Лекция 2. Жизненный цикл программного продукта

2.1. Немного истории

Появление понятия жизненного цикла ПО было связано с кризисом программирования, который наметился в конце 60-х – начале 70-х годов прошлого века. Суть кризиса состояла в том, что программные проекты все чаще стали выходить из-под контроля: нарушались сроки, превышались запланированные объемы финансирования, результаты не соответствовали требуемым. Многие проекты вообще не доводились до завершения. Кроме того, оказалось, что недостаточно разработать программу, а надо ее еще сопровождать и этап сопровождения часто требует больше средств, чем разработка.

Ситуация была вызвана ростом сложности проектов. Масштабы ее нарастали. Необходимо было принимать меры для радикального усовершенствования принципов и методов разработки ПО с учетом его развития и сопровождения. Заговорили о том, что надо обратиться к опыту промышленного проектирования и производства, где был накоплен опыт успешной разработки не менее сложных проектов.

Методологическую основу промышленной инженерии составляет понятие жизненного цикла изделия (продукта) как совокупности всех действий, которые надо выполнить на протяжении всей «жизни» изделия. Смысл жизненного цикла состоит во взаимосвязанности всех этих действий. Например, вы решили построить садовый домик, имея некоторый опыт строительных работ. Ориентируясь на здравый смысл, вы купили материалы, инструменты и построили дом. Через год он начал оседать одним углом – вы забыли его спроектировать и рассчитать фундамент. Через два года жена сказала, что лестница на мансарду должна быть не на веранде, где и так мало места, а с улицы. Лестница у вас была сработана фундаментально, а с улицы надо вырубать и перекрывать козырьки – вы не предусмотрели модификацию конструкции. Через три года у вас сгорела проводка, которая была «надежно» спрятана и вам пришлось снимать всю облицовку, которую вы «надежно» прибивали 100 мм гвоздями – вы не предусмотрели ремонтопригодность. Если бы вы изначально рассматривали строительство с позиции жизненного цикла, то на этапах проектирования и строительства вы подумали бы о надежности, модифицируемости и ремонтопригодности.

Итак, жизненный цикл промышленного изделия:

- последовательность этапов (фаз, стадий): проектирования, изготовления образца, организация производства, серийное производство, эксплуатация, ремонт, вывод из эксплуатации;
- состоящих из технологических процессов, действий и операций.

Организация промышленного производства с позиции жизненного цикла позволяет рассматривать все его этапы во взаимосвязи, что ведет к сокращению сроков, стоимости и трудозатрат.

Впервые о жизненном цикле ПО заговорили в 1968 г. в Лондоне, где состоялась встреча 22-х руководителей проектов по разработке ПО. На встрече анализировались проблемы и перспективы проектирования, разработки, распространения и поддержки программ. Применяющиеся принципы и методы разработки ПО требовали постоянного усовершенствования. Именно на этой встрече была предложена концепция жизненного цикла ПО (SLC – Software Lifetime Cycle) как последовательности шагов-стадий, которые необходимо выполнить в процессе создания и эксплуатации ПО.

Вокруг этой концепции было много споров. В 1970 г. У.У. Ройс (W.W. Royce) произвел идентификацию нескольких стадий в типичном цикле и было высказано предположение, что контроль выполнения стадий приведет к повышению качества ПО и сокращению стоимости разработки.

2.1.1. История. Стандарты и проблемы ЖЦ ПО

Разрабатывались стандарты ЖЦ ПО. Наиболее известными являются:

1985 (уточнен в 1988 г.) DOD-STD-2167 A – Разработка программных средств для систем военного назначения. Первый формализованный и утвержденный стандарт жизненного цикла для проектирования ПС систем военного назначения по заказам Министерства обороны США. Этим документом регламентированы 8 фаз (этапов) при создании сложных критических ПС и около 250 типовых обязательных требований к процессам и объектам проектирования на этих этапах.

1994г. MIL-STD-498. Разработка и документирование программного обеспечения. Принят Министерством обороны США для замены DOD-STD-2167 A и ряда других стандартов. Он предназначен для применения всеми организациями и предприятиями, получающими заказы Министерства обороны США. В 1996 г. утверждено очень подробное (407 стр.) руководство “Применение и рекомендации к стандарту MIL-STD-498”. Основную часть составляют 75 подразделов — рекомендаций по обеспечению и реализации процессов ЖЦ сложных критических ПС высокого качества и надежности, функционирующих в реальном времени.

1995г. IEEE 1074. Процессы жизненного цикла для развития программного обеспечения. Охватывает полный жизненный цикл ПС, в котором выделяются шесть крупных базовых процессов. Эти процессы детализируются 16 частными процессами. В последних имеется еще более мелкая детализация в совокупности на 65 процессов-работ. Содержание каждого частного процесса начинается с описания общих его функций и задач и перечня действий — работ при последующей детализации. Для каждого процесса в стандарте представлена входная и результирующая информация о его выполнении и краткое описание сущности процесса. В стандарте внимание сосредоточено преимущественно на непосредственном создании ПС и на процессах предварительного проектирования. В приложении представлены четыре варианта адаптации максимального состава компонентов ЖЦ ПС к конкретным особенностям типовых проектов.

Между тем, разработка стандартов ЖЦ и их практическое применение сталкивались с рядом проблем:

- Внедрение стандартов требовало вложения значительных средств, что не всегда окупалось.
- Было неясно, все ли требуемые процессы надо выполнять и в какой мере
- Различные типы ПО (ИС, реального времени, бизнес системы), различные требования
- Высокая динамика отрасли и устаревание стандартов
- Терминологическая неоднозначность различных корпоративных стандартов
- Во многих случаях применение стандартов было вызвано только требованиями заказчиков, хотя на практике превращалось в тормоз и гробило выполнение проектов.

Подробнее:

В. Липаев. Стандарты, регламентирующие жизненный цикл сложных программных комплексов. http://www.pcweek.ru/year1998/N24/CP1251/Reviews/chapt1.htm

Итеративная и инкрементальная разработка: краткая история. http://www.sibinfo.ru/news/03_10_14/iid_history.shtml

2.2. ISO 12207 (15504) Жизненный цикл ПП: структура и организация

2.2.1. Стандарт ISO/IEC 12207

Разрешением проблем стандартизации ЖЦ ПО явилась разработка и принятие в 1995 г. стандарта ISO/IEC 12207 - Information Technology - Software Life Cycle Processes (ISO - International Organization of Standardization - Международная организация по стандартизации; IEC - International Electrotechnical Commission - Международная электротехническая комиссия). В 2000 г. он был принят как ГОСТ 12207. Процессы жизненного цикла программных средств.

Стандарт ISO 12207 разрабатывался с учетом лучшего мирового опыта на основе вышеперечисленных стандартов. Основными результатами стандарта ISO 12207 являются:

- Введение единой терминологии по разработке и применению ПО (предназначен не только для разработчиков, но и для заказчиков, пользователей, всех заинтересованных лиц).
- Разделение понятий ЖЦ ПО и модели ЖЦ ПО. ЖЦ ПО в стандарте вводится как полная совокупность всех процессов и действий по созданию и применению ПО, а модель ЖЦ – конкретный вариант организации ЖЦ, обоснованно (разумно) выбранный для каждого конкретного случая
- Описание организации ЖЦ и его структуры (процессов)
- Выделение процесса адаптации стандарта для построения конкретных моделей ЖЦ

2.2.2. ISO 12207. Основные определения

В стандарте ISO 12207 даются следующие определения:

Программный продукт (software product): Набор машинных программ, процедур и, возможно, связанных с ними документации и данных.

Жизненный цикл программного продукта (software life cycle) – это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации

Процесс (process): Набор взаимосвязанных работ, которые преобразуют исходные данные в выходные результаты.

Стандарт определяет организацию ЖЦ программного продукта как совокупность процессов, каждый из которых разбит на действия, состоящие из отдельных задач.

Устанавливает структуру (архитектуру) ЖЦ программного продукта в виде перечня процессов, действий и задач.

2.2.3. ISO 12207. Структура ЖЦ ПО

В соответствии со стандартом ISO 12207 процессы ЖЦ делятся на три группы:

- Основные
- Вспомогательные
- Организационные

Отдельно описан процесс адаптации стандарта, содержащий основные работы, которые должны быть выполнены при адаптации настоящего стандарта к условиям конкретного программного проекта

К числу основных относятся процессы:

- Заказа. Определяет работы заказчика, то есть организации, которая приобретает систему, программный продукт или программную услугу.

- Поставки. Определяет работы поставщика, **то** есть организации, которая поставляет систему, программный продукт или программную услугу заказчику.
- Разработки. Определяет работы разработчика, то есть организации, которая проектирует и разрабатывает программный продукт.
- Эксплуатации. Определяет работы оператора, то есть организации, которая обеспечивает эксплуатационное обслуживание вычислительной системы в заданных условиях в интересах пользователей.
- Сопровождения. Определяет работы персонала сопровождения, то есть организации, которая предоставляет услуги по сопровождению программного продукта, состоящие в контролируемом изменении программного продукта с целью сохранения его исходного состояния и функциональных возможностей. Данный процесс охватывает перенос и снятие с эксплуатации программного продукта.

Вспомогательными процессами являются:

- Документирования. Определяет работы по описанию информации, выдаваемой в процессе жизненного цикла.
- Управления конфигурацией. Определяет работы по управлению конфигурацией.
- Обеспечения качества. Определяет работы по объективному обеспечению того, чтобы программные продукты и процессы соответствовали требованиям, установленным для них, и реализовывались в рамках утвержденных планов. Совместные анализы, аудиторские проверки, верификация и аттестация могут использоваться в качестве методов обеспечения качества.
    - Верификации. Определяет работы (заказчика, поставщика или независимой стороны) по верификации программных продуктов по мере реализации программного проекта.
    - Аттестации. Определяет работы (заказчика, поставщика или независимой стороны) по аттестации программных продуктов программного проекта.
    - Совместного анализа. Определяет работы по оценке состояния и результатов какой-либо работы. Данный процесс может использоваться двумя любыми сторонами, когда одна из сторон (проверяющая) проверяет другую сторону (проверяемую) на совместном совещании.
    - Аудита. Определяет работы по определению соответствия требованиям, планам и договору. Данный процесс может использоваться двумя сторонами, когда одна из сторон (проверяющая) контролирует программные продукты или работы другой стороны (проверяемой).
- Решения проблем. Определяет процесс анализа и устранения проблем (включая несоответствия), независимо от их характера и источника, которые были обнаружены во время осуществления разработки, эксплуатации, сопровождения или других процессов.

Организационные процессы жизненного цикла:

- Управления. Определяет основные работы по управлению, включая управление проектом, при реализации процессов жизненного цикла.
- Создания инфраструктуры. Определяет основные работы по созданию основной структуры процесса жизненного цикла.
- Усовершенствования. Определяет основные работы, которые организация (заказчика, поставщика, разработчика, оператора, персонала сопровождения или администратора другого процесса) выполняет при создании, оценке, контроле и усовершенствовании выбранных процессов жизненного цикла.
- Обучения. Определяет работы по соответствующему обучению персонала.

2.2.4. ISO 15504. Процессы ЖЦ ПО

Стандарт ISO 12207 разрабатывался 9 лет и достаточно быстро устарел. В 1998г. выходит новый стандарт ISO/IEC TR 15504: Information Technology - Software Process Assessment (Оценка процессов разработки ПО). В этом документе рассматриваются вопросы аттестации, определения зрелости и усовершенствования процессов жизненного цикла ПО. Один из разделов документа содержит новую классификацию процессов жизненного цикла, являющуюся развитием стандарта ISO 12207.

Связь со стандартом ISO 12207 состоит в том, что все процессы стандарта ISO 15504 принадлежат к одной из следующих типов:

- базовый — процесс из 12207;
- расширенный — расширение процесса из 12207;
- новый — процесс, не описанный в 12207;
- составляющий — часть процесса из 12207;
- расширенный составляющий — расширенная часть проц. из 12207

2.2.5. ISO 15504. Классификация процессов

В соответствии с новой классификацией в трех группах процессов вводятся пять категорий процессов:

- Основные процессы:
  - CUS: Потребитель-поставщик
  - ENG: Инженерная
- Вспомогательные процессы:
  - SUP: Вспомогательная
- Организационные процессы:
  - MAN Управленческая
  - ORG: Организационная

2.2.5.1. ISO 15504. CUS: Потребитель-поставщик

Категория Потребитель-Поставщик состоит из процессов, непосредственно влияющих на потребителя, поддерживающих процесс разработки программного средства и его передачи потребителю и обеспечивающих возможность корректного использования программного средства или услуги.

Включает следующие процессы:

- CUS.1 Процесс приобретения (Acquisition process)
  - CUS.1.1 Процесс подготовки приобретения (Acquisition preparation process)
  - CUS.1.2 Процесс выбора поставщика (Supplier selection process)
  - CUS.1.3 Процесс мониторинга поставщика (Supplier Monitoring process)
  - CUS.1.4 Процесс приемки (Customer Acceptance process)
- CUS.2 Поставки (Supply process)
- CUS.3 Процесс выявления требований (Requirements process)
- CUS.4 Эксплуатации (Operation process)
  - CUS.4.1 Процесс эксплуатационного использования (Operational use process)
  - CUS.4.2 Процесс поддержки потребителя(Customer support process)

2.2.5.2. ISO 15504. ENG: Инженерные процессы

Инженерная категория процессов состоит из процессов, которые непосредственно определяют, реализуют или поддерживают программный продукт, его взаимодействие с системой и документацию на него. В тех случаях, когда система целиком состоит из программных средств, инженерные процессы имеют отношение только к созданию и поддержанию этих программных средств.

Включает следующие процессы:

- ENG.1 Процесс разработки (Development process)
  - ENG.1.1 Процесс анализа требований и разработки системы (Systemrequirements analysis and design process)
  - ENG.1.2 Процесс анализа требований к программным средствам (Software requirements analysis process)
  - ENG.1.3 Процесс проектирования программных средств (Software design process)
  - ENG.1.4 Процесс конструирования программных средств (Software construction process)
  - ENG.1.5 Процесс интеграции программных средств (Software integration process)
  - ENG.1.6 Процесс тестирования программных средств (Software testing process)
  - ENG.1.7 Процесс интеграции и тестирования системы (System integration andtesting process)
- ENG.2 Процесс сопровождения системы и программных средств(System and software maintenance process)

2.2.5.3. ISO 15504. SUP: Вспомогательные процессы

Вспомогательная категория состоит из процессов, которыми могут пользоваться любые другие процессы (включая другие вспомогательные процессы) в различные моменты жизненного цикла программных средств.

Включает следующие процессы:

- SUP.1 Процесс документирования (Documentation process)
- SUP.2 Процесс управления конфигурацией (Configuration management process)
- SUP.3 Процесс обеспечения качества (Quality assurance process)
- SUP.4 Процесс верификации (Verification process)
- SUP.5 Процесс проверки соответствия (Validation process)
- SUP.6 Процесс совместных проверок (Joint review process)
- SUP.7 Процесс аудита (Audit process)
- SUP.8 Процесс разрешения проблем (Problem resolution process)

2.2.5.4. ISO 15504. MAN: Управленческие процессы

Управленческая категория состоит из процессов, содержащих практики общего характера, которые могут быть использованы каждым, кто управляет любым проектом или процессом в ходе жизненного цикла программных средств.

К управленческой категории относятся следующие процессы:

- MAN.1 Процесс административного управления (Management process)
- MAN.2 Процесс управления проектами (Project management process)
- MAN.3 Процесс управления качеством (Quality Management process)
- MAN.4 Процесс управления рисками (Risk Management process)

2.2.5.5. ISO 15504. ORG: Организационные процессы

Организационная категория процессов состоит из процессов, которые устанавливают цели функционирования организации и создают активы процессов, продуктов и ресурсов, которые, будучи использованы в проектах организации, способствуют выполнению ее целей. Хотя организационные практики в целом относятся не только к процессам, относящимся к программным средствам, последние выполняются в общем контексте организации, и для их эффективного использования необходимо соответствующее окружение. Кратко, эти организационные процессы:

- создают инфраструктуру организации;
- используют все лучшее из того, что имеется (передовой опыт) во всех частях организации (эффективные процессы, лучшие навыки, качественный программный код, хорошие средства поддержки);
- делают это общедоступным в рамках всей организации;
- создают базу для постоянного совершенствования во всей организации.

Организационной категории принадлежат процессы:

- ORG.1 Процесс организационных установок (Organizational alignment process)
- ORG.2 Процесс усовершенствования (Improvement process)
  - ORG.2.1 Процесс создания процессов (Process establishment process)
  - ORG.2.2 Процесс аттестации процессов (Process assessment process)
  - ORG.2.3 Процесс усовершенствования процессов (Process improvement process)
- ORG.3 Процесс административного управления кадрами (Human resource management process)
- ORG.4 Процесс создания инфраструктуры (Infrastructure process)
- ORG.5 Процесс измерения (Measurement process)
- ORG.6 Процесс повторного использования (Reuse process)

2.3. Модель жизненного цикла программного продукта

Модель жизненного цикла ПО описывается набор фаз (этапов, стадий) проекта по созданию ПО, в которых выполняются отдельные процессы, разбитые на операции и задачи. В глоссарии PMI (PMI. Глоссарий http://www.pmi.ru/glossary/) даются следующие определения этих понятий:

Жизненный цикл проекта Набор обычно последовательных фаз проекта, количество и состав которых определяется потребностями управления проектом организацией или организациями, участвующими в проекте.

Фаза проекта Объединение логически связанных операций проекта, обычно завершающихся достижением одного из основных результатов.

Процесс Набор взаимосвязанных ресурсов и работ, благодаря которым входные воздействия преобразуются в выходные результаты.

Операция, работа Элемент работ проекта. У операций обычно имеется ожидаемая длительность, потребность в ресурсах, стоимость. Операции могут далее подразделяться на задачи.

В этих определениях существенным является следующее:

- Состав, количество и, можно добавить, порядок выполнения фаз определяется особенностью проекта.
- Каждая фаза завершается получением одного из основных результатов, в то время как процесс или задача – просто значимого результата

Разберемся немного подробнее.

2.3.1. Схема модели ЖЦ ПО

Для схемы модели жизненного цикла ПО характерно следующее:

Результатом выполнения каждой фазы является некоторая модель ПО. Описание требований – модель того, что должен делать программный продукт; результат анализа – модель основных архитектурных решений и GUI, …

Результат выполнения каждой фазы является входом следующей фазы и фазы должны выполняться в определенной моделью ЖЦ последовательности.

Некоторые процессы могут выполняться на нескольких фазах, некоторые – в пределах одной.

В стандарте ISO 12207 модель жизненного цикла (life cycle model) определяется как структура, состоящая из процессов, работ и задач, включающих в себя разработку, эксплуатацию и сопровождение программного продукта, охватывающая жизнь системы от установления требований к ней до прекращения ее использования. При этом, конкретные модели определяются особенностью задач, ограничениями на ресурсы, опытом разработчиков и т.д. Между тем, известны некоторые типовые модели ЖЦ ПО, которые проявили себя в определенных условиях, имеют определенные преимущества, недостатки и условия применимости. Эти типовые модели устанавливают некоторые принципы организации модели жизненного цикла ПО.

2.3.2. Каскадная модель. Принципы

Каскадная модель (водопад - waterfall) включает выполнение следующих фаз:

- **исследование концепции** — происходит исследование требований, разрабатывается видение продукта и оценивается возможность его реализации.
- **определение требований** — определяются программные требования для информационной предметной области системы, предназначение, линии поведения, производительность и интерфейсы.
- **разработка проекта** — разрабатывается и формулируется логически по-следовательная техническая характеристика программной системы, включая структуры данных, архитектуру ПО, интерфейсные представления и процессуаль-ную (алгоритмическую) детализацию;
- **реализация** — эскизное описание ПО пре-вращается в полноценный программный продукт. Результат: исходный код, база данных и документация. В реализации обычно выделяют два этапа: реализацию компонент ПО и интеграцию компонент в готовый продукт. На обоих этапах выполняется кодирование и тестирование, которые тоже иногда рассматривают как два подэтапа.
- **эксплуатация и поддержка** - подразумевает запуск и текущее обеспечение, включая предоставление технической помощи, обсуждение возникших вопросов с пользователем, регистрацию запросов пользователя на модернизацию и внесение изменений, а также корректирование или устранение ошибок;
- **сопровождение —** устранение программных ошибок, неис-правностей, сбоев, модернизация и внесение изменений. Состоит из итераций разработки.

Основными принципами каскадной модели являются:

- Строго последовательное выполнение фаз:
- Каждая последующая фаза начинается лишь тогда, когда полностью завершено выполнение предыдущей фазы
- Каждая фаза имеет определенные критерии входа и выхода: входные и выходные данные.
- Каждая фаза полностью документируется

- Переход от одной фазы к другой осуществляется посредством формального обзора с участием заказчика
- Основа модели – сформулированные требования (ТЗ), которые меняться не должны
- Критерий качества результата – соответствие продукта установленным требованиям.

2.3.3. Каскадная модель. Преимущества и недостатки

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

При использовании каскадной модели для «неподходящего» проекта могут проявляться следующие ее основные недостатки:

- Попытка вернуться на одну или две фазы назад, чтобы исправить какую-либо проблему или недостаток, приведет к значительному увеличению затрат и сбою в графике;
- Интеграция компонент, на которой обычно выявляется большая часть ошибок, выполняется в конце разработки, что сильно увеличивает стоимость устранения ошибок;
- Запаздывание с получением результатов – если в процессе выполнения проекта требования изменились, то получится устаревший результат

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.

2.3.4. Каскадная модель. Применимость

Каскадная модель впервые четко сформулирована в 1970 году Ройсом (W.W. Royce. Managing the Development of Large Software Systems. http://facweb.cti.depaul.edu/jhuang/is553/Royce.pdf).

На начальном периоде она сыграла ведущую роль как метод регулярной разработки сложного ПО. В семидесятых-восьмидесятых годах XX века модель была принята как стандарт министерства обороны США. Со временем недостатки каскадной модели стали проявляться все чаще и возникло мнение, что она безнадежно устарела. 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 (автомати-зированного бухгалтерского учета, начисления зарплаты, …)
- Выпуск новой версии уже существующего продукта, если вносимые изменения вполне определены и управляемы (перенос уже существующего продукта на новую платформу)
- И наконец, принципы каскадной модели находят применение как элементы моделей других типов, о чем речь пойдет ниже.

Подробнее о каскадной модели - см. [1, стр. 135-141]

2.3.5. Спиральная модель. Принципы

На практике, при решении достаточно большого количества задач, разработка ПО имеет циклический характер, когда после выполнения некоторых стадий приходится возвращаться на предыдущие. Можно указать две основные причины таких возвратов:

- Ошибки разработчиков, допущенные на ранних стадиях и выявленные на поздних стадиях – ошибки анализа, проектирования, кодирования, выявляемые, как правило, на стадии тестирования.
- Изменение требований в процессе разработки («ошибки» заказчиков). Это или неготовность заказчиков сформулировать требования («Сказать, что должна делать программа я смогу только после того, как увижу как она работает»), или изменения требований, вызванные изменениями ситуации в процессе разработки (изменения рынка, новые технологии, …).

Циклический характер разработки ПО отражен в спиральной модели ЖЦ, описанной Б. Боэмом в 1988 году (Barry Boehm, "A Spiral Model of Software Development and Enhancement", IEEE Computer, Vol.21, No.5, pp. 61-72, 1988. www.computer.org/computer/homepage/misc/Boehm/r5061.pdf). Спиральная модель была предложена как альтернатива каскадной модели, учитывающая повторяющийся характер разработки ПО. Основные принципы спиральной модели можно сформулировать следующим образом:

- Разработка вариантов продукта, соответствующих различным вариантам требований с возможностью вернуться к более ранним вариантам
- Создание прототипов ПО как средства общения с заказчиком для уточнения и выявления требований
- Планирование следующих вариантов с оценкой альтернатив и анализом рисков, связанных с переходом к следующему варианту
- Переход к разработке следующего варианта до завершения предыдущего в случае, когда риск завершения очередного варианта (прототипа) становится неоправданно высок.
- Использование каскадной модели как схемы разработки очередного варианта
- Активное привлечение заказчика к работе над проектом. Заказчик участвует в оценке очередного прототипа ПО, уточнении требований при переходе к следующему, оценке предложенных альтернатив очередного варианта и оценке рисков.

2.3.6. Спиральная модель. Схема

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 начинается с планирования требований и деталей ЖЦ продукта для оценки затрат. На фазе определения целей устанавливаются альтернативные варианты требований, связанные с аранжировкой требований по важности и стоимости их выполнения. На фазе оценки устанавливаются риски вариантов требований. На фазе разработки – спецификация требований (с указанием рисков и стоимости), готовится демо-версия ПО для анализа требований заказчиком.

Следующий цикл – разработка проекта – начинается с планирования разработки. На фазе определения целей устанавливаются ограничения проекта (по срокам, объему финансирования, ресурсам, …), 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 вариантном (прототипом) продукта.

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.

2.3.7. Спиральная модель. Преимущества и недостатки

Спиральная модель (по отношению к каскадной) имеет следующие очевидные преимущества:

- Более тщательное проектирование (несколько начальных итераций) с оценкой результатов проектирования, что позволяет выявить ошибки проектирования на более ранних стадиях.
- Поэтапное уточнение требований в процессе выполнения итераций, что позволяет более точно удовлетворить требованиям заказчика
- Участие заказчика в выполнении проекта с использованием прототипов программы. Заказчик видит, что и как создается, не выдвигает необоснованных требований, оценивает реальные объемы финансирования.
- Планирование и управление рисками при переходе на следующие итерации позволяет разумно планировать использование ресурсов и обосновывать финансирование работ.
- Возможность разработки сложного проекта «по частям», выделяя на первых этапах наиболее значимые требования.

Основные недостатки спиральной модели связаны с ее сложностью:

- Сложность 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
- Бесконечность модели – на каждом витке заказчик может выдвигать новые требования, которые приводят к необходимости следующего цикла разработки.

2.3.8. Спиральная модель. 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.
-

2.3.9. Другие типы моделей ЖЦ ПО

Каскадная и спиральная модели устанавливают некоторые принципы организации жизненного цикла создания программного продукта. Каждая из них имеет преимущества, недостатки и области применимости. Каскадная модель проста, но применима в случае, когда требования известны и меняться не будут. Спиральная модель учитывает такие важные показатели проекта как изменяемость требований, невозможность оценить заранее объем финансирования, риски выполнения проекта. Но спиральная модель сложна и требует больших затрат на сопровождение.

Существуют некоторые другие типы моделей, которое можно рассматривать как «промежуточные» между каскадной и спиральной моделями. Эти модели используют отдельные преимущества каскадной и спиральной моделей и достигают успеха для определенных типов задач.

2.3.9.1. Итерационная модель

Итерационная модель жизненного цикла является развитием классической каскадной модели и предполагает возможность возвратов на ранее выполненные этапы. При этом, причинами возвратов в классической итерационной модели являются выявленные ошибки, устранение которых и требует возврата на предыдущие этапы в зависимости от типа (природы) ошибки – ошибки кодирования, проектирования, спецификации или определения требований. Реально итерационная модель является более жизненной, чем классическая (строгая) каскадная модель, т.к. создание ПО всегда связано с устранением ошибок. Следует отметить, что уже в первой статье, посвященной каскадной модели, Боэм отмечал это обстоятельство и описал итерационный вариант каскадной модели. практически все применяемые модели жизненного цикла имеют итерационный характер, но цели итераций могут быть разными.

2.3.9.2. V-образная модель

V-образная модель была создана как итерационная разновидность каскадной модели. Целями итераций в этой модели является обеспечение процесса тестирования. Тестирование продукта обсуждается, проектируется и планируется на ранних этапах жизненного цикла разработки. План испытания приемки заказчиком разрабатывается на этапе планирования, а компоновочного испытания системы - на фазах анализа, разработки проекта и т.д. Этот процесс разработки планов испытания обозначен пунктирной линией между прямоугольниками V-образной модели. Помимо планов, на ранних этапах разрабатываются также и тесты, которые будут выполняться при завершении параллельных этапов.

2.3.9.3. Инкрементная (пошаговая) модель

Инкрементная разработка представляет собой процесс поэтапной реализации всей системы и поэтапного наращивания (приращения) функциональных возможностей. На первом шаге необходим полный заранее сформулированный набор требований, которые делятся по некоторому признаку на части. Далее выбирается первая группа требований и выполняется полный проход по каскадной модели. После того, как первый вариант системы, выполняющий первую группу требований сдан заказчику, разработчики переходят к следующему шагу (второму инкременту) по разработке варианта, выполняющего вторую группу требований.

Особенностью инкрементной модели является разработка приемочных тестов на этапе анализа требований, что упрощает приемку варианта заказчиком и устанавливает четкие цели разработки очередного варианта системы.

Инкрементная модель особенно эффективна в случае, когда задача разбивается на несколько относительно независимых подзадач (разработка подсистем «Зарплата», «Бухгалтерия», «Склад», «Поставщики»). Кроме того, инкрементная модель может для внутренней итерации может использовать не только каскадную, но и другие типы моделей.

2.3.9.4. Модель быстрого прототипирования

Модель быстрого протитипирования предназначена для быстрого создания прототипов продукта с целью уточнения требований и поэтапного развития прототипов в конечный продукт. Скорость (высокая производительность) выполнения проекта обеспечивается планированием разработки прототипов и участием заказчика в процессе разработки.

Начало жизненного цикла разработки помещено в центре эллипса. Совместно с пользователем разрабатывается предварительный план проекта на основе предварительных требований. Результат начального планирования - документ, описывающий в общих чертах примерные графики и результативные данные.

Следующий уровень – создание исходного прототипа на основе быстрого анализа, проекта база данных, пользовательского интерфейса и некоторых функций. Затем начинается итерационный цикл быстрого прототипирования. Разработчик проекта демонстрирует очередной прототип, пользователь оценивает его функционирование, совместно определяются проблемы и пути их преодоления для перехода к следующему прототипу. Этот процесс продолжается до тех пор, пока пользователь не согласится, что очередной прототип в точности отображает все требования.

Получив одобрение пользователя, быстрый прототип преобразуют детальный проект, и систему настраивают на производственное использование. Именно на этом этапе настройки ускоренный прототип становится полностью действующей системой.

При разработке производственной версии программы, может понадобиться более высокий уровень функциональных возможностей, различные системные ресурсы, необходимых для обеспечения полной рабочей нагрузки, или ограничения во времени. После этого следуют тестирование в предельных режимах, определение измерительных критериев и настройка, а затем, как обычно, функциональное сопровождение.

2.4. Модели жизненного цикла MSF, RUP, XP

В настоящее время широкое применение получают так называемые промышленные технологии создания программного продукта. Эти технологии были разработаны фирмами, накопившими большой опыт создания ПО. Технологии представлены описаниями принципов, методов, применяемых процессов и операций. Такие технологии, как правило, поддерживаются набором CASE-средств, охватывают все этапы жизненного цикла продукта и успешно применяются для решения практических задач.

Хороший обзор моделей жизненного цикла промышленных технологий можно прочитать: Скопин И.Н.Основы менеджмента программных проектов. Лекция №11: Модели жизненного цикла в некоторых реальных методологиях программирования. http://www.intuit.ru/department/se/msd/11/1.html. Здесь мы рассмотрим особенности моделей жизненного цикла трех наиболее известных промышленных технологий:

- Microsoft Solution Framework (MSF) – разработка фирмы Microsoft, предназначенная для решения широкого круга задач. Технология масштабируема, т.е. настраиваема на решение задач любой сложности коллективом любой численности.
- Rational Unified Process (RUP) – разработка фирмы Rational, долгое время успешно занимавшейся созданием CASE-средств, применяемых на различных этапах жизненного цикла продукта от анализа до тестирования и документирования. Аналогично MSF, RUP универсальна, масштабируема и настраиваема на применение в конкретных условиях.
- Extreme Programming (XP) – активно развивающаяся в последнее время технология, предназначенная для решения относительно небольших задач, относительно небольшими коллективами профессиональных разработчиков в условиях жестко ограниченного времени.

Каждая из этих технологий имеет свои особенности организации модели жизненного цикла создания продукта.

2.4.1. Модель Microsoft Solution Framework

Одна из особенностей технологии MSF состоит в том, что она ориентирована не просто на создание программного продукта, удовлетворяющего перечисленным требованиям, а на поиск решения проблем, стоящих перед заказчиком. Различие состоит в том, что перечисляемые заказчиком требования являются проявлениями некоторых более глубоких проблем и неточность, неполнота, изменение требований в процессе разработки – следствие недопонимания проблем. Поэтому, в технологии MSF большое внимание

уделяется анализу проблем заказчика и разработке вариантов системы для поиска решения этих проблем.

Модель жизненного цикла MSF является некоторым гибридом каскадной и спиральной моделей, сочетая простоту управления каскадной модели с гибкостью спиральной. Схема модели жизненного цикла MSF (модели процессов) представлена на слайде.

Модель жизненного цикла MSF ориентирована на “вехи” (milestones) – ключевые точки проекта, характеризующие достижение какого-либо существенного результата. Этот результат может быть оценен и проанализирован, что подразумевает ответ на вопрос: “А достигли ли мы целей, поставленных на этом шаге?». В модели предусматривается наличие основных вех (завершение главных фаз модели) и промежуточных, отражающих внутренние этапы главных фаз.

Основными фазами модели MSF являются:

- Создание общей картины приложения (Envisioning).На этом этапе решаются следующие основные задачи: оценка существующей ситуации; определение состава команды, структуры проекта, бизнес-целей, требований и профилей пользователей; разработка концепции решения и оценка риска. Устанавливаются две промежуточные вехи: "Организован костяк команды" и "Создана общая картина решения".
- Планирование (Panning). Включает планирование и проектирование продукта. На основе анализа требований разрабатывается проект и основные архитектурные решения, функциональные спецификации системы, планы и календарные графики, среды разработки, тестирования и пилотной эксплуатации. Этап состоит из трех стадий: 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 интерфейсы.
- Разработка (Developing). Создается вариант решения проблемы, в виде кода и документации очередного прототипа, включая спецификации и сценарии тестирования. Основная веха этапа - "Окончательное утверждение области действия проекта". 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.
- Стабилизация (Stabilizing). Подготовка к выпуску окончательной версии продукта, доводка его до заданного уровня качества. Здесь выполняется комплекс работ по тестированию (обнаружение и устранение дефектов), проверяется сценарий развертывания продукта. Когда решение становится достаточно устойчивым, проводится его пилотная эксплуатация в тестовой среде с привлечением пользователей и применением реальных сценариев работы.
- Развертывание (Deploying). 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.

Подробнее:

1. Модель процессов MSF, версия 3.1. http://www.microsoft.com/Rus/Download.aspx?file=/Msdn/Msf/MSF_process_model_rus.doc
2. Андрей Колесов. Введение в методологию Microsoft Solutions Framework. http://www.bytemag.ru/Article.asp?ID=2866

2.4.2. Модель Rational Unified Process

Модель жизненного цикла RUP является довольно сложной, детально проработанной итеративно-инкрементной моделью с элементами каскадной модели. В модели RUP выделяются 4 основные фазы, 9 видов деятельности (процессов). Кроме того, в модели описывается ряд практик, которые следует применять или руководствоваться для успешного выполнения проекта. RUP ориентирован на поэтапное моделирование создаваемого продукта с помощью языка UML.

Основными фазами RUP являются:

- **Фаза начала проекта (Inception).** Определяются основные цели проекта, бюджет проекта, основные средства его выполнения — технологии, инструменты, ключевой персонал, составляются предварительные планы проекта. Основная цель этой фазы — достичь компромисса между всеми заинтересованными лицами относительно задач проекта.
- **Фаза проработки (Elaboration).** Основная цель этой фазы — на базе основных, наиболее существенных требований разработать стабильную базовую архитектуру продукта, которая позволяет решать поставленные перед системой задачи и в дальнейшем используется как основа разработки системы.
- **Фаза построения (Construction).** Основная цель этой фазы — детальное прояснение требований и разработка системы, удовлетворяющей им, на основе спроектированной ранее архитектуры.
- **Фаза передачи (Transition).** Цель фазы — сделать систему полностью доступной конечным пользователям. Здесь происходит окончательное развертывание системы в ее рабочей среде, подгонка мелких деталей под нужды пользователей.

В рамках каждой фазы возможно проведение нескольких итераций, количество которых определяется сложностью выполняемого проекта.

Деятельности (основные процессы) RUP делятся на пять рабочих и четыре поддерживающие. К рабочим деятельностям относятся:

- Моделирование предметной области (бизнес-моделирование, Business Modeling). 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 (Requirements). Цели — понять, что должна делать система, определить границы системы и основу для планирования проекта и оценок ресурсозатрат в нем.
- Анализ и проектирование (Analysis and Design). Выработка архитектуры системы на основе ключевых требований, создание проектной модели, представленной в виде диаграмм UML, описывающих продукт с различных точек зрения.
- Реализация (Implementation). Разработка исходного кода, компонент системы, тестирование и интегрирование компонент.
- Тестирование (Test). Общая оценка дефектов продукта, его качество в целом; оценка степени соответствия исходным требованиям.

Поддерживающими деятельностями являются:

- 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.

2.4.3. Модель Extreme Programming

Экстремальное программирование является примером так называемого метода «живой» разработки (Agile Development Method). В группу «живых» входят, помимо экстремального программирования, входит еще ряд методов, о чем подробнее можно прочитать: Мартин Фаулер. Новые методологии программирования. http://www.maxkir.com/sd/newmethRUS.html.

2.4.3.1. Схема модели XP

Модель жизненного цикла XP является итерационно-инкрементной моделью быстрого создания (и модификации) протопопов продукта, удовлетворяющих очередному требованию (user story). Особенности этой модели представлены на слайде. Основными фазами модели можно считать:

- «Вброс» архитектуры – начальный этап проекта, на котором создается видение продукта, принимаются основные решения по архитектуре и применяемым технологиям. Результатом начального этапа является метафора (metaphor) системы, которая в достаточно простом и понятном команде виде должна описывать основной механизм работы системы.
- Истории использования (User Story) – этап сбора требований, записываемых на специальных карточках в виде сценариев выполнения отдельных функций. Истории использования являются требованиями для планирования очередной версии и одновременной разработки приемочных тестов (Acceptance tests) для ее проверки.
- Планирование версии (релиза). Проводится на собрании с участием заказчика путем выбора User Stories, которые войдут в следующую версию. Одновременно принимаются решения, связанные с реализацией версии. Цель планирования - получение оценок того, что и как можно сделать за 1-3 недели создания следующей версии продукта.
- Разработка проводится в соответствии с планом и включает только те функции, которые были отобраны на этапе планирования.
- Тестирование проводится с участием заказчика, который участвует в составлении тестов.
- Выпуск релиза – разработанная версия передается заказчику для использования или бета-тестирования.
  По завершению цикла делается переход на следующую итерацию разработки

2.4.3.2. Extreme Programming. Принципы

Особенности модели жизненного цикла XP проясняют следующие принципы этого метода. Прежде всего, это принципы «живой» разработки ПО, зафиксированные в манифесте «живой» разработки:

- Люди их общение более важны, чем процессы и инструменты
- Работающая программа более важна, чем исчерпывающая документация

- Сотрудничество с заказчиком более важно, чем обсуждение деталей контракта
- Отработка изменений более важна, чем следование планам

Кроме того, в XP есть несколько правил (техник), характеризующих особенности модели его жизненного цикла:

- Живое планирование (planning game) - как можно быстрее определить объем работ, который нужно сделать до следующей версии ПО. Решение принимается на основе, в первую очередь, бизнес-приоритетов заказчика и, во-вторую, технических оценок. Планы изменяются, как только они начинают расходится с действительностью или пожеланиями заказчика.
- Частая смена версий (small releases) - первая работающая версия должна появиться как можно быстрее, и тут же должна начать использоваться. Следующие версии подготавливаются через достаточно короткие промежутки времени.
- Простые проектные решения (simple design) - в каждый момент времени система должна быть сконструирована так просто, насколько это возможно. Новые функции добавляются только после ясной просьбы об этом. Вся лишняя сложность удаляется, как только обнаруживается.
- Разработка на основе тестирования (test-driven development) - сначала пишутся тесты, потом реализуются модули так, чтобы тесты срабатывали. Заказчики заранее пишут тесты, демонстрирующие основные возможности системы, чтобы можно было увидеть, что система действительно заработала.
- Постоянная переработка (refactoring) - системы для устранения излишней сложности, увеличения понятности кода, повышения его гибкости. При этом предпочтение отдается более элегантным и гибким решениям, по сравнению с просто дающими нужный результат.
- Программирование парами (pair programming) - весь код пишется двумя программистами на одном компьютере, что повышает его качество (отсутствие ошибок, понятность, читаемость,…).
- Постоянная интеграция (continuous integration) - система собирается и проходит интеграционное тестирование как можно чаще, по несколько раз в день, каждый раз, когда пара программистов оканчивает реализацию очередной функции.
- 40-часовая рабочая неделя - сверхурочная работа рассматривается как признак больших проблем в проекте. Не допускается сверхурочная работа 2 недели подряд — это истощает программистов и делает их работу значительно менее продуктивной.

Более подробно об экстремальном программировании можно почитать здесь: http://www.xprogramming.ru/index.html.

Классификацию современных моделей жизненного цикла ПО по типам проектов можно найти в обзоре: Рассел Арчибальд. Модели жизненного цикла высокотехнологичных проектов. http://www.pmprofy.ru/content/rus/107/1073-article.asp

Вопросы для контроля

1. Что такое жизненный цикл программного продукта?
2. Что такое процесс, действие, задача?
3. Какие типы процессов и конкретные процессы вы запомнили?
4. Что такое модель жизненного цикла ПО?
5. Какие типы моделей вы знаете? В чем их преимущества, недостатки, область применимости?
6. Что вы можете сказать об особенностях моделей жизненного цикла MSF, RUP, XP?

Рекомендуемая литература

- **Основная**
  - Шафер Д, Фатрел Р, Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат.: Пер. с англ. - М.: Вильямс., 2003. - 1136с. (стр.31; 135-175)
  - ГОСТ Р ИСО/МЭК 12207-99. Процессы жизненного цикла программных средств. http://www.staratel.com/iso/InfTech/DesignPO/ISO12207/ISO12207-99/ISO12207.htm
  - Оценка и аттестация зрелости процессов создания и сопровождения программных средств и информационных систем (ISO/IEC TR 15504) ISBN: 5-212-00884-0/ Изд: АйТи, Книга и бизнес. http://www.ntrlab.ru/rus/method/iso15504/ Глава 2. Раздел 5. Измерение «процесс»
- **Дополнительная**
  - В.Липаев. Стандарты, регламентирующие жизненный цикл сложных программных комплексов http://www.pcweek.ru/year1998/N24/CP1251/Reviews/chapt1.htm
  - В. В. Кулямин. Технологии программирования. Компонентный подход. Лекция 2. Жизненный цикл и процессы разработки ПО. МГУ. ВМК. Каф. Системного программирования. http://www.ispras.ru/~RedVerst/RedVerst/Lectures and training courses/Software Development Technologies/Lecture02.doc

Использованные источники

При разработке материалов лекции использовались следующие источники:

№	*Источник*	*Темы лекции*	*Слайды*
1	Шафер Д, Фатрел Р, Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат.: Пер. с англ. - М.: Вильямс., 2003. - 1136с		
1.1	Введение (стр.31)	Немного истории	3
1.2	Модели жизненного цикла разработки ПО (стр. 135-175)		
3	В.Липаев. Стандарты, регламентирующие жизненный цикл сложных программных комплексов http://www.pcweek.ru/year1998/N24/CP1251/Reviews/chapt1.htm	История…	4
4	ГОСТ Р ИСО/МЭК 12207-99. Процессы жизненного цикла программных средств. http://www.staratel.com/iso/InfTech/DesignPO/ISO12207/ISO12207-99/ISO12207.htm	ISO 12207. Структура ЖЦ ПО	7
5	Оценка и аттестация зрелости процессов создания и сопровождения программных средств и информационных систем (ISO/IEC TR 15504) ISBN: 5-212-00884-0/ Изд: АйТи, Книга и бизнес. http://www.ntrlab.ru/rus/method/iso15504/ Глава 2. Раздел 5. Измерение «процесс»	ISO 15504 …	8-15
	С.Н.Баранов. Управление программным проектом. http://www.exams.icqinfo.ru/edu/tp-part2.rar	Схемы моделей ЖЦ спиральной, V-образной	
	В. В. Кулямин. Технологии программирования. Компонентный подход. Лекция 2. Жизненный цикл и процессы разработки ПО. http://www.ispras.ru/~RedVerst/RedVerst/Lectures and training	Модель ЖЦ RUP, XP	

	courses/Software Development Technologies/Lecture02.doc МГУ. ВМК. Каф. Системного программирования.		
	Андрей Колесов. Введение в методологию Microsoft Solutions Framework. http://www.bytemag.ru/Article.asp?ID=2866	Модель ЖЦ MSF	