МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ
КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. И.АРАБАЕВА
ФАКУЛЬТЕТ ФИЗИКО-МАТЕМАТИЧЕСКОГО ОБРАЗОВАНИЯ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

«Утверждаю»
Декан ФФМО и ИТ
доц. Бексултанов Ж.Т.
_____
«___»_____201__г.

Учебно-методический комплекс по дисциплине «Разработка и стандартизация программных средств и информационных технологий»

Составитель: ст. преподаватель: Асанбекова Н.О.
Рассмотрена и утверждена на заседании кафедры ПИ
Протокол №__ от «__» _____201__г.

Бишкек-2018 год

РАЗРАБОТКА И СТАНДАРТИЗАЦИЯ ПРОГРАММНЫХ СРЕДСТВ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

I полугодие

СОДЕРЖАНИЕ

1. Понятие и общие положения о стандарте. Стандарты и методологии в жизненном цикле программного обеспечения
2. Стандартизация в области технологии разработки ПО
3. Нормативные документы по стандартизации и виды стандартов. Стандарты в области программного обеспечения
4. Международные организации, разрабатывающие стандарты. Национальные организации, разрабатывающие стандарты
5. Основные, вспомогательные и организационные процессы жизненного цикла программного продукта
6. Технология разработки программного обеспечения
7. Общие принципы моделирования жизненного цикла программных средств. Модели ЖЦ ПС.
8. Проектирование программного продукта. Структурное программирование и объектно-ориентированное проектирование

ТЕМА №1. 1. ПОНЯТИЕ И ОБЩИЕ ПОЛОЖЕНИЯ О СТАНДАРТЕ. СТАНДАРТЫ И МЕТОДОЛОГИИ В ЖИЗНЕННОМ ЦИКЛЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Промышленное применение компьютеров и растущий спрос на программы поставили актуальные задачи существенного повышения *производительности разработки ПО*, разработки индустриальных методов планирования и проектирования программ, переноса организационно-технических, технико-экономических и социально-психологических приемов, закономерностей и методов из сферы материального производства в сферу применения компьютеров. *Комплексный подход* к процессам разработки, эксплуатации и сопровождения ПО выдвинул ряд насущных проблем, решение которых исключит «узкие места» в проектировании программ, уменьшит сроки завершения работ, улучшит выбор и адаптацию существующих программ, а может быть и определит судьбу систем со встроенными ЭВМ. В программных проектах, больших и малых, методология разработки программы используется для проектирования, разработки и сопровождения приложения. Эта методология может полностью отсутствовать при реализации малых проектов. В таких проектах главная идея программы обсуждается одним программистом и конечным пользователем, некоторые детали заносятся на бумагу, и проект реализуется в течение нескольких дней или недель. Совершенно иначе выглядят проекты, в которых задействованы команды разработчиков и группы конечных пользователей, а сроки исполнения проектов исчисляются месяцами и годами совместной работы обеих сторон. В данном случае необходима строгая методология создания и реализации проектов, называемая жизненным циклом разработки программ, или ЖЦРП.

Из истории развития самых разных областей техники известно, что наиболее стратегичным и эффективным методом борьбы с недостатком ресурсов были стандартизация и унификация. Их внедрение, как правило, позволяло в разных отраслях сокращать совокупные затраты на разработку, производство и эксплуатацию изделий.

Кому и зачем нужны стандарты?

Основная масса специалистов – в том числе и в отрасли информационных технологий – это специалисты средней квалификации. А системы должны работать надежно, должны быть совместимы с другими системами, нормально эксплуатироваться, поэтому нужно создать технические и технологические условия для решения этих вопросов. Прежде всего, конечно, нужно обобщать, формализовать и использовать лучший опыт, накопленный в отрасли.

Стандарты **удешевляют совокупную стоимость** владения системами, облегчают возможность расширения, модификации и масштабирования систем, а следовательно, увеличивают срок их жизни и окупаемость инвестиций. Помогает снизить квалификационные

требования к персоналу, сформировать четкие программы обучения, лучше подготовить персонал к решению практических задач.

Стандартизация выгодна всем: и производителям, и потребителям ИС. Она позволяет потребителям ИС:

- не тратить лишних средств на закупку нестандартного оборудования, что может вызвать дополнительные проблемы;
- формализовать и снизить требования к квалификации эксплуатационного персонала без ухудшения качества работ, сохранить независимость от персонала (от «незаменимых» сотрудников);
- иметь возможность выбора поставщиков, которые предоставляют стандартизованные решения.

Итак, стандарты нужны:

– потребителям информационных систем (ИС) – для выбора техники, для упорядочения своей деятельности и взаимодействия с поставщиками;

– поставщикам продуктов и услуг – для снижения себестоимости продукции и следования требованиям рынка;

– разработчикам и эксплуатационникам ИС – для повышения качества решений и обеспечения совместимости с другими системами, а также для применения повторно используемых решений, для снижения трудоемкости и себестоимости работ, повышения их качества.

1.2. Какова структура нормативной базы предприятия и как ее выбрать?

Достаточно часто, особенно на крупных предприятиях, применяются системы иностранного производства; иногда ИТ-служба заказывает разработку ИС; почти всегда ИТ-служба (и/или компания-разработчик) вынуждена сопровождать сразу несколько систем.

Разработка больших проектов, как правило, связана с работой коллективов из нескольких десятков и даже сотен человек. Иногда такие разработки ведет кооперация, включающая несколько организаций. Разумеется, в этих условиях разработка и сопровождение создаваемого программного обеспечения (ПО) немыслимы без совокупности нормативно-методических документов, регламентирующих различные аспекты деятельности людей, методик их поведения и взаимодействия на различных стадиях жизненного цикла ПО.

В обоих случаях комплекс документов, регламентирующих требования к объектам проектирования, порядку взаимодействия людей и организаций в процессе их деятельности, называют нормативно-методическим обеспечением или нормативно-методической базой [3].

Каждая компания применяет продукцию в соответствии не только с ее свойствами, но и с достигнутой культурой использования.

Весьма важно – особенно с позиций формирования этой культуры, сокращения совокупных затрат на использование ИТ – сформировать набор стандартов, шаблонов документов и регламентов, или нормативно-методическую базу, которая охватывала бы набор основных объектов регламентации и поэтапно привела бы к унификации архитектурных решений и эксплуатационной документации; появлению документации сопровождения; формированию правил взаимодействия подразделений между собой и поставщиками.

Рассмотрим основные принципы формирования нормативной базы (с акцентом на разработку, сопровождение и развитие ПО ИС).

1.3. Цели, задачи и состав нормативно-методического обеспечения

Нормативно-методическое обеспечение (НМО) представляет собой комплекс документов, регламентирующих:

– порядок разработки, сопровождения, внедрения и развития ПО ИС;

– общие требования к составу и связям между входящими в него составными частями;

– виды, состав и содержание проектной и программной документации.

Целью НМО является установление общих правил ведения и оформления разработки ПО, обеспечивающих единую нормативную и методическую основу для взаимодействия групп специалистов заказчика и подрядчиков, осуществляющих разработку, эксплуатацию, внедрение и сопровождение ПО ИС. Следование требованиям НМО позволит создавать ИС, которые отличает высокое качество, сопровождаемость, соответствие требованиям международных стандартов, а также поможет снизить затраты на создание и использование ПО ИС.

Основными задачами НМО являются:

– регламентация общего порядка, состава и содержания процессов создания, сопровождения, внедрения и развития ПО ИС;

– регламентация общих требований к ПО ИС для обеспечения их высокого качества, унификации построения, оформления, повторного использования, а также для снижения совокупных затрат на их жизненном цикле (ЖЦ);

– формирование методических материалов, обеспечивающих различным коллективам, участвующим в работах на ЖЦ, возможность использовать наиболее эффективные приемы и методы работы, выполнять их по единой схеме и получать единообразные результаты;

– регламентация состава и форм проектных материалов и программной документации.

Все входящие в состав НМО документы должны быть определены по:

– виду регламентации (стандарт, руководящий документ, положение, инструкция и т. п.);

– статусу регламентирующего документа (международный, отраслевой, предприятия);

– области действия документа (отрасль, организация-заказчик/подрядчик, проект);

– объекту регламентации или методического обеспечения.

Нормативное обеспечение должно определять:

– классификацию ПО;

– базовые термины и определения;

– требования к составу и связям ПО ИС, порядку их формирования и развития;

– общие правила ведения работ;

– требования к сопровождению и эксплуатации;

– правовые отношения держателей подлинников, дубликатов и рабочих копий.

Все ли надо стандартизировать?

Любой регламентирующий документ (в частности, стандарт) имеет два важнейших параметра: объект стандартизации (регламентации) и область действия.

Объекты, подлежащие регламентации, можно подразделить на две большие группы:

– архитектура ИС (или ПО ИС), входящих в область ответственности информационной службы или разрабатывающего коллектива;

– процессы создания, подлежащие регламентации.

Соответственно, **целесообразно рассматривать**:

– элементы НМО, относящиеся к конструкции ИС: общую архитектуру, протоколы, интерфейсы;

– процессы создания, интеграции, сопровождения, развития, обслуживания, эксплуатации ИС и т. д.

Областью действия элементов НМО может быть вся организация либо отдельный проект.

Все входящие в состав НМО документы должны быть **определены по**:

– виду регламентации (стандарт, руководящий документ, положение, инструкция и т. п.);

– статусу (международный, отраслевой, предприятия);

– области действия документа (заказчик, подрядчик, проект);

– объекту регламентации или методического обеспечения.

В качестве объектов регламентации и методического обеспечения могут выступать:

– объект работ, его архитектура, протоколы и интерфейсы;

– стадии работ и их результаты;

– процессы жизненного цикла и их отдельные элементы (задачи, работы и т. п.);

– этапы работ по стадиям и процессам;

– регламенты выполнения работ и отдельные процедуры;

– роли персонала, обязанности и ответственность за конечный результат;

– результаты работ (проектные и программные документы);

– метрики измерения сложности, качества и трудоемкости работ.

НМО может также определять порядок взаимодействия организации-заказчика с соисполнителями. В качестве документа, входящего в состав НМО, могут рассматриваться и организационно-распорядительные документы: положения, регламенты и пр., – касающиеся взаимодействия организации-заказчика с соисполнителями.

Нормативно-методическая документация должна регламентировать и методически поддерживать процессы жизненного цикла по ГОСТ Р ИСО/МЭК 12207-99, в первую очередь это планирование проектных работ, управление графиком и ресурсами для их выполнения; метрика и методика оценки трудоемкости работ на всех стадиях выполнения проекта; управление рисками; управление качеством ПО; управление требованиями; конфигурационное управление; управление изменениями; тестирование ПО; порядок испытаний; документирование.

Все документы, входящие в состав НМО, являются стандартами. Стандартизации подлежат только объекты и процессы, характерные для предприятия в целом, прошедшие длительную апробацию, проверку временем. Такие документы закрепляются в виде стандартов предприятия.

Методическое обеспечение должно поддерживать комплексную методологию эффективного ведения работ. Желательно, чтобы эта методология максимально охватывала весь жизненный цикл ПО и была применима к объектам и видам деятельности организации, стремящейся ее использовать. Очень часто организации применяют методологии, не ориентированные на архитектуру ПО, нужную организации, либо на классы решаемых ею задач, либо на виды деятельности (например, заказывающая организация использует непосредственно методологию разработки без ее адаптации к своим условиям). Именно в этом случае требуется привлечение консалтинговых организаций.

В соответствии **с требованиями ГОСТ Р ИСО/МЭК 12207-99** методическая база должна формироваться по процессам, их задачам или шагам (в зависимости от используемой методологии) и ролям персонала в целях определения наиболее эффективных приемов работы и обеспечения качества. В дополнение к методологии обычно необходимо разрабатывать внутренние регламенты работ и взаимодействия участников проектов. Регламенты могут распространяться как на конкретный проект (или нескольких проектов), так и на организацию в целом. Регламенты могут устанавливаться и для отдельных объектов технологии создания, сопровождения и развития ПО ИС. Должны быть сформированы правила изменения регламентов. Опыт отечественных и зарубежных организаций показывает, что наибольший

эффект (повышение качества, сокращение сроков разработки) достигается, если методическое обеспечение поддержано инструментальными средствами.

Тема №2. Стандартизация в области технологии разработки ПО

Основой любых технологических дисциплин, будь то в строительстве, машиностроении, приборостроении и т.п., как правило, является некоторый набор базовых стандартов. В этом смысле не является исключением и такая дисциплина, как «Технология разработки программного обеспечения».

Во всех индустриально развитых странах стандартам, нацеленным на обеспечение качества ПС, уделяется большое внимание. В первую очередь это относится к проблеме использования ПС в критических приложениях, таких как национальная оборона, космос, энергетика (особенно ядерная), производство (потенциально опасное для жизни и здоровья людей, окружающей среды), транспорт и коммуникации. В настоящее время качество программных средств поддерживается и обеспечивается широкой номенклатурой международных (ISO, IEC), а также национальных стандартов США (DOD, MIL, NIST, ANSI, IEEE), Германии (DIN), Франции (NFZ), Японии (JIS), России (ГОСТ) и др.

За рубежом разработка стандартов по технологии производства ПС идет непрерывно: последовательно публикуются проекты и новые версии этих стандартов, происходит их публичное обсуждение, согласование и утверждение. Таким образом, многие стандарты поэтапно углубляются и детализируются, иногда разрастаясь до целых групп стандартов. Особенно данный подход характерен для общественных организаций, занимающихся вопросами стандартизации, таких, как ISO, IEC, ANSI, IEEE.

В целом процесс стандартизации в этой области находится еще на начальном этапе своего развития: большинство проектов и стандартов носит рекомендательный характер. Очень динамичное развитие стандартов в этой области за рубежом в настоящее время вызывает некоторые трудности при их изучении и анализе. Однако оно внушает определенный оптимизм: поскольку, рано или поздно все самое ценное и наилучшее останется и станет фактическим стандартом для всех производителей и потребителей.

Безусловно, есть области (например, связанные с государственной безопасностью), которые не могут регулироваться международными нормативными документами. Здесь должны существовать наши стандарты.

Основными результатами деятельности по стандартизации должны быть повышение степени соответствия продуктов (услуг), процессов их функциональному назначению, устранение технических барьеров в международном товарообмене, содействие научно-техническому прогрессу и сотрудничеству в различных областях.

Стандартизация связана с такими понятиями, как «объект стандартизации» и «область стандартизации». Объектом стандартизации обычно называют продукцию, процесс, услугу, для которых разрабатывают те или иные требования, характеристики, параметры, правила и т.п.

Стандартизация может касаться либо объекта в целом, либо его отдельных составляющих (характеристик). Областью стандартизации называют совокупность взаимосвязанных объектов стандартизации. Стандартизация осуществляется на разных уровнях. Уровень стандартизации зависит от того, участники какого географического, экономического, политического региона мира принимают стандарт. Если участие в стандартизации открыто для соответствующих органов любой страны, то это международная стандартизация. Региональная стандартизация – деятельность, открытая только для соответствующих органов государств одного географического, политического или экономического региона. Региональная и международная стандартизация осуществляется специалистами стран, представленных в соответствующих региональных и международных организациях (рис.1).

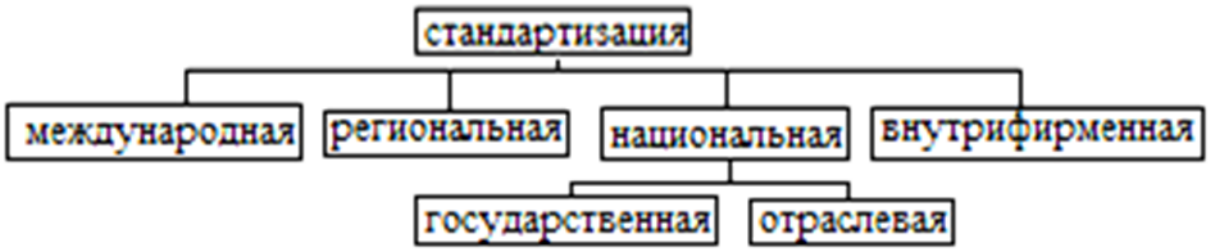


Рис.1. Схема уровней стандартизации

При этом национальная стандартизация также может осуществляться на разных уровнях: на государственном, отраслевом, в том или ином секторе экономики (например, на уровне министерств), на уровне ассоциаций, производственных фирм, предприятий (фабрик, заводов) и учреждений.

Стандартизацию, которая проводится в административно-территориальной единице (провинции, крае и т.п.), принято называть административно-территориальной стандартизацией.

Тема №3. Нормативные документы по стандартизации и виды стандартов. Стандарты в области программного обеспечения

В процессе стандартизации вырабатываются нормы, правила, требования, характеристики, касающиеся объекта стандартизации, которые оформляются в виде нормативного документа.

Рассмотрим разновидности нормативных документов, которые рекомендуются руководством 2-й Международной организации по стандартизации и Международной электротехнической комиссии (ИСО/МЭК), а также принятые в государственной системе стандартизации. Руководство ИСО/МЭК рекомендует стандарты, документы технических условий, своды правил, регламенты (технические регламенты), положения (рис. 2.2).

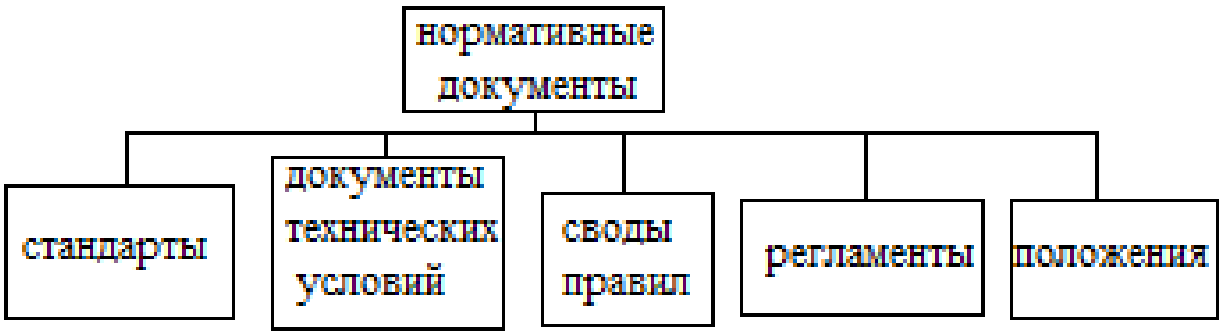


Рис. 2.2. Схема разновидностей нормативных документов

Стандарты бывают международными, региональными, национальными, административно-территориальными. Они принимаются соответственно международными, региональными, национальными, территориальными органами по стандартизации. Все эти категории стандартов предназначены для широкого круга потребителей. По существующим нормам стандартизации стандарты периодически пересматриваются для внесения изменений, чтобы их требования соответствовали уровню научно-технического прогресса, или, согласно терминологии ИСО/МЭК, стандарты должны представлять собой «признанные технические правила». Нормативный документ, в том числе и стандарт, считается признанным техническим правилом, если он разработан в сотрудничестве с заинтересованными сторонами путем консультаций и на основе консенсуса. Указанные выше категории стандартов называют общедоступными. Другие же категории стандартов, такие, как фирменные или отраслевые, не являясь таковыми, могут, однако, использоваться и в нескольких странах согласно существующим там правовым нормам. Отраслевые стандарты разрабатываются применительно к продукции определенной отрасли. Их требования не должны противоречить обязательным требованиям государственных стандартов, а также правилам и нормам безопасности, установленным для отрасли. Принимают такие стандарты государственные органы управления (например, министерства), которые несут ответственность за соответствие требований отраслевых стандартов обязательным требованиям ГОСТ Р.

Диапазон применяемости отраслевых стандартов ограничивается предприятиями, подведомственными государственному органу управления, принявшему данный стандарт. На добровольной основе возможно использование этих стандартов субъектами хозяйственной деятельности иного подчинения. Степень обязательности соблюдения требований стандарта отрасли определяется тем предприятием, которое применяет его, или по договору между изготовителем и потребителем. Контроль за выполнением обязательных требований организует ведомство, принявшее данный стандарт. Стандарты предприятий разрабатываются и

принимаются самими предприятиями. Кроме того, стандартизация на предприятии может затрагивать и продукцию, производимую этим предприятием. Тогда объектами стандарта предприятия будут составные части продукции, технологическая оснастка и инструменты, общие технологические нормы процесса производства этой продукции. Стандарты предприятий могут содержать требования к различного рода услугам внутреннего характера

Стандарты в области программного обеспечения

Стандарты в области программного обеспечения дают возможность разработчикам программного обеспечения использовать данные и программы других разработчиков, осуществлять экспорт/импорт данных.

Такие стандарты регламентируют взаимодействие между различными программами. Для этого предназначены стандарты межпрограммного интерфейса, например OLE (Object Linking and Embedding – «связывание и встраивание объектов»). Без таких стандартов программные продукты были бы «закрытыми» друг для друга.

Стандарты занимают все более значительное место в направлении развития индустрии информационных технологий. Более 250 подкомитетов в официальных организациях по стандартизации работают над стандартами в области информационных технологий. Более 1000 стандартов или уже приняты этими организациями, или находятся в процессе разработки. Процесс стандартизации информационных технологий далеко не закончен (и по нашему мнению, вряд ли когда-либо будет закончен, так как область информационных технологий постоянно динамично развивается).

Все компании-разработчики должны обеспечить приемлемый уровень качества выпускаемого программного обеспечения (ПО). Для этих целей предназначены стандарты качества программного обеспечения или отдельные разделы в стандартах разработки программного обеспечения, посвященные требованиям к качеству программного обеспечения.

С точки зрения пользователя все многообразие ПО должно управляться единообразно. Должна быть единообразная навигация – перемещение по программе, единообразные органы управления ПО и единая реакция программного обеспечения на пользователя. Для этого разработаны стандарты на пользовательский интерфейс – GUI (Graphical User Interface). Все это регламентируется стандартами, действующими в сфере информационных технологий.

Необходимость стандартизации разработки программного обеспечения наиболее удачно описана во введении в стандарт ISO/IEC 12207: «Программное обеспечение является неотъемлемой частью информационных технологий и традиционных систем, таких, как транспортные, военные, медицинские и финансовые. Имеется множество разнообразных стандартов. 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. Международный стандарт обеспечивает такой общий порядок».

Порядок в многообразии стандартов, действующих в сфере ИТ, и классификация их представлены на рисунке 2.3.

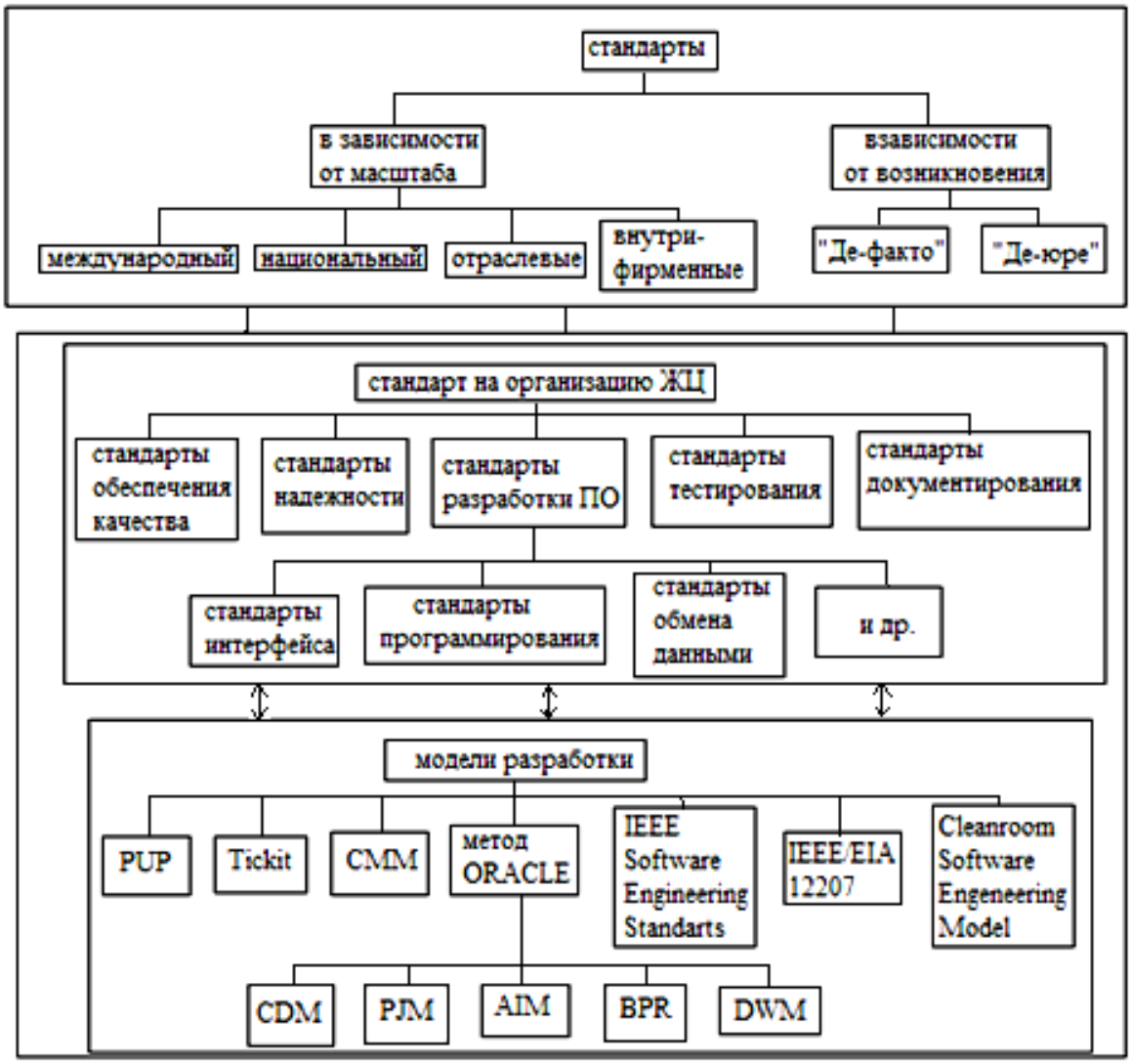


Рис. 2.3. Стандарты, действующие в сфере ИТ

Как видно, верхняя часть классификации напоминает указанные выше виды стандартов. Однако здесь появляются и свои особенности. Это относится прежде всего к стандартам «де-юре» и «де-факто».

Стандарт «де-факто» в 60-е и 70-е годы XX века ставил пользователей в зависимое от производителей положение при использовании основных средств обработки данных и телекоммуникаций. Важный аспект сегодняшней работы по стандартизации – преодоление этой зависимости через продвижение стандартных интерфейсов. Долгое время такими стандартами

были SQL (Structured Query Language) и язык диаграмм Д. Росса SADT (Structured Analysis and Design Technique).

Стандарт «де-юре» создается формально признанной стандартизующей организацией. Он не может быть изменен, не пройдя процесс согласования под контролем организации, разрабатывающей стандарты. Стандарты OSI (Open Systems Interconnection reference model), Ethernet, POSIX, SQL и большинство стандартов языков – примеры такого рода стандартов.

В настоящее время он стал одним из главных стандартов в области информационных систем и обеспечил технологию базового языка для целого поколения СУБД, основанных на реляционной модели. Несмотря на то, что он был коммерчески реализован в начале 80-х годов лишь для небольшой группы программных продуктов, SQL, бесспорно, получил признание с принятием ANSI и ISO стандарта SQL-S6. В качестве примера можно взять переход стандарта «де-факто» в стандарт «де-юре» и то, как это отразилось в развитии стандартизации языка SQL (это рассмотрено в работе [3]).

Одной из причин преуспевания SQL послужило формирование Американским национальным институтом стандартов (American National Standards Institute, ANSI) комитета ХЗН2, учрежденного для разработки стандартов языков баз данных. Представитель IBM предложил использовать в качестве предварительных спецификаций реляционного языка результаты ранее проведенной IBM работы над SEQUEL/2, и разработчики стандарта приступили к работе. Документ, озаглавленный «SQL», представлял собой по большей части трактат о различных формах SQL, используемых в коммерческих программных продуктах.

Международная организация по стандартизации (International Standards Organization, ISO) в рамках технического комитета ТС97 (называемого теперь ISO/IEC JTC1-Joint Technical Committee) также вела работу по созданию стандарта языков реляционных баз данных. В середине 80-х годов как ANSI, так и ISO одобрили стандарты SQL (ANSI – в 1986 г., а ISO – в начале 1987 г.). Объединенный технический комитет (JTC1) предназначен для формирования всеобъемлющей системы базовых стандартов в области ИТ и их расширений для конкретных сфер деятельности.

JTC1 покрывает все: от техники программного обеспечения до языков программирования, компьютерной графики и обработки изображения, соединения оборудования, методов защиты и т.д. Работа над стандартами ИТ в JTCI тематически распределена по подкомитетам (Subcommittees – SC).
В дополнение создана специальная группа по функциональным стандартам (Special Group on Functional Standards – SGFS) для обработки предложений по международным стандартизованным профилям (International Standardized Profiles – ISPs), представляющим определения профилей ИТ.

В качестве еще одного примера появления стандарта можно привести появление ныне популярного UML – Unified Modeling Language. Основные разработки по методам объектно-ориентированного анализа и проектирования появились между 1988 и 1992 гг. К 1994 г. было большое количество неформальных лидеров разработчиков-практиков (около полутора десятков), которые продвигали свои методологии. Все их методы были схожи, при этом зачастую отличия между ними заключались во второстепенных деталях. Назревал разговор о стандартизации. Команда из OMG пыталась рассмотреть проблему стандартизации, но в ответ получила открытое письмо с протестом от всех авторов. В 1996 г. три ведущих специалиста в области объектно-ориентированного анализа и проектирования Джеймс Рамбо (James Rumbaugh), Гради Буч (Grady Booch), Ивар Якобсон (Ivar Jacobson) объединились, и появился на свет Унифицированный метод версии 0.8, в 1996 г. «трое друзей» работали над своим методом, который получил название Unified Modeling Language. В январе 1997 г. различные организации представили свои предложения по стандартизации методов, предусматривающие в первую очередь возможность обмена информацией между различными моделями. В результате сейчас имеется единственное предложение – стандарт UML.

Тема №4. Международные организации, разрабатывающие стандарты. Национальные организации, разрабатывающие стандарты

ИСО еще занимается и проблемами сертификации. Она определяет свои задачи как содействие развитию стандартизации и смежных видов деятельности в мире с целью обеспечения международного обмена товарами и услугами, а также развития сотрудничества в интеллектуальной, научно-технической и экономической областях.

Вопросы информационной технологии, микропроцессорной техники и т. п. входят в область совместных разработок ИСО/МЭК. В последние годы ИСО уделяет много внимания стандартизации систем обеспечения качества. Практическим результатом усилий в этих направлениях являются разработка и издание международных стандартов. При их разработке ИСО учитывает ожидания всех заинтересованных сторон: производителей продукции (услуг), потребителей, правительственных кругов, научно-технических и общественных организаций.

На сегодняшний день в состав ИСО входит 120 стран со своими национальными организациями по стандартизации.

Международные стандарты ИСО не имеют статуса обязательных для всех стран-участниц. Любая страна мира вправе применять или не применять их. Решение связано в основном со степенью участия страны в международном разделении труда и состоянием ее внешней торговли.

Среди национальных организаций, разрабатывающих стандарты, в России и к Кыргызстане действует Государственный комитет по стандартизации.

Согласно Руководству ИСО/МЭК деятельность по стандартизации осуществляют соответствующие органы и организации. Под органом, занимающимся стандартизацией, подразумевается орган, деятельность которого в области стандартизации является общепризнанной на национальном, региональном или международном уровне. Основные функции такого органа – разработка и утверждение нормативных документов, доступных широкому кругу потребителей. Однако он может выполнять немало других функций, что особенно характерно для национального органа по стандартизации.

Национальным органом по стандартизации в России и в КР является Государственный комитет по стандартизации и метрологии (Госстандарт России, КР). Это орган исполнительной власти, осуществляющий межотраслевую координацию, а также функциональное регулирование в области стандартизации, метрологии и сертификации. В ведении Государственного комитета по стандартизации и метрологии находятся службы по надзору за государственными стандартами и обеспечением единства измерений, а также центры стандартизации, метрологии и сертификации, предприятия, учреждения, учебные заведения и иные организации.

В организационной структуре Госстандарта предусмотрены подразделения для реализации значительного объема работ: научно-исследовательские институты, опытные заводы, типографии, учебные заведения, более 100 территориальных центров стандартизации, метрологии и сертификации (ЦСМ). На базе территориальных органов Госстандарта создаются органы по сертификации и испытательные лаборатории.

Работы по государственной стандартизации планируются. Составление планов находится в ведении Госстандарта, который является основным заказчиком по государственным основополагающим стандартам, стандартам общих технических условий и технических условий в части их обязательных требований, по исследованиям в области международных и региональных стандартов относительно принятия и применения их в качестве государственных. Госстандарт определяет стратегические направления по государственной стандартизации, анализирует все заказы, планы работы технических комитетов, предложения от субъектов хозяйственной деятельности и разрабатывает планы по государственной стандартизации, как правило, годовые. Приоритетными считаются задания по гармонизации отечественных нормативных документов с международными (региональными), национальными зарубежными стандартами, а также по разработке требований безопасности к объектам стандартизации и защите прав потребителей. Выполнение планов государственной стандартизации финансируется из государственного бюджета и контролируется Госстандартом.

Технические комитеты по стандартизации. Постоянными рабочими органами по стандартизации являются технические комитеты (ТК), занимаются стандартизацией, как в инициативном порядке, так и по договорам на выполнение такого задания в соответствии с программами ТК и планами государственной стандартизации.

Они специализируются в зависимости от объекта стандартизации. В рамках этой специализации в ТК проводится также работа и по международной (региональной) стандартизации. По линии международной стандартизации ТК занимаются вопросами гармонизации отечественных стандартов с международными, способствуя принятию государственных стандартов в качестве международных, участвуют в организации проведения в России заседаний международных организаций по стандартизации и др.

Научно-технической базой для создания ТК обычно служат предприятия или организации, профиль деятельности которых соответствует специализации технического комитета. В их число включаются и научно-исследовательские институты Госстандарта. Правовой основой для создания ТК служит решение этих государственных органов. Заинтересованные предприятия, организации могут проявлять инициативу по участию их специалистов в работе технического комитета. Госстандарт привлекает к работе в ТК ведущих ученых и специалистов. Общества потребителей имеют право участвовать в работе технических комитетов по определению требований к качеству объекта стандартизации и выбору методов его оценки, в разработке новых и обновлении действующих стандартов. Участие в деятельности технических комитетов всех заинтересованных сторон добровольное.

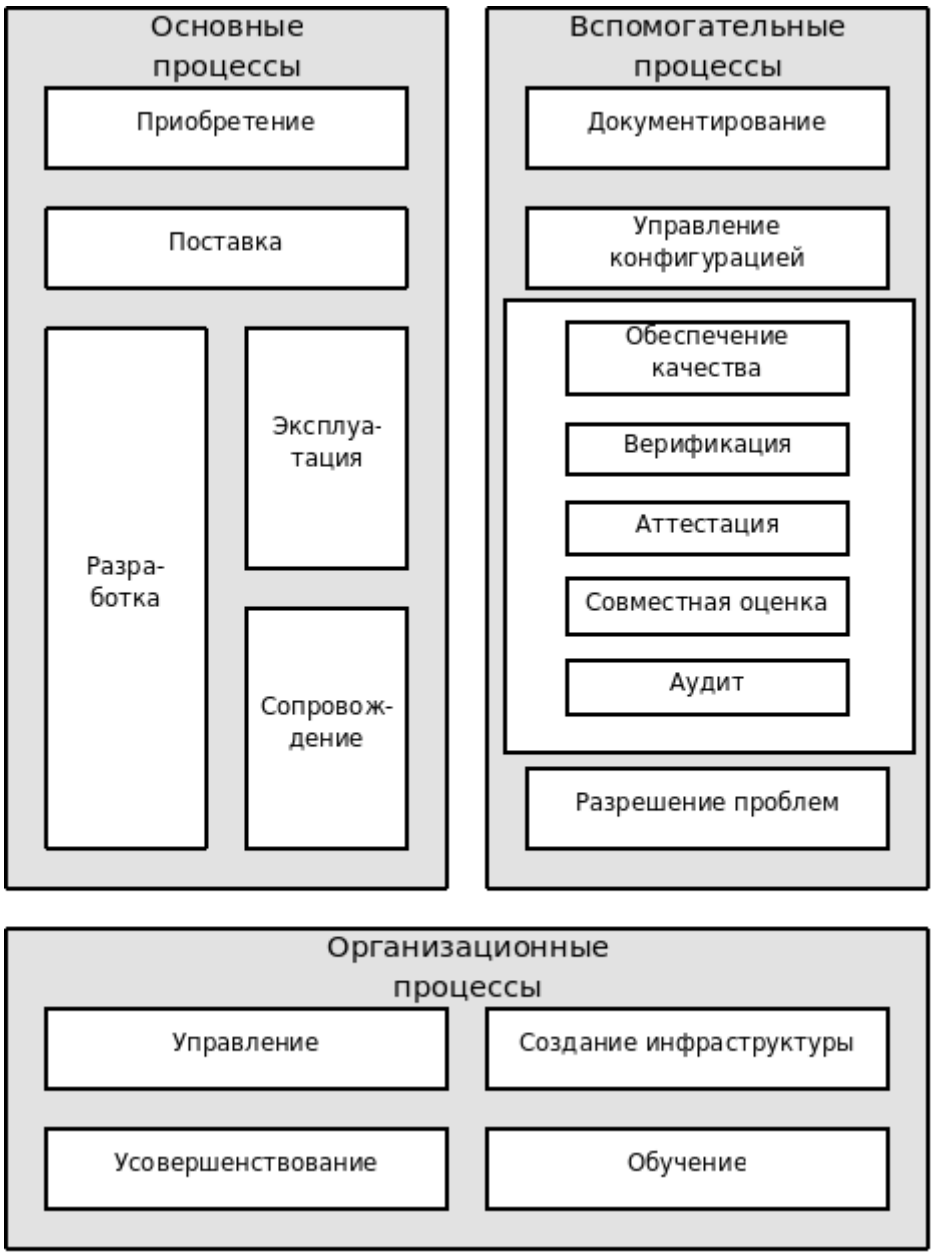
Тема №5. Основные, вспомогательные и организационные процессы жизненного цикла программного продукта

В соответствии со стандартом ГОСТ Р ИСО/МЭК 12207-99 все процессы ЖЦ ПО разделены на три группы:

1. Основные процессы
    1. **приобретение**
    2. **поставка**
    3. **разработка**
    4. **эксплуатация**
    5. **сопровождение** (модификация и исправление системы в случае обнаружения неполадок или возникновении новых требований)
2. Вспомогательные процессы

1. **документирование** (формализованное описание информации, созданной в течение ЖЦ ПО)
2. **управление конфигурацией** (применение административных и технических процедур на всем протяжении ЖЦ ПО для определения состояния компонентов ПО в системе, управления модификациями ПО, описания и подготовки отчетов о состоянии компонентов ПО и запросов на модификацию, обеспечения полноты, совместимости и корректности компонентов ПО, управления хранением и поставкой ПО)
3. **обеспечение качества** (обеспечение соответствующих гарантии того, что ПО и процессы его ЖЦ соответствуют заданным требованиям и утвержденным планам)
4. **верификация** (определение того, что программные продукты, являющиеся результатами некоторого действия, полностью удовлетворяют требованиям или условиям, обусловленным предшествующими действиями)
5. **аттестация** (подтверждение и оценка достоверности проведенного тестирования ПО)
6. **совместная оценка** (оценка состояния работ по проекту и ПО, создаваемому при выполнении данных работ)
7. **аудит** (определение соответствия требованиям, планам и условиям договора)
8. **разрешение проблем**

3. организационные процессы
    1. **управление** (управление выпуском продукта, управление проектом и задачами соответствующих процессов, таких, как приобретение, поставка, разработка, эксплуатация, сопровождение и др.)
    2. **инфраструктура** (выбор и поддержка (сопровождение) технологии, стандартов и инструментальных средств, выбор и установка аппаратных и программных средств, используемых для разработки, эксплуатация или сопровождение ПО)
    3. **усовершенствование** (оценка, измерение, контроль и усовершенствование процессов ЖЦ ПО)
    4. **обучение** (первоначальное обучение и последующее постоянное повышение квалификации персонала)



Основные процессы ЖЦ реализуются ответственным субъектом, вовлеченным в ЖЦ ПС. Ответственным субъектом является одно из юридических лиц (или подразделений, или должностных физических лиц), которое реализует соответствующий процесс. Ответственными субъектами могут быть заказчик, поставщик, разработчик, эксплуатационный оператор и сопровождающий персонал. Перечислим основные процессы ЖЦ ПС и дадим их краткие определения.

Процесс заказа – это работы заказчика (субъекта, приобретающего систему, ПС или получающего программную услугу).

Процесс поставки – это работы поставщика (субъекта, поставляющего систему, ПС или программную услугу заказчику).

Процесс разработки – это работы разработчика (субъекта, проектирующего и разрабатывающего ПС).

Процесс эксплуатации – это работы эксплуатационного персонала (субъекта, обеспечивающего эксплуатационное обслуживание вычислительной системы в заданных условиях в интересах пользователей).

Процесс сопровождения – это работы персонала сопровождения (субъекта, предоставляющего услуги по сопровождению ПС, обеспечивающие контролируемое изменение программного продукта в целях сохранения его исходного состояния и функциональных возможностей). Данный процесс охватывает перенос ПС в другую среду и снятие его с эксплуатации.

Основные процессы жизненного цикла программного продукта

Основные процессы включают в себя набор определенных действий и связанных с ними задач, которые должны быть выполнены в течение жизненного цикла ПП. К ним относятся процессы приобретения, поставки, разработки, эксплуатации и сопровождения.

Процесс приобретения (acquisition process) охватывает действия заказчика по приобретению ПП. К этим действиям относятся:

– инициирование приобретения;

– подготовка заявочных предложений;

– подготовка и корректировка договора;

– надзор за деятельностью поставщика;

– приемка и завершение работ.

Процесс поставки (supply process) охватывает действия и задачи поставщика при снабжении заказчика ПП или услугой. К этим действиям относятся:

– инициирование поставки;

– подготовка ответа на заявочные предложения;

– подготовка договора;

– планирование;

– выполнение и контроль;

– проверка и оценка;

– поставка и завершение работ [3].

Процесс разработки (development process) охватывает действия и задачи разработчика и предусматривает следующие основные направления работ:

– создание ПП и его компонентов в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации;

– подготовку материалов, необходимых для проверки работоспособности и качества ПП;

– подготовку материалов, необходимых для организации обучения персонала и т.д.

Процесс эксплуатации (operation process) охватывает действия и задачи оператора – организации, занимающейся эксплуатацией разработанных ПП или системы. К этим действиям относятся:

- подготовительная работа;
- эксплуатационное тестирование;
- эксплуатация системы;
- поддержка пользователей [3].

Процесс сопровождения (raintenance process) охватывает действия и задачи сопровождающей организации (службы сопровождения). Данный процесс активизируется при изменениях (модификациях) ПП и соответствующей документации, вызванных возникшими проблемами или потребностями в модернизации либо адаптации ПП. В соответствии со стандартом IEEE-90 (IEEE – Institute of Electrical and Electronics Engineers – Институт инженеров по электротехнике и электронике) под сопровождением понимается внесение изменений в ПП в целях исправления ошибок, повышения производительности либо адаптации к изменившимся условиям работы или требованиям [2, 3].

Вспомогательные (поддерживающие) процессы жизненного цикла программного продукта

Основной целью вспомогательных (поддерживающих) процессов является создание надежного, полностью удовлетворяющего требованиям заказчика ПП в установленные договором сроки. К вспомогательным относятся процессы документирования, управления конфигурацией, обеспечения качества, верификации, аттестации, совместной оценки, аудита, разрешения
проблем.

Процесс документирования (documentaton process) предусматривает формализованное описание информации, созданной в течение жизненного цикла ПП. Данный процесс состоит из набора действий, с помощью которых планируют, проектируют, разрабатывают, выпускают, редактируют, распространяют и сопровождают документы, необходимые для всех заинтересованных лиц, таких как руководство, технические специалисты и пользователи системы.

Процесс документирования включает в себя подготовительную работу, проектирование и разработку документации, выпуск документации, сопровождение [3].

Процесс управления конфигурацией (configuration management process) предполагает применение административных и технических процедур на всем протяжении жизненного цикла ПП.

Согласно стандарту IEEE-90 под конфигурацией прогроммного продукта понимается совокупность его функциональных и физических характеристик, установленных в технической документации и реализованных в ПП.

Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в ПП на всех стадиях жизненного цикла. Общие принципы и рекомендации по управлению конфигурацией ПП отражены в стандарте ISO/IEC CD 12207-2: 1995 «Information Technology – Software Life Cycle Processes. Part 2. Configuration Management for Software» – Информационные технологии – Процессы жизненного цикла программ. Часть 2. Управление конфигурацией программ».

Процесс управления конфигурацией включает в себя подготовительную работу, идентификацию конфигурации, контроль конфигурации, учет состояния конфигурации, оценку конфигурации; управление выпуском и поставку [3].

Процесс обеспечения качества (quality assurance process) обеспечивает соответствующие гарантии того, что ПП и процессы его жизненного цикла соответствуют заданным требованиям и утвержденным планам. Под качеством ПП понимается совокупность свойств, которые характеризуют способность ПП удовлетворять заданным требованиям. Для получения достоверных оценок создаваемого ПП процесс обеспечения его качества должен происходить независимо от субъектов, непосредственно связанных с разработкой ПП. При этом могут использоваться результаты других вспомогательных процессов, таких как верификация, аттестация, совместная оценка, аудит и разрешение проблем.

Процесс обеспечения качества включает в себя подготовительную работу; обеспечение качества продукта; обеспечение качества процесса; обеспечение прочих показателей качества системы [3].

Обеспечение прочих показателей качества системы осуществляется в соответствии с условиями договора и стандартом качества 180-9001.

Процесс верификации (verification process) состоит в доказательстве того, что ПП, являющиеся результатами некоторого действия, полностью удовлетворяют требованиям или условиям, зависящим от предшествующих действий.

Верификация может проводиться самим исполнителем или другим специалистом данной организации, а также специалистом сторонней организации. Тут возможны различные вариации, в соответствии с которыми можно говорить о различной степени независимости верификации. Если процесс верификации осуществляется организацией, не зависящей от поставщика, разработчика, оператора или службы сопровождения, то он называется процессом независимой верификации.

Процесс верификации включает в себя подготовительную работу и собственно верификацию. Верификация в узком смысле означает формальное доказательство правильности ПП. Данный процесс может включать в себя анализ, оценку и тестирование.

В процессе верификации проверяются непротиворечивость требований к системе и степень учета потребностей пользователей; возможности поставщика выполнить заданные требования; соответствие выбранных процессов жизненного цикла ПП условиям договора; адекватность стандартов, процедур и среды разработки процессам жизненного цикла ПП; соответствие проектных спецификаций ПП 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 ПП в систему; адекватность, полнота и непротиворечивость документации.

Процесс аттестации (validation process) предусматривает определение полноты соответствия заданных требований к создаваемой системе или ПП функциональному назначению последних. Под аттестацией обычно понимают подтверждение и оценку достоверности проведенного тестирования ПП. Аттестация должна гарантировать полное соответствие ПП спецификациям, требованиям и документации, а также возможность его безопасного и надежного применения пользователем. Аттестацию рекомендуется выполнять путем тестирования во всех возможных ситуациях и использовать при этом независимых специалистов. Аттестация так же, как и верификация, может осуществляться с различными степенями независимости.

Процесс аттестации включает в себя подготовительную работу и собственно аттестацию.

Аттестация позволяет определить полноту соответствия разработанных требований к создаваемому ПП или системе функциональному назначению последних.

Процесс совместной оценки (joint review process) предназначен для оценки состояния работ по проекту и ПП, создаваемому при выполнении данных работ. Он заключается в основном в контроле за планированием и управлением ресурсами, персоналом, аппаратурой и инструментальными средствами проекта.

Оценка выполняется как на уровне управления проектом, так и на уровне его технической реализации и проводится в течение всего срока действия договора. Данный процесс может выполняться двумя любыми сторонами, участвующими в договоре, при этом одна сторона проверяет другую.

Процесс совместной оценки включает в себя подготовительную работу; оценку управления проектом; техническую оценку.

Процесс аудита (audit process) представляет собой определение соответствия требованиям, планам и условиям договора как хода выполнения работ по созданию ПП, так и самого ПП. Аудит может выполняться двумя любыми сторонами, участвующими в договоре, когда одна сторона проверяет другую.

Аудит служит для установления соответствия реальных работ и отчетов требованиям, планам и контракту. Аудиторы (ревизоры) не должны иметь прямой зависимости от разработчиков ПП. Они оценивают состояние работ, использование ресурсов, соответствие документации спецификациям и стандартам, корректность проводимого тестирования. Процесс аудита включает в себя подготовительную работу и собственно аудит [3].

Процесс разрешения проблем (problem resolution process) предусматривает анализ и решение проблем (включая выявленные несоответствия), обнаруженных в ходе разработки, эксплуатации, сопровождения и других процессов, независимо от их происхождения или источника. Каждая обнаруженная проблема должна быть идентифицирована, описана, проанализирована и разрешена.

Процесс разрешения проблем включает в себя подготовительную работу и собственно разрешение проблем.

Организационные процессы жизненного цикла программного продукта

Основной целью организационных процессов является организация процесса разработки надежного, полностью удовлетворяющего требованиям заказчика ПП, к установленным договором относятся процессы управления, создания инфраструктуры, усовершенствования, обучения [3].

Процесс управления (management process) состоит из действий и задач, которые могут выполняться любой стороной, управляющей своими процессами. Данная сторона (менеджер) отвечает за управление выпуском продукта, проектом и задачами соответствующих процессов, таких как приобретение, поставка, разработка, эксплуатация, сопровождение и др.

Процесс управления включает в себя инициирование и определение области управления; планирование; управление работами по созданию ПП и контроль за их выполнением; проверку и оценку; завершение работ.

Процесс создания инфраструктуры (infrastructure process) охватывает выбор и поддержку (сопровождение) технологии, стандартов и инструментальных средств, выбор и установку аппаратных и программных средств, используемых для разработки, эксплуатации или сопровождения ПП. Инфраструктура должна модифицироваться и сопровождаться в соответствии с изменениями требований к соответствующим процессам. Она является одним из объектов управления конфигурацией.

Процесс усовершенствования (improvement process) предусматривает оценку, измерение, контроль и усовершенствование процессов жизненного цикла ПП. Данный процесс включает в себя создание процесса; оценку процесса; усовершенствование процессов жизненного цикла ПП.

Процесс обучения (training process) охватывает первоначальное обучение и последующее постоянное повышение квалификации персонала. Приобретение, поставка, разработка, эксплуатация и сопровождение программного продукта в значительной степени зависят от уровня знаний и квалификации персонала. Например, разработчики ПП должны пройти необходимое обучение методам и средствам программной инженерии. Содержание процесса обучения определяется требованиями к проекту. Для этого процесса должны быть запланированы необходимые ресурсы и технические средства обучения, кроме того должны быть разработаны и представлены методические материалы, необходимые для обучения пользователей в соответствии с учебным планом.

Тема №6. Технология разработки программного обеспечения

Как известно, первые программы появились одновременно с первыми компьютерами в конце 40-х – начале 50-х годов XX века. И писались они в те времена теми же людьми, что создавали первые компьютеры, т.е. физиками, математиками, электронщиками и другими учеными-исследователями, открывшими тогда для себя новую отрасль науки – информатику. Эти программы, как правило, носили научно-исследовательский характер и использовались самими же их авторами. Никто тогда не задумывался над вопросами: «За какие сроки эти программы написаны? Сколько средств затрачено на их изготовление? Наконец, насколько качественно они написаны, т.е. удачен ли их интерфейс, содержат ли они ошибки?».

Однако прошло каких-то 10–15 лет и компьютеры из научных лабораторий пришли в промышленность, медицину, военную технику. Теперь уже эксплуатировать компьютеры и установленное на них программное обеспечение стали не те, кто их разрабатывал. Т. е. компьютеры и программы стали обычным продуктом. В это время программное обеспечение (ПО) уже стали разрабатывать специальные фирмы или специальные подразделения многопрофильных компьютерных фирм сначала на заказ и в основном для военных и для космоса, а потом, с появлением и распространением персональных компьютеров, ПО стали производить и просто на продажу.

В середине 60-х годов некоторые исследователи заметили, что процесс создания ПО обычно не хотел укладываться в запланированные сроки и бюджет, а качество продукта, как правило, оставляло желать лучшего. Эти проблемы программной индустрии тогда же получили

интегральное название «software crisis», т.е. в переводе на русский – кризис программного обеспечения, или кризис программной индустрии.

Например, процент успешно завершаемых в отведенные для них сроки проектов, в машиностроении или архитектуре обычно превышает 90 %, а аварии или катастрофы из-за ошибок в проектировании таких объектов единичны. Что же касается программирования, то тут ситуация диаметрально противоположная. По данным Пентагона о своих подрядчиках, лишь 17 % всех проектов по разработке ПО укладываются в срок и рамки бюджета, остальные проекты превышают бюджет в среднем на 189 %, сроки – на 222 %, а реализуется в них всего 61 % требуемых заказчиком возможностей.

Если же говорить о качестве ПО, т.е. в первую очередь о его надежности и безопасности, то и тут выяснилось, что оно обычно значительно уступает соответствующим показателям аппаратного обеспечения. Например, сбои в программных системах управления космическими аппаратами НАСА или Российского Космического Агентства (РКА) приводят к многомиллионным убыткам (авариям на станции «Мир» или к потере Вояджеров, направленных к Марсу). В этой связи широкую известность получил случай крушения ракеты «Ариан 5» Европейского Космического Агентства, который произошел 4 июня 1996 г., и, как выяснилось позднее, причиной его было именно некачественное программное обеспечение. Совокупные убытки от данной аварии оценивались примерно в 500 млн долларов. Часто ценой ошибки в программном обеспечении становится человеческая жизнь. Наконец, недаром одно только недоверие к качеству разработанного ранее программного обеспечения породило так называемую «Проблему 2000 г.», из-за раздувания ажиотажа вокруг которой многие фирмы понесли значительные убытки, а некоторые заработали.

Под термином «программирование» в таком случае естественно понимать процесс создания программы на конкретном алгоритмическом языке (подчеркнем, только программы, а не программного средства и тем более не программного продукта). Тогда, с учетом обычного толкования термина, *технология* – это набор правил, методик, инструментов, позволяющих наладить производственный процесс выпуска какого-либо продукта, включая процессы планирования, измерения, оценки качества, ответственность исполнителя и многое другое). Под *технологией разработки программного обеспечения* следует понимать весь комплекс мер экономического, организационного и технического характера, направленных на получение качественного программного продукта в установленные сроки с наименьшими затратами, т. е. это, по сути дела, и есть аналог английского термина Software Engineering.

Таким образом, под термином *технология программирования* в данном контексте естественно понимать технологические аспекты процессов непосредственного создания самих

программ, т. е. технология разработки программного обеспечения в этом случае будет более широким понятием и как часть будет включать в себя технологию программирования.

Достаточно часто в технической литературе встречаются также термины «методология программирования» и «парадигма программирования». Эти понятия по смыслу достаточно близки между собой и обозначают основную идею (парадигму) или методы, лежащие в основе тех или иных способов разработки программ, например, идеи объектно-ориентированного, функционального, логического, структурного, визуального и т. п. программирования. Очевидно, эти понятия будут составными частями более широких понятий: *технологии программирования* и, следовательно, *технологии разработки программного обеспечения*.

Тема №7. Общие принципы моделирования жизненного цикла программных средств. Модели ЖЦ ПС

Ранее уже встречалось понятие «модели жизненного цикла программного средства», когда при адаптации стандарта ГОСТ Р ИСО/МЭК 12207 к конкретному проекту предлагалось задаться такой моделью.

Модель жизненного цикла включает

– интеграцию и тестирование ПС;

– системное тестирование;

– инсталляцию и аттестационное тестирование (приемо-сдаточные испытания).

Ответственность является особенностью структуры ЖЦ применительно к условиям проекта, в который закономерно может быть вовлечено множество субъектов. Безусловно, применение ГОСТ Р ИСО/МЭК 12207 требует от соответствующих субъектов определенных усилий по его адаптации к условиям реализации конкретных проектов. Тем не менее, можно с уверенностью полагать, что внедрение данного стандарта в практическую деятельность должно облегчить упорядочение взаимоотношений между субъектами, вовлеченными в ЖЦ ПС.

Понятие модели жизненного цикла разработки программного продукта

Под моделью жизненного цикла разработки ПП понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении жизненного цикла разработки ПП. Модель жизненного цикла зависит от специфики и сложности выполняемого проекта, а также от условий, в которых создается и будет функционировать ПП.

Стандарт ISO/IEC 12207 не предлагает конкретные методы разработки и модель жизненного цикла ПП. Положения стандарта являются общими для любых моделей жизненного

цикла, методов и технологий разработки ПП. Стандарт описывает структуру процессов жизненного цикла ПП, но не уточняет, как выполнить действия и задачи, включенные в эти процессы.

Модель жизненного цикла любого конкретного ПП определяет характер процесса его создания, который представляет собой совокупность упорядоченных во времени, взаимосвязанных и объединенных в этапы работ, выполнение которых необходимо и достаточно для создания ПП, соответствующего заданным требованиям. Наибольшее распространение получили следующие модели жизненного цикла разработки ПП: каскадная модель, или «водопад» (Waterfall model); V-образная модель (V-shaped model); модель прототипирования (Prototype model); модель быстрой разработки приложений, или RAD-модель (RAD – Rapid Application Development model); многопроходная модель (Incremental model); спиральная модель (Spiral model).

Краткие характеристики каждой из перечисленных моделей приведены в таблице 5.1.

Таблица 5.1

Модели жизненного цикла разработки программного продукта

Название	Характеристики
1	2
Каскадная модель	Прямолинейная и простая в использовании. Необходим постоянный жесткий контроль за ходом работы. Разрабатываемое программное обеспечение не доступно для изменений
V-образная модель	Простая в использовании. Особое значение придается тестированию и сравнению результатов фаз тестирования и проектирования

Окончание табл. 5.1

1	2
Модель прототипирования	Создается «быстрая» частичная реализация системы до составления окончательных требований. Обеспечивается обратная связь между пользователями и разработчиками в процессе выполнения проекта.

	Используемые требования не полные
Модель быстрой разработки приложений	Проектные группы небольшие (3–7 человек) и составлены из высококвалифицированных специалистов. Уменьшенное время цикла разработки (до 3 мес.) и улучшенная производительность. Повторное использованис кода и автоматизация процесса разработки
Многопроходная модель	Быстро создается работающая система. Уменьшается возможность внесения изменений в процессе разработки. Невозможен переход от текущей реализации к новой версии в течение построения текущей частичной реализации
Спиральная модель	Охватывает каскадную модель. Расчленяет фазы на меньшие части. Позволяет гибко выполнять проекгирование. Анализирует риски и управляет ими. Пользователи знакомятся с ПП на более раннем этапе благодаря прототипам.

Классическая каскадная, или «водопадная» модель

В однородных информационных системах 1970-х и 1980-х годов прикладные ПП представляли собой единое целое. Для разработки такого типа ПП применялась классическая каскадная, или «водопадная» модель ЖЦ ПС (по-английски waterfall model, рисунок 5.1). Создана по образу и подобию методик, наработанных в других инженерных областях, где существует стандартная практика поэтапного создания продукта, начиная с составления технического задания (спецификаций) и заканчивая поставкой заказчику. Такая модель реализует, по сути, принцип однократного выполнения каждого вида деятельности в виде заранее ограниченных и однозначно упорядоченных во времени стадий, этапов или фаз проекта, осуществляемых как бы в их естественных границах, например, как показано на рисунке 5.1.

Преимущества каскадного способа: на каждой стадии формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности; выполняемые в логичной последовательности стадии работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

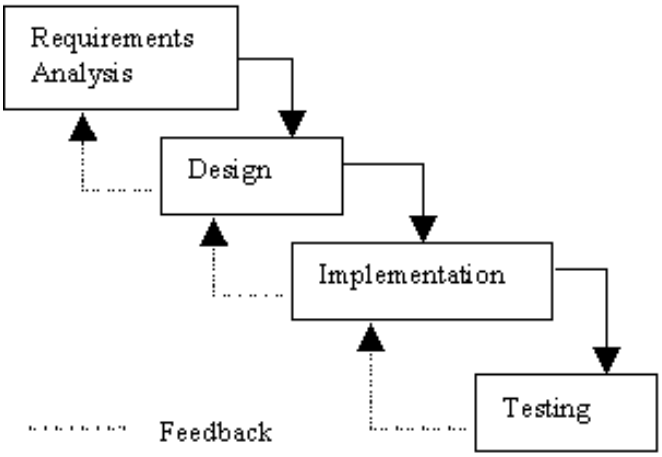


Рис. 5.1. Каскадная модель

Каскадный подход хорошо зарекомендовал себя при построении информационных систем, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования с целью предоставить разработчикам свободу реализовать их технически как можно лучше.
В эту категорию попадают сложные системы с большим числом задач вычислительного характера, системы реального времени и др.

В то же время данный подход обладает рядом существенных недостатков, обусловленных прежде всего тем, что реальный процесс разработки ПП никогда не укладывается и такую жесткую схему. Этот процесс носит, как правило, итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних стадиях. Таким образом, постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс разработки принимает иной вид (см. рис. 5.2).

Модифицированная каскадная, или модель «водоворота»

Более реальной и близкой к практике программирования представляется модифицированная каскадная модель, или модель «водоворота», изображенная на рисунке 5.2.

В отличие от классической, данная модель допускает параллельное выполнение отдельных работ (их перекрытие), а также возвраты назад, в том числе и на несколько фаз, в случае обнаружения ошибок. Таким образом, в данных моделях есть место проверкам и аттестации.

Заметим, что процессы сопровождения и эксплуатации обычно реализуют после процесса разработки. Процессы же заказа, поставки, а также вспомогательные и организационные процессы обычно выполняют параллельно с процессом разработки.

Данная модель ЖЦ стандартизована Министерством обороны США (DOD STD 2167A) для ПС военного назначения и является документально управляемой, т.е. каждая фаза такой

модели считается окончательно завершенной только в том случае, если оформлены все оговоренные для нее стандартом документы.

Модифицированная каскадная модель ЖЦ является наиболее подходящей для промышленной разработки относительно больших ПС с заранее четко определенными функциями и требованиями к качеству. Такая ситуация обычно имеет место при разработке систем военного назначения, аэрокосмических систем или систем управления технологическими процессами в реальном времени, а также других критических систем. Однако и у такой модели ЖЦ есть свои недостатки, проявляющиеся особенно ярко при разработке ПС в условиях неопределенности исходных требований, что часто имеет место при проектировании информационных систем, например, экономического характера. В такой ситуации огромное значение приобретает этап формулирования требований, составления спецификаций и создания плана проекта. Системные аналитики несут личную ответственность за все последующие изменения проектных решений. Поэтому объем документации исчисляется тысячами страниц, а количество утверждающих заседаний, отнимающих рабочее время многих людей, просто огромно. И очень важным становится момент принятия окончательного решения о передаче проекта в разработку, потому что необходимость что-либо изменить впоследствии может оказаться фатальной для чьей-нибудь карьеры. Именно по этой причине многие проекты при использовании каскадной модели ЖЦ ПС так никогда и не покинули фазу планирования, впав в так называемый «паралич анализа» [3].

Модель «сделал-исправил»

В англоязычных источниках такую модель называют Build and Fix Model, или Ad-Hoc model. Это самая простая и, наверное, самая распространенная модель ЖЦ ПС (особенно среди студентов). Иногда ее даже не считают за модель, называя такой способ разработки ПС кустарным или программированием «на коленке». Графически данная модель представлена на рисунке 5.3.

Таким образом, при кустарной разработке, минуя фазы подробного анализа требований пользователя, составления формальных и документальных спецификаций, а также проектирования, обычно сразу приступают к программированию и создают первую версию ПС, которую затем модифицируют до тех пор, пока она не удовлетворит пользователя.

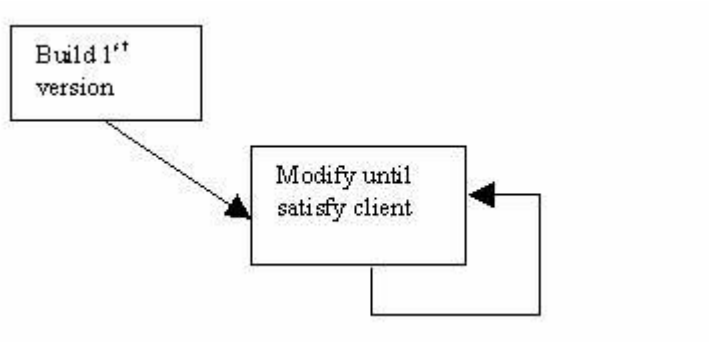


Рис. 5.2. Модель «сделал-исправил»

Такая схема достаточно хорошо работает при изготовлении очень маленьких и простых ПС, например, при программировании на лабораторном практикуме или в ходе курсового проектирования.

Прототипирование

Модель прототипирования (рис. 5.3) позволяет создать прототип ПП до или в течение этапа составления требований к ПП.

Рис. 5.3. Модель прототипирования

Потенциальные пользователи работают с этим прототипом, определяя его сильные и слабые стороны, и о результатах сообщают разработчикам ПП. Таким образом, обеспечивается обратная связь между пользователями и разработчиками, которая используется для изменения или корректировки спецификации требований к ПП. В результате такой работы продукт будет отражать реальные потребности пользователей.

Модель прототипирования обладает целым рядом преимуществ:

– взаимодействие заказчика с разрабатываемой системой начинается на раннем этапе;

– благодаря реакции заказчика на прототип сводится к минимуму число неточностей в требованиях;

– снижается вероятность возникновения путаницы, искажения информации или недоразумений при определении требований к ПП, что приводит к созданию более качественного ПП;

– в процессе разработки всегда можно учесть новые, даже неожиданные требования заказчика;

– прототип представляет собой формальную спецификацию, воплощенную в ПП;

– прототип позволяет очень гибко выполнять проектирование и разработку, включая несколько итераций на всех фазах жизненного цикла разработки;

– заказчик всегда видит прогресс в процессе разработки ПП;

– возможность возникновения противоречий между разработчиками и заказчиками сведена к минимуму;

– уменьшается число доработок, что снижает стоимость разработки:

– возникающие проблемы решаются на ранних стадиях, что резко сокращает расходы на их устранение;

– заказчики принимают участие в процессе разработки на протяжении всего жизненного цикла и в конечном итоге в большей степени довольны результатами работы.

Кроме указанных достоинств модели прототипирования присутсвуют и целый ряд недостатков: решение сложных задач может отодвигаться на будущее, заказчик может предпочесть получить прототип, а не законченную полную версию ПП, прототипирование может неоправданно затянуться, перед началом работы неизвестно, сколько итераций придется выполнить.

Модель прототипирования рекомендуется применять в следующих случаях: требования к ПП заранее неизвестны; требования не постоянны или неудачно сформулированы; требования необходимо уточнить; нужна проверка концепции; существует потребность в пользовательском интерфейсе; выполняется новая, не имеющая аналогов разработка; разработчики не уверены в том, какое решение следует выбрать.

Спиральная модель ЖЦ ПС

При более детальном уровне рассмотрения ЖЦ отдельно могут быть выделены итерационные циклические формы, основанные на технике макетирования систем и называемые спиральными моделями Б. Боэма. Графически такая модель представлена на рисунке 5.4.

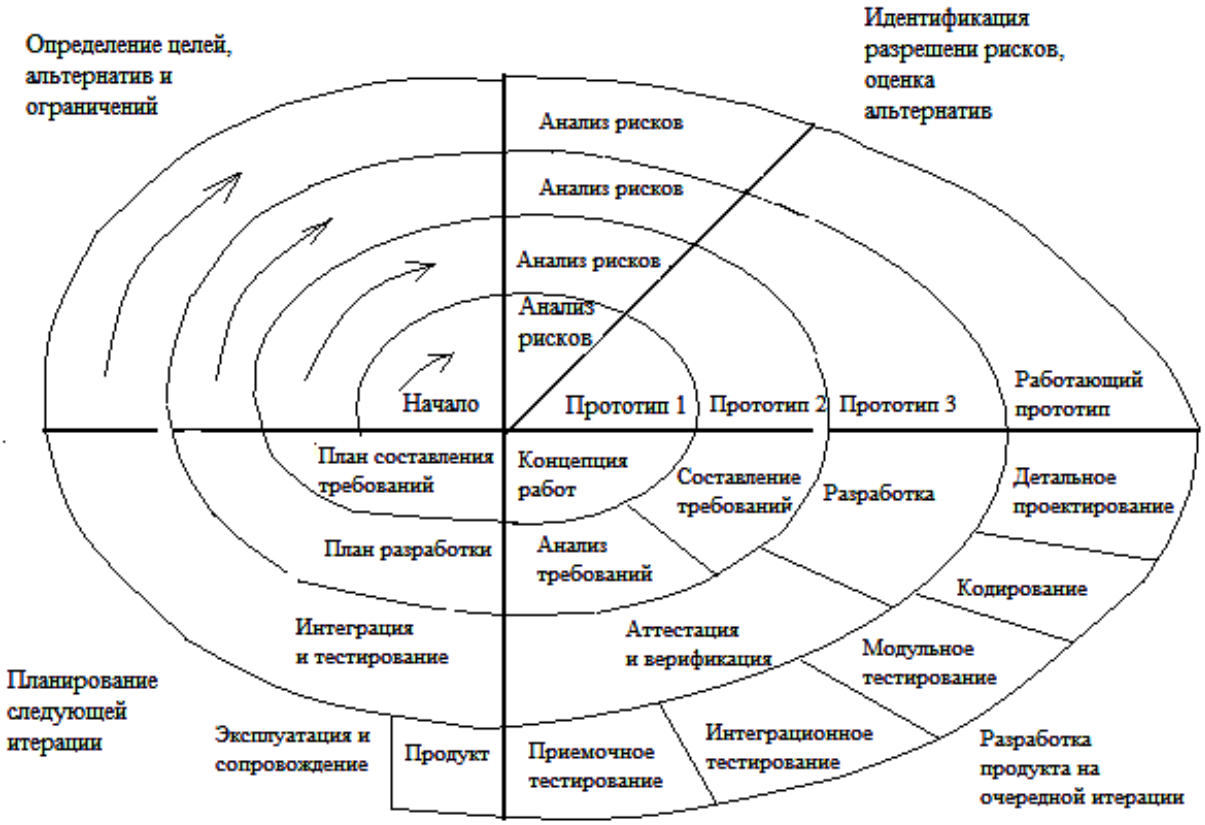


Рис. 5.4. Спиральная модель

При этом весь ЖЦ разбивается на 4 циклически повторяемых основных этапа: определение целей, альтернатив и ограничений, оценка альтернатив и связанных с их принятием рисков, разработка и тестирование, а также планирование дальнейших работ. Такая модель ЖЦ последовательностью основных процессов напоминает классическую каскадную модель. Однако есть и важнейшее отличие, заключающееся в том, что перед каждым из основных процессов ЖЦ, таких как разработка концепции ПС, разработка требований, а также проектирование и программирование с соответствующими проверками и тестированием (имевших место и в каскадной модели), всегда выполняются процедуры определения целей, альтернатив и ограничений, а также оценки альтернатив и связанных с их принятием рисков. После же выполнения любых процедур разработки и тестирования всегда выполняется планирование дальнейших работ. Причем основной упор в данной модели делается именно на процедурах определения целей, альтернатив и ограничений, оценки альтернатив и связанных с их принятием рисков, а также планировании дальнейших работ, которые на каждом новом витке спирали выполняются на все более и более высоком уровне, постоянно усложняясь [1, 2].

Спиральная модель обладает следующими достоинствами: заказчик имеет возможность увидеть разрабатываемый ПП на ранних стадиях разработки; заказчики принимают активное

участие в разработке ПП; в модели воплощены преимущества каскадной и многопроходной моделей.

Недостатки спиральной модели: усложненная структура; спираль может продолжаться до бесконечности, так как каждая ответная реакция заказчика может породить новый цикл.

В качестве модели жизненного цикла разработки программного продукта большое распространение получила улучшенная спиральная модель, показанная на рисунке 5.5. В отличие от ранее рассмотренной спиральной модели эта модель использует каскадный подход на завершающих этапах разработки ПП. Использование спиральной модели целесообразно, если существует хотя бы одна из следующих причин: целесообразно создание прототипа; организация обладает навыками, требуемыми для адаптации модели; требуется выполнять проекты со средней и высокой степенями риска; заказчик не уверен в своих требованиях; требования слишком сложные; проект очень большой.

Рис. 5.5. Улучшенная спиральная модель с указанием вспомогательных процессов

Другие модели ЖЦ ПС

Надо сказать, что приведенными выше типами моделей ЖЦ все их разнообразие не исчерпывается. Существует еще несколько общеизвестных, но относительно редко применяемых модификаций моделей ЖЦ ПС, а также большое количество частно - фирменных моделей с разной степенью детализации процессов.

От каскадной модели V-образная модель унаследовала последовательную структуру, в соответствии с которой каждая последующая фаза начинается только после успешного завершения фазы предыдущей. Данная модель основана на систематическом подходе к проблеме, для решения которой определены четыре базовых шага: анализ, проектирование, разработка и обзор. При выполнении анализа осуществляются планирование проекта и составление требований. Проектирование разделяется на высокоуровневое и детальное (низкоуровневое). Разработка включает в себя кодирование, а обзор – различные виды тестирования.

На модели хорошо просматриваются взаимосвязи между аналитическими фазами и фазами проектирования, которые предшествуют кодированию и тестированию. Модель включает в себя следующие фазы: составление требований к проекту и планирование; составление требований к продукту и их анализ; высокоуровневое проектирование; детальное проектирование; модульное тестирование; интеграционное тестирование; системное тестирование; эксплуатация и сопровождение.

На этой фазе в ПП могут вноситься поправки и может выполняться его модернизация.

Преимущества V-образной модели: большая роль придается верификации и аттестации ПП, начиная с ранних стадий его разработки, все действия планируются; предполагаются аттестация и верификация не только самого ПП, но и всех полученных внутренних и внешних данных; ход выполнения работы может легко отслеживаться, так как завершение каждой фазы является контрольной точкой.

Кроме перечисленных достоинств модель обладает и рядом недостатков: не учитываются итерации между фазами; нельзя вносить изменения на разных этапах жизненного цикла; тестирование требований происходит слишком поздно, поэтому внесение изменений влияет на выполнение графика работ.

Данную модель целесообразно использовать при разработке программных продуктов, главным требованием для которых является высокая надежность.

Например, при разработке сложных компьютерных систем иногда используют так называемую V-модель, которая, в общем, является разновидностью каскадной модели ЖЦ ПС. Однако в такой модели каждому процессу анализа или разработки на каждом уровне абстрагирования поставлен в соответствие процесс интеграции или тестирования, как это показано на рисунке 5.6. В результате внешний вид модели напоминает латинскую букву V, за что она и получила свое название.

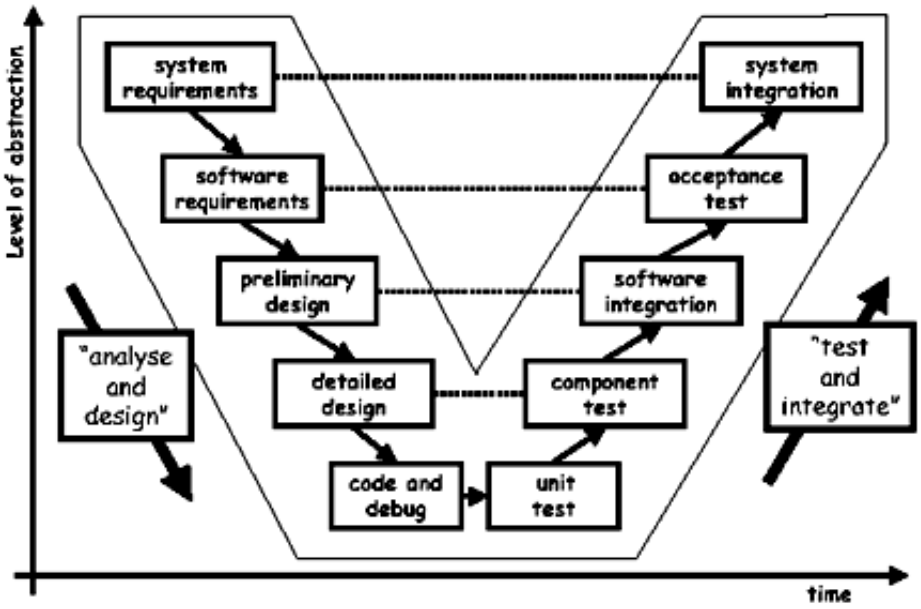


Рис. 5.6. V-модель

Некоторые особенности в моделировании ЖЦ ПС могут быть связаны и с используемым методом проектирования, например, объектно-ориентированным подходом. Однако эти особенности, как правило, не носят глобального характера и не затрагивают основных этапов и последовательности их выполнения в известных моделях ЖЦ [1, 2].

Тема №8. Проектирование программного продукта. Структурное программирование и объектно-ориентированное проектирование

Появление первых ЭВМ ознаменовало новый этап в развитии техники вычислений. Возникла идея, что достаточно разработать последовательность элементарных действий, каждое из которых преобразовать в понятные ЭВМ инструкции, и любая вычислительная задача может быть решена. Этот подход оказался настолько жизнеспособным, что долгое время доминировал над всеми другими в процессе разработки программ. Появились специальные языки программирования, которые позволили преобразовывать отдельные вычислительные операции в соответствующий программный код.

Основой данной методологии разработки программ стала процедурная, или алгоритмическая организация структуры программного кода. Это было настолько естественно для решения вычислительных задач, что ни у кого не вызывала сомнений целесообразность такого подхода. Исходным в этой методологии являлось понятие «алгоритм», под которым в общем случае подразумевалось некоторое предписание выполнить точно определенную последовательность действий, направленных на достижение заданной цели или решение поставленной задачи.

С этой точки зрения вся история математики тесно связана с разработкой тех или иных алгоритмов решения актуальных для своей эпохи задач. Более того, само понятие «алгоритм» стало предметом соответствующей теории – теории алгоритмов, которая занимается изучением их общих свойств. Со временем содержание этой теории стало настолько абстрактным, что соответствующие результаты понимали только специалисты. Как дань этой традиции какой-то период времени языки программирования назывались алгоритмическими, а первое графическое средство документирования программ получило название «блок-схема алгоритма». Соответствующая система графических обозначений была зафиксирована в ГОСТ 19.701–90, который регламентировал использование условных обозначений в схемах алгоритмов, программ, данных и систем.

Однако потребности практики не всегда требовали установления вычислимости конкретных функций или разрешимости отдельных задач.
В языках программирования возникло и закрепилось новое понятие – «процедура», которое конкретизировало общее понятие «алгоритм» применительно к решению задач на компьютерах. Так же, как и алгоритм, процедура представляет собой законченную последовательность действий или операций, направленных на решение отдельной задачи. В языках программирования появилась специальная синтаксическая конструкция, которая получила название «процедура».

Со временем разработка больших программ превратилась в серьезную проблему и потребовала их разбиения на более мелкие фрагменты. Основой для такого разбиения и стала процедурная декомпозиция, при которой отдельные части программы, или модули, представляли собой совокупность процедур для решения некоторой совокупности задач. Главная особенность процедурного программирования заключается в том, что программа всегда имеет начало во времени, или начальную процедуру (начальный блок), и окончание (конечный блок). При этом вся программа может быть представлена визуально в виде направленной последовательности графических примитивов, или блоков.

Важным свойством таких программ является необходимость завершения всех действий предшествующей процедуры для начала действий последующей процедуры. Изменение порядка выполнения этих действий даже в пределах одной процедуры потребовало включения в языки программирования специальных условных операторов типа if–then–else и goto для реализации ветвления вычислительного процесса в зависимости от промежуточных результатов решения задачи.

Появление и интенсивное использование условных операторов и оператора безусловного перехода стало предметом острых дискуссий среди специалистов по программированию. Дело в том, что бесконтрольное применение в программе оператора

безусловного перехода goto способно серьезно осложнить понимание кода. Соответствующие программы стали сравнивать со спагетти, называя их bowl of spaglletti, имея в виду многочисленные переходы от одного фрагмента программы к другому или, что еще хуже, возврат от конечных операторов программы к ее начальным операторам.

Ситуация казалась настолько драматичной, что в литературе зазвучали призывы исключить оператор goto из языков программирования. Именно с этого времени принято считать хорошим стилем программирование без оператора goto.

Рассмотренные идеи способствовали становлению некоторой системы взглядов на процесс разработки программ и написания программных кодов, которая получила название «методология структурного программирования». Основой данной методологии является процедурная декомпозиция программной системы и организация отдельных модулей в виде совокупности выполняемых процедур. В рамках данной методологии получило развитие нисходящее проектирование программ, или программирование «сверху вниз». Период наибольшей популярности идей структурного программирования приходится наконец 1970-х – начало 1980-х годов.

В качестве вспомогательного средства структуризации программного кода было рекомендовано использование отступов в начале каждой строки, которые должны выделять вложенные циклы и условные операторы. Все это призвано способствовать пониманию или читабельности самой программы. Данное правило со временем было реализовано в современных инструментариях разработки программ.

6.4. Объектно-ориентированное проектирование

Основными понятиями объектно-ориентированного подхода являются «объект» и «класс». Объект определяется как осязаемая реальность (tangible entity). Объект характеризуется состоянием, поведением и индивидуальностью; структура и поведение схожих объектов определяют общий для них класс. Термины «экземпляр класса» и «объект» являются эквивалентными. Состояние объекта характеризуется перечнем всех возможных (статических) свойств данного объекта и текущими (динамическими) значениями каждого из этих свойств. Поведение объекта полностью определяется его действиями. Определенное воздействие одного объекта на другой с целью вызвать соответствующую реакцию называется операцией. Как правило, в объектных и объектно-ориентированных языках операции, выполняемые над данным объектом, называются методами и являются составной частью определения класса. Класс – это множество объектов, связанных общностью структуры и поведения. Любой объект является экземпляром класса. Следующую группу важных понятий объектно-ориентированного подхода составляют наследование и полиморфизм. Полиморфизм можно интерпретировать как способность класса принадлежать более чем одному типу. Наследование означает построение

новых классов на основе существующих с возможностью добавления или переопределения данных и методов.

Объектно-ориентированная система изначально строится с учетом ее эволюции. Наследование и полиморфизм обеспечивают возможность определения новой функциональности классов с помощью создания производных классов – потомков. Потомки наследуют характеристики родительских классов без изменения их первоначального описания и добавляют при необходимости собственные структуры данных и методы. Определение производных классов, при котором задаются только различия или уточнения, в огромной степени экономит время и усилия при производстве и использовании спецификаций и программного кода.

Понятие «объект» впервые было использовано около 30 лет назад при попытках отойти от традиционной архитектуры Фон Неймана и преодолеть барьер между высоким уровнем программных абстракций и низким уровнем абстрагирования. С объектно-ориентированной архитектурой также тесно связаны объектноориентированные операционные системы. Наиболее значительный вклад в объектный подход был внесен объектными и объектно-ориентированными языками программирования Simula, Smalltalk, C++, ObjectPascal. На объектный подход оказали влияние также развивавшиеся достаточно независимо методы моделирования баз данных, в особенности подход «сущность – связь».

Концептуальной основой объектно-ориентированного подхода служит объектная модель. Основными ее элементами являются абстрагирование (abstraction), инкапсуляция (encapsulation), модульность (modularity), иерархия (hierarchy).

Кроме основных имеются еще три дополнительных элемента: типизация (typing), параллелизм (concurrency), устойчивость (persistence) – не являюшихся, в отличие от основных, строго обязательными.

Важным качеством объектно-ориентированного подхода является согласованность моделей деятельности организации и моделей проектируемой системы, начиная со стадии формирования требований и заканчивая стадией реализации. Требование согласованности моделей выполняется благодаря возможности применения абстрагирования, модульности, полиморфизма на всех стадиях разработки. Модели ранних стадий могут быть непосредственно подвергнуты сравнению с моделями реализации. По объектным моделям может быть прослежено отображение реальных сущностей моделируемой предметной области (организации) в объекты и классы информационной системы.

Большинство существующих методов объектно-ориентированного анализа и проектирования (ООАП) включает в себя как язык моделирования, так и описание процесса моделирования. Язык моделирования – это нотация (в основном графическая), которая

используется методом для описания проектов. Графическая нотация представляет собой совокупность графических объектов, которые используются в моделях; она является синтаксисом языка моделирования. Например, нотация диаграммы классов определяет, каким образом представляются такие элементы и понятия, как класс, ассоциация и множественность. Процесс – это описание шагов, которые необходимо выполнить при разработке проекга.

Унифицированный язык моделирования UML (Unified Modelling Language) – это преемник того поколения методов ООАП, которые появились в конце 1980-х – начале 1990-х годов. Создание UML фактически началось в конце 1994 г., когда Гради Буч и Джеймс Рамбо приступили к работе по объединению методов Booch и OMT (Object Modeling Technique) с помощью А. Джекобсона и под эгидой компании Rational Software. К концу 1995 г. они создали первую спецификацию объединенного метода, названного ими Unified Method, версия 0.8. Тогда же, в 1995 г., к ним присоединился создатель метода OOSE (Object-Oriented Software Engineering) Ивар Якобсон. Таким образом, UML является прямым объединением и унификацией методов Г. Буча, Д. Рамбо, А. Джекобсона и И. Якобсона. Язык UML принят на вооружение практически всеми крупнейшими компаниями – производителями ПП (Microsoft, IBM, Hewlett-Packard, Oracle, Sybase и др.). Создатели UML представляют его как язык для определения, представления, проектирования и документирования программных, организационно-экономических, технических и других систем.

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Основная литература

1. А.М.Минитаева Разработка и стандартизация программных средств и информационных технологий, ОМСК, 2011

2. Липаев В.В. Проектирование программных средств. Учебное пособие. – М.: Высшая школа, 1990. – 303 с. (86 экз.)

3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. – М.: Финансы и статистика, 2002. – 349 с. (30 экз.)

4. Миньков С.Л. Разработка и применение ППП в экономике: Учебное пособие. – Томск: ТМЦДО, 2002. – 231 с. (100 экз.)

5. Миньков С.Л. Пользовательский интерфейс: Методические указания к выполнению лабораторных работ. – Томск: ТУСУР, 2006. – 48 с. (50 экз.)

Дополнительная литература

1. Стандартизация разработки программных средств: Учебное пособие / В.А Благодатских, В.А. Волнин, К.Ф. Поскакалов; Под ред О.С. Разумова. – М.: Финансы и статистика, 2003. – 288 с.

2. Экономика, разработка и использование программного обеспечения ЭВМ. Учебное пособие. / В.А. Благодатских, М.А. Енгибарян, Е.В. Ковалевская и др. - М.: Финансы и статистика, 1995. - 288 с.

3. Информационные системы в экономике: Учебник / Под ред. В.В. Дика. – М.: Финансы и статистика, 1996. – 272 с.

4. Липаев В.В. Документирование и управление конфигурацией программных средств. Методы и стандарты. - М.: СИНТЕГ, 1998. – 220 с.

5. Липаев В.В. Надежность программных средств. – М.: СИНТЕГ, 1998. – 232 с.

6. Характеристики качества программного обеспечения / Боэм Б., Браун Дж., Каспар Х. и др. – М.: Мир, 1981. – 206 с.

7. Единая система программной документации // М.: Изд-во стандартов, 1994.–157 с.

8. Ван-Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ. – М.: Мир, 1981. – 320 с.

9. Йордон Э. Управление сложными Интернет-проектами. – М.: ЛОРИ, 2003. – 344 с.