Национальный Открытый Университет "ИНТУИТ": www.intuit.ru
Екатерина Лаврищева, Владимир Петрухин
Лекция 3. Модели жизненного цикла для разработки программных систем

За десятилетия опыта построения программных систем был наработан ряд типичных схем последовательности выполнения *работ* при проектировании и разработки ПС. Такие схемы получили название моделей ЖЦ.

Модель жизненного цикла – это схема выполнения *работ* и задач в рамках процессов, обеспечивающих разработку, эксплуатацию и сопровождение программного продукта, и отражающая эволюцию ПС, начиная от формулировки требований к ней до прекращения пользоваться ею [1.14], [[2.2- 2.6].

Исторически в эту схему *работ* включают:

- разработку требований или технического задания;
- разработку системы или технического проекта;
- программирование или рабочее проектирование;
- пробную эксплуатацию;
- сопровождение и улучшение;
- снятие с эксплуатации.

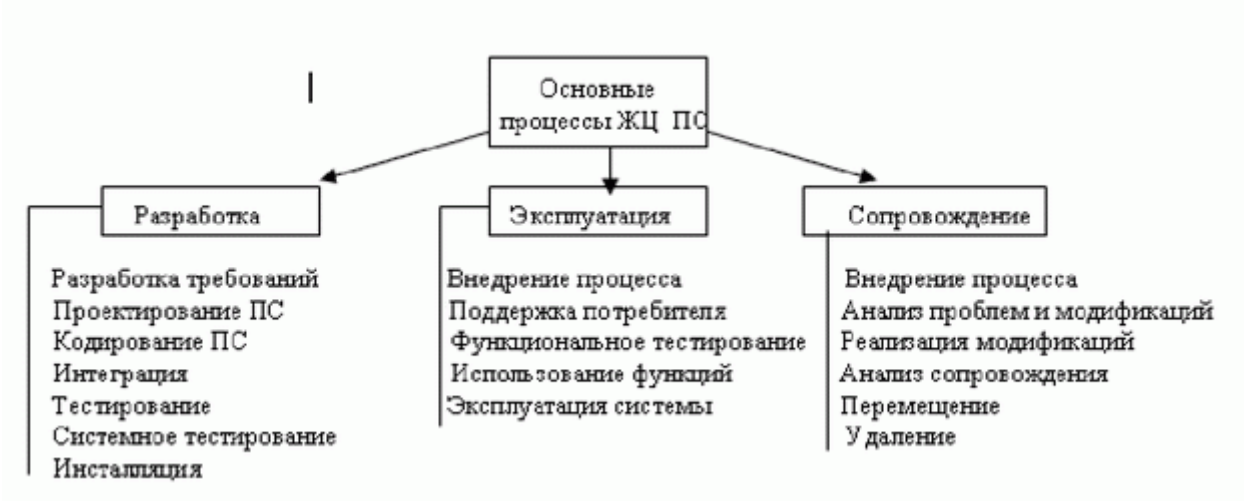
Основное назначение моделей ЖЦ состоит в следующем:

- планирование и распределение работ между разработчиками и ресурсов, а также управление программным проектом;
- обеспечение взаимодействия между разработчиками проекта и заказчиком;
- наблюдение и контроль работ, оценивание промежуточных продуктов ЖЦ на соблюдение спецификаций требований, правильное их выполнение, оценивание продукта и реальных затрат, в том числе и по применяемым программным средствам и инструментам;
- согласование промежуточных результатов с заказчиком;
- проверка правильности конечного продукта путем его тестирования на запланированных и согласованных с заказчиком наборах тестов;
- оценивание соответствия характеристик качества полученного продукта заданным требованиям;
- обсуждение используемых процессов ЖЦ в плане оценки их возможностей и недостатков, проявившихся при их применении, а также определение направлений усовершенствования или модернизации ЖЦ и др.

В связи с такими задачами, возлагаемыми на модель ЖЦ, необходимо сделать правильный выбор процессов, их задач и действий для построения модели ЖЦ ПС, которая удовлетворяет концептуальной идее проектируемой системы с учетом ее сложности и масштаба *работ*. В модель ЖЦ обязательно включаются процессы реализации *работ* и задач, обеспечивающие создание промежуточного продукта и переход к следующему процессу модели.

2.1. Процессы ЖЦ стандарта ISO/IEC 12207

При выборе схемы модели ЖЦ для конкретной *предметной области*, решаются вопросы включения важных для создаваемого продукта видов *работ* или не включения несущественных *работ*. На сегодня основой формирования новой модели ЖЦ для конкретной прикладной системы является стандарт *ISO*/IEC 12207, который задает полный набор процессов (более 40), охватывающий все возможные виды *работ* и задач, связанных с построением ПС, начиная с анализа *предметной области* и кончая изготовлением соответствующего продукта. Данный стандарт содержит основные и вспомогательные процессы (рис. 2.1 и рис. 2.2).



увеличить изображение

Рис. 2.1. Схема основных процессов ЖЦ ПС

На рис. 2.1. представлены процессы, связанные непосредственно с разработкой ПС. К категории основных процессов относятся также "первичные" процессы, определяющие порядок подготовки договора на разработку ПС, *мониторинг* деятельности поставщиков ПС заказчику.

Стандарт *ISO*/IEC 12207 предоставляет структуру процессов ЖЦ, но не обязывает использовать все процессы в модели ЖЦ *ПО* или в конкретной методологии разработки *ПО*.

увеличить изображение

Рис. 2.2. Схема вспомогательных процессов ЖЦ ПС

Являясь стандартом высокого уровня, он не задает детали того, как надо выполнять действия или задачи, составляющие процессы. Он также не задает требований к формату и содержанию документов, выпускаемых на разных процессах.

Процессы, действия и задачи приведены в стандарте в наиболее общей естественной последовательности. Это не значит, что в такой же последовательности они должны быть применены в конкретной модели ЖЦ ПС. В зависимости от проекта процессы, действия и задачи стандарта выбираются, упорядочиваются и включаются в модель ЖЦ. При применении они могут перекрывать, прерывать друг друга, выполняться итерационно или рекурсивно. Это определяет "динамический" характер стандарта и позволяет реализовать с его помощью произвольную модель ЖЦ ПС. Поэтому организации, которая намерена применить этот стандарт в своей работе, понадобятся дополнительные стандарты или процедуры, определяющие разные детали *по* применению выбранных элементов ЖЦ. Отметим, что комитет *ISO* выпускает руководства и процедуры, дополняющие стандарт 12207.

Кроме этого, стандарт *ISO*/IEC 12207 предоставляет основу для принятия ряда других связанных с ним стандартов, таких как

стандарты *по* управлению *ПО*, обеспечению качества, верификации и валидации, управлению конфигурацией, метриками *ПО* и т.д.

Из данного стандарта можно выбрать только те процессы, которые более всего подходят для реализации конкретной ПС. Обязательными являются основные процессы, которые присутствуют во всех известных моделях ЖЦ. В зависимости от целей и задач *предметной области* они могут быть пополнены дополнительными (*документирование*, обеспечение качества, *верификация* и *валидация* и т.п.) и организационными (планирование, управление и др.) процессами этого стандарта. Разработчик принимает решение о включении в новую создаваемую модель ЖЦ процесса обеспечения качества компонентов и системы управления проектом или определения набора проверочных (верификационных) процедур для обеспечения правильности продукта и соответствия его заданным требованиям.

Процессы, включенные в модель ЖЦ, предназначены для реализации стандартных задач процессов ЖЦ и могут привлекать другие процессы для реализации специализированных задач системы (например, защиты данных). Интерфейсы (входы и выходы) любых двух процессов ЖЦ должны быть минимальными и каждый из них должен удовлетворять следующим правилам:

- если процесс A вызывается процессом B и только процессом B, то A принадлежит B ;
- если функция вызывается более чем одним процессом, то она становится отдельным процессом;
- проверка любой функции в ЖЦ является обязательной.

Иными словами, если задача требуется более чем одному процессу, то она может стать процессом, используемым однократно или многократно на протяжении жизни конкретной системы. Каждый процесс должен иметь внутреннюю структуру, установленную в соответствии с тем, что он должен выполнять.

Процессы модели ЖЦ ориентированы на разработчика системы. Он может выполнять один или несколько процессов и процесс может быть выполнен одним или несколькими разработчиками. При этом один из них является ответственным за один процесс или за все процессы модели, даже если отдельные работы выполняет другой разработчик.

Создаваемая модель ЖЦ увязывается с конкретными методиками разработки систем и соответствующими стандартами в области программной инженерии либо разрабатываются самостоятельно для проекта с учетом его возможностей и особенностей. Иными словами, каждый процесс ЖЦ подкрепляется выбранными для реализации задач ПС средствами, методами программирования и методикой их применения и выполнения.

Важную роль при формировании модели ЖЦ имеют организационные аспекты:

- планирование последовательности работ и сроков их исполнения;
- подбор и подготовка ресурсов (людских, программных и технических) для выполнения работ;
- оценка возможностей реализации проекта в заданные сроки, стоимость и ресурсы.

Внедрение модели ЖЦ в практическую *деятельность по* созданию программного продукта позволяет упорядочить взаимоотношения между субъектами процесса разработки ПС и учитывать динамику модификации требований к проекту и к системе.

Пример. ЖЦ разработки ПС с задачами и действиями для процесса тестирования. Основное назначение процесса тестирования ЖЦ – выполнение задач процесса на основе входов (*входные данные* для выполнения задач процесса) и выходов при завершении задач, а также ролей и действий исполнителей этих задач.

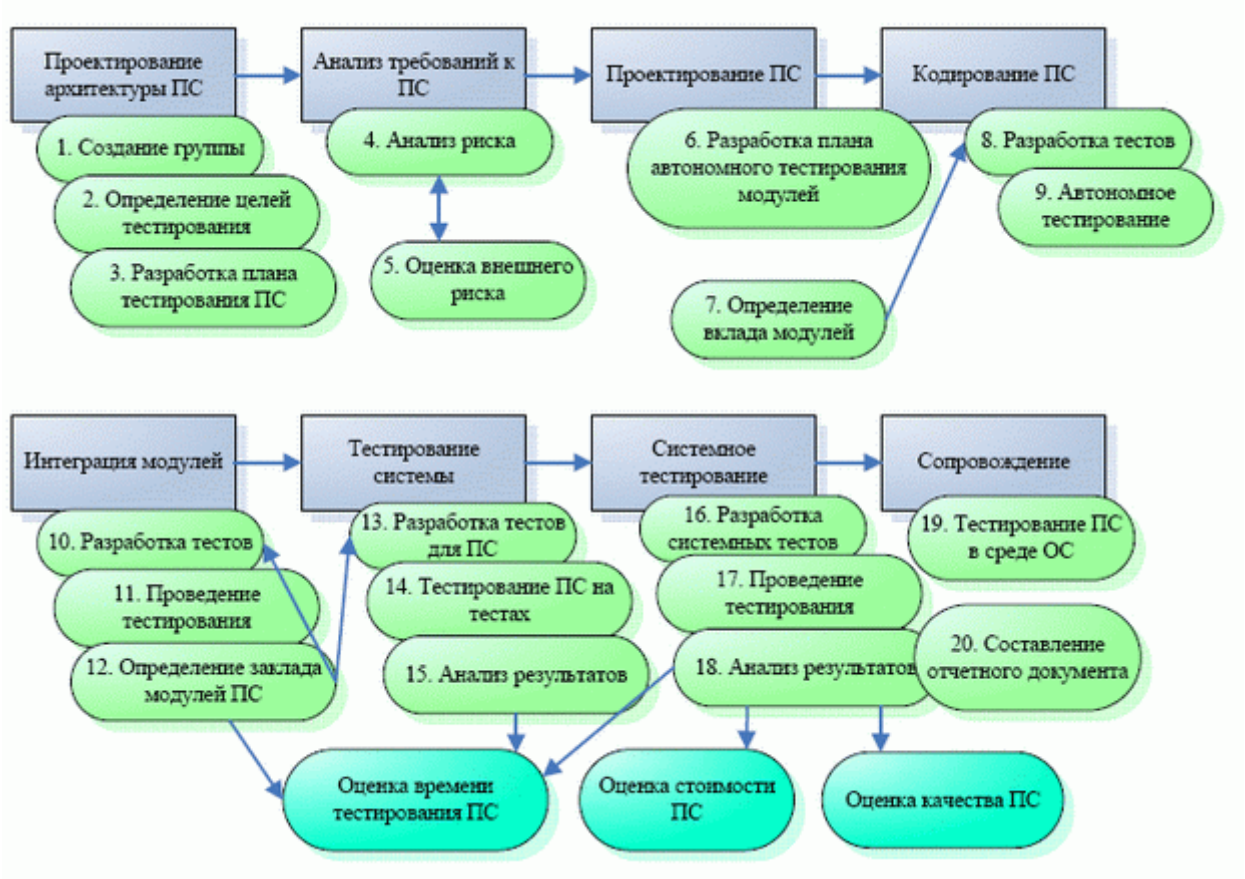
В соответствии со стандартом *ISO*/IEC 12207 были выявлены задачи тестирования и распределены *по* процессам ЖЦ ПС. В результате был получен единый непрерывный процесс тестирования разных ПС, задачами которого являются *подготовка*, *проведение* и *оценивание* результатов тестирования, которые распределились *по* 20 действиям (шагам) процесса разработки [2.7, 2.9]. Данный подход к тщательному тестированию ПС целесообразно применять, например, для систем реального времени.

На шаге *подготовки* осуществляется *анализ* рабочих продуктов процесса разработки ПС (входных для данного шагу процесса тестирования) для определения целей, объектов, сценариев и ресурсов тестирования, адекватных шагу тестирования. Результаты выполнения шагов подготовки тестирования должны фиксироваться в планах тестирования.

На шаге *выполнения* осуществляется фиксация результатов выполнения тестов, их сравнение с ожидаемыми результатами, *определение* текущего состояния *рабочего продукта* ПС и принятие решения о достаточности тестирования.

Каждый шаг процесса *разработки* состоит из набора решаемых задач, распределение *по* процессам и подпроцессам ЖЦ (рис. 2.3).

Шаги процесса и отдельные задачи могут выполняться циклически для разных объектов ПС при их тестировании.



увеличить изображение

Рис. 2.3. ЖЦ разработки ПС с конкретизированными задачами на подпроцессах тестирования

Описание семантики задач и шагов процесса тестирования представлено в табл. 2.1.

Для подключения задач тестирования ко всем процессам ЖЦ проводится: *распределение обязанностей* между участниками процесса с учетом требований относительно их профессиональной подготовки; *определение* стандартов на *представление* окончательных документов, метрик процесса, критериев начала и завершения задач и перехода к следующему шагу процесса; подбор методов тестирования для выбранного класса ПС для проверки правильности выполнения задач тестирования; разработка специальных шаблонов документов для документирования процесса тестирования относительно каждого шага процесса тестирования.

Таблица 2.1. Состав задач процесса тестирования

Шаг процесса	Задачи процесса тестирования
1. Создание группы тестирования	1.1. Определение участников процесса тестирования
	1.2. *Распределение обязанностей* в группе и формирование плана тестирования
2. Анализ риска	2.1. Идентификация рисков
	2.2. Упорядочение рисков
	2.3. Распределение ресурсов
3. Определения целей тестирования	3.1. Идентификация целей тестирования
	3.2. Определения критериев прохождения тестов
	3.3. Приведения в порядок целей тестирования по оценкам риска
4. Разработка планов тестирования	4.1. Разработка плана тестирования ПС
	4.2. Разработка плана интеграционного тестирования

	4.3. Разработка плана автономного тестирования
	4.4. Разработка плана комплексного тестирования
5. Разработка тестов	5.1. Проектирование и разработка тестов
	5.2. Подготовка тестовых данных
	5.3. Проверка тестовых документов
6. Автономное и интеграционное тестирование	6.1. Автономное тестирование модулей и анализ результатов
	6.2. Интеграционное тестирование
	6.3. Повторное тестирование после устранения дефектов
	6.4. Анализ результатов интеграционного тестирования
7. Тестирования ПС	7.1. Утверждение среды и ресурсов тестирования
	7.2. Тестирование ПС
	7.3. Повторное тестирование ПС после устранения дефектов
	7.4. Анализ результатов завершения тестирования ПС
	7.5. Тестирование инсталляции ПС
8. Составление документа по тестированию ПС и подготовка отчета	8.1. Сбор и анализ данных о результатах тестирования
	8.2. Подготовка решений и рекомендаций по использованию ПС
	8.3. Подготовка итогового документа о результатах тестирования
	8.4. Проверка решений и подготовка документа отчета

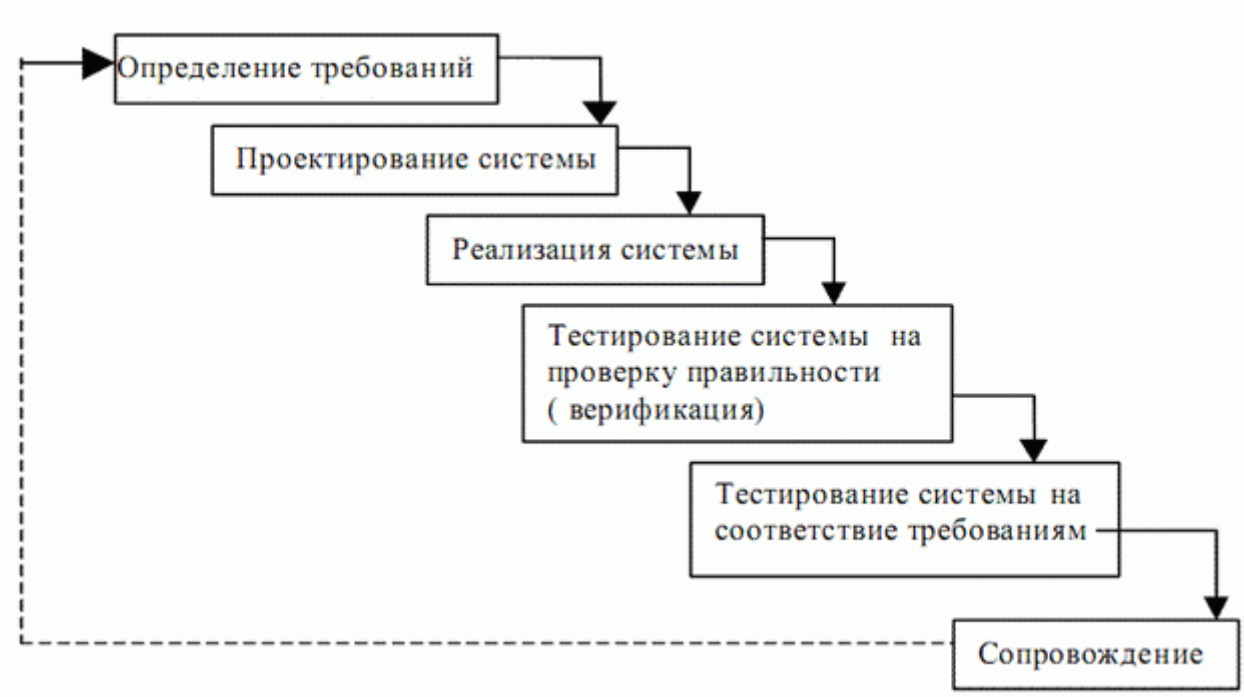
При завершении тестирования ПС для определения времени тестирования, стоимости *работ* учитываются результаты тестирования процесса разработки ПС и оформляется отчетный документ *по* изготовлению ПС. *Оценивание* риска отказов проводится на этапе подготовки тестирования и на шагах анализа.

2.2. Типы моделей ЖЦ

Рассмотренные вопросы послужили источником формирования различных видов моделей ЖЦ, основанных на процессном подходе к разработке программных проектов. К широко используемым типам моделей ЖЦ относятся следующие: каскадная, спиральная, инкрементная, эволюционная, стандартизованная и др.

2.2.1. Каскадная модель ЖЦ

Одной из первых стала применяться *каскадная модель*, в которой каждая работа выполняется один раз и в том порядке, как это представлено в модели (рис. 2.4).



увеличить изображение

Рис. 2.4. Каскадная модель ЖЦ программных систем

Т.е. делается предположение, что каждая работа будет выполнена настолько тщательно, что после ее завершения и перехода к следующему этапу возвращения к предыдущему не потребуется.

Разработчик проверяет промежуточный результат разными известными методами верификации и фиксирует его в качестве готового эталона для следующего процесса.

Согласно данной модели ЖЦ работы и задачи процесса разработки обычно выполняются последовательно, как это представлено в схеме. Однако вспомогательные и организационные процессы (контроль требований, управление качеством и др.) обычно выполняются параллельно с процессом разработки. В данной модели возвращение к начальному процессу предусматривается после сопровождения и исправления ошибок.

Особенность такой модели состоит в фиксации последовательных процессов разработки программного продукта. В ее основу положена модель фабрики, где продукт проходит стадии от замысла до производства, затем передается заказчику как готовое изделие, изменение которого не предусмотрено, хотя возможна замена на другое подобное изделие в случае рекламации или некоторых ее деталей, вышедших из строя.

Недостатки этой модели:

- процесс создания ПС не всегда укладывается в такую жесткую форму и последовательность действий;
- не учитываются изменившиеся потребности пользователей, изменения во внешней среде, которые вызовут изменения требований к системе в ходе ее разработки;
- большой разрыв между временем внесения ошибки (например, на этапе проектирования) и временем ее обнаружения (при сопровождении), что приводит к большой переделке ПС.

При применении каскадной модели могут иметь место следующие факторы риска:

- требования к ПС недостаточно четко сформулированы, либо не учитывают перспективы развития ОС, сред и т.п.;
- большая система, не допускающая компонентной декомпозиции, может вызвать проблемы с размещением ее в памяти или на платформах, не предусмотренных в требованиях;
- внесение быстрых изменений в технологию и в требования может ухудшить процесс разработки отдельных частей системы или системы в целом;
- ограничения на ресурсы (человеческие, программные, технические и др.) в ходе разработки могут сузить отдельные возможности реализации системы;

полученный продукт может оказаться плохим для применения по причине недопонимания разработчиками требований или функций системы или недостаточно проведенного тестирования. Преимущества реализации системы с помощью каскадной модели следующие:

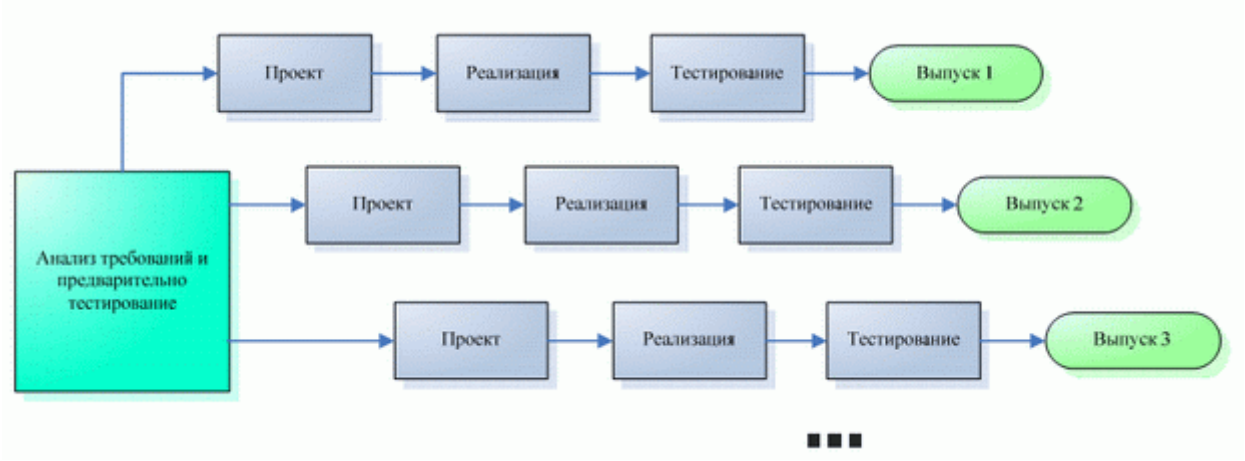
- все задачи подсистем и системы реализуются одновременно (т.е. ни одна задача не забыта), а это способствует установлению стабильных связей и отношений между ними;

полностью разработанную систему с документацией на нее легче сопровождать, тестировать, фиксировать ошибки и вносить изменения не беспорядочно, а целенаправленно, начиная с требований (например, добавить или заменять некоторые функции) и повторить процесс.

Каскадную модель можно рассматривать как модель ЖЦ, пригодную для создания первой версии ПО с целью проверки реализованных в ней функций. При сопровождении и эксплуатации могут быть обнаружены разного рода ошибки, исправление которых потребует повторного выполнения всех процессов, начиная с уточнения требований.

2.2.2. Инкрементная модель ЖЦ

Первая создаваемая промежуточная версия системы (выпуск 1) реализует часть требований, в последующую версию (выпуск 2) добавляют дополнительные требования и так до тех пор, пока не будут окончательно выполнены все требования и решены задачи разработки системы. Для каждой промежуточной версии на этапах ЖЦ выполняются необходимые процессы, работы и задачи, в том числе, анализ требований и создание новой архитектуры, которые могут быть выполнены одновременно.



увеличить изображение

Рис. 2.5. Инкрементная модель ЖЦ

Процессы разработки технического проекта ПС, его программирование и тестирование, сборка и квалификационные испытания ПС выполняются при создании каждой последующей версии.

В соответствии с данной моделью ЖЦ, процессы которой практически такие же, что и в каскадной модели, ориентир делается на разработку некоторой законченной промежуточной версии, а задачи процесса разработки выполняются последовательно или частично параллельно для ряда отдельных промежуточных структур версии.

Работы и задачи процесса разработки следующей версии системы с дополнительными требованиями или функциями могут выполняться неоднократно в той же последовательности для всех промежуточных версий системы. Процессы сопровождения и эксплуатации могут быть реализованы параллельно с процессом разработки версии путем проверки частично реализованных требований в каждой промежуточной версии и так до получения законченного варианта системы. Вспомогательные и организационные процессы ЖЦ обычно выполняются параллельно с процессом разработки версии системы и к концу разработки будут собраны данные, на основании которых может быть установлен уровень завершенности и качества изготовленной системы.

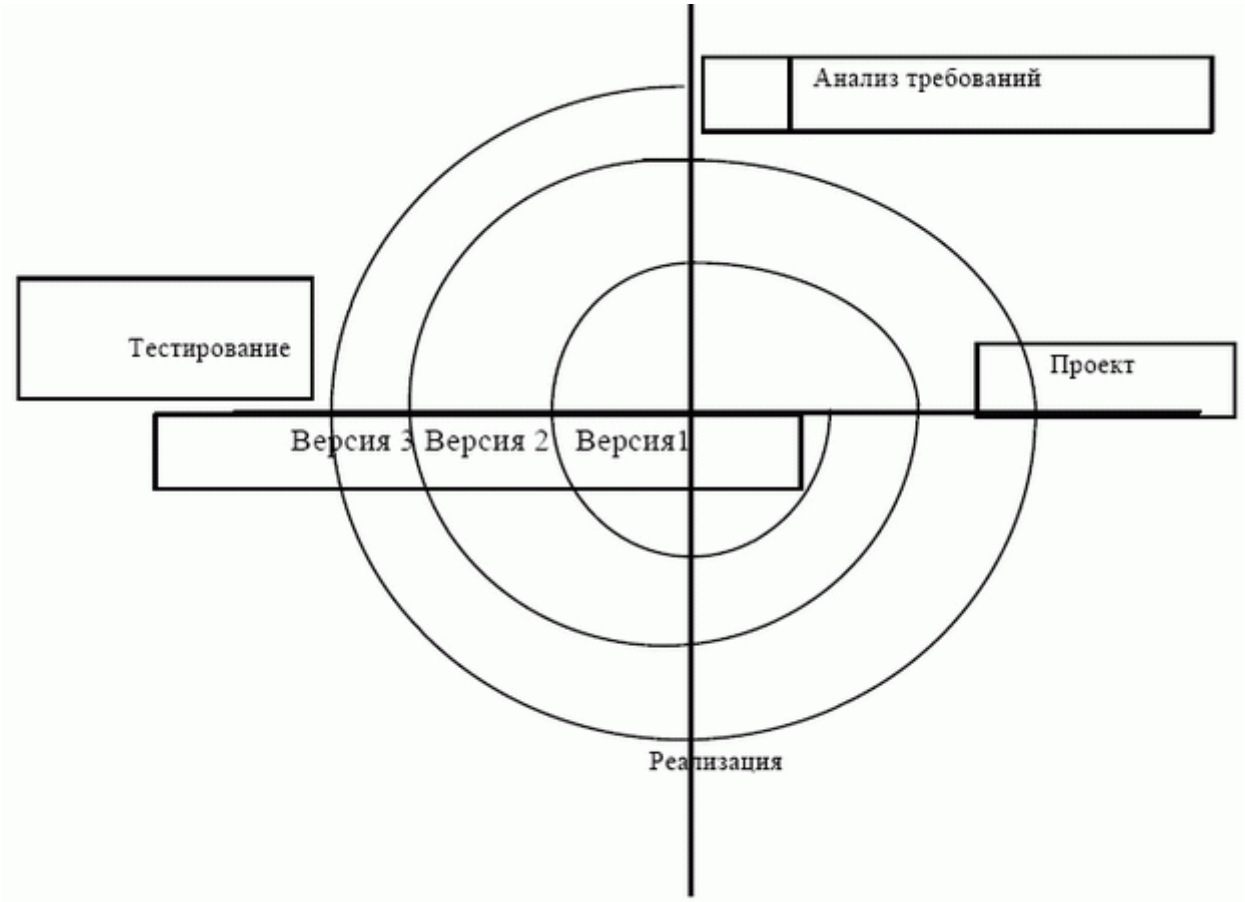
При применении данной модели необходимо учитывать следующие факторы риска:

- требования составлены с учетом возможности их изменения при реализации продукта;
- все возможности системы требуется реализовать с начала;
- быстрое изменение технологии и требований к системе может привести к нарушению полученной структуры системы;
- ограничения в ресурсном обеспечении (исполнители, финансы) могут привести к затягиванию сроков сдачи системы в эксплуатацию.

Данную модель ЖЦ целесообразно использовать, в случаях когда:

- желательно реализовать некоторые возможности системы быстро за счет создания промежуточной версии продукта;
- система декомпозируется на отдельные составные части, которые можно реализовывать как некоторые самостоятельные промежуточные или готовые продукты;
- возможно увеличение финансирования на разработку отдельных частей системы.

2.2.3. Спиральная модель



увеличить изображение

Рис. 2.6. Спиральная модель ЖЦ разработки программных систем

Исходя из возможности внесения изменений, как в процесс, так и в создаваемый промежуточный продукт была создана *спиральная модель* (рис. 2.6). Внесение изменений ориентировано на удовлетворение потребности пользователей сразу, как только будет установлено, что созданные артефакты или элементы документации (описание требований проекта, комментарии различного вида и т.п.), не соответствуют действительному состоянию разработки.

Данная модель ЖЦ допускает анализ продукта на витке разработки, его проверку, оценку правильности и принятия решения двигаться на следующий виток или опуститься на предыдущий виток для доработки на нем промежуточного продукта.

Отличие этой модели от каскадной модели состоит в возможности обеспечивать многоразовое возвращение к процессу формулирования требований и к повторной разработке с любого процесса выполнения работ. На изображенной модели (рис. 2.6), каждый виток спирали соответствует одной из версий разработки системы.

При необходимости внесения изменений в систему на каждом витке с целью получения новой версии системы обязательно вносятся изменения в предварительно зафиксированные требования, после чего происходит возврат на предыдущий виток спирали для продолжения реализации новой версии системы с учетом изменений.

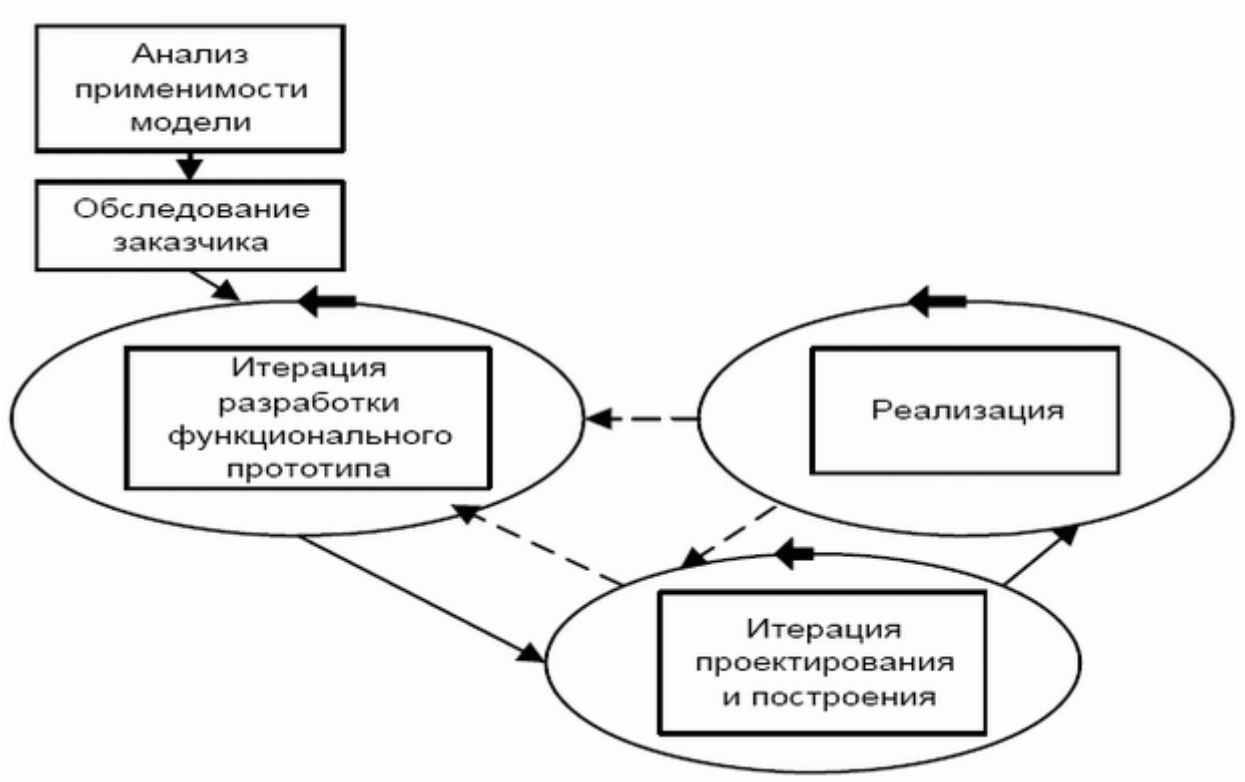
2.2.4. Эволюционная модель ЖЦ

В случае эволюционной модели система разрабатывается в виде последовательности блоков структур (конструкций). В отличие от инкрементной модели ЖЦ подразумевается, что требования устанавливаются частично и уточняются в каждом последующем промежуточном блоке структуры системы.

Использование эволюционной модели предполагает проведение исследования предметной области для изучения потребностей заказчика проекта и анализа возможности применения этой модели для реализации. Модель применяется для разработки несложных и не критических систем, для которых главным требованием является реализация функций системы. При этом требования не могут быть определены сразу и полностью. Тогда разработка системы проводится итерационно путем ее эволюционного развития с получением некоторого варианта системы – прототипа, на котором проверяется реализация требований. Иными словами, такой процесс по своей сути является итерационным, с повторяющимися этапами разработки, начиная от

измененных требований и до получения готового продукта. В некотором смысле к этому типу модели можно отнести спиральную модель.

Развитием этой модели является модель эволюционного прототипирования в рамках всего ЖЦ разработки (рис. 2.7). В литературе она часто называется моделью быстрой разработки приложений RAD (*Rapid Application Development*). В данной модели приведены действия, которые связаны с анализом ее применимости для конкретного вида системы, а также обследование заказчика для определения потребностей пользователя для разработки плана создания прототипа.



увеличить изображение

Рис. 2.7. Модель эволюционного прототипирования

В модели есть две главные итерации разработки функционального прототипа, проектирования и реализации системы. Проверяется, удовлетворяет ли она всем функциональным и нефункциональным требованиям. Основной идеей этой модели является моделирование отдельных функций системы в прототипе и постепенное эволюционная его доработка до выполнения всех заданных функциональных требований.

Итераций по получению промежуточных вариантов прототипа может быть несколько, в каждой из которых добавляется функция и повторно моделируется работа прототипа. И так до тех пор, пока не будут промоделированы все функции, заданные в требованиях к системе. Потом выполняется еще итерация – окончательное программирование для получения готовой системы.

Эта модель применяется для систем, в которых наиболее важными являются функциональные возможности, и которые необходимо быстро продемонстрировать на CASE-средствах.

Так как промежуточные прототипы системы *соответствуют реализации* некоторых функциональных требований, то их можно проверять и при сопровождении и эксплуатации, т.е. параллельно с процессом разработки очередных прототипов системы. При этом вспомогательные и организационные процессы могут выполняться параллельно с процессом разработки и накапливать сведения по данным количественных и качественных оценок на процессах разработки.

При этом учитываются такие факторы риска:

- реализация всех функций системы одновременно может привести к громоздкости;
- ограниченные человеческие ресурсы заняты разработкой в течение длительного времени.

Преимущества применения данной модели ЖЦ следующие:

- быстрая реализация некоторых функциональных возможностей системы и их апробирование;
- использование промежуточного продукта в следующем прототипе;
- выделение отдельных функциональных частей для реализации их в виде прототипа;
- возможность увеличения финансирования системы;
- обратная связь устанавливается с заказчиком для уточнения функциональных требований;

упрощение внесения изменений в связи с заменой отдельной функции.

Модель развивается в направлении добавления нефункциональных требований к системе, связанных с защитой и безопасностью данных, несанкционированным доступом к ним и др.

2.2.5. Стандартизация модели ЖЦ

Типичный ЖЦ системы начинается с формулировки идеи или потребности, проходит все процессы разработки, производства, эксплуатации и сопровождения системы. Стандартный ЖЦ состоит из процессов, каждый процесс характеризуется видами деятельности и задачами, которые выполняются на нем. Переход от одного процесса

к другому должен быть санкционирован и определены входные и выходные данные.

Модель данного ЖЦ включает в себя процессы:

определение требований;

разработка (проектирование, конструирование);

верификация, валидация, тестирование;

изготовление;

эксплуатация;

сопровождение.

Данной модели соответствуют все виды деятельности, начиная с разработки проекта или концепции программного продукта и заканчивая его изготовлением. Как было сказано выше, стандарт ISO/IEC 12207 объединяет эти виды деятельности в следующие три категории: основные, организационные и вспомогательные процессы, которые и составляют стандартный ЖЦ.

Процессы приобретения, поставки и разработки используются для анализа и определения системных требований и решений верхнего уровня проектирования системы и предварительного определения требований к компонентам системы, включая ПО. Процесс разработки может быть использован для анализа, демонстрации, прототипирования требований и проектных решений.

На этапе проектирования разрабатывается техническое, программное, организационное обеспечение системы, а также проектируются, разрабатываются, интегрируются, тестируются и оцениваются ее компоненты. Результатом этого процесса является система, которая разрабатывалась согласно контракту или договора.

Стандарт разработан так, чтобы его можно было применить полностью или частично. Действия и задачи основных процессов отбираются, адаптируются и применяются при разработке или модификации системы. Процесс разработки может включать одну или более итераций. Результатом являются требования к ПО, проект и реализованный продукт.

Если разрабатываемое ПО – часть системы, то к ней могут применяться все действия процессов разработки, и если эта часть – автономное ПО, то некоторые общие действия на уровне системы могут не использоваться при его разработке.

Во время процесса изготовления система готовится для поставки заказчику и покупателям. Цель процесса – тиражирование (производство) и установка работающей системы у заказчика для

сопровождения. Данный процесс заключается в копировании изготовленного продукта и документации на соответствующие носители пользователей. К видам деятельности на процессе относится достижение качества реализации и создания конфигурации (версии) системы. Другие вспомогательные процессы и действия (например, сбор данных о результатах контроля) могут применяться по мере необходимости.

Изготовленная система, начиная с первой ее версии, передается заказчику или продается желающим покупателям. Другие процессы (приобретения, поставки и разработки) могут использоваться при инсталляции и проверке разработанной или модифицированной системы.

Процесс *эксплуатации* включает использование системы ее покупателями. Когда система больше не удовлетворят пользователей, она утилизируется, т.е. удаляются из употребления путем уничтожения кодов, архивов, процедур и т.п.

Во время *сопровождения* система модифицируется вследствие обнаруженных ошибок и недостатков в ее разработке либо по требованиям пользователя, который желает ее адаптировать к новой среде или усовершенствовать отдельные ее функции.

2.3. Сопоставление ЖЦ стандарта ISO/IEC 12207 и областей SWEBOK

Каждая область ядра знаний SWEBOK *по* существу соответствует одному или нескольким процессам, которые определены в стандарте *ISO*/IEC 12207. В связи с этим проведен сравнительный *анализ* областей SWEBOK и процессов модели ЖЦ упомянутого стандарта. Для этого вначале рассмотрим процессы ЖЦ, а потом области SWEBOK.

2.3.1. Характеристика процессов стандарта ISO/IEC 12207

Процессы данного стандарта разбиты по группам: основные, вспомогательные и организационные.

К основным процессам стандарта относятся:

- приобретение (*acquisition*);
- поставка (supply);
- разработка (development);
- эксплуатация (operation);
- сопровождение (maintenance).

Процесс приобретения инициирует ЖЦ ПО и определяет действия организации-покупателя (или заказчика), которая приобретает автоматизированную систему, программный продукт или сервис.

Процесс поставки определяет действия предприятия-поставщика, которое снабжает покупателя системой, программным продуктом или сервисом.

Процесс разработки состоит в изготовлении исполнителем проекта программного продукта на процессах ЖЦ: разработка требований, проектирование, кодирование, тестирование и интеграция.

Процесс эксплуатации определяет действия оператора по обслуживанию системы, использованию ее пользователями, изучившими ее возможности для удовлетворения своих потребностей в плане обработки данных или вычислений.

Процесс сопровождения состоит в выполнении предписанных действий по инсталляции системы, запуску функций, а также по управлению модификациями и поддержанием системы в рабочем состоянии.

К вспомогательным процессам стандарта относятся процессы:

- документирования (documentation);
- управления конфигурацией (configuration management);
- обеспечения качества (*quality assurance*);
- верификации (verification);
- валидации (validation);
- совместного анализа (оценки) (joint review);
- аудита (audit).

Вспомогательные процессы поддерживают реализацию основных процессов и способствуют получению требуемого качества ПО. Они инициируются другими процессами.

К организационным процессам стандарта относятся пр оц **ессы :**

- управления (management);
- создания инфраструктуры (infrastructure);
- усовершенствования (improvement);
- обучения (training).

За каждым процессом стандарта наблюдает определенный участник разработки или руководитель в части выполнения предусмотренных

видов деятельности и задачи, которые в него входят, и проверки результатов. В табл. 2.2. приведено общее количество определенных в стандарте процессов, действий и задач.

Таблица 2.2. **Общий перечень процессов ЖЦ стандарта 12207**

Класс	Процесс	Действие	Задача
Основные процессы	5	35	135
Вспомогательные процессы	8	25	70
Организационные процессы	4	14	27
Итого	**17**	**74**	**232**

Из этого множества процессов стандарта далее будут сравниваться только те процессы, которые имеют аналоги областям знаний в ядре знаний SWEBOK.

2.3.2. Характеристика областей знаний SWEBOK

В ядре знаний SWEBOK определено 10 областей знаний. Среди них выделим базовые области, методы и средства которых соответствуют *процессам разработки* ПС:

1. Разработка требований;
2. Проектирование;
3. Конструирование;
4. Тестирование;
5. Сопровождение.

Эти области знаний по своим базовым концепциям и методам, определенным в SWEBOK, соответствуют задачам и выполняемым действиям следующих *процессов разработки* ЖЦ стандарта ISO/IEC – 12207:

1. Разработка требований;
2. Проектирование;
3. Кодирование;
4. Тестирование;
5. Интеграция;
6. Интеграционное тестирование;
7. Эксплуатация;
8. Сопровождение.

Эти процессы задают последовательность задач и действий при разработке разных типов ПС с применением методов и средств, которые представлены в ядре знаний для перечисленных пяти областей SWEBOK. Фактически процессы и области совпадают по смыслу и названию, но содержание действий на процессах определяются методами и средствами пяти областей, которые приведены выше.

В табл. 2.3. приведен сопоставительный перечень основных областей SWEBOK, их задач и соответственно задач ЖЦ стандарта. При этом процессы приобретения и поставки из состава основных процессов исключены, поскольку они не относятся к процессам разработки ПО.

Остальные пять областей ядра SWEBOK относятся к числу процессов обеспечения и управления разработкой проекта, при которых проводится верификация, сбор данных для проведения оценки качества и др. И хотя области ядра знаний явно не содержат названий процессов ЖЦ, функционально и содержательно они соответствуют процессам, относящимся к категории основных, вспомогательных и организационных.

Перечень процессов ЖЦ категории вспомогательных и организационных приведены на рис. 2.2, а соответствующие им области знаний SWEBOK таковы:

управление конфигурацией,

управление инженерией ПО (или управление проектом),

процесс инженерии ПО (инфраструктура процесса разработки),

методы и средства инженерии;

инженерия качества (управление качеством).

Данные области знаний включают методы и средства разработки ПС, а также управление проектом, рисками, конфигурацией, качеством создаваемого продукта. Они соответствуют отдельным задачам вспомогательных и организационных процессов ЖЦ стандарта и предназначены для управления проектом, конфигурацией и качеством.

Таблица 2.3. **Задачи основных областей SWEBOK и процессов ЖЦ**

Область SWEBOK	Задачи области SWEBOK	Задачи процессов ЖЦ стандарта ISO/IEC 12207
Разработка требований	Инженерия требований	Подготовка заказа
	Выявление требований	Выявление требований
	Анализ требований	Анализ требований к системе
	Спецификация требований	Анализ требований к ПО
	Проверка требований	Описание документа
	Управление требованиями	
Проектирование	Разработка архитектуры ПО	Проектирование:

ПО	Нотация	архитектуры системы
	Анализ качества проектирования	архитектуры ПО
	Использование стратегии и	ПО
	методов проектирования	Кодирование ПО
		Тестирование ПО
Конструирование ПО	Снижение сложности	Конструирование
	Предупреждение отклонений от стиля	структуры системы
	Структуризация системы для проверок	Кодирование элементов структуры и ПО
	Использование внешних стандартов	Интеграция элементов
		Применение стандартов программной инженерии
Тестирование ПО	Тестирование элементов и системы	Тестирование ПО
	Тестирование спецификаций, структуры и системы на наборах данных	Интеграционное тестирование
		Квалификационное тестирование
	Метрическое измерение тестирования	Интеграция системы
	Планирование и оценка качества	*Системное тестирование*
		Установка и приемка ПО
Сопровождение ПО	Запуск ПО	Инсталляция ПО
	Нахождение ошибок, планирование исправлений	Анализ проблем и модификация
	Внесение изменений	Реализация модификаций
		Анализ сопровождения
		Миграция, удаление ПО
Эксплуатация системы	Методы обеспечения эксплуатации системы	Внедрение процесса
		Функциональное тестирование
		Эксплуатация системы
		Поддержка пользователя

В табл. 2.4 приведен перечень областей ядра SWEBOK и соответствующие задачи вспомогательных (организационных и дополнительных) процессов ЖЦ стандарта ISO/IEC 12207.

Таблица 2.4. **Задачи областей SWEBOK и вспомогательных процессов ЖЦ**

Области SWEBOK	Задачи областей SWEBOK	Задачи процессов стандарта 12207
Управление конфигурацией	Процесс управления конфигурацией.	Определение и контроль конфигурации.
	Идентификация элементов.	Учет состояния и оценка конфигурации.
	Учет статуса, аудит.	Управление реализацией и поставкой версии.
	Контроль конфигурации.	
	Управление версиями.	
Управление проектом	Организационное управление.	Инициация и определение области применения.

	Планирование проектом. Управления процессами и проектом. Инженерия измерения ПО. Управление риском.	Планирование. Выполнение и контроль. Анализ управления проектом: технический анализ; аудит (ревизия).
Управление качеством	Концепция качества ПО. Определение и планирование качеством. Верификация и валидация. Измерение в анализе качества ПО.	Внедрение процесса. Обеспечение производства и качества. Процесс верификации и валидации. Анализ и оценивание качества.
Методы и средства инженерии	Методы инженерии. Инструменты инженерии.	Процесс усовершенствования: определение процесса; *оценка процесса*; – улучшение процесса.
Процесс инженерии ПО	Инфраструктура процесса. Определение процесса. Измерение процесса. Анализ проекта. Выполнение изменений. Оценки стоимости и затрат.	Создание инфраструктуры. Сопровождение инфраструктуры. Внедрение процесса. Завершение.

Сопоставление концепций, методов и средств областей SWEBOK с задачами процессов ЖЦ позволяет регламентировать поиск, обнаружение ошибок и внесение изменений в требования к системе.

Контрольные вопросы и задания

1. Охарактеризуйте понятие модели ЖЦ и назовите основные виды моделей ЖЦ.
2. Опишите каскадную и *спиральную модели* ЖЦ?
3. Дайте характеристику эволюционной модели ЖЦ.
4. Назовите другие виды моделей ЖЦ.
5. Какие общие черты имеют инкрементная и эволюционная модели?
6. Перечислите основные процессы ЖЦ стандарта.
7. Как построить новую модель ЖЦ на основе стандарта?
8. Перечислите процессы категории организационных процессов ЖЦ стандарта.
9. Назовите задачи и методы тестирования ПС.
10. Назовите основные задачи управления качеством и проектом.
11. Проведите сравнительный анализ модели процессов ЖЦ стандарта 12207 и областей ядра знаний SWEBOK.
12. Определите основные цели областей SWEBOK и процессов ЖЦ.

Внимание! Если Вы увидите ошибку на нашем сайте, выделите её и нажмите Ctrl+Enter.