходимых исправлений и время, необходимое для доведения ПО до рабочего состояния.

**Литературные источники**

1. A Guide to the Project Management Body of Knowledge (PMBOK Guide). <http://www.cs.bilkent.edu.tr/~cagatay/cs413/PMBOK.pdf>
2. Методы оценки надежности прикладного программного обеспечения технического назначения.

<http://cyberleninka.ru/article/n/metody-otsenki-nadezhnosti-prikladnogo-programmnogo-obespecheniya-tehnicheskogo-naznacheniya>

1. Н.В.Василенко, В.А.Макаров. Модели оценки надежности программного обеспечения.
2. ISO 9000:2015. Системы менеджмента качества – Основные положения и словарь.
3. Формальные методы обеспечения качества ПО.

<http://kspt.icc.spbstu.ru/media/files/2010/course/softwarequality/lec2.pdf>