УДК 338.242, 004.9

ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Лопоха Д.В. [1]

[1] Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики», факультет технологического менеджмента и инноваций
(197101, Россия, Санкт-Петербург, пр. Кронверкский, д.49), e-mail: ftmi@mail.ifmo.ru

Современные информационные технологии с их стремительно растущим потенциалом и быстро снижающимися издержками становятся сегодня одним из ключевых показателей конкурентоспособности практически любых форм организации труда и занятости. Одновременно с усложнением и растущими возможностями цифровой техники для бизнеса революционным образом усложняется, а с другой стороны, становиться более комфортным, удобным и всеохватывающим программное обеспечение. Каждая разработка требует слаженной работы профессиональной команды. Очень сложно выполнять сложную командную работу, такую как разработка программного обеспечения, без какого либо плана. Основой такого создания программного обеспечения является методология разработки. Существует много споров о том, какой метод лучше всего подходит для определённого типа ПО и соответственно множество стандартов, которые регламентируют жизненный цикл ПО, а в некоторых случаях даже процессы обработки. В работе рассмотрены такие популярные стандарты как ГОСТ 34.601-90 для проектирования автоматизированных систем, Международный стандарт ISO/IEC 12207 (ГОСТ Р ИСО/МЭК 12207: 1995) , который является базовым стандартом процессов жизненного цикла ПО, ISO / IEC 12207: 2008 для создания систем и программных продуктов и услуг. В работе рассмотрены разные модели жизненного цикла программного обеспечения, которые определяют характер процесса его создания, представляющий собой совокупность упорядоченных во времени, взаимосвязанных и объединённых в стадии (фазы) работ, выполнение которых необходимо и достаточно для создания ПО, соответствующего заданным требованиям.

Ключевые слова: программное обеспечение, жизненный цикл программного обеспечения, модель жизненного цикла, стандарт программного обеспечения.

SOFTWARE LIFECYCLE

Lopoha L.V. [1]

[1] Federal State Autonomous Educational Institution of Higher Education "St. Petersburg National Research University of Information Technologies, Mechanics and Optics," Faculty of Technological Management and Innovation
(197101, Russia, St. Petersburg, Ave. Kronverksky, 49), e-mail: ftmi@mail.ifmo.ru

Modern information technologies, with their rapidly growing potential and rapidly declining costs, are now becoming a key indicator of the competitiveness of virtually any form of labour and employment organization. Along with the growing complexity and opportunities of digital technology for business, it becomes revolutionized, and on the other hand, more comfortable, convenient and inclusive software. Each development requires the coherent work of a professional team. It is very difficult to do complex teamwork such as software development without any plan. The basis for such software development is the development methodology. There is much debate as to which method is best suited for a particular type of software and accordingly a variety of standards that govern the software lifecycle, and in some cases even processing processes. The work deals with such popular standards as GOST 34.601-90 for the design of automated systems, International standard ISO/IEC 12207 (GOST

R ISO/IEC 12207:1995), which is the basic standard of software life cycle processes, ISO/IEC 12207:2008 for the creation of systems and software products and services. The work deals with different models of software life cycle, which define the nature of the process of its creation, which is a set of ordered in time, interconnected and combined in the stage (phase) of works, the performance of which is necessary and sufficient to create software that meets the specified requirements.
Keywords: software, life cycle of the software, model of life cycle, standard of the software.

ВВЕДЕНИЕ

Программное обеспечение - это набор инструкций, данных или программ, используемых для управления компьютерами и выполнения определенных задач. В отличие от аппаратного обеспечения, которое описывает физические аспекты компьютера, программное обеспечение - это общий термин, используемый для обозначения приложений, сценариев и программ, работающих на устройстве. Программное обеспечение можно рассматривать как переменную часть компьютера, а аппаратную часть - неизменную часть [1].

У программного обеспечения есть свой жизненный цикл - ряд событий, происходящих с системой в процессе ее создания и дальнейшего использования. Его этапы охватывают полный жизненный цикл программного обеспечения, то есть с момента его создания до вывода продукта из эксплуатации.

Присоединение к процессу жизненного цикла приводит к разработке программного обеспечения на систематической и дисциплинированной основе.

В данном реферате мы подробнее рассмотрим известные модели жизненного цикла ПО, их преимущества и недостатки , стандарты программного обеспечения, разработку ПО, а также выделим наиболее удобную модель ЖЦПО и сделаем вывод о успешной реализации разработки программного обеспечения.

Стандарты жизненного цикла ПО

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

Существует множество стандартов, которые регламентируют жизненный цикл ПО, а в некоторых случаях даже процессы обработки.

Среди наиболее популярных стандартов, можно выделить следующие:

- ГОСТ 34.601-90 – данный стандарт распределяется на автоматизированные системы, которые применяются в разнообразных видах деятельности. Например, в различных исследованиях, проектированиях , управлениях. Немаловажным является и то,

что стандарт устанавливает стадии и этапы развития АС. Этапы работ и их стадии , которые закреплены в стандарте, в большей степени относятся к каскадной модели жизненного цикла [2].

- Международный стандарт ISO/IEC 12207 (ГОСТ Р ИСО/МЭК 12207: 1995) [3] является базовым стандартом процессов жизненного цикла ПО, который ориентируется на различные виды программного обеспечения и типы проектов ИС. Данный стандарт определяет стратегию и общий порядок в создании и эксплуатации.
- ISO / IEC 12207: 2008 применяется к приобретению систем и программных продуктов и услуг, к поставке, разработке, эксплуатации, техническому обслуживанию и утилизации программных продуктов и программной части системы, независимо от того, выполняются ли они внутри или снаружи организации. Включены те аспекты определения системы, которые необходимы для обеспечения контекста для программных продуктов и услуг.

Этапы разработки жизненного цикла ПО

Очень сложно выполнять сложную командную работу, такую как разработка программного обеспечения, без какого либо плана. Каждая методология разработки программного обеспечения является его основой. Существует много споров о том, какой метод лучше всего подходит для определенного типа ПО.

Рассмотрим следующие этапы разработки жизненного цикла программного обеспечения:

1. Планирование

Это самый первый этап, который сначала определяет, существует ли необходимость в новой системе для достижения стратегических целей бизнеса. После чего разрабатывается план высокого уровня с бизнес-намерениями закупить ресурсы, необходимые для создания, изменения или обновления службы или решения. Цель этого шага - определить масштаб проблемы и найти решения. Термин ресурсы относится к требованию затратам, времени, выгодам и нескольким другим важным элементам.

На этом этапе также происходит идентификация деятельности по обеспечению качества (обеспечение качества), оценка рисков проекта. Далее, технико-экономическое обоснование является своего рода техническим анализом, целью которого является поиск наиболее эффективных способов завершить проект без какого-либо риска.

2. Анализ

Этот этап начинается со сбора команды и оценки функциональных требований проекта. Это один из начальных и критических этапов SDLC. Он проводится старшими разработчиками / тестировщиками команды с информацией от клиента, предпродажных, маркетинговых исследований и специалистов в данной области. Эти входные данные помогают в планировании проектного подхода и проведении технико-экономического обоснования на основе финансовых, операционных и технических аспектов.

После прохождения анализа требований следующее действие- опубликовать четкое определение каждого из них в форме документа. Вы можете поделиться им с клиентом или бизнес-аналитиком для утверждения. Этот артефакт является спецификацией требований к программному обеспечению. В нем перечислены все требования к продукту с достаточным количеством деталей, необходимых для начала проектирования и разработки.

3. Проектирование

На этом этапе проводится проектирование программы на низком уровне, иными словами, здесь проектируются классы и методы, рассматриваются, оцениваются и сравниваются различные варианты и причины выбора окончательных подходов и способов реализации.

При разработке небольших программ программисты обычно сами проектируют программу на данном уровне, это выглядит как написание псевдокода или рисование схем, поэтому часто этот этап рассматривается как часть непосредственного кодирования и в таких случаях итоговый документ (если того требует формальность) состоит преимущественно из различных набросков и заметок программистов. Но при реализации крупных проектов данному процессу отводится отдельный этап и проектирование в этом случае проводится с очень высокой степенью детальности.

4. Реализация

Эта стадия имеет много названий, таких как этап разработки или кодирования или реализации. Именно здесь начинается настоящее развитие, следуя руководящим указаниям по проектированию. Команда разработчиков пишет код для каждого модуля согласно определению, установленному DDS. Хорошо написанный проектный документ, который имеет достаточные, структурированные и подходящие детали, может сделать кодирование относительно простым и помочь разработчику завершить работу вовремя.

Каждая организация имеет своего рода стандарты кодирования, руководящие принципы и лучшие практики, которые предназначены для производства качественного и многократно используемого кода. Все программисты должны знать и практиковать их, работая над задачей разработки. Он или она должны помнить об IDE, компиляторах

(например, GCC / MSVC), интерпретаторах (например, Python LINT) и отладчиках (например, WINDBG, GDB). Выбор языка программирования зависит от природы программного обеспечения для сборки, а также от его способности развиваться быстрее.

5. Тестирование

Тестирование происходит практически на всех этапах жизненного цикла разработки программного обеспечения. Например: проверка SRS, DDS, модульное тестирование отдельных модулей, все такие действия являются некоторыми формами проверки [4 6].

Несмотря на то, что продукт требует тщательного тестирования, чтобы подтвердить, что каждый компонент и все функции работают в соответствии с требованиями заказчика, данный этап, на котором команда разрабатывает план тестирования, пишет тестовые случаи, регистрирует дефекты, выполняет регрессию и следит за тем, чтобы программное обеспечение достигло наивысшего из стандартов качества.

6. Сопровождение, внесение изменений, оптимизация.

После запуска программы в промышленную эксплуатацию осуществляется сопровождение этой программы, т.е. внесение изменений на основе выявленных недочетов в процессе эксплуатации системы, а также проводится оптимизация функционала или добавление нового.

Модели жизненного цикла ПО

Модель ЖЦ любого конкретного ПО определяет характер процесса его создания, который представляет собой совокупность упорядоченных во времени, взаимосвязанных и объединенных в стадии (фазы) работ, выполнение которых необходимо и достаточно для создания ПО, соответствующего заданным требованиям.

Процесс жизни любой системы или программного продукта может быть описан посредством модели жизненного цикла, состоящей из стадий. Модель жизненного цикла представляется в виде последовательности стадий, которые могут перекрываться и (или) повторятся циклически в соответствии с областью применения, сложность действий, размером, а также потребностью в изменениях и возможностями. Каждая стадия является важной и непосредственно описывается формулировкой цели и выходов. Процессы и действия жизненного цикла отбираются и исполняются на этих стадиях для полного удовлетворения цели и результатов этой стадии.

Каскадная модель жизненного цикла

Данная модель обязана своим появлением У. Ройсу (1970 г.). Модель имеет и другое название – водопад (waterfall). Особенность данной модели проявляется в том, что переход на следующую ступень осуществляется только после того, как будет полностью завершена работа на предыдущей стадии; возвратов на пройденные стадии не предусматривается.

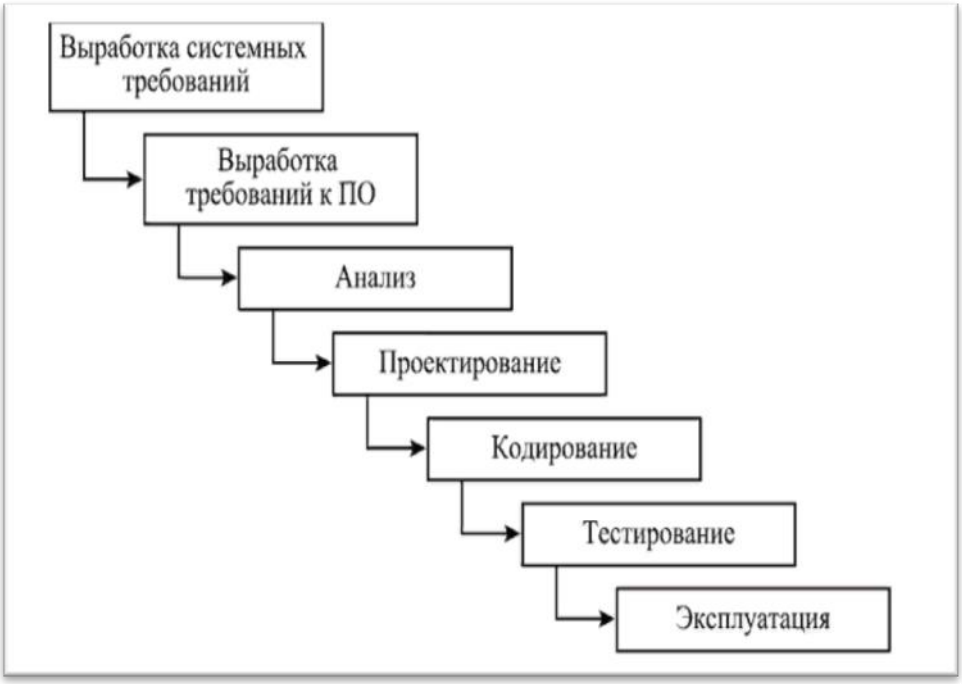

Рис.1- Каскадная модель ПО(водопад)

Этапы каскадной модели:

1. Выработка системных требований – начальный этап разработки ПО, установка и создание условий, при которых будет работать ПО.

2. Выработка требований к ПО – разработка целей, для которых создается программное обеспечение.

3. Анализ – анализ стоимости создания, а также всех положительных и отрицательных сторон разработки ПО.

4. Проектирование – процесс разработки ПО, при котором создаются все его функции, удовлетворяющие целям и средствам проекта.

5. Кодирование – перевод ПО в двоичную систему для того, чтобы он мог выполнять необходимые задаваемые программы.

6. Тестирование – проверка в работе ПО, проверка всех функций, для которого оно создавалось.

7. Эксплуатация – выпуск ПО в глобальное пользование, контроль за работой, создание необходимых обновлений [5].

Преимущества:

- Последовательное выполнение этапов проекта в строгом фиксированном порядке
- Позволяет оценивать качество продукта на каждом этапе

Недостатки:

- Отсутствие обратных связей между этапами
- Не соответствует реальным условиям разработки программного продукта

Спиральная модель жизненного цикла

Спиральная модель представляет собой комбинацию итерационной модели и модели жизненного цикла программного обеспечения. Это можно сравнить с тем , будто вы выбираете данную модель и комбинируете ее с итеративной моделью циклического процесса.

Эта модель учитывает риск, который часто остается незамеченным большинством других моделей. Модель начинается с определением целей и ограничений программного обеспечения в начале одной итерации. Следующая фаза прототипирования программного обеспечения. Это включает в себя анализ рисков. Тогда одна стандартная модель жизненного цикла ПО используется для построения программного обеспечения. На четвертом этапе готовится план следующей итерации

Преимущества:

- Быстрое получение результата;
- Повышение конкурентоспособности;
- Изменяющиеся требования — не проблема.

Недостатки:

- Отсутствие регламентации стадий.

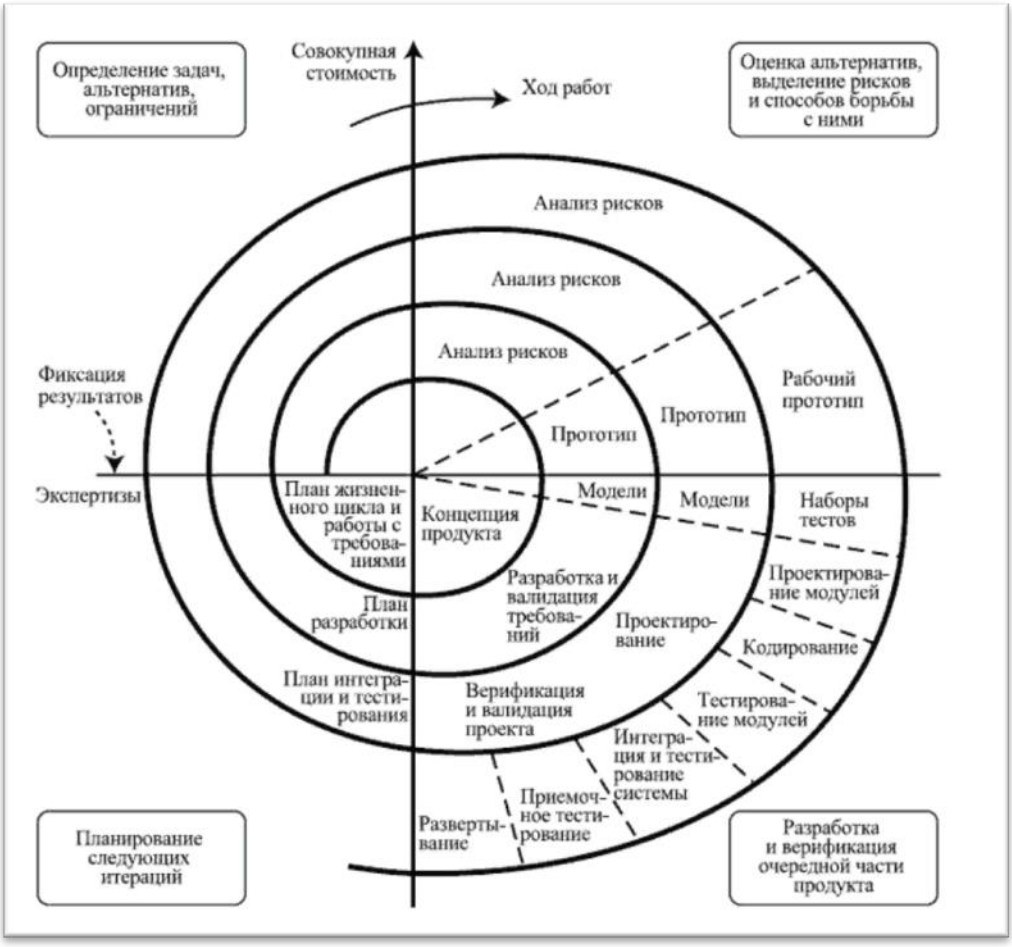

Рис.2 – Спиральная модель ПО

V-модель жизненного цикла

Модель V является одной из методологий жизненного цикла программного обеспечения , где разработка проекта завершается линейным образом, образуя V-образную форму. Эта модель схожа с каскадной моделью, так как следующий шаг начинается только после выполнения предыдущего. Данная модель используется при разработке систем, для которых требуется высокая надежность [4]. Данная модель, её форма, демонстрирует взаимосвязь между каждым этапом жизненного цикла разработки и связанной с ней этапом тестирования. V-модель ввела концепцию ранней проверки и верификации, и её можно использовать повторно , а также комбинировать с другими моделями.

Преимущества:

- планирование на ранних стадиях разработки системы ее тестирования;
- обеспечение аттестации и верификации всех промежуточных результатов разработки;

- упрощение (по сравнению с каскадной моделью) отслеживания хода процесса разработки, возможность более реального использования графика проекта;
- простота в использовании.

Недостатки:

- сложность поддержки параллельных событий;
- поздние сроки тестирования требований в жизненном цикле, что оказывает существенное влияние на график выполнения проекта при необходимости выполнить их изменения;
- отсутствие в модели действий, направленные на анализ рисков.

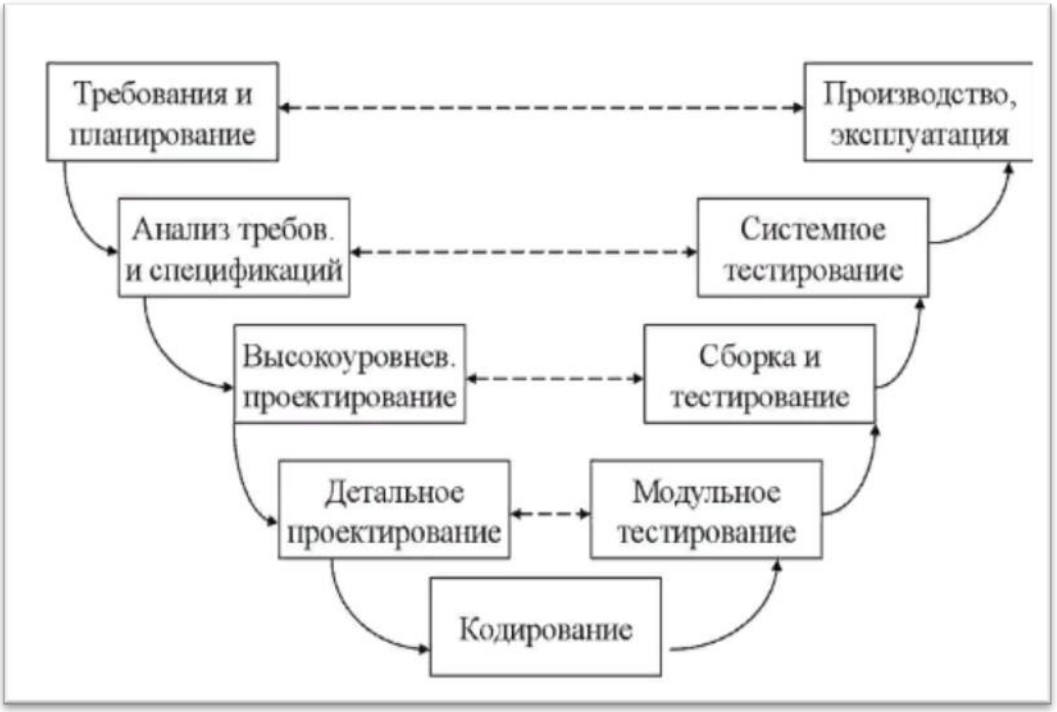

Рис.3 V-модель жизненного цикла ПО

V-модель жизненного цикла ПО состоит из 9 этапов:

1. Планирование проектов и требований – заключается в разработке целей, для которых создается программное обеспечение, создаются условия и требования для его работы.

2. Анализ требований продукта и спецификаций – в данном этапе анализируется стоимость создания, а также все положительные и отрицательные стороны разработки ПО.

3. Разработка архитектурного проекта на высоком уровне – создание макета проекта с учетом всех требований и условий использования.

4. Детализированная разработка проекта – это пошаговое создание проекта ПО по заданному макету.

5. Кодирование – перевод ПО в двоичную систему для того, чтобы он мог выполнять необходимые задаваемые программы.

6. Модульное тестирование – проверка работоспособности каждого из модулей ПО.

7. Интеграция и тестирование – проверка верности работы при вводе в устройство.

8. Системное и приемочное тестирование – это тестирование заданных функций, а также правильности приема команд и программ.

9. Производство, эксплуатация и сопровождение - выпуск ПО в глобальное пользование, контроль за работой, создание необходимых обновлений.

Заключение

В результате проделанной работы , мы познакомились с несколькими моделями программного обеспечения, с их преимуществами и недостатками , стандартами ПО, а также с этапами разработки программного обеспечения.

По моему мнению, самая практичная и театрализованная модель – это V-модель жизненного цикла ПО. Данная модель является таковой , так-как она требуется для более тщательного тестирования, обеспечивает гибкость в размещении новых проектов независимо от их размера, сложности или сроков.

Можно сделать вывод, что ПО должно быть ясным, полным по своему содержанию и пригодным для работы как с маленькими, так и с большими проблемами в соответствии со своим предназначением. Несколько ловушек могут превратить реализацию программного обеспечения скорее в препятствие для развития, чем инструмент, который может нам помочь. Неспособность учесть потребности клиентов и всех пользователей может привести к плохому пониманию системных требований, что влечет плохой результат.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Матс И.К. Модели жизненного цикла программного обеспечения – 2011. // [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/111674/. – Дата обращения: 10.12.2019.

2. Попов И.М. Что такое программное обеспечение? – 2018.// [Электронный ресурс]. – Режим доступа: http://procomputer.su/comp-gramotnost/79-chto-takoe-programmnoe-obespechennie. – Дата обращения: 13.12.2019

3. ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы стадии создания. // Москва, Стандартинформ, 2009г.

4. ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. // Дата актуализации: 01.01.2019

5. Бодренко К. Разработка требований к программному обеспечению- 2004. // [Электронный ресурс].- Режим доступа: http://bodrenko.org/ssukpo/ssukpo-l3.htm . - Дата обращения: 13.12.2019.

6. Иванов И.Д. Этапы разработки ПО-2015. // [Электронный ресурс].- Режим доступа: https://info-comp.ru/programmirovanie/724-stages-of-program-development.html/. – Дата обращения: 14.12.2019