**Стандартизация в области информационных технологий**

**В области ИТ наиболее значимые с точки зрения практики стандарты публикуются следующими организациями:**

* **Институт инженеров по электротехнике и радиоэлектронике** (IEEE, [www.ieee.org](http://www.ieee.org)) в течение многих лет остается лидирующей научно-технической организацией, в том числе, в создании стандартов документации программного обеспечения. Большинство стандартов разработаны различными комитетами, состоящими из опытных и ответственных инженеров-профессионалов.
* **Международная организация по стандартизации (ISO)** имеет огромное влияние во всем мире, особенно среди организаций производителей, имеющих дело с Евросоюзом (ЕС). В настоящее время фактически все современные стандарты в области ИТ, переведенные и принятые в РФ – это стандарты, подготовленные ISO совместно с международной электротехнической комиссией – МЭК (IEC). Вы знаете, что особое внимание уделяется вопросам обеспечения качества продукции на международном уровне, поэтому, согласно постановления правительства РФ №113 от 02.02.1998 соблюдение требований ISO 9000 (серия стандартов, регламентирующих управление качеством (менеджмент качества) на предприятиях) – обязательное условие для получения госзаказа.
* **Институт технологий разработки программного обеспечения** (SoftwareEngineeringInstitute – SEI, [sei.cmu.edu](http://sei.cmu.edu) – более 1000 статей) был учрежден Министерством обороны США в университете Карнеги-Меллон для поднятия уровня технологии программного обеспечения у подрядчиков Министерства обороны. Работа SEI также была принята многими коммерческими компаниями, которые считают улучшение процесса разработки программного обеспечения своей стратегической корпоративной задачей. Мы обратимся к одному из стандартов, разработанному SEI, который называется Моделью зрелости возможностей (СММ).
* **Консорциум по технологии манипулирования объектами** (ObjectManagementGroup, [www.omg.org](http://omg.org)) является некоммерческой организацией, в которую в качестве членов входят около 700 компаний. OMG устанавливает стандарты для распределенных объектно-ориентированных вычислений. Нужно заметить, что OMG использует унифицированный язык моделирования UML в качестве своего стандарта для описания проектов. UML мы будем изучать детально, т.к. использование этого языка совместно с унифицированным процессом фирмы Rational является основой при проработке ядра курсового проекта.

**Понятие жизненного цикла системы**

Необходимость стандартизации разработки программного обеспечения наиболее удачно описана во введении стандарта **ГОСТ Р ИСО/МЭК 12207-99 «Информационная технология. Процессы жизненного цикла программных средств»**: «Программное обеспечение является неотъемлемой частью информационных технологий и традиционных систем, таких, как транспортные, военные, медицинские и финансовые. Имеется множество разнообразных стандартов, процедур, методов, инструментальных средств и типов операционной среды для разработки и управления программным обеспечением. Это разнообразие создает трудности при проектировании и управлении программным обеспечением, особенно при объединении программных продуктов и сервисных программ. Стратегия разработки программного обеспечения требует перехода от этого множества к общему порядку, который позволит специалистам, практикующимся в программном обеспечении, «говорить на одном языке» при разработке и управлении программным обеспечением. Этот международный стандарт обеспечивает такой общий порядок».

**Стандарт ГОСТ Р ИСО/МЭК 12207-99** определяет базовое понятие программной системы (**ЖЦ ПС**)– «жизненный цикл» (ГОСТ Р ИСО/МЭК ТО 15271-2002 «Информационная технология. Руководство по применению ГОСТ Р ИСО/МЭК 12207»).

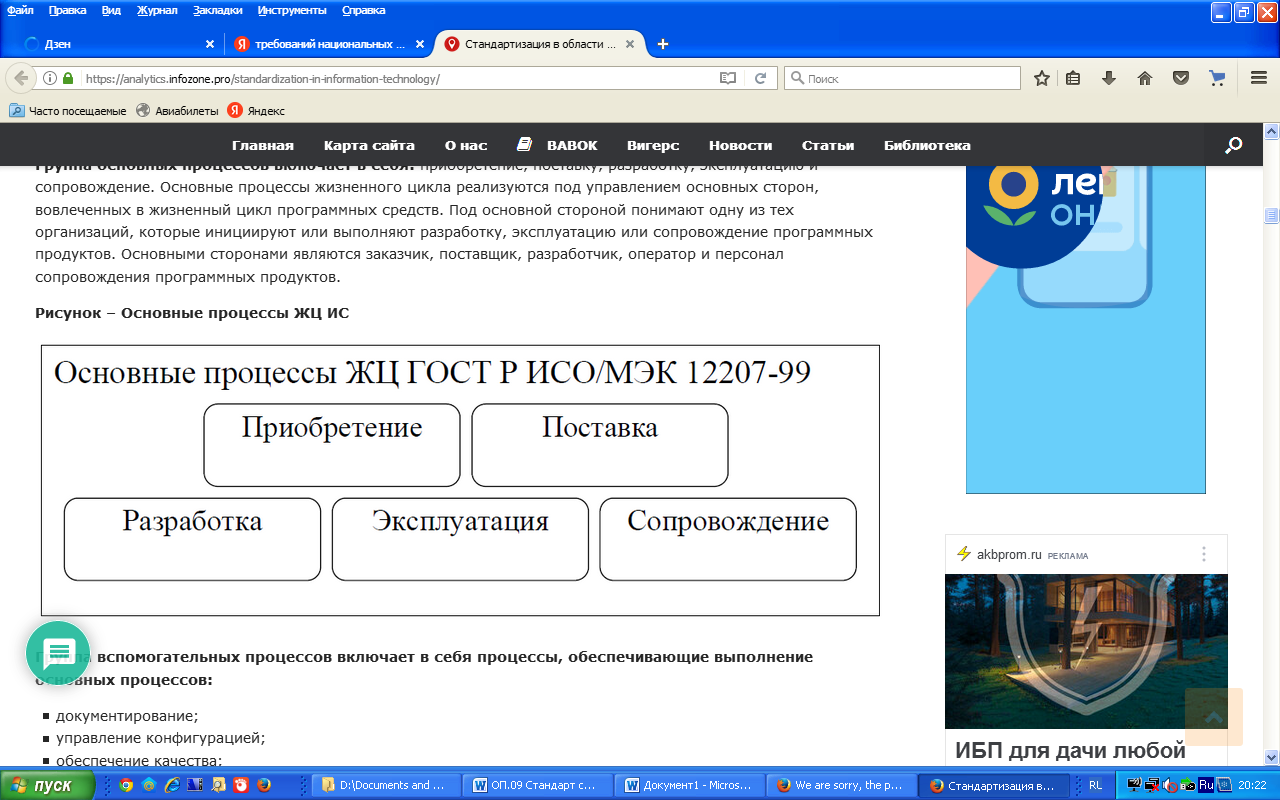
**ГОСТ Р ИСО/МЭК 12207-99** вводит понятие модели жизненного цикла как структуры, состоящей из процессов, и охватывающей жизнь системы от установления требований к ней до прекращения ее использования. **Предлагается это определение подкорректировать и разделить на два определения:**

1. **жизненный цикл** – совокупность процессов, разделенных на работы и задачи, и включающих в себя разработку, эксплуатацию и сопровождение программного продукта, охватывающих жизнь системы от установления требований к ней до прекращения ее использования.
2. **модель жизненного цикла** – структура, определяющая последовательность осуществления процессов, работ и задач, выполняемых на протяжении жизненного цикла программной системы, а также взаимосвязи между ними.

**Процессы ЖЦ разделены на три группы:** основные, вспомогательные и организационные.

**Группа основных процессов включает в себя:** приобретение, поставку, разработку, эксплуатацию и сопровождение. Основные процессы жизненного цикла реализуются под управлением основных сторон, вовлеченных в жизненный цикл программных средств. Под основной стороной понимают одну из тех организаций, которые инициируют или выполняют разработку, эксплуатацию или сопровождение программных продуктов. Основными сторонами являются заказчик, поставщик, разработчик, оператор и персонал сопровождения программных продуктов.

**Рисунок – Основные процессы ЖЦ**



**Группа вспомогательных процессов включает в себя процессы, обеспечивающие выполнение основных процессов:**

* документирование;
* управление конфигурацией;
* обеспечение качества;
* верификация;
* аттестация;
* оценка;
* аудит;
* решение проблем.

**Группа организационных процессов включает в себя процессы:**

* управление проектами;
* создание инфраструктуры проекта;
* определение, оценка и улучшение самого ЖЦ;
* обучение.

**В тексте ГОСТ 12207-99** работы, входящие в состав основных, вспомогательных и организационных процессов охарактеризованы очень общно, фактически намечены только их направления, поэтому для того, что бы приступить к проектированию понадобятся стандарты и дополнительная литература, раскрывающая содержание каждого отдельного процесса  
Из группы основных процессов наибольший интерес представляет процесс разработки.  
Следует отметить, что ГОСТ 34.601-90 «Автоматизированные системы.

**Стадии создания процесса создания автоматизированной системы (АС) разделяет на следующие стадии:**

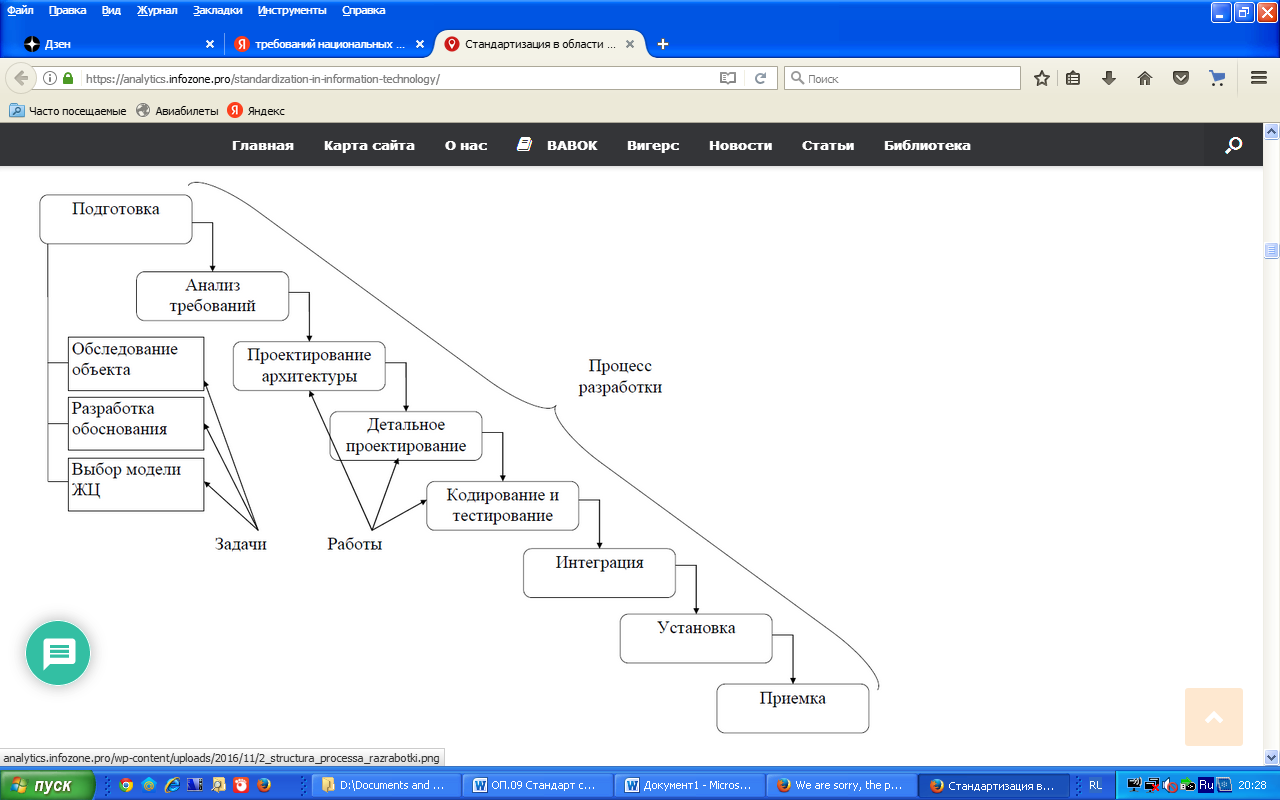
* формирование требований к АС,
* разработка концепции АС,
* техническое задание,
* эскизный проект,
* технический проект,
* рабочая документация,
* ввод в действие,
* сопровождение.

Стадии разделены на этапы, содержание которых перекликается с содержанием ряда задач, описанных в ГОСТ 12207-99.

**Процесс разработки**

**Процесс разработки (developmentprocess)** предусматривает действия и задачи, выполняемые разработчиком, и охватывает работы по созданию ПС и его компонентов в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации; подготовку материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала, и т. д.

**Рисунок – Структура процесса разработки**



**Подготовительная работа**

начинается с выбора модели ЖЦ ПС (жизненный цикл программной системы), соответствующей масштабу, значимости и сложности проекта. Действия и задачи процесса разработки должны соответствовать выбранной модели. Разработчик должен выбрать, адаптировать к условиям проекта и использовать согласованные с заказчиком стандарты, методы и средства разработки, а также составить план выполнения работ.

**Анализ требований**

**Анализ требований к ПС предполагает определение следующих характеристик для каждого компонента ПС:**

* функциональных возможностей, включая характеристики производительности и среды функционирования компонента;
* внешних интерфейсов;
* спецификаций надежности и безопасности;
* эргономических требований;
* требований к используемым данным;
* требований к установке и приемке;
* требований к пользовательской документации;
* требований к эксплуатации и сопровождению.

**Проектирование архитектуры**

системы на высоком уровне заключается в определении компонентов ее оборудования, ПС и операций, выполняемых эксплуатирующим систему персоналом. Архитектура системы должна соответствовать требованиям, предъявляемым к системе, а также принятым проектным стандартам и методам.  
**Проектирование архитектуры ПС включает следующие задачи (для каждого компонента ПС):**

* трансформацию требований к ПС в архитектуру, определяющую на высоком уровне структуру ПС и состав его компонентов;
* разработку и документирование программных интерфейсов ПС и баз данных;
* разработку предварительной версии пользовательской документации;
* разработку и документирование предварительных требований к тестам и плана интеграции ПС.

**Детальное проектированиеПС включает следующие задачи:**

* описание компонентов ПС и интерфейсов между ними на более низком уровне, достаточном для их последующего самостоятельного кодирования и тестирования;
* разработку и документирование детального проекта базы данных;
* обновление (при необходимости) пользовательской документации;
* разработку и документирование требований к тестам и плана тестирования компонентов ПС;
* обновление плана интеграции ПС.

**Кодирование и тестированиеПС охватывают следующие задачи:**

* разработку (кодирование) и документирование каждого компонента ПС и базы данных, а также совокупности тестовых процедур и данных для их тестирования;
* тестирование каждого компонента ПС и базы данных на соответствие предъявляемым к ним требованиям. Результаты тестирования компонентов должны быть документированы;
* обновление (при необходимости) пользовательской документации;
* обновление плана интеграции ПС.

Для каждого из агрегированных компонентов разрабатываются наборы тестов и тестовые процедуры, предназначенные для проверки каждого из квалификационных требований при последующем квалификационном тестировании. Квалификационное требование — это набор критериев или условий, которые необходимо выполнить, чтобы квалифицировать программный продукт как соответствующий своим спецификациям и готовый к использованию в условиях эксплуатации.

**ГОСТ Р ИСО/МЭК 12119-2000 «Информационная технология. Пакеты программ. Требования к качеству и тестирование»** содержит указания, которые определяют порядок тестирования продукта на соответствие его требованиям к качеству. Тестирование является трудоемким процессом. Согласно оценкам некоторых специалистов процентное  
распределение времени между процессами проектирование – разработка – тестирование находится в отношении 40-20-40. В этой связи широкое распространение получают системы автоматизации тестирования. В стандарте IEEE 1209-1992 «RecommendedPracticefortheEvaluationandSelectionof CASE Tools» сформулированы общие требования к функциям средств автоматизации тестирования.

**Интеграция системы**заключается в сборке всех ее компонентов, включая ПС и оборудование. После интеграции система, в свою очередь, подвергается квалификационному тестированию на соответствие совокупности требований к ней. При этом также производятся оформление и проверка полного комплекта документации на систему.

**Установка системы**осуществляется разработчиком в соответствии с планом в той среде и на том оборудовании, которые предусмотрены договором. В процессе установки проверяется работоспособность ПС и баз данных.

**Приемка системы**предусматривает оценку результатов квалификационного тестирования ПС и системы и документирование результатов оценки, которые проводятся заказчиком с помощью разработчика. Разработчик выполняет окончательную передачу ПС заказчику в соответствии с договором, обеспечивая при этом необходимое обучение и поддержку. Наш курс преимущественно нацелен на детальное рассмотрение первых четырех работ процесса разработки ПС. Каждой из этих работ будет посвящен отдельный раздел, а сейчас для дальнейшего изложения необходимо несколько слов сказать о моделях ЖЦ ПС.

**Модели жизненного цикла программного средства**

**Модель жизненного цикла** – структура, определяющая последовательность осуществления процессов, работ и задач, выполняемых на протяжении жизненного цикла информационной системы, а также взаимосвязи между ними.

**К настоящему времени наибольшее распространение получили две основные модели жизненного цикла:**

* каскадная (водопадная) модель;
* спиральная модель.

**Каскадная модель**

**Каскадная модель** демонстрирует классический подход к разработке различных систем в различных прикладных областях. Для разработки информационных систем данная модель широко использовалась в 70-х и первой половине 80-х годов. Именно каскадная модель положена в основу ГОСТ серии 34.xxx и стандарта Министерства обороны США DOD-STD-2167A. **Процессы ГОСТ 12207-99 в ГОСТ 34.601-90 «Автоматизированные системы. Стадии создания»** названы стадиями.  
Каскадная модель предусматривает последовательную организацию процессов. Причем переход к следующему процессу происходит только после того, как полностью завершены все работы на предыдущем. Каждый процесс завершается выпуском полного комплекта документации, достаточной для того, чтобы работа могла быть продолжена другой командой разработчиков.

***Главный недостаток*** каскадной модели заключается в том, что ошибки и недоработки на любом из этапов проявляются, как правило, на последующих этапах работ, что приводит к необходимости возврата назад. По сведениям консалтинговой компании TheStandishGroup в США более 28 % проектов корпоративных информационных систем (IT-проектов) заканчивались неудачей; почти 46% IT-проектов завершались с перерасходом бюджета (в среднем на 189%); и только 26% проектов укладывается и в выделенный срок, и бюджет.

**Кроме того, к недостаткам каскадной модели следует отнести:**

* сложность распараллеливания работ;
* сложность управления проектом;
* высокий уровень риска и ненадежность инвестиций (возврат на предыдущие стадии может быть связан не только с ошибками, но и с изменениями, произошедшими в предметной области или в требованиях заказчика во время разработки. Фактически это означает, что существует вероятность того, что процесс разработки «зациклится» и система никогда не дойдет до сдачи в эксплуатацию. Расходы на проект будут постоянно расти, а сроки сдачи готового продукта постоянно откладываться).

**Спиральная модель**

В отличие от каскадной, предполагает итерационный процесс разработки информационной системы. Спиральную модель предложил в середине 1980-х годов Барри Боэм. Каждый виток спирали соответствует созданию фрагмента или версии программного изделия, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы на следующем витке спирали.

На каждой итерации углубляются и последовательно конкретизируются детали проекта, собираются метрические данные, которые используются для оптимизации последующих итераций. Однако усложняется механизмы обеспечения целостности документации (когда то или иное требование или определение приводится в тексте только один раз).  
**Принципиальные особенности спиральной модели:**

* отказ от фиксации требований и назначение приоритетов пользовательским требованиям;
* разработка последовательности прототипов, начиная с требований наивысшего приоритета;
* идентификация и анализ риска на каждой итерации;
* оценка результатов по завершении каждой итерации и планирование следующей итерации.

**Быстрая разработка приложений**

На основе спиральной модели была основана практическая технология, получившая название «быстрая разработка приложения» — RAD (RapidApplicationDevelopment).

**При этом ЖЦ состоял из четырех стадий:**

* анализ и планирование требований,
* проектирование,
* реализация,
* внедрение.

**Основные принципы RAD:**

* разработка приложений по определенной системе;
* необязательность полного завершения работ на каждой из стадий жизненного цикла ПО;
* обязательность вовлечения пользователей в процесс разработки;
* применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
* использование прототипирования, позволяющее полнее выяснить и удовлетворить потребности пользователей;
* тестирование и развитие проекта, осуществляемые одновременно с разработкой;
* ведение разработки немногочисленной хорошо управляемой командой профессионалов;
* грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

В начале 2001 г. ряд ведущих специалистов в области программной инженерии (Мартин Фаулер, Кент Бек и др.) сформировали группу под названием AgileAlliance для поддержания и развития нового подхода к проектированию – «быстрая разработка ПО» (AgileSoftwareDevelopment). Одной из реализаций этого подхода является «Экстремальное программирование» (ExtremeProgramming — XP).

**Принципы экстремального программирования заключаются в следующем:**

1. В команде работает от трех до десяти программистов. Один или несколько заказчиков должны иметь возможность непрерывного обеспечения текущей экспертизы.
2. Программы разрабатываются трехнедельными итерациями. На каждой итерации производится работающий, протестированный код, который может сразу использоваться заказчиками. Собранная система переправляется к конечным пользователям в конце каждого периода выпуска версий, который может занимать от двух до пяти итераций.
3. Единицей собираемых требований к ПО является «пользовательская история» (userstory), записанная на индексной карточке, и, описывающая с точки зрения пользователя функциональность, которая может быть разработана за одну итерацию. Заказчики и программисты планируют работы на следующей итерации таким образом:
   * программисты оценивают время для завершения работы с каждой карточкой;
   * заказчики расставляют приоритеты, изменяют и пересматривают их при необходимости. Разработка истории начинается с ее обсуждения программистами и экспертом-заказчиком.
4. Программисты работают парами и следуют строгим стандартам кодирования, установленным ими в начале проекта. Они создают модульные тесты для всего, что они пишут, и добиваются, чтобы эти тесты выполнялись каждый раз при сдаче кода на обязательный контроль версий и в систему управления конфигурацией.
5. В то время как программисты работают, заказчики посещают программистов, чтобы прояснять идеи, пишут приемочные тесты системы для прогона во время итерации и в ее конце выбирают истории для реализации в следующей итерации.
6. Каждый день команда проводит оперативные совещания, на которых программисты рассказывают, над чем они работают, что продвигается хорошо и в чем требуется помощь. В конце каждой итерации проводится другое совещание, на котором они оценивают, что было сделано хорошо, и над чем нужно работать в следующий раз. Этот перечень вывешивается, чтобы все могли его видеть, работая во время следующей итерации.
7. Один человек в команде назначается «наставником». Вместе с участниками команды он оценивает использование ими основных приемов: парного программирования и тестирования, ротации пар, поддержания простоты проектных решений, коммуникации и т.д. с целью постоянного совершенствования процесса разработки.

Подход быстрой разработки ПО не является универсальным и применим только в проектах определенного класса. Для характеристики таких проектов АлистерКоберн ввел два параметра — критичность и масштаб.  
**Критичность определяется последствиями, вызываемыми дефектами в ПО, и может иметь один из четырех уровней:**

* С – дефекты вызывают потерю удобства;
* D – дефекты вызывают потерю возместимых средств (материальных или финансовых);
* Е – дефекты вызывают потерю невозместимых средств;
* L – дефекты создают угрозу человеческой жизни.

**Масштаб определяется количеством разработчиков, участвующих в проекте:**

* от 1 до 6 человек – малый масштаб;
* от 6 до 20 человек – средний масштаб;
* свыше 20 человек – большой масштаб.

По оценке Коберна, быстрая разработка ПО применима только в проектах малого и среднего масштаба с низкой критичностью (С или D). Это означает, что технологии RAD и XP наиболее хорошо подходят для относительно небольших проектов, разрабатываемых для конкретного заказчика, и не применимы для построения сложных расчетных программ, операционных систем или программ управления сложными объектами в реальном масштабе времени, а также систем, от которых зависит безопасность людей.

**Унифицированный процесс разработки ПО**

В настоящее время продолжаются работы по созданию некоторого универсального процесса разработки ИС. В 1999г. сотрудниками компании RationalАйваром Джекобсоном, ГрадиБучем и Джеймсом Рамбо была издана книга UnifiedSoftwareDevelopmentProcess (Унифицированный процесс разработки ПО), которая была переведена на русский язык и издана издательством «Питер» в 2002. Унифицированный процесс представляет собой попытку объединения водопадной и итеративной моделей ЖЦ.

**При этом, ЖЦ разделен на 4 фазы:**

1. **Начало (inception):** осуществляется первичная оценка проекта.
   * создается упрощенная модель вариантов использования, содержащая наиболее критичные с точки зрения реализации прецеденты;
   * создается пробный вариант архитектуры, содержащей наиболее важные подсистемы;
   * проводится идентификация и определение приоритетов рисков;
   * планируется фаза проектирования;
   * грубо оценивается весь проект в целом;
2. **уточнение (elaboration):** детально описываются большинство вариантов использования и разрабатывается архитектура системы. В конце фазы проектирования менеджер проекта выполняет подсчет ресурсов, необходимых для завершения проекта. Необходимо ответить на вопрос: достаточно ли проработаны варианты использования, архитектура и план, чтобы можно было бы давать контрактные обязательства на выполнение работы и переходить к подготовке и подписанию «Технического задания»?;
3. **построение (construction)** – создание продукта. В конце этой фазы продукт включает в себя все варианты использования, которые разработчики и заказчик решили включить в текущий выпуск;
4. **внедрение (transition)** – выпуск продукта. Проводится тестирование бета-версии продукта отделом тестирования компании и одновременно организуется пробная эксплуатация системы пользователями. После этого разработчики исправляют обнаруженные ошибки и вносят некоторые из предложенных улучшений в главный выпуск, подготавливаемый для широкого распространения.

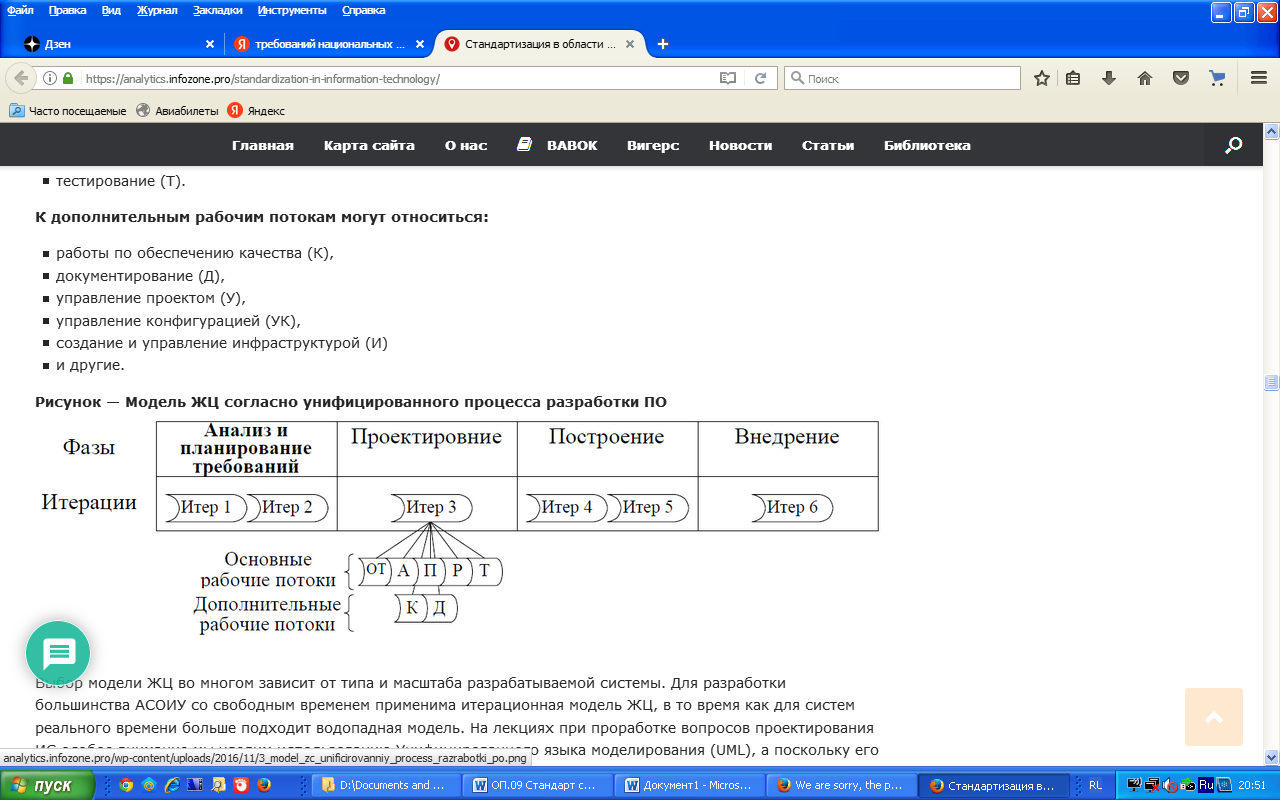
Каждая фаза USDP может включать в свой состав одну или несколько итераций в зависимости от размера проекта. На протяжении каждой итерации может выполняться 5 основных и некоторое количество дополнительных рабочих потоков.  
**К основным рабочим потокам в USDP относятся:**

* определение требований (ОТ);
* анализ (А);
* проектирование (П);
* реализация (Р);
* тестирование (Т).

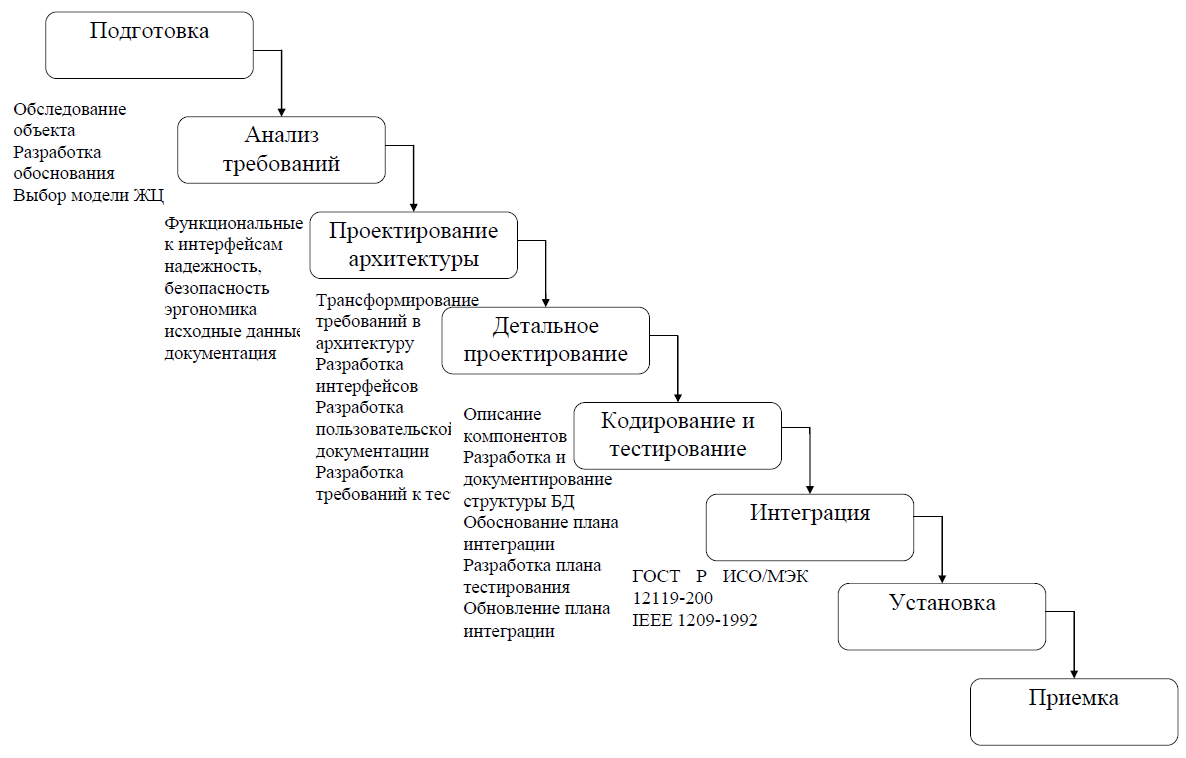
**К дополнительным рабочим потокам могут относиться:**

* работы по обеспечению качества (К),
* документирование (Д),
* управление проектом (У),
* управление конфигурацией (УК),
* создание и управление инфраструктурой (И)
* и другие.

**Рисунок — Модель ЖЦ согласно унифицированного процесса разработки ПО**



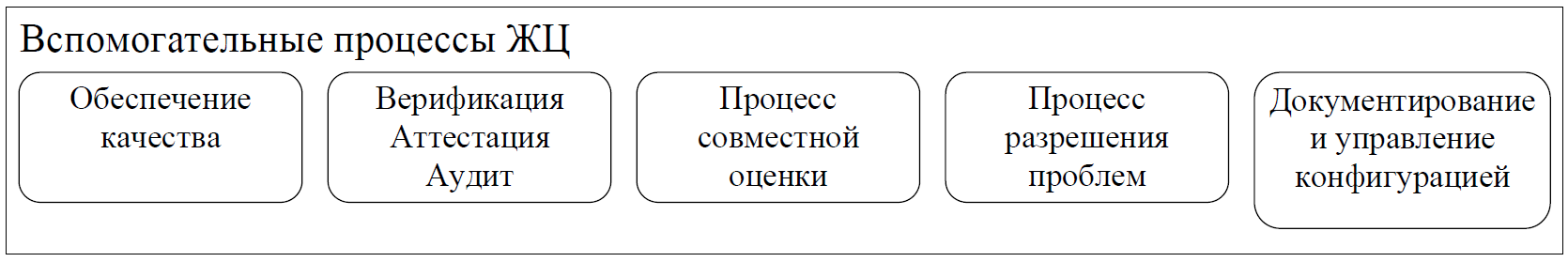
Выбор модели ЖЦ во многом зависит от типа и масштаба разрабатываемой системы. Для разработки большинства АСОИУ со свободным временем применима итерационная модель ЖЦ, в то время как для систем реального времени больше подходит водопадная модель. На лекциях при проработке вопросов проектирования ИС особое внимание мы уделим использованию Унифицированного языка моделирования (UML), а поскольку его создателями являются Гради Буч и Джеймс Рамбо, то мы будем обращаться и к идеологии Унифицированный процесс разработки.

**Рисунок – Нормативные документы, сопровождающие процесс разработки**  


**Вспомогательные процессы жизненного цикла**

**Процесс обеспечения качества**

**Процесс обеспечения качества (qualityassuranceprocess)** обеспечивает соответствующие гарантии того, что ПС и процессы его ЖЦ соответствуют заданным требованиям и утвержденным планам. Под качеством ПС понимается совокупность свойств, которые характеризуют способность ПС удовлетворять заданным требованиям.

**Рисунок – Структура вспомогательных процессов ЖЦ**  


В контексте ГОСТ Р ИСО/МЭК 9126-93. **«Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению»** под характеристикой качества понимается «набор свойств (атрибутов) программной продукции, по которым ее качество описывается и оценивается».

**Стандарт определяет шесть комплексных характеристик, которые с минимальным дублированием описывают качество ПС:**

* **функциональные возможности** – набор атрибутов, относящихся к сути набора функций и их конкретным свойствам. Функциями являются те, которые реализуют установленные или предполагаемые потребности;
* **надежность** – набор атрибутов, относящихся к способности программного обеспечения сохранять свой уровень качества функционирования при установленных условиях за установленный период времени;
* **практичность** – набор атрибутов, относящихся к объему работ, требуемых для использования и индивидуальной оценки такого использования определенным или предполагаемым кругом пользователей;
* **эффективность** – набор атрибутов, относящихся к соотношению между уровнем качества функционирования программного обеспечения и объемом используемых ресурсов при установленных условиях
* **сопровождаемость** – набор атрибутов, относящихся к объему работ, требуемых для проведения конкретных изменений (модификаций);
* **мобильность** – набор атрибутов, относящихся к способности программного обеспечения быть перенесенным из одного окружения в другое.

**ГОСТ 28195-89 «Оценка качества программных средств. Общие положения»** на верхнем, первом, уровне выделяет 6 показателей – факторов качества: надежность, корректность, удобство применения, эффективность, универсальность и сопровождаемость. Эти факторы детализируются в совокупности 19 критериями качества на втором уровне. Дальнейшая детализация показателей качества представлена метриками и оценочными элементами, которых насчитывается около 240. Каждый из них рекомендуется экспертно оценивать в пределах от 0 до 1. Состав используемых факторов, критериев и метрик предлагается выбирать в зависимости от назначения, функций и этапов жизненного цикла ПС.

**В стандарте ГОСТ 28806-90 «Качество программных средств. Термины и определения»** формализуются общие понятия программы, программного средства, программного продукта и их качества. Даются определения 18 наиболее употребляемых терминов, связанных с оценкой характеристик программ. Уточнены понятия базовых показателей качества, приведенных в ГОСТ 28195-89.  
Вопрос обеспечения качества ПС требует особого внимания, поскольку согласно постановления правительства РФ №113 от 02.02.1998 соблюдение требований международного стандарта обеспечения и управления качеством ISO 9000 – обязательное условие для получения госзаказа.  
На современном этапе недостаточно иметь только методы оценки качества произведенного и используемого программного средства (выходной контроль), необходимо иметь возможность планировать качество, измерять его на всех этапах жизненного цикла программного средства и корректировать процесс производства программного обеспечения для улучшения качества.

Стандарты серии ISO 9000 являются слишком общими. Каждая компания, производящая программное обеспечение и желающая внедрить у себя действенную систему управления качеством на основе стандартов ISO 9000-й серии, должна учесть специфику своей отрасли и разработать систему показателей качества, которая бы отражала реальное влияние факторов качества на программный продукт. С этой целью многие организации определили процесс раздельной систематической и полной проверки – контроль качества (QualityAssurance), который начинается вместе с запуском проекта, предусматривает инспектирование и тестирование и проводится в идеале некоторой независимой организацией.

**Цель инспектирования состоит в проверке частей проекта на наличие дефектов:**

* документации,
* требований,
* результатов анализа,
* проектирования,
* листингов и т.д.

Актуальность инспектирования показывает сравнение стоимости и обнаружения и исправления дефекта во время инспектирования и во время интеграции по данным Fagin, M., «DesignandCodeInspectionstoReduceErrorsinProgramDevelopment, IBM SystemsJournal. Некоторые авторы считают эти данные весьма заниженными.

|  |  |  |
| --- | --- | --- |
|  | **Дефект, найденный в процессе инспектирования** | **Дефект, найденный в процессе интеграции** |
| Количество часов на отыскание | 0,7 – 2 | 0,2 – 10 |
| Количество часов на исправление | 0,3 – 1,2 | > 9 |
| Всего | 1,0 – 3,2 | 9,2 – 19 и больше |

К вопросам отыскания дефектов стали относится намного серьезнее после того как американский спутник стоимостью несколько миллиардов долларов, посланный на Венеру, потерял управление из-за опечатки в подпрограмме коррекции траектории – вместо запятой была поставлена точка с запятой.  
Оценка и улучшение качества выполняется посредством использования метрик – количественных характеристик некоторых показателей процесса.

**Для проведения инспектирования требуется выполнение следующих шагов:**

1. **Процесс инспектирования начинается с планирования.** Разрабатывается классификация дефектов по описанию, степени серьезности и типу. Выполняется выбор метрик, по которым будет проводиться инспектирование, выбор инструментов для сбора и анализа полученных данных, а также распределение ролей между проверяющими:
   * Ведущий ответственен за правильное проведение инспектирования.
   * Корректор отвечает за деятельность команды и направляет ее в нужное русло. Корректор принимает участие в инспектировании.
   * Регистратор отвечает за учет описания и классификацию дефектов, как это принято в команде. Регистратор также участвует в инспектировании.
   * Специализированный инспектирующий – специалист в некоторой узкой области, к которой принадлежит инспектируемый фрагмент.
2. При необходимости может быть организован обзорный семинар для лучшего понимания объекта инспектирования.
3. Проведение инспектирования. Инспектирующие проверяют работу в полном объеме на своих рабочих местах (например, проверяют, соответствует ли инспектируемый программный код детальному проекту). Инспектирующие обычно заносят все дефекты в базу данных (например, доступную через сеть) вместе с описаниями и классификацией. Инспектируемые части системы должны быть логически завершенными.
4. Проводится инспекционное собрание, в ходе которого участники представляют свои результаты.
5. Автор исправляет дефекты (фаза доработки).
6. На окончательном собрании по завершению работы корректор и автор убеждаются в том, что все дефекты исправлены. Однако это не предполагает детальной ревизии всей работы корректором. Все исправления остаются на совести автора, ответственного за свою работу.
7. Как и после других процессов, группа встречается для обсуждения самого процесса инспектирования и решает, как он может быть улучшен.

В компании ведется учет времени, потраченного на инспектирование и объема проверенной работы с целью их дальнейшего использования при оценке инспектирования в будущем. В условиях жесткого временного ограничения используется т.н. система «опеки», при которой каждый член команды опекается коллегой.  
Для учета всех факторов контроля качества удобно пользоваться списками контрольных вопросов. Такие списки содержат пункты, которые необходимо последовательно проверить.  
**Например, план контроля качества программного обеспечения (SoftwareQualityAssurancePlan – SQAP) в соответствии со стандартом IEEE 739-1989 определяет:**

* кто будет нести ответственность за качество – физическое лицо, менеджер, группа, организация и т.п.;
* какая документация требуется;
* какие методы будут использоваться для гарантии качества – инспектирование, тестирование и т.д.;
* какие мероприятия должны быть проведены в ходе управления процессом – собрания, аудиты, обзоры и т. п.

Содержание стандарта и пример его отработки приведен в книге Э. Брауде. Вопросы использования метрик рассматриваются в разделе «Организационные процессы. Процесс усовершенствования».

**Надежность и безопасность**

Одной из наиболее значимых характеристик, входящих в понятие качество, является свойство надежности.  
По определению, установленному в ГОСТ 13377-75 «Надежность в технике. Термины и определения», надежность – свойство объекта выполнять заданные функции, сохраняя во времени значения установленных эксплуатационных показателей в заданных пределах, соответствующих заданным режимам и условиям использования, технического обслуживания, ремонта, хранения и транспортирования. Таким образом, надежность является внутренним свойством системы, заложенным при ее создании и проявляющимся во времени при функционировании и эксплуатации.  
Надежность функционирования ПС наиболее широко характеризуется устойчивостью, или способностью к безотказному функционированию, и восстанавливаемостью работоспособного состояния после произошедших сбоев или отказов.  
Контроль надежности и безопасности создаваемых и модифицируемых программ должен сопровождать весь жизненный цикл ПС посредством специально организованной эффективной технологической системы обеспечения их качества. Проверка и подтверждение качества сложных и критических ПС должна обеспечиваться сертификацией аттестованными проблемно-ориентированными сертифицированными лабораториями.

**Стандарты в области информационной безопасность делят на две группы:**

* оценочные стандарты, предназначенные для оценки и классификации ИС и средств защиты по требованиям безопасности – стандарт Министерства обороны США «Критерии оценки доверенных компьютерных систем», «Гармонизированные критерии Европейских стран», международный стандарт «Критерии оценки безопасности информационных технологий», Руководящие документы Гостехкомиссии России;
* спецификации, регламентирующие различные аспекты реализации и использования средств и методов защиты, публикуемые «Тематической группой по технологии Internet» (InternetEngineeringTaskForce, IETF) и ее подразделений – рабочей группой по безопасности.

**К наиболее значимым оценочным стандартам можно отнести:**

* Гостехкомиссия России. Руководящий документ. Средства вычислительной техники. Межсетевые экраны. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации. – Москва, 1997 – классифицирует межсетевые экраны в соответствие с уровнем фильтрации потока данных эталонной семиуровневой модели.
* ИСО/МЭК 15408:1999 «Критерии оценки безопасности информационных технологий».

**Ко второй группе можно отнести следующие документы:**

* Х.800 «Архитектура безопасности для взаимодействия открытых систем». Выделены основные сетевые сервисы безопасности: аутентификация, управление доступом, обеспечение конфиденциальности и/или целостности данных. Для реализации сервисов предусмотрены следующие сетевые механизмы безопасности и их комбинации: шифрование, электронная цифровая подпись, управление доступом, контроль целостности данных, аутентификация, дополнение трафика, управление маршрутизацией, нотаризация.
* Спецификация Internet-сообщества RFC 1510 «Сетевой сервис аутентификации Kerberos (V5)» рассматривает проблему аутентификации в разнородной распределенной среде с поддержкой концепции единого входа в сеть;
* Х.500 «Служба директорий: обзор концепций, моделей и сервисов», Х.509 «Служба директорий: каркасы сертификатов открытых ключей и атрибутов».

**Процессы верификации, аттестации и аудита**

Верификация, аттестация и аудит являются составной частью плана контроля качества SQAP IEEE 739-1989.  
Верификация отвечает на вопрос: «Делаем ли мы на данном этапе то, что запланировано?». Аттестация и аудит отвечает на вопрос: «Отвечает ли строящийся объект нуждам заказчика?».  
Стандарт IEEE 1012-1986 SoftwareVerificationandValidationPlan (SVVP) объединяет процессы аттестации и аудита под названием валидация и определяет порядок их проведения.

**В процессе верификации проверяются следующие условия:**

* непротиворечивость требований к системе и степень учета потребностей пользователей;
* возможности поставщика выполнить заданные требования;
* соответствие выбранных процессов ЖЦ ПС условиям договора;
* адекватность стандартов, процедур и среды разработки процессам ЖЦ ПС;
* соответствие проектных спецификаций ПС заданным требованиям;
* корректность описания в проектных спецификациях входных и выходных данных, последовательности событий, интерфейсов, логики и т.д.;
* соответствие кода проектным спецификациям и требованиям;
* тестируемость и корректность кода, его соответствие принятым стандартам кодирования;
* корректность интеграции компонентов ПС в систему;
* адекватность, полнота и непротиворечивость документации.