МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский Нижегородский
государственный университет им. Н.И. Лобачевского»
Институт информационных технологий, математики и механики

Е.А. Кумагина
Е.А. Неймарк

МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА И ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Учебно-методическое пособие

Рекомендовано методической комиссией ИИТММ для студентов ННГУ,
обучающихся по направлению подготовки 09.04.03 «Прикладная информатика»

Нижний Новгород
2016

УДК 004.415.2(076)
ББК 32.81я73
К 88

К 88 Кумагина, Е.А., Неймарк, Е.А. Модели жизненного цикла и технологии проектирования программного обеспечения: учебно-методическое пособие / Е.А. Кумагина, Е.А. Неймарк. – Нижний Новгород: Изд-во ННГУ, 2016. – 41 с.

Рецензент: к.ф.-м.н., доцент **К.А. Баркалов**

Учебно-методическое пособие предназначено для студентов ИИТММ направления подготовки «Прикладная информатика», изучающих курсы «Проектирование информационных систем», «Информационные ресурсы общества», «Менеджмент информационноемких технологий».

В пособии расссмотрены как теоретические вопросы проектирования информационных систем: процессы жизненного цикла информационной системы, типы моделей жизненного цикла. Так и промышленные технологии создания программных продуктов.

УДК 004.415.2(076)
ББК 32.81я73
К 88

Оглавление

1. Структурный анализ

Структурный анализ – метод исследования систем, включающий их общий обзор и дальнейшую детализацию, порождающий иерархическую структуру модели исследуемого объекта. Все методологии структурного анализа базируются на ряде общих принципов. В качестве основных выделяются два принципа:

- принцип "разделяй и властвуй",
- принцип иерархического упорядочивания.

Первый принцип показывает, что решение трудных задач достигается путем разбиения их на множество меньших независимых задач, более легких для понимания и решения.

Второй принцип декларирует, что иерархическое упорядочивание резко повышает понимаемость проблемы при организации ее частей в виде древовидной иерархической структуры. Каждый новый уровень добавляет новые детали. Тем самым повышается понимаемость системы.

Таким образом, структурный анализ, исходя из функционального описания системы в целом, позволяет разделить её на функциональные части, выделить функциональные описания отдельных частей, исследовать в них информационные потоки и формализовать структуры данных. Функции должны детализироваться сверху вниз в виде иерархи ческой структуры таким образом, чтобы процедуры сбора, хранения и переработки информации, рассматриваемые сначала как нечто единое целое, расчленялись на отдельные элементы данных, компонентов и действия, совершаемые над этими данными

Проектирование и производство заказных технических систем и программных продуктов регламентируют четыре крупных комплексов международных стандартов[8]:

1. Стандарт ГОСТ Р ИСО/МЭК 12207-2010 – Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств;
2. CMMI – Система и модель оценки зрелости, управление проектами программных средств;
3. ГОСТ Р ИСО/МЭК 9000:2000 – Стандарты менеджмента (административного управления) качеством систем;
4. Стандарт ISO 19759:2005 – SWEBOK, Совокупность знаний о разработке программных средств. Руководство.

Стандарт ГОСТ Р ИСО/МЭК 12207-2010 устанавливает общую структуру процессов жизненного цикла программных средств, на которую можно ориентироваться в программной индустрии [3]. Настоящий стандарт **определяет процессы, виды деятельности и задачи**, которые используются при приобретении программного продукта или услуги, а также при поставке, разработке, применении по назначению, сопровождении и прекращении

применения программных продуктов. Понятие программного средства включает в себя встроенный фирменный программный компонент.

В стандарте 12207 **не детализируются процессы жизненного цикла** в терминах методов или процедур, необходимых для удовлетворения требований и достижения результатов процесса.

Стандарт 12207 устанавливает также процесс, который может использоваться при определении, управлении и совершенствовании процессов жизненного цикла программных средств. Настоящий стандарт **не устанавливает конкретной модели** жизненного цикла системы или программных средств, разработки методологии, методов, моделей или технических приемов.

Стороны, применяющие настоящий стандарт, отвечают за выбор модели жизненного цикла для программных проектов и отображение процессов, действий и задач, представленных в настоящем стандарте, на эту модель. Стороны также ответственны за выбор и применение методов разработки программных средств и за выполнение действий и задач, подходящих для программного проекта.

1.1. Декомпозиция процессов

Каждый процесс, представленный в стандарте, удовлетворяет общему описанию. С целью более подробного описания процессы иногда подвергают декомпозиции на более мелкие части. Некоторые процессы декомпозируют в совокупность действий и (или) в процессы более низкого уровня. Процесс более низкого уровня описывается тогда, когда декомпозированная часть процесса сама удовлетворяет критериям, характеризующим процесс. Деятельность используется тогда, когда определенная часть процесса, полученная в результате декомпозиции, не является процессом. Деятельность может рассматриваться просто как набор задач.

Иногда полезно выполнить декомпозицию процессов на процессы более низкого уровня с более четким уровнем детализации. Некоторые процессы более низкого уровня описываются исключительно для целей оценки. Такие процессы не представлены в основной части настоящего стандарта.

Задача выражается в форме требования, рекомендации или допустимого действия, предназначенных для поддержки достижения выходов процесса. Для этой цели в настоящем стандарте используются вспомогательные глаголы "должен", "следует" и "может", чтобы подчеркнуть различие между разными формами задач. Глагол "должен" используется для выражения условия, требуемого для соответствия, "следует" – для выражения рекомендации среди других возможностей и "может" – для того, чтобы отразить направление допустимых действий в пределах ограничений настоящего стандарта.

Дополнительный справочный материал представлен в виде необязательных примечаний или необязательных приложений.[3]

1.2. Модели и стадии жизненного цикла

Процесс жизни любой системы или программного продукта может быть описан посредством **модели жизненного цикла**, состоящей из **стадий**. Модели могут использоваться для представления всего жизненного цикла от замысла до прекращения применения или для представления части жизненного цикла, соответствующей текущему проекту.

Модель жизненного цикла представляется в виде последовательности стадий, которые могут перекрываться и (или) повторяться циклически в соответствии с областью применения, размером, сложностью, потребностью в изменениях и возможностях.

Каждая стадия описывается формулировкой цели и выходов. Процессы и действия жизненного цикла отбираются и исполняются на этих стадиях для полного удовлетворения цели и результатам каждой стадии. Различные организации могут использовать различные стадии в пределах жизненного цикла. Однако каждая стадия реализуется организацией, ответственной за эту стадию, с надлежащим рассмотрением информации, имеющейся в планах жизненного цикла и решениях, принятых на предшествующих стадиях. Аналогичным образом организация, ответственная за текущую стадию, ведет записи принятых решений и записи допущений, относящихся к последующим стадиям данного жизненного цикла.

Стандарт не требует использования какой-либо конкретной модели жизненного цикла. Однако он требует, чтобы в каждом проекте определялась подходящая модель жизненного цикла, предпочтительно та, которая уже выбиралась организацией для применения в различных проектах. Применение модели жизненного цикла обеспечивает средства для установления зависимой от времени последовательности, необходимой для менеджмента проекта.

Кроме того, стандарт не содержит требований использования какой-либо заданной совокупности стадий. Пример совокупности стадий жизненного цикла системы включает в себя стадии концепции, разработки, производства, применения по назначению, поддержки и прекращения применения. Примером совокупности стадий жизненного цикла программного продукта является разработка, применение по назначению и сопровождение.

Далее будут описаны различные типы или классы моделей жизненного цикла. Примеры этих типов моделей известны под такими наименованиями, как каскадная и спиральная. Следует отметить, что простой выбор наименования типа модели не удовлетворяет требованию определить модель, состоящую из стадий с определенными целями и результатами, достигнутыми посредством процессов стандарта.

2. Процессы жизненного цикла

2.1. Основные определения

Жизненный цикл (life cycle) – развитие системы, продукта, услуги, проекта или других изготовленных человеком объектов, начиная со стадии разработки концепции и заканчивая прекращением применения.

Модель жизненного цикла (life cycle model) – структура процессов и действий, связанных с жизненным циклом, организуемых в стадии, которые также служат в качестве общей ссылки для установления связей и взаимопонимания сторон.

Процесс (process) – совокупность взаимосвязанных или взаимодействующих видов деятельности, преобразующих входы в выходы.

Цель процесса (process purpose) – цель высокого уровня выполнения процесса и вероятные выходы эффективной реализации процесса.

Выход процесса (process outcome) – наблюдаемый результат успешного достижения цели процесса. Формулировка выхода процесса описывает один из следующих результатов:

- изготовление какого-либо артефакта;
- существенное изменение состояния;
- удовлетворение заданных ограничений, например требований, конечных целей и т.п.

Продукт (product) – результат процесса.

Проект (project) –попытка действий с определенными начальными и конечными сроками, предпринимаемая для создания продукта или услуги в соответствии с заданными ресурсами и требованиями.

Проект может рассматриваться как уникальный процесс, включающий в себя скоординированные и управляемые виды деятельности, а также может быть комбинацией видов деятельности из процессов проекта и технических процессов, определенных в настоящем стандарте.

2.2. Критерии для процессов

В стандарте 12207 процессы, входящие в них действия и задачи располагаются в виде упорядоченной последовательности, подходящей для пояснений. Эта последовательность не предполагает или не устанавливает какой-либо зависимости от времени. Из-за невозможности достичь единого мнения или применить универсальную, развернутую во времени последовательность пользователь настоящего стандарта может самостоятельно выбирать и назначать процессы, виды деятельности и задачи как наиболее подходящие и эффективные.

Стандарт способствует выполнению итераций между действиями и рекурсий в пределах отдельных действий для того, чтобы нейтрализовать нежелательное влияние любой подразумевающейся последовательности действий и задач. Стороны, применяющие настоящий стандарт, ответственны

за выбор модели жизненного цикла для проекта и отображение процессов, действий и задач в этой модели.

Стандарт 12207 устанавливает структуру работ в пределах жизненного цикла программных средств. Жизненный цикл начинается от замысла или потребности, которая может быть удовлетворена полностью или частично программным средством, и завершается прекращением применения этого программного средства. Такая архитектура создается совокупностью процессов и взаимосвязями между этими процессами. Определение процессов жизненного цикла основывается на двух базовых принципах: связности и ответственности.

Принцип связности: процессы жизненного цикла являются связными и соединяются оптимальным образом, считающимся практичным и выполнимым.

Принцип ответственности: процесс передается под ответственность какой-либо организации или стороне в пределах жизненного цикла программного средства.

2.3. Описание процессов

Процессы в настоящем стандарте описываются способом, подобным способу, представленному в ИСО/МЭК 15288, для того, чтобы обеспечить использование обоих стандартов в одной организации или проекте.

Каждый процесс настоящего стандарта описывается в терминах следующих атрибутов:

- наименование – передает область применения процесса как целого;
- цель – описывает конечные цели выполнения процесса;
- выходы – представляют собой наблюдаемые результаты, ожидаемые при успешном выполнении процесса;
- деятельность – является перечнем действий, используемых для достижения выходов;
- задачи – представляют собой требования, рекомендации или допустимые действия, предназначенные для поддержки достижения выходов процесса.

2.4. Категории процессов жизненного цикла

Настоящий стандарт группирует различные виды деятельности, которые могут выполняться в течение жизненного цикла программных систем, в семь групп процессов. Каждый из процессов жизненного цикла в пределах этих групп описывается в терминах цели и желаемых выходов, списков действий и задач, которые необходимо выполнять для достижения этих результатов.

Цели и результаты процессов жизненного цикла образуют эталонную модель процессов.

Таблица 1 Виды деятельности процессов ЖЦ

процессы	виды деятельности
процессы соглашения	1. приобретение , 2. поставка.

процессы организационного обеспечения проекта	1. менеджмент модели ЖЦ, 2. менеджмент инфраструктуры, 3. менеджмент портфеля проектов, 4. менеджмент людских ресурсов, 5. менеджмент качества.
процессы проекта	1. планирование проекта, 2. оценка проекта и процес управления, 3. менеджмент решений, 4. менеджмент рисков, 5. менеджмент конфигурации, 6. менеджмент информации, 7. процесс измерений
технические процессы	1. определение требований правообладателей, 2. анализ системных требований, 3. проектирование архитектуры системы, 4. реализация, 5. комплексирование системы, 6. квалификационное тестирование, 7. инсталляция программных средств 8. поддержка приемки программных средств, 9. функционирования программных средств, 10. сопровождение программных средств, 11. прекращения применения программных средств.
процессы реализации программных средств	1. реализация программных средств, 2. анализ требований программных средств, 3. проектирование архитектуры программных средств, 4. детальное проектирование программных средств, 5. конструирование программных средств, 6. комплексирование программных средств, 7. квалификационное тестирование.
процессы поддержки программных средств	1. менеджмент программной документации, 2. менеджмент конфигурации, 3. обеспечение гарантий качества программных средств,

	4. верификация программных средств, 5. валидация программных средств, 6. ревизия программных средств, 7. аудит программных средств, 8. решение проблем в программных средствах.
процессы повторного применения программных средств	1. проектирование доменов, 2. мереджмент повторного применения активов, 3. менеджмент повторного применения программ.

Процессы соглашения

Процессы соглашения определяют действия, необходимые для выработки соглашений между двумя организациями. Если реализуется процесс приобретения, то он обеспечивает средства для проведения деловой деятельности с поставщиком продуктов, предоставляемых для применения в функционирующей системе, услугах поддержки этой системы или элементах системы, разработанных в рамках проекта. Если реализуется процесс поставки, то он обеспечивает средства для проведения проекта, в котором результатом является продукт или услуга, поставляемые приобретающей стороне.

Процессы организационного обеспечения проекта

Процессы организационного обеспечения проекта осуществляют менеджмент возможностей организаций приобретать и поставлять продукты или услуги через инициализацию, поддержку и управление проектами. Эти процессы обеспечивают ресурсы и инфраструктуру, необходимые для поддержки проектов, и гарантируют удовлетворение организационных целей и установленных соглашений. Они не претендуют на роль полной совокупности деловых процессов, реализующих менеджмент деловой деятельности организации.

Процессы проекта

В настоящем стандарте проект выбран как основа для описания процессов, относящихся к планированию, оценке и управлению. Принципы, связанные с этими процессами, могут применяться в любой области менеджмента организаций.

Существуют две категории процессов проекта. Процессы менеджмента проекта используются для планирования, выполнения, оценки и управления продвижением проекта. Процессы поддержки проекта обеспечивают выполнение специализированных целей менеджмента. Обе категории процессов проекта описаны ниже.

Технические процессы

Технические процессы используются для определения требований к системе, преобразования требований в полезный продукт, для разрешения постоянного копирования продукта (где это необходимо), применения

продукта, обеспечения требуемых услуг, поддержания обеспечения этих услуг и изъятия продукта из обращения, если он не используется при оказании услуги.

Процессы реализации программных средств

Процессы реализации программных средств используются для создания конкретного элемента системы (составной части), выполненного в виде программного средства. Эти процессы преобразуют заданные характеристики поведения, интерфейсы и ограничения на реализацию в действия, результатом которых становится системный элемент, удовлетворяющий требованиям, вытекающим из системных требований.

Процессы поддержки программных средств

Процессы поддержки программных средств предусматривают специально сфокусированную совокупность действий, направленных на выполнение специализированного программного процесса. Любой поддерживающий процесс помогает процессу реализации программных средств как единое целое с обособленной целью, внося вклад в успех и качество программного проекта.

Процессы повторного применения программных средств

Группа процессов повторного применения программных средств состоит из трех процессов, которые поддерживают возможности организации использовать повторно составные части программных средств за границами проекта. Эти процессы уникальны, поскольку, в соответствии с их природой, они используются вне границ какого-либо конкретного проекта.

3. Модели жизненного цикла

Понятие жизненного цикла является одним из базовых понятий методологии проектирования информационных систем. Жизненный цикл информационной системы представляет собой непрерывный процесс, начинающийся с момента принятия решения о создании информационной системы и заканчивающийся в момент полного изъятия ее из эксплуатации.

Жизненный цикл информационных систем регламентирует международный стандарт ISO/IEC 12207-2010.

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, размером, сложностью, потребностью в изменениях и возможностями. Каждая стадия описывается формулировкой цели и выходов. Процессы и действия жизненного цикла отбираются и исполняются на этих стадиях для полного удовлетворения цели и результатов этой стадии.

3.1. Каскадная модель ЖЦ

Каскадная модель предусматривает последовательную организацию работ. Демонстрируется классический подход к разработке различных систем из любой ПО.

В ранее существовавших ИС каждое приложение представляло собой единое целое. Для разработки такого типа приложений применялся каскадный способ. Его основной характеристикой является разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем. Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

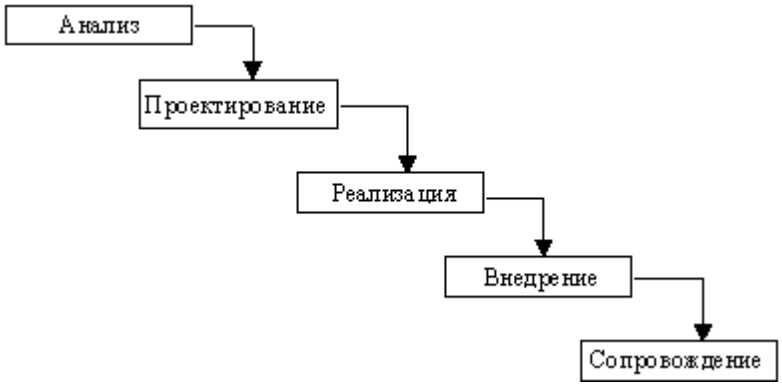

Рис. 1. Основные этапы разработки по каскадной модели

1 этап, анализ – исследование проблемы, четкая формулировка требований заказчика. Результат – техническое задание (задание на разработку), согласованное со всеми заинтересованными сторонами.

2 этап, проектирование – разработка проектных решений, удовлетворяющих всем требованиям технического задания. Результат – комплект проектной документации, содержащей все необходимые данные для реализации проекта.

3 этап, реализация –программирование в соответствии с проектными решениями, полученными на предыдущем этапе. Методы, используемые для реализации, не имеют принципиального значения. На этом этапе также производится проверка ПО на предмет соответствия требованиям технического задания, выявление скрытых недостатков, проявляющихся в реальных условиях работы ИС. Результат – готовый программный продукт.

4 этап, внедрение – сдача готового проекта. Главная задача – убедить заказчика, что все его требования реализованы в полной мере.

5 этап, сопровождение – в ходе сопровождения в программу вносятся изменения, с тем, чтобы исправить обнаруженные в процессе использования дефекты и недоработки (первичное сопровождение), а также для добавления новой функциональности, с целью повысить удобство использования и применимость ПО (зрелое сопровождение).

Преимущества

Каскадная модель имеет следующие преимущества:

Проста и понятна заказчикам, т.к часто используется другими организациями для отслеживания проектов, не связанных с разработкой ПО

Проста и удобна в применении:

- процесс разработки выполняется поэтапно.
- ее структурой может руководствоваться даже слабо подготовленный в техническом плане или - неопытный персонал;
- она способствует осуществлению строгого контроля менеджмента проекта;
- каждую стадию могут выполнять независимые команды (все документировано);
- позволяет достаточно точно планировать сроки и затраты.

Положительные стороны каскадного подхода заключаются в следующем:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты,
- в самом начале разработки достаточно точно и полно сформулировать все требования, с тем, чтобы предоставить разработчикам свободу в выборе реализации, наилучшей с технической точки зрения.

Недостатки

Недостатки каскадной модели связаны с тем, что реальный процесс создания ИС никогда не укладывается в такую жесткую схему. Недостатки каскадной модели особо остро проявляются в случае, когда трудно (или невозможно) сформулировать требования или требования могут меняться в процессе выполнения проекта. В этом случае разработка ПО имеет принципиально циклический характер.

Основные недостатки:

- существенная задержка получения результатов,
- ошибки и недоработки на любом этапе выясняются, как правило, на последующих этапах работы, что приводит к необходимости возврата на предыдущие стадии,
- сложность распараллеливания работ по проекту,
- чрезмерная информационная перенасыщенность каждого из этапов,
- сложность управления проектом,
- высокий уровень риска и ненадежность инвестиций.

Согласование результатов с пользователями производится только в точках, планируемых после завершения каждого этапа работ. Следовательно, требования к ИС "заморожены" в виде технического задания на все время ее создания. Таким образом, пользователи могут внести свои замечания только после того, как работа над системой будет полностью завершена. В случае неточного изложения требований или их изменения в течение длительного периода создания ПО, пользователи получают систему, не удовлетворяющую их потребностям. Причем, несоответствия требованиям могут возникать на любом этапе разработки – искажения могут непреднамеренно вноситься и проектировщиками-аналитиками и программистами, т.к. они не обязательно хорошо разбираются в ПО, для которой производится разработка ИС.

Кроме того, модели (как функциональные, так и информационные) автоматизируемого объекта могут устареть за время их разработки.

Ошибки, допущенные на более ранних стадиях обнаруживаются, как правило, не сразу на следующем уровне, а позднее (через несколько этапов), а это означает, что часть проекта должна быть возвращена на начальный уровень работы.

Сложность параллельного ведения работ связана с необходимостью постоянного согласования различных частей проекта. Пока одни работают, другие простаивают. Кроме того, после того, как передали проект на другую стадию, может найтись лучшее решение, но оно уже не может быть использовано.

Информационная перенасыщенность – при внесении изменений в одну из частей проекта необходимо информировать всех разработчиков, которые используют эту часть в своей работе. А это может потребовать повторного тестирования и даже внесения изменений в уже готовые части проекта.

И ещё один серьёзный недостаток – конфликт между разработчиками. Он обусловлен тем, что возврат на предыдущую стадию обычно сопровождается поиском причин и виновных.

Применимость

Между тем, каскадная модель не утратила своей актуальности при решении следующих типов задач:

Требования и их реализация максимально четко определены и понятны; используется неизменяемое определение продукта и вполне понятные технические методики. Это задачи типа:

- научно-вычислительного характера (пакеты и библиотеки научных программ типа расчета несущих конструкций зданий, мостов, …),
- операционные системы и компиляторы,
- системы реального времени управления конкретными объектами.

Кроме того, каскадная модель применима в условиях:

- повторная разработка типового продукта (автоматизированного бухгалтерского учета, начисления зарплаты, …),
- выпуск новой версии уже существующего продукта, если вносимые изменения вполне определены и управляемы (перенос уже существующего продукта на новую платформу).

Принципы каскадной модели находят применение как элементы моделей других типов, о чем речь пойдет ниже.

3.2. Спиральная модель ЖЦ

Предлагает итерационный процесс разработки ИС. Основной упор делается на начальные этапы ЖЦ – анализ и проектирование, так как именно здесь проверяется и обосновывается реализуемость технических решений путем создания прототипов.

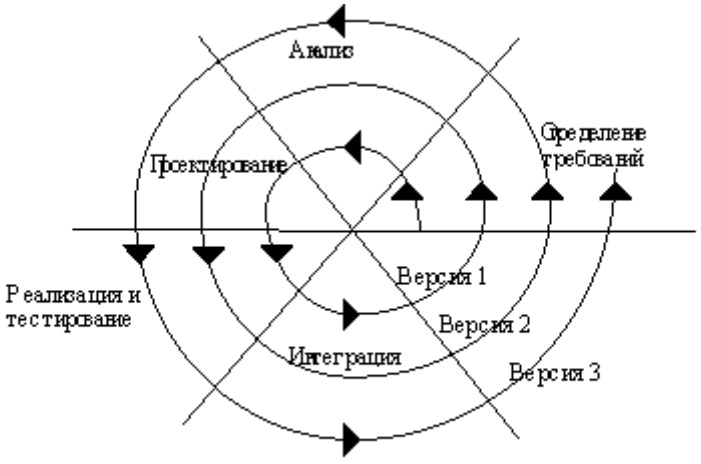

Рис. 2. Спиральная модель ЖЦ

Схема работы спиральной модели выглядит следующим образом. Разработка вариантов продукта представляется как набор циклов раскручивающейся спирали. Каждому циклу спирали соответствует такое же

количество стадий, как и в модели каскадного процесса. При этом начальные стадии, связанные с анализом и планированием, представлены более подробно с добавлением новых элементов. В каждом цикле выделяются четыре базовые фазы:

- определение целей, альтернативных вариантов и ограничений,
- оценка альтернативных вариантов, идентификация и разрешение рисков,
- разработка продукта следующего уровня,
- планирование следующей фазы.

"Раскручивание" проекта начинается с анализа общей постановки задачи на разработку ПО. Здесь на первой фазе определяются общие цели, устанавливаются предварительные ограничения, определяются возможные альтернативы подходов к решению задачи. Далее проводится оценка подходов, устанавливаются их риски. На шаге разработки создается концепция (видение) продукта и путей его создания.

Следующий цикл – разработка проекта – начинается с планирования разработки. На фазе определения целей устанавливаются ограничения проекта (по срокам, объему финансирования, ресурсам и т.д.), определяются альтернативы проектирования, связанные с альтернативами требований, применяемыми технологиями проектирования, привлечением субподрядчиков. На фазе оценки альтернатив устанавливаются риски вариантов и делается выбор варианта для дальнейшей реализации. На фазе разработки выполняется проектирование и создается демо-версия, отражающая основные проектные решения.

Следующий цикл –реализация ПО – также начинается с планирования. Альтернативными вариантами реализации могут быть применяемые технологии реализации, привлекаемые ресурсы. Оценка альтернатив и связанных с ними рисков на этом цикле определяется степенью «отработанности» технологий и «качеством» имеющихся ресурсов. Фаза разработки выполняется по каскадной модели с выходом – действующим вариантом (прототипом) продукта.

Особенности

Отметим некоторые особенности спиральной модели:

До начала разработки ПО есть несколько полных циклов анализа требований и проектирования.

Количество циклов модели (как в части анализа и проектирования, так и в части реализации) не ограничено и определяется сложностью и объемом задачи

В модели предполагаются возвраты на оставленные варианты при изменении стоимости рисков.

Каждая итерация представляет собой законченный цикл разработки, приводящий к выпуску внутренней или внешней версии и которая совершенствуется от итерации к итерации, чтобы стать законченной системой.

Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его

качество и планируются работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

Разработка итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная же задача – как можно быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований.

Преимущества

Существенно упрощается внесение изменений в проект при изменении требований заказчика,

Отдельные элементы ИС интегрируются в единое целое постепенно, причем, процесс интеграции производится фактически непрерывно,

Уменьшение уровня рисков, т.к. риски обнаруживаются именно в процессе интеграции. Риск максимален в начале разработки проекта, по мере продвижения он уменьшается,

Гибкость в управлении проектом. Например, можно сократить сроки разработки за счет уменьшения функциональности системы или использовать в качестве составных частей вместо своих разработок продукцию других фирм,

Итерационный подход упрощает повторное использование компонентов. Анализ проекта после проведения нескольких начальных итераций позволяет выявить общие, многократно используемые компоненты, которые на последующих итерациях будут совершенствоваться,

Спиральная модель позволяет получить более надежную и устойчивую систему, т.к. ошибки и слабые места обнаруживаются и исправляются на каждой итерации.

Недостатки

Основные недостатки спиральной модели связаны с ее сложностью:

- слобжность анализа и оценки рисков при выборе вариантов,
- сложность поддержания версий продукта (хранение версий, возврат к ранним версиям, комбинация версий),
- сложность оценки точки перехода на следующий цикл,
- есконечность модели – на каждом витке заказчик может выдвигать новые требования, которые приводят к необходимости следующего цикла разработки.

Основная проблема спирального цикла – определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Применимость

Спиральную модель целесообразно применять при следующих условиях:

- когда пользователи не уверены в своих потребностях или когда требования слишком сложны и могут меняться в процессе выполнения проекта и необходимо прототипирование для анализа и оценки требований,
- когда достижение успеха не гарантировано и необходима оценка рисков продолжения проекта,
- когда проект является сложным, дорогостоящим и обоснование его финансирования возможно только в процессе его выполнения,
- когда речь идет о применении новых технологий, что связано с риском их освоения и достижения ожидаемого результата,
- при выполнении очень больших проектов, которые в силу ограниченности ресурсов можно делать только по частям.

3.3. Сводная таблица групп процессов ЖЦ

Ф – формирование требований, П – проектирование, Р – реализация, Т – тестирование, В – ввод в действие, Э – эксплуатация, С – снятие с эксплуатации

Таблица 2. Взаимосвязь процессов и стадий разработки

процессы			стадии						
			ф	п	р	т	в	э	с
Основные процессы	Приобретение	Инициирование приобретения	●						
		Подготовка заявочных предложений	●						
		Подготовка и корректировка договора	●						
		Надзор за деятельностью поставщика	●	●	●	●	●		
		Приемка и завершение работ				●	●		
	Поставка	Инициирование поставки	●						
		Подготовка ответа на заявочные предложения	●						
		Подготовка договора	●						
		Планирование	●						
		Выполнение и контроль	●	●	●	●	●		
		Проверка и оценка	●	●	●	●	●		
		Поставка и завершение работ				●	●		
	Разработка	Подготовительная работа	●						
		Анализ требований к системе	●	●					
		Проектирование архитектуры системы		●					
		Детальное проектирование		●					
		Кодирование и тестирование			●				

		Интеграция ПО			●				
		Квалификационное тестирование			●	●			
		Установка					●		
		Приемка					●		
	Эксплуатация	Подготовительные работы					●		
		Эксплуатационное тестирование					●		
		Эксплуатация						●	
		Поддержка пользователей						●	
	Сопровождение	Подготовительная работа				●	●		
		Анализ проблем и запросов модификации					●	●	
		Модификация ПО					●	●	
		Проверка и приемка					●	●	
		Снятие с эксплуатации							●
Вспомогательные процессы	Документирование	Подготовительная работа	●	●	●	●	●	●	●
		Проектирование и разработка	●	●	●	●	●	●	●
		Выпуск документации	●	●	●	●	●	●	●
		Сопровождение	●	●	●	●	●	●	●
	Управление конфигурацией	Подготовительная работа	●						
		Идентификация конфигурации		●					
		Контроль конфигурации		●	●	●	●	●	
		Учет состояния конфигурации		●	●	●	●	●	
		Оценка конфигурации		●	●	●	●	●	
		Управление выпуском				●	●	●	
		Верификация	●	●	●	●	●	●	
		Аттестация				●	●	●	
	Оценка	Оценка управления проектом	●	●	●	●	●	●	
		Техническая оценка	●	●	●	●	●	●	
		Аудит	●	●	●	●	●	●	
		Разрешение проблем	●	●	●	●	●	●	●

Организационные процессы	Управление	Определение области управления	●	●	●	●	●	●	
		Планирование	●	●	●	●	●	●	●
		Выполнение и контроль	●	●	●	●	●	●	●
		Проверка и оценка	●	●	●	●	●	●	●
		Завершение						●	●
	Создание инфраструктуры	Создание инфраструктуры	●	●					
		Сопровождение инфраструктуры	●	●	●	●	●	●	
	Усовершенствование	Создание процесса	●	●					
		Оценка процесса	●	●	●	●	●	●	
		Усовершенствование		●	●	●	●	●	
	Обучение	Разработка учебных материалов	●	●	●				●
		Реализация плана обучения	●	●	●	●	●	●	●

4. Технологии проектирования ПО

Разработка программного обеспечения (ПО) – довольно трудоемкая задача, причем сложность возрастает многократно при увеличении объема работы. Можно, конечно, разрабатывать ПО на интуитивном уровне, основываясь на знаниях и умениях менеджера текущего проекта и его команды. Но такой подход не позволит разрабатывать сложные системы, поскольку риски высоки, а процесс управления рисками не отработан.

Согласно стандарту CMMI, организации, которым можно доверять создание ПО, должны иметь уровень три или выше. Тогда вероятность получения продукта в поставленные сроки с заданной функциональностью и бюджетом достаточно велика. Организации, находящиеся на данном уровне имеют корпоративные стандарты управления проектами, что позволяет им добиваться успеха с большой долей вероятности. В настоящее время все ведущие компании-разработчики программных продуктов отработали свои технологические процессы создания ПО, многие создавали стандарты собственными силами, другие – адаптируя приобретенные технологии.

Очевидно, что, имея отработанную схему процесса создания программного продукта, эти организации могут ее продавать как отдельное "know-how". для этого имеющаяся схема реализуется в виде технологии создания программного продукта.

Технология разработки ПО – это совокупность процессов и методов создания программного продукта. Промышленные технологии создания программных продуктов имеют несколько обязательных характеристик, среди них:

- обеспечение поддержки жизненного цикла информационной системы, то есть каждая технология основана на какой-либо модели жизненного цикла,
- гарантия достижения целей разработки, то есть достижение требований к системе при соблюдении сроков и бюджета,
- соответствие принципам управляемости,
- не зависят от средств реализации ИС.

Обычно промышленная технология представлена в виде упорядоченной совокупности взаимосвязанных технологических процессов в рамках ЖЦ ПО, а сам технологический процесс в виде пошаговой процедуры, определяющей последовательность технологических операций проектирования. Технологические операции заданы при помощи графических и текстовых нотаций, кроме того, для оценки результатов выполнения технологических операций, используются критерии и правила, которые также заданы в промышленной технологии. Саму технологическую операцию можно представить в виде черного ящика с тремя входами и одним выходом (Рис. 3). На вход поступают некоторые исходные данные в стандартном представлении, на выходе получаем результаты, кроме того входом являются исполнители и

технические средства, а также инструкции, стандарты и критерии оценки результатов.

Технологические операции объединяются в технологическую цепочку, результатом которой является продукт, ценный для потребителя.

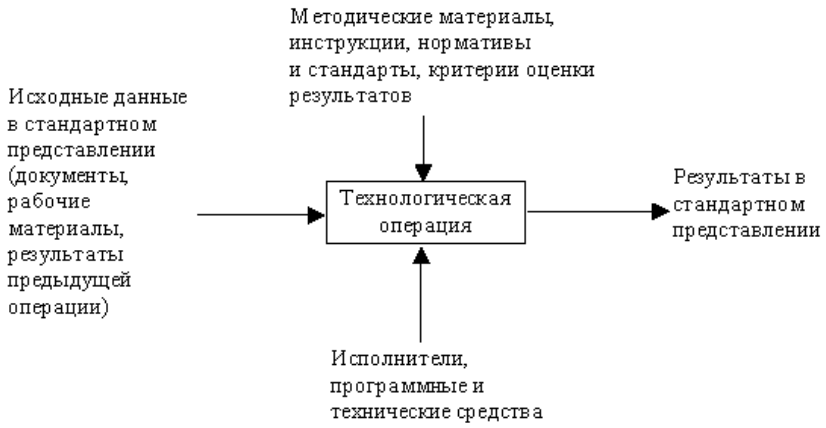

Рис. 3. Схема технологической операции

Выбор конкретной технологии разработки программного продукта зависит от многих факторов, оценка которых должна привести к адекватному решению по данному вопросу. Результатом такой оценки будет одна из альтернатив:

- выбор конкретной технологии разработки ПО и ее приобретение;
- вывод об отсутствии адекватных технологий.

Во втором случае решением может быть модификация одной из существующих технологий, наиболее подходящей по требованиям, разработка собственной технологии или отказаться от внедрения.

Оценивать технологию разработки ПО можно по следующим технико-экономическим характеристикам:

- функциональные характеристики процессов жизненного цикла,
- функциональные характеристики применения (среда функционирования, совместимость с другими ТС ПО, соответствие технологическим стандартам),
- характеристики качества (надежность, удобство использования, эффективность, как осуществляется сопровождение, переносимость),
- общие характеристики (затраты на технологию, лицензионная политика, оценочный эффект от внедрения ТС ПО, инфраструктура, требуемая для внедрения ТС ПО, доступность и качество обучения, сертификация поставщика, поддержка поставщика).

Приведем ниже несколько примеров промышленных технологий создания программных продуктов.

4.1. Rational Unified Process RUP (IBM)

Rational Unified Process является одним из корпоративных стандартов в области создания программного обеспечения. Данная технология была разработана компанией Rational Software, входящей в настоящее время в состав IBM.[13]

Основой данной технологии является поэтапное моделирование продукта средствами UML, в ней реализуется итерационный и инкрементный подход к созданию ПО. Разработка системы выполняется в виде нескольких краткосрочных мини-проектов фиксированной длительности (от 2 до 6 недель), называемых итерациями. Каждая итерация включает свои собственные этапы анализа требований, проектирования, реализации, тестирования, интеграции и завершается созданием работающей системы.

Данная технология достаточно гибкая и имеет возможность масштабирования, то есть подходит как для малых проектов и рабочих групп, так и для больших. Технология состоит из 4 основных этапов, сопровождающимися 9-ю видами деятельности (процессами).

Основные этапы и связанные с ними затраты по ресурсам и времени представлены графически на Рис. 4. Как следует из рисунка, фаза Конструирование является наиболее затратной как по ресурсам, так и по времени.

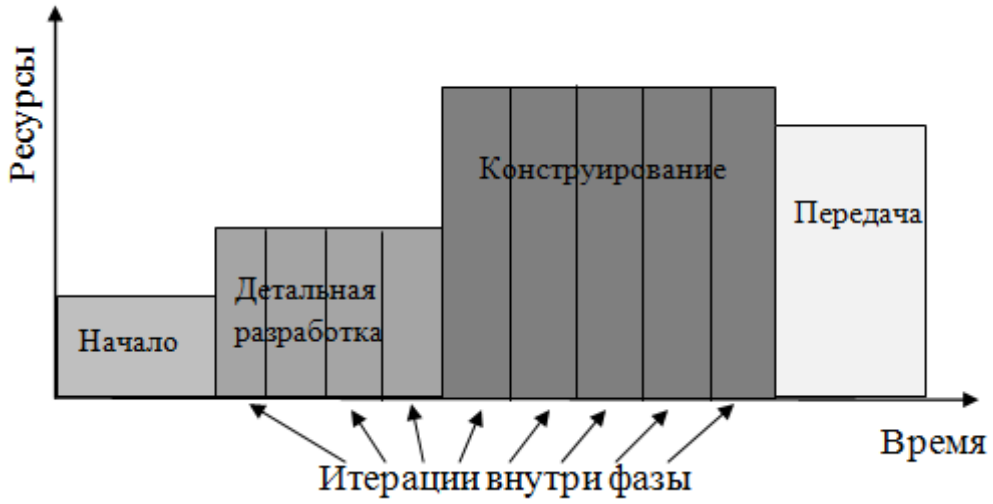

Рис. 4. Основные фазы RUP

При разработке ПО согласно технологии RUP, жизненный цикл разбивается на несколько итераций, на каждой создается дополненная версия ПО, внутри каждой итерации обязательно содержаться все четыре фазы технологии. Каждая фаза, в свою очередь, также может быть разбита на итерации в зависимости от трудоемкости самой фазы. Все фазы проходятся последовательно, каждая фаза завершается оценкой полученных результатов в

четко определенной контрольной точке, здесь принимается решение о дальнейшей разработке и передача управления на следующую фазу.

Перечислим основные задачи, которые решаются в рамках каждой фазы.

Начало (Inception) Во время начальной фазы вырабатывается бизнес-план проекта - определяется, сколько приблизительно он будет стоить и какой доход принесет. Определяются также границы проекта, и выполняется некоторый начальный анализ для оценки размеров проекта.

Основная цель— достичь компромисса между всеми заинтересованными лицами относительно задач проекта.

При этом:

- формируются видение и границы проекта,
- создается экономическое обоснование (business case),
- определяются основные требования, ограничения и ключевая функциональность продукта,
- создается базовая версия,
- оцениваются риски.

Результатами начальной фазы являются:

- общее описание системы: основные требования к проекту, его характеристики и ограничения,
- начальная модель вариантов использования (степень готовности - 10-20%),
- начальный проектный глоссарий (словарь терминов),
- начальный бизнес-план,
- план проекта, отражающий стадии и итерации,
- один или несколько прототипов.

Детальная разработка (Elaboration). На этой фазе выявляются более детальные требования к системе, выполняется высокоуровневый анализ предметной области и проектирование для построения базовой архитектуры системы, создается план конструирования и устраняются наиболее рискованные элементы проекта.

Основная цель— на базе основных, наиболее существенных требований разработать стабильную базовую архитектуру продукта. архитектура включает модель предметной области, которая отражает понимание бизнеса и служит отправным пунктом для формирования основных классов предметной области и технологическую платформу, определяющую основные элементы технологии реализации системы и их взаимодействие.

На этой фазе осуществляется:

- Документирование требований (включая детальное описание для большинства прецедентов использования).
- Получение спроектированной, реализованной и оттестированной исполняемой архитектуры.
- Обновление экономического обоснования и получение более точных оценок сроков и стоимости.

- Снижение основных рисков.

Результатами фазы разработки являются:

- модель вариантов использования (завершенная, по крайней мере, на 80%), определяющая функциональные требования к системе;
- перечень дополнительных требований, включая требования нефункционального характера и требования, не связанные с конкретными вариантами использования;
- описание базовой архитектуры будущей системы;
- работающий прототип;
- уточненный бизнес-план;
- план разработки всего проекта, отражающий итерации и критерии оценки для каждой итерации.

Основными признаками завершения стадии разработки являются два события:

- разработчики в состоянии оценить с достаточно высокой точностью, сколько времени потребуется на реализацию каждого варианта использования;
- идентифицированы все наиболее серьезные риски, и степень понимания наиболее важных из них такова, что известно, как справиться с ними.

Построение (Construction) Во время этой фазы происходит реализация большей части функциональности продукта. Это самая масштабная по времени и ресурсрзатратам часть проекта.

Основная цель— детальное прояснение требований и разработка системы

Эта фаза обычно разбивается на несколько итераций, каждая из которых носит инкрементальный характер, то есть вся работа делится на несколько этапов, каждый из которых является продолжением и усложнением предыдущего. Также итерации могут носить повторяющийся характер, если задача этапа не выполнена или выполнена не в полной мере.

- итерации являются инкрементными в соответствии с той функцией, которую они выполняют. Каждая итерация добавляет очередные конструкции к вариантам использования, реализованным во время предыдущих итераций;
- итерации являются повторяющимися по отношению к разрабатываемому коду. На каждой итерации некоторая часть существующего кода переписывается с целью сделать его более гибким.

Фаза Построение завершается первым внешним релизом системы и вехой *начальной функциональной готовности*

Передача (Transition). В ходе этой фазы продукт, полученный на предыдущей фазе, передается конечному пользователю или заказчику.

Основная цель— передать систему конечным пользователям

В ходе передачи:

- оценивается качество продукта,
- создается финальная версия продукта.

Финальная версия передается от разработчика к заказчику (β-версия, обучение пользователей).

Фаза Передача завершается полным переводом готового ПО в эксплуатацию на стороне заказчика.

Технология RUP определяет 9 процессов, из них 6 основных и 3 вспомогательных. Рапределение интенсивности процессов по разным фазам технологии представлено на Рис. 5.

Основные процессы

Моделирование предметной области

Цели этой деятельности — понять бизнес-контекст, в котором должна будет работать система (и убедиться, что все заинтересованные лица понимают его одинаково), понять возможные проблемы, оценить возможные их решения и их последствия для бизнеса организации, в которой будет работать система.

Определение требований

Цели — понять, что должна делать система, определить границы системы и основу для планирования проекта и оценок ресурсозатрат в нем.

Анализ и проектирование

Выработка архитектуры системы на основе ключевых требований, создание проектной модели, представленной в виде диаграмм UML, описывающих продукт с различных точек зрения.

Реализация

Разработка исходного кода, компонент системы, тестирование и интегрирование компонент.

Тестирование

Общая оценка дефектов продукта, его качество в целом; оценка степени соответствия исходным требованиям.

Поддерживающие (вспомогательные)процессы:

Развертывание (Deployment).

Цели — развернуть систему в ее рабочем окружении и оценить ее работоспособность.

Управление конфигурациями и изменениями (Configuration and Change Management).

Определение элементов, подлежащих хранению и правил построения из них согласованных конфигураций, поддержание целостности текущего состояния системы, проверка согласованности вносимых изменений.

Управление проектом (Project Management).

Включает планирование, управление персоналом, обеспечения связей с другими заинтересованными лицами, управление рисками, отслеживание текущего состояния проекта.

Управление средой проекта (Environment).

Настройка процесса под конкретный проект, выбор и смена технологий и инструментов, используемых в проекте.

Для автоматизации генерации кода и создания документации используется инструментальные средства комплекса Rational Rose. В основе работы Rational Rose лежит построение диаграмм и спецификаций UML, определяющих архитектуру системы, ее статические и динамические аспекты.

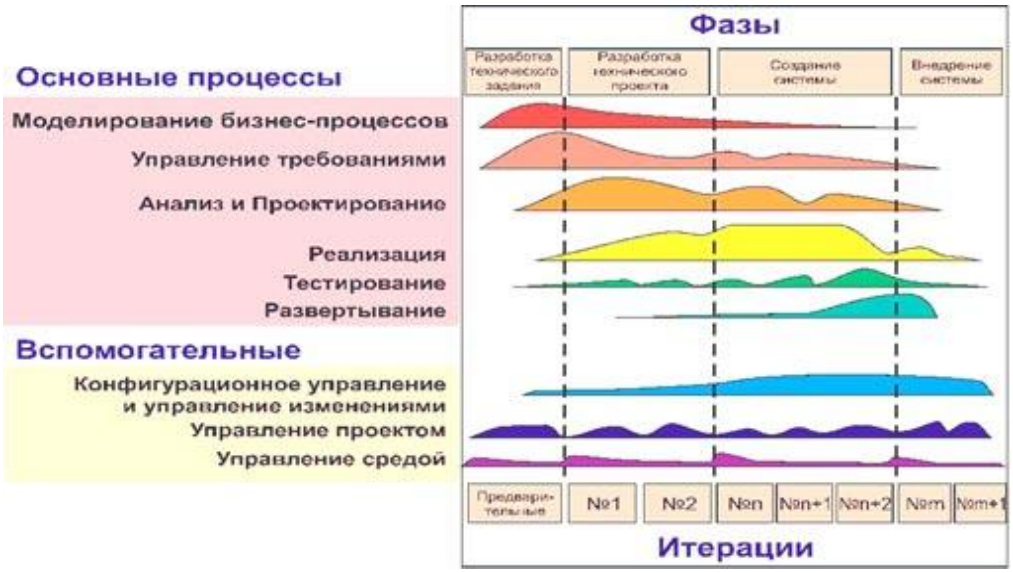

Рис. 5. Интенсивность процессов PUP на разных фазах

В результате разработки проекта с помощью Rational Rose формируются следующие документы:

- диаграммы UML, представляющие собой модель разрабатываемой программной системы;
- спецификации классов, объектов, атрибутов и операций;
- заготовки текстов программ.

Тексты программ являются заготовками для последующей работы программистов. Состав информации, включаемой в программные файлы, определяется либо по умолчанию, либо по усмотрению пользователя. В дальнейшем эти исходные тексты развиваются программистами в полноценные программы.

Подводя итоги описания технологии RUP, подчеркнем основные аспекты ЖЦ, положенного в ее основу. В данной технологии используется каскадная модель ЖЦ с элементами инкрементальной. Черты инкрементальной модели выражаются в разбивке каждого этапа на итераций, целью которых является последовательное осмысление стоящих проблем, наращивание эффективных решений и снижение риска потенциальных ошибок в проекте. Каскадной – в переходе к следующей фазе после достижения заданного уровня качества, наличии четких правил перехода и использовании результатов предыдущей фазы для начала текущей.

4.2. Custom Development Method CDM (Oracle)

Данная технология основана на использовании инструментального комплекса Oracle Developer Suite. Технология в основном ориентирована на разработку ПО, в котором приоритетным является разработка и использование базы данных, в том числе конверсия базы данных при переходе на новое ПО. [10, 12]

При этом технология имеет две разновидности: CDM-classic и CDM-fast track, состоит из 6-ти основных фаз и 11 процессов.

CDM-classic – это технология разработки, рассчитанная на крупномасштабные проекты с временем от восьми месяцев до трех лет, CDM-fast track – для разработки маленьких проектов со временем функционирования от четырех до 16 месяцев.

Опишем более подробно оба вида технологии CDM.

CDM classic

В данной технологии определяется 6 фаз, каждая из которых может быть осуществлена в несколько этапов.

Определение (**Definition**) на данной фазе определяются цели создания системы, приоритеты и ограничения, разрабатывается системная архитектура и составляется план разработки.

Основная цель— получение согласия высшего руководства для перехода на следующую фазу.

- определение требований высокого уровня;
- определение бизнес-требований;
- получение области (scope) проекта.

Анализ (Analysis) здесь строятся модель информационных потребностей (диаграмма "сущность-связь"), диаграмма функциональной иерархии (на основе функциональной декомпозиции системы), матрица перекрестных ссылок и диаграмма потоков данных.

Основная цель— получение детальных требований к системе.

- детальное изучение бизнес-требований;
- моделирование бизнес-процессов;
- перевод моделей в требования к ИС;
- определение архитектуры;
- определение возможностей конверсии.

Дизайн (**Design**) на данном этапе разрабатывается подробная архитектура системы, проектируются схема реляционной БД и программные модули, устанавливаются перекрестные ссылки между компонентами системы для анализа их взаимного влияния и контроля за изменениями.

Основная цель— перевод требований в спецификации

- получение спецификаций;
- базирование на технологии.

Построение (**Build**). При построении создается БД, строятся прикладные системы, производится их тестирование, проверка качества и соответствия требованиям пользователей. Создается системная документация, материалы для обучения и руководства пользователей.

Основная цель— получение продукта

- кодирование;
- тестирование.

Передача (**Transition**) здесь анализируются производительность и целостность системы.

Основная цель— установка системы у клиента и подготовка к стадии production

- подготовка персонала;
- проведение приемочных тестов;
- установка;
- переведение в рабочее состояние.

Работа (**Production**) на данной фазе выполняется поддержка и, при необходимости, модификация системы.

Основная цель— поддержание работоспособности системы

- наблюдение за ИС;
- поддержка ИС;
- планирование следующих изменений;
- переведение в рабочее состояние.

CDM-classic определяет 11 процессов:

1. определение бизнес-требований, или постановка задачи (Business Requirements Definition);
2. исследование существующих систем (Existing Systems Examination). Выполнение этого процесса должно обеспечить понимание состояния существующего технического и программного обеспечения для планирования необходимых изменений;
3. определение технической архитектуры (Technical Architecture);
4. проектирование и реализация базы данных (Database Design and Build). Процесс предусматривает проектирование и реализацию реляционной базы данных, включая создание индексов и других объектов БД;
5. проектирование и реализация модулей (Module Design and Build). Этот процесс является основным в проекте. Он включает непосредственное проектирование приложения и создание кода прикладной программы;
6. конвертирование данных (Data Conversion). Цель этого процесса - преобразовывать, перенести и проверить согласованность и непротиворечивость данных, оставшихся в наследство от "старой" системы и необходимых для работы в новой системе;
7. документирование (Documentation);
8. тестирование (Testing);
9. обучение (Training);

10. внедрение, или переход к новой системе (Transition). Этот процесс включает решение задач установки, ввода новой системы в эксплуатацию, прекращения эксплуатации старых систем;
11. поддержка и сопровождение (Post-System Support).

Распределение процессов по фазам показано на Рис. 6, здесь схематически представлено присутствие процесса на этапе и объем работ, связанных с этим процессом. Так, например, процесс определения бизнес-требований будет нужен только на первых двух этапах, а процесс конвертирования данных распределен по всем этапам, кроме эксплуатации, практически равномерно.

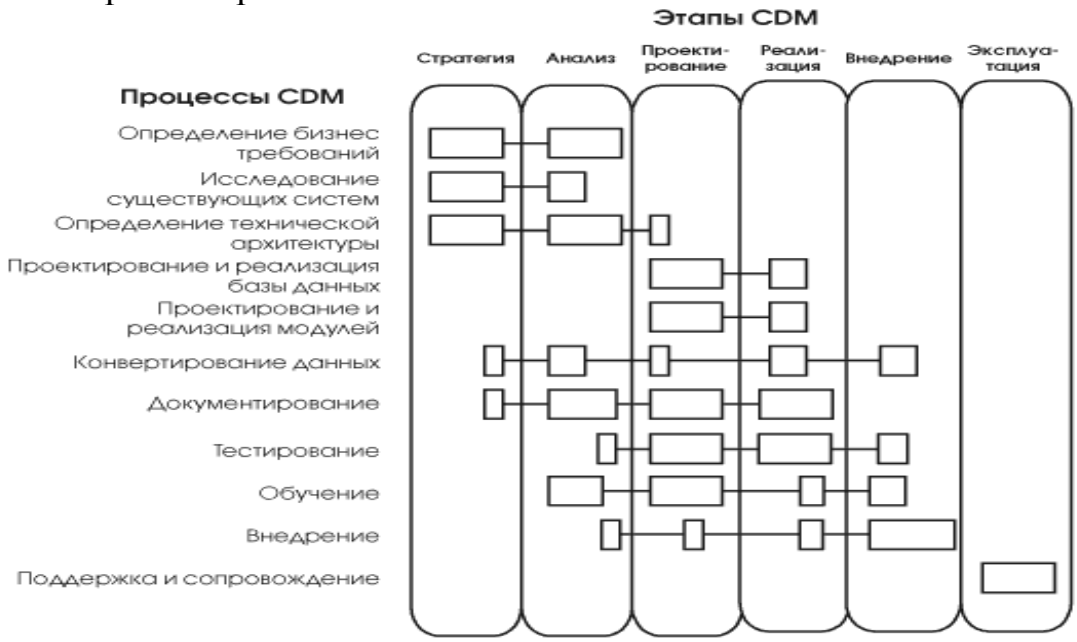

Рис. 6. распределение процессов по фазам

CDM fast track

Как уже было сказано выше, данная технология рассчитана для небольших проектов с временными рамками от четырех до шестнадцати месяцев. Поэтому основными характеристиками данного продукта является ориентация на быстрое получение продукта, следовательно, разработка имеет короткий цикл и должна содержать небольшой объем работ.

Данная технология, благодаря своим характеристикам, входит в группу Agile (подвижный, сообразительный) методологий. Она имеет меньше фаз, чем CM-classic, но такое же количество процессов. При этом все фазы CM-classic включены в данную методологию, просто некоторые фазы объединены для ускорения получения результата и перехода на следующую фазу.

Перечислим фазы технологии CDM fast track и основные задачи, которые решаются внутри фазы. Графически фазы и переходы между ними представлены на Рис. 7.

Определение (Definition)

- Получение области (scope) проекта;
- Определение бизнес-требований;
- Установление приоритета требований.

Моделирование требований (Requirements Modeling)

- Моделирование бизнес-требований;
- Построение функционального прототипа (БД, модули);
- Создание списка приоритетов по функциональной модели.

Дизайн и построение (System Design and Generation)

- Развитие системы в несколько итераций согласно уточнениям требований;
- Работа по MoSCoW List ;
- Проработка решения с конечным пользователем.

Передача в работу (Transition to Production)

- Установка и запуск у конечного пользователя;
- Подготовка к следующей версии.

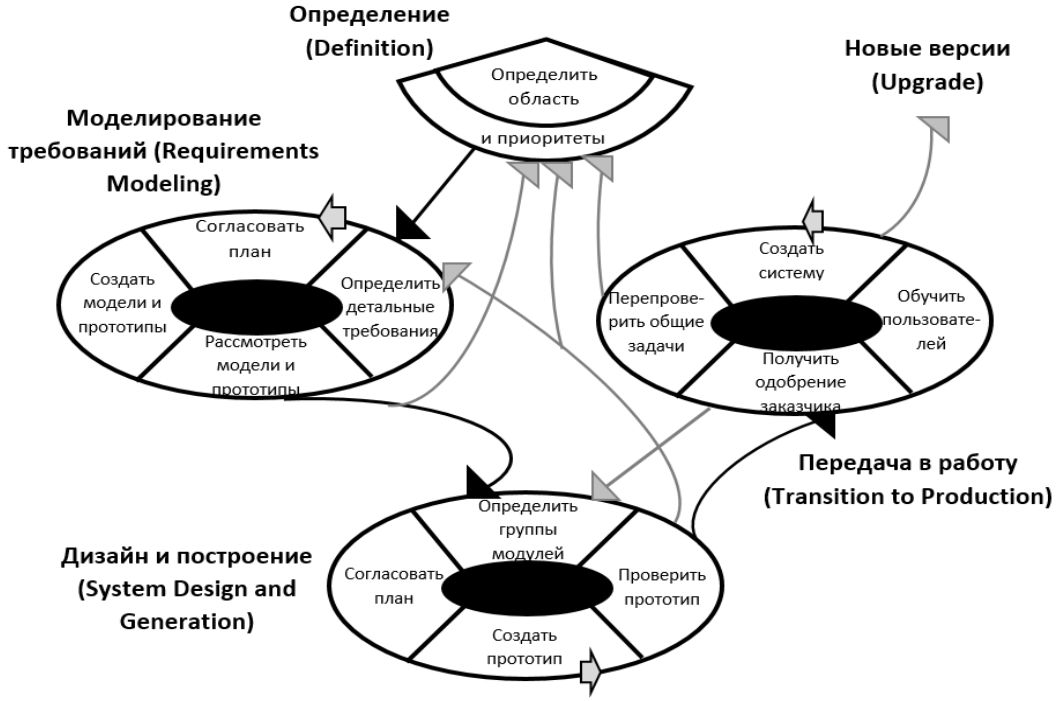

Рис. 7. Графическое представление фаз технологии CDM fast track

Теперь проведем анализ модели ЖЦ, используемой в каждой из технологий. CDM-classic является классическим примером методологии основанной на каскадной модели ЖЦ, здесь четко определены фазы и задачи для каждой фазы, есть точки перехода на следующую фазу. Вариант технологии CDM fast track представляет собой итерационную модель с элементами спирального или инкрементального развития на каждом из этапов. То, что основой технологии является итерационная модель, четко просматривается на Рис. 7, т.е. каждая из фаз имеет возможность возврата на предыдущие шаги для проработки результатов. Спиральная или

инкрементальная схема может использоваться внутри фазы для детальной проработки результатов и снижения неопределенности.

4.3. Microsoft Solution Framework MSF (MicroSoft)

Основным девизом команды Microsoft при продвижении данной технологии является лозунг: "Лучшее из двух миров!" – это значит, что технология MSF должна совместить лучшие черты из двух основных, и конкурирующих между собой, моделей ЖЦ ПО – каскадной и спиральной. Из каскадной модели данная технология позаимствовала чёткость целей и переходов для каждого из этапов, а из спиральной – минимизацию рисков на каждом из них.[11]

Каскадная модель предусматривает четкий переход от этапа к этапу: работы следующего этапа начинаются только после выполнения всех задач предыдущего. Такой стиль подходит для проектов, в которых проектные требования четко определяются заранее и с большой вероятностью не будут корректироваться потом. Данная схема организации разработки очень удобна с точки зрения управления проектом, так как позволяет четко сформулировать состав и обязанности его участников и контролировать графики выполнения проекта.

Спиральная модель обычно ориентируется на крайний случай, когда требования и параметры проекта непрерывно корректируются, а новые требования формулируются лишь по мере необходимости выполнения конкретных работ. Такая схема часто ассоциируется с понятием "экстремальной разработки"; при этом исполнитель и заказчик работают в постоянном тесном сотрудничестве, клиент привлекается на каждом этапе, формулируя свои соображения по поводу созданных компонентов. Однако при такой организации очень велик риск, что процесс разработки выйдет из-под контроля, поэтому реально данная модель используется лишь в относительно небольших проектах.

Однако проблема заключается в том, что чаще всего все требования на задание действительно практически невозможно определить заранее, к тому же даже сформулированные требования подвергаются коррекции. Но тогда требуется повысить уровень управляемости проектом, без чего создание сложного ПО просто невозможно. Компромисс между этими противоречивыми требованиями и предоставляет модель процессов MSF, в которой сочетаются каскадная и спиральная модели разработки: проект реализуется поэтапно, с наличием соответствующих ключевых (контрольных) точек, а сама последовательность этапов может повторяться по спирали.

Данная технология претендует на всеобщность применения, т.е. предполагается, что с ее помощью можно работать над проектом любого масштаба, не внося существенных изменений в процесс применения.

Технология MSF состоит из 4х этапов, каждый из которых завершается "вехой" – ключевой точкой, в которой производится оценка достигнутого, каждый из этапов сопровождается группой из 6и процессов, в зависимости от

этапа в процессе смещается точка фокуса. Веха каждого из этапов – это определенные задачи с необходимым уровнем качества их выполнения. После достижения заданного уровня, можно переходить на выполнение следующего этапа.

На каждом этапе, кроме основной вехи, определен еще ряд задач, выполнение которых ведет к выполнению основных требований этапа.

Выработка концепции (Envisioning)

Промежуточные задачи этапа:

- оценка существующей ситуации;
- определение состава команды;
- определение структуры проекта;
- определение бизнес-целей;
- определение требований и профилей пользователей;
- разработка концепции решения;
- создание документа общей картины и области действия проекта;
- оценка риска.

Промежуточные вехи:

- Организован костяк команды;
- Создана общая картина решения.

Финальная веха:

- Утверждение документа общей картины.

Для организации костяка команды не обязательно нужен полный поименный список команды, но в документе структуры проекта должны быть определены роли и обязанности каждого члена команды, а также описана иерархия отчетности и ответственности в группе, каналы взаимодействия с заказчиком и общая структура команды.

Когда говориться о создании общей картины решения, речь идет о разработке концепции решения, которым должна руководствоваться команда для достижения долгосрочных бизнес-целей проекта. Область действия проекта определяет, что включается в контекст проекта, а что выходит за его рамки. На этой временной точке речь идет о создании первой версии документа, который находится в стадии рецензирования участниками команды и согласования заказчиком.

Планирование (Panning)

Промежуточные задачи этапа:

- анализ и документирование требований;
- разработка проект и основные архитектурные решения;
- функциональные спецификации системы;
- планы и календарные графики (тестирования и пилотной эксплуатации);
- выбор среды разработки.

Промежуточные вехи или стадии проектирования:

- концептуальное,

- логическое,
- физическое.

Финальная веха:

- утверждение проектных планов.

Промежуточные вехи делят данный этап на три стадии: концептуальное, логическое и физическое проектирование. На стадии концептуального проектирования задача рассматривается с точки зрения пользовательских и бизнес-требований и заканчивается определением набора сценариев использования системы. При **логическом** проектировании задача рассматривается с точки зрения проектной команды, решение представляется в виде набора сервисов. И уже на стадии **физического** проектирования задача рассматривается с точки зрения программистов, уточняются используемые технологии и программные интерфейсы.

Проектные планы включают в себя несколько различных аспектов:

- генеральный план и календарный график проекта;
- план управления рисками.

Разработка (Developing)

На этапе разработки создается решение, в том числе пишется и документируется код. В начале этого этапа команда проверяет выполнение всех задач, характерных для предыдущих этапов, а затем приступает к решению промежуточных задач.

Промежуточные задачи этапа:

- создание компонент решения (документация, код);
- разработка инфраструктуры

Промежуточная веха - Разработка завершена:

- решение готово к тестированию и стабилизации;
- выявление всех оставшихся проблем.

Финальная веха

- окончательное утверждение области действия проекта.

В этот момент все функции продукта готовы и прошли тестирование в рамках своего модуля. После этого продукт готов к внешнему тестированию и стабилизации. Кроме того, заказчики, пользователи, сотрудники службы поддержки и сопровождения, а также ключевые участники проекта могут предварительно оценить продукт и указать все недостатки, которые нужно устранить до его поставки.

Результаты этапа предполагают следующие элементы:

- исходный текст кода и исполняемые файлы;
- сценарии установки и конфигурации для развертывания;
- окончательная функциональная спецификация;
- элементы поддержки решения;
- спецификации и сценарии тестирования.

Стабилизация (Stabilizing)

Данный этап - подготовка к выпуску окончательной версии продукта, доводка его до заданного уровня качества. Здесь выполняется комплекс работ по тестированию (обнаружение и устранение дефектов), а также проверяется сценарий развертывания продукта и проводится пилотная эксплуатация.

Промежуточные задачи этапа:

- подготовка к выпуску окончательной версии продукта;
- доведение до заданного уровня качества;
- определение состава команды;
- обнаружение и устранение дефектов;
- пилотная эксплуатация в тестовой среде.

Финальная веха

- подтверждение готовности проекта к выпуску.

Когда решение становится достаточно устойчивым, проводится его пилотная эксплуатация в тестовой среде с привлечением пользователей и применением реальных сценариев работы.

Один из главных показателей этапа стабилизации - число обнаруженных ошибок. Сходимость этой величины в сторону устойчивого уменьшения - признак того, что близится завершение работ над продуктом. Важнейшая промежуточная контрольная точка - появление версии, в которой усилиями самой проектной команды не обнаружено ни одной ошибки. Далее следуют выпуски кандидат-релизов продукта для их исследования в условиях пилотной эксплуатации. Завершающая контрольная точка - подтверждение готовности продукта к выпуску и полноценному развертыванию в промышленной среде.

Развертывание (Deploying)

На этом этапе выполняется установка решения и необходимых компонентов окружения, проводится его стабилизация в промышленных условиях и передача проекта в руки группы сопровождения. Кроме того, анализируется проект в целом на предмет уровня удовлетворенности заказчика.

Промежуточные задачи этапа:

- установка решения и необходимых компонентов окружения;
- стабилизация в промышленных условиях;
- передача проекта в руки группы сопровождения;
- анализ проекта в целом на предмет уровня удовлетворенности заказчика

Финальная веха

- Подтверждение завершения внедрения

Следует подчеркнуть, что момент завершения данного этапа бывает достаточно сложно формально определить, так как выявление неполадок может продолжаться и в ходе промышленной эксплуатации. Именно поэтому необходимо четко сформулировать критерии для завершающей контрольной точки этапа развертывания и не пытаться отладить абсолютно все.

Технология MSF определяет следующие процессы, сопровождающие создание программного продукта:

- Управление продуктом
- Управление программой
- Разработка
- Удовлетворение потребителя
- Тестирование
- Управление выпуском

Также эти процессы называются кластерами и играют важную роль при создании команды.

Надо заметить, что в данной технологии созданию команды отведено важное место. При создании команды предлагается процессно-ролевой или кластерный подход, то есть каждый член команды имеет своей целью поддерживать один или несколько процессов, называемых кластерами. Модель проектной группы MSF подчеркивает важность построения ролевых кластеров в соответствии с нуждами бизнеса. Группировка связанных областей компетенции, каждая из которых имеет свою специфику, обеспечивает хорошую сбалансированность команды. Четкое определение целей повышает уровень ответственности и способствует лучшему их восприятию проектной командой, что незамедлительно сказывается наилучшим образом на качестве выпускаемого продукта. Поскольку каждая из целей одинаково необходима для успешности проекта, все роли находятся в равноправных партнерских взаимоотношениях с равной значимостью при принятии решений.

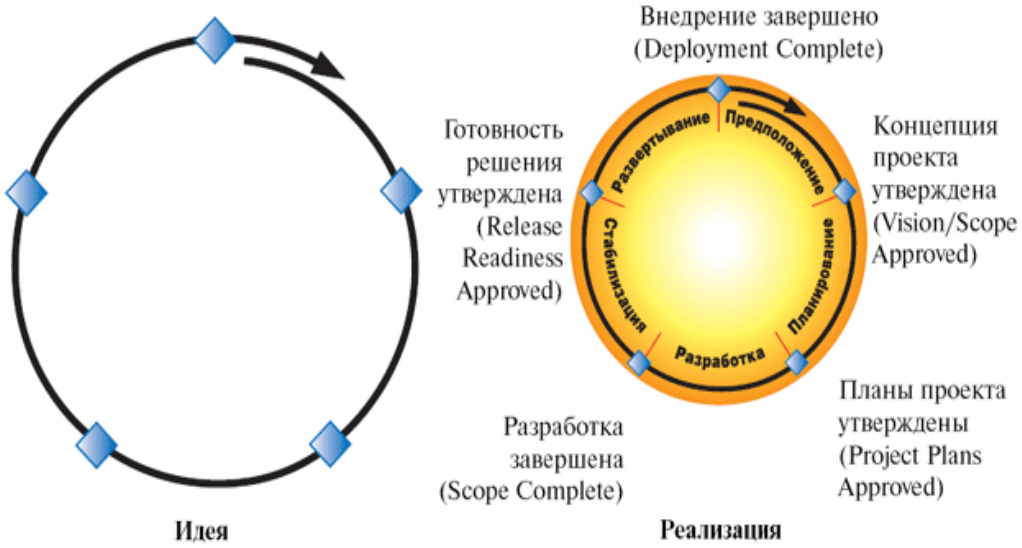

Рис. 8. Графическое представление фаз технологии MSF, этапы и финальные вехи

Заметим, что использование ролевых кластеров не подразумевает и не навязывает никакой специальной структуры организации или обязательных должностей. Административный состав ролей может широко варьироваться в разных организациях и проектных группах. Чаще всего роли распределяются

среди различных подразделений одной организации, но иногда часть их отводится сообществу потребителей или внешним по отношению к организации консультантам и партнерам. Ключевым моментом является четкое определение работников, ответственных за каждый ролевой кластер, их функций, ответственности и ожидаемого вклада в конечный результат.

Подводя итоги, проанализируем какие МЖЦ легли в основу данной технологии проектирования: декларативно данная технология является сочетанием каскадной и спиральной модели (лучшее из двух миров), при этом из спиральной модели взят контроль рисков и выход на новые версии, а из каскадной все остальное. Графическое представление технологии MSF показано на Рис. 8.

4.4. Extreme Programming XP

Данная технология входит в группу Agile (подвижный, сообразительный) методологий.[1] Еще одним примером этой группы технология является технология RAD (Rapid Application Development), которая в данной работе не рассматривается. Эти две технологии имеют очень много общего, причем настолько, что во многих книгах они сливаются в одну.

Основными или общими характеристиками данной группы технологий является:

- получение быстрых результатов в малом проекте с ограниченными ресурсами;
- малая рабочая группа (до 50 человек);
- короткий цикл разработки (до 6 месяцев).

При этом данный подход предполагает, что первые несколько версий наверняка не будут полностью работоспособны, поскольку для реализации выбирается несколько задач, возможно не связанных друг с другом. Гораздо важнее показать, что процесс движется в правильном направлении, с минимумом затрат по ресурсам и времени. Тем не менее, создание каждой версии предполагает последовательное прохождение четырех этапов, графическое представление которых показано на Рис. 9.

Планирование

- Осуществляется на основе
  - бизнес-приоритетов заказчика
  - технических возможностей
- Сбор/ Отбор User Story
- Определяется время окончания версии

Дизайн

- Вброс архитектуры
- Выбор технологий
- Создание metaphor системы

Кодирование

- Только отобранные User Stories

- Осуществляется в парах (2 программиста на одном компьютере)
- Максимальная простота кода

Тестирование

- Тесты на основании User Story (создаются на этапе планирования)
- Вместе с заказчиком

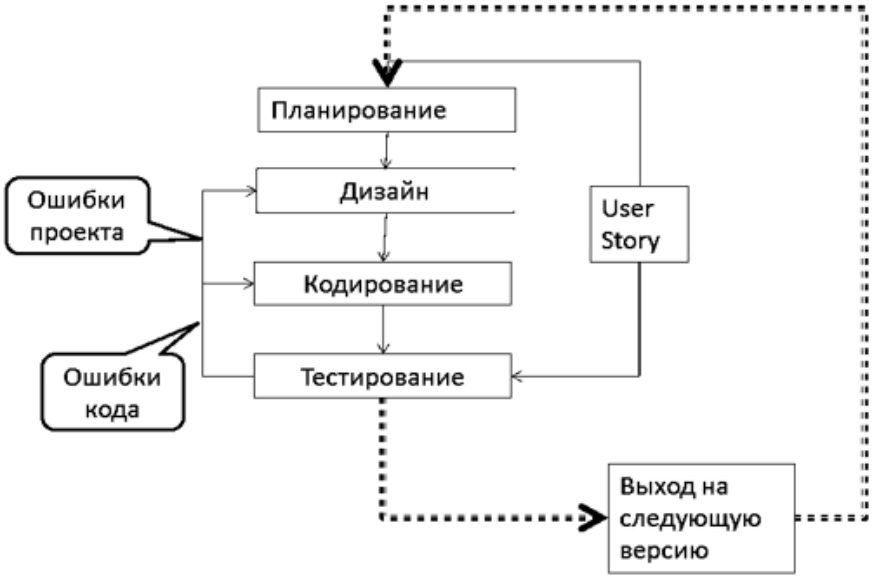

Рис. 9. Графическое представление фаз технологии XP

Важным и непременным требованием при реализации технологии XP является постоянное тесное взаимодействие с конечным пользователем на каждом этапе, особенно на этапах планирования и тестирования. Если присутствие конечного пользователя в процессе невозможно, то его роль исполняет один из членов команды. При этом важно отметить, что взаимодействие осуществляется не с заказчиком, а именно с пользователем, то есть с тем, кто непосредственно будет использовать ПО и максимально заинтересован в удобстве и эффективности его работы.

При описании данной технологии ее автор К.Бек охарактеризовал ее как разработка через тестирование. То есть при планировании цикла сначала планируются тесты, для этого формируется набор User Story, предполагаемых для реализации. С пользователем обсуждается какие User Story будут включены в итерацию и какие результаты тестирования должны быть получены после реализации каждой из выбранных User Story.

Еще одной особенностью данной технологии является "парное программирование", этот подход позволяет одновременно писать и делать ревизию кода, что позволяет выявить большинство ошибок не на стадии тестирования, а уже в процессе написания кода.

Кроме того, данная технология накладывает определенные ограничения на квалификацию команды разработчиков. Группа должна состоять из профессионалов, имеющих большой опыт в анализе, проектировании, написании и тестировании ПО.

Предполагается, что код пишется в рамках ООП, поскольку данный подход позволяет решить несколько задач: локализация ошибок, повышение надежности, повторное использование кода и сокращение рутинной работы.

Данная технология является ярким представителем воплощения инкрементальной модели, поскольку каждый из инкрементов имеет в своей основе определенный набор User Story, причем, в рамках данной технологии не обязательно использование MoSCoW list для ранжирования (выбора) User Story для каждого из инкрементов. Конкретные User Story для реализации выбираются соглашением между заказчиком и командой.

Список литературы

1. Бек К. Экстремальное программирование: разработка через тестирование — Питер, 2003 – 224 с.: ил.
2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2006. – 544 с: ил.
3. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств.– М.: Стандартинформ, 2012 год
4. Избачков Ю. С, Петров В. Н. Информационные системы: Учебник для вузов. 2-е изд. – СПб.: Питер, 2006. – 656 с: ил.
5. Карпенко С.Н. Введение в программную инженерию. Учебно-методические материалы по программе повышения квалификации «Информационные технологии и компьютерное моделирование в прикладной математике». Нижний Новгород, 2007, 103 с.
6. Липаев В.В. Проектирование и производство сложных заказных программных продук- тов. – М.: СИНТЕГ, 2011. – 408 с.
7. Орлов С.А. Технологии разработки программного обеспечения: Учебник – СПб.: Питер, 2002. – 464 с.: ил.
8. Электронный фонд правовой и нормативно-технической документации . [электронный ресурс] http://docs.cntd.ru/document/gost-r-iso-mek-12207-2010
9. Barry W. Boehm, A Spiral Model of Software Development and Enhancement, Computer, May 1988, IEEE, pp.61-72
10. CDM - метод разработки информационных систем фирмы Oracle//Oracle Magazine / Russian Edition 2, 1997.
11. Microsoft Solutions Framework White Paper, 2002. [электронный ресурс] http://www.uml.org.cn/SoftWareProcess/MSFProcessModelv.3.1.pdf
12. Oracle CDM Method Handbook. Oracle corp. 1996.
13. Rational Unified Process: Best Practices for Software development Teams. Rational Software White Paper TP026B, Rev 11/01. [электронный ресурс] https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Елена Александровна **Кумагина**
Елена Александровна **Неймарк**

МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА И ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Учебно-методическое пособие

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородскийгосударственный университет им. Н.И. Лобачевского»
603950, Нижний Новгород, пр. Гагарина, 23.