РОССИЙСКАЯ ФЕДЕРАЦИЯ
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК

А.Г. Ивашко, М.В. Григорьев, И.И. Коломиец

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Учебное пособие

Издательство
Тюменского государственного университета
2007

УДК 004.41

А.Г. ИВАШКО, М.В. ГРИГОРЬЕВ, И.И. КОЛОМИЕЦ Проектирование информационных систем

Учебное пособие. - Тюмень: Тюменский государственный университет, 2007. – 394 с.

Проектирование информационных систем является одной из важных дисциплин в подготовке студентов по специальности «Прикладная информатика (по областям)».

В учебном пособии представлены как технологии применения функционального моделирования при разработке информационных систем, так и методологии объектно-ориентированного проектирования с использованием UML. Рассматриваются вопросы программной инженерии, управления программными проектами, а также agile методологии RUP и eXtreme Programing.

В учебном пособии приведено описание оригинальной «Виртуальной среды исследования бизнес процессов» с примерами ее реализации. Описана методика выполнения лабораторных работ по дисциплине «Проектирование информационных систем» в формате командной разработки программных проектов. Данная методика основана на методологии RUP и использует неформальный набор артефактов.

РЕЦЕНЗЕНТЫ: Кафедра Информационной безопасности
Тюменского государственного университета;
А.А. Захаров, зав. кафедрой, профессор, д.т.н.

Александр Григорьевич Ивашко, Михаил Викторович Григорьев,
Инна Ивановна Коломиец

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Учебное пособие

Редактор

Технический редактор

Компьютерная верстка

Компьютерный дизайн обложки

Трафаретная печать

Офсетная печать

Подписано в печать . Тираж экз.

Объем 24,4 п.л. Формат 60x84/16. Заказ.

Издательство Тюменского государственного университета

625003, г. Тюмень, ул. Семакова, 10.

Содержание

Предисловие

Данное методическое пособие задумывалось, как конспект лекций по предмету «Проектирование информационных систем» для специальности «Прикладная информатика в экономике». Кафедра информационных систем Тюменского государственного университета имеет довольно большой опыт, как в преподавании этого предмета, так и в разработке информационных систем. Только одно то, что на кафедре более 100 студентов ежегодно выполняют дипломные работы, львиная доля которых посвящена проектированию и разработке информационных систем, говорит о многом.

В учебном пособии представлены современные тенденции развития данной дисциплины, управления программными проектами и вопросы программной инженерии; детально освящены методологии структурного анализа и моделирования информационных систем; рассматривается история развития объектно-ориентированного подхода и методологии объектно-ориентированного проектирования с использованием UML; изложена оригинальная методика разработки информационных систем, основанная на методологии RUP; представлена виртуальная среда исследования бизнес-процессов с примерами ее реализации.

Авторы выражают благодарность всем преподавателям кафедры, принявшим участие в обсуждении вопросов разработки информационных систем, а также студентам, участвовавшим в апробации предлагаемых методик. Хотелось бы выразить особую благодарность Хамидулину Р.Н., принимавшего активное участие в разработке методик преподавания предмета.

1. Терминология

Давайте разберемся с терминологией. Казалось бы, все довольно просто – «Проектирование информационных систем» определяет наши действия по созданию проекта, причем проекта какой- либо информационной системы.

Что означают термины «проектировать» и «проект»? Вот некоторые их определения в различных источниках.

- «Проектирование (от лат. projectus, буквально - брошенный вперёд), процесс создания проекта - прототипа, прообраза предполагаемого или возможного объекта, состояния. Проект - комплект указанной документации и материалов (определённого состава)» /1/.
- «Проект — это уникальная (в отличие от операций) деятельность, имеющая начало и конец во времени, направленная на создание определённого, уникального продукта или услуги. В силу своей уникальности проектная деятельность связана со многими рисками. Существует отдельная дисциплина в менеджменте — управление проектами (project management). Типичные проекты — создание или проектирование различных потребительских продуктов (здания, машины, электронные устройства, программное обеспечение)» /2/.
- «Проект - это временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов» /3/.
- «Проект - это уникальный процесс, состоящий из набора взаимосвязанных и контролируемых работ с датами начала и окончания и предпринятый для достижения цели соответствия конкретным требованиям, включая ограничения по времени, затратам и ресурсам» /4/.

Из стандартов ЕСКД /5/ следует, что проектная деятельность имеет стадии, которые называются: техническое предложение, эскизный проект, технический проект, рабочая конструкторская документация. Из стандартов в

области информационных технологий следуют еще более удивительные вещи. Так, например, в ЕСПД нет отдельного вида деятельности проектирования, а есть стадии разработки программ и программной документации: техническое задание, эскизный проект, технический проект, рабочий проект, внедрение. При этом этапы стадии «рабочий проект»: разработка программы, разработка программной документации, испытания программы, указывают на то, что результатом этой стадии является не проект, как «прообраз предполагаемого или возможного объекта», а непосредственно объект - программа и программная документация.

По-нашему мнению, наиболее простое определение, которое мы и будем использовать в дальнейшем - «Проект - это временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов» /3/. Проектирование - это совокупность действий, которые и реализуют этот проект. Такое толкование проектирования делает этот термин синонимом термина «Разработка», что и согласуется с определением ЕСПД.

Результатом проекта в нашем случае должен быть уникальный продукт – информационная система. Наступает второй шаг наших терминологических исследований.

Дело в том, что в отечественной литературе нет единого, сложившегося определения «информационная система», хотя это, наверное, одно из самых распространенных словосочетаний, встречающихся в Интернете. Причиной этого является, то, что этот термин в стандартах, посвященных информационным технологиям, не определяется.

Ярким примером является определение, данное на русскоязычном сайте Википедия /2/ : «Информационная система (ИС) — это система, предназначенная для ведения информационной модели, чаще всего — какой-либо области человеческой деятельности. Эта система должна обеспечивать средства для протекания информационных процессов: хранение, передача, преоб-

разование информации. По мнению одних авторов, ИС включает в себя персонал, её эксплуатирующий, по мнению других — нет».

В то же время, на англоязычном сайте Википедия /5/ дается более конкретное определение этого термина - **Information System** (IS) is the system of persons, data records and activities that process the data and information in a given organization, including manual processes or automated processes[1] (в данном пособии мы будем придерживаться этого определения). Обычно термин используется ошибочно как синоним машинных информационных систем[2] (computer-based information systems), который является только компонентом информационных технологий в информационной системе. Машинные информационные системы - область изучения информационных технологий (ИТ). Не нужно эти системы рассматривать отдельно от ИС в широком понятии, которая всегда содержит машинные информационные системы. Машинная информационная система, по определению Langefors /6/, является технологической средой для регистрации, хранения и распространения данной информации. Информационная система - также социальная система, чье поведение наряду с технологией определяется целями, ценностями и мнениями, как отдельных личностей, так и целых групп.

В книге Ciborra /7/ определяются «информационные системы», как область, в которой мы «имеет дело с развертыванием информационной технологии в организациях, учреждениях и обществе в целом».

Вы видите, как термин ИС постепенно используется для определения информационных систем, которые используют информационные технологии, для автоматизации основных функций. Так, например, в литературе встреча-

---

[1] Информационная Система (ИС) - система людей, хранимых данных и процессов обработки данных и информации в контексте конкретной организации. Обработка включает ручные или автоматизированные процессы

[2] Наиболее близкий термин в русскоязычной литературе – автоматизированные информационные системы

ется следующее определение: Информационная система (ИС) – это организационно-упорядоченная взаимосвязанная совокупность средств и методов ИТ, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели /8/. Такое понимание информационной системы предполагает использование в качестве основного технического средства переработки информации ЭВМ и средств связи, реализующих информационные процессы и выдачу информации, необходимой в процессе принятия решений задач из любой области. ИС является средой, составляющими элементами которой являются компьютеры, компьютерные сети, программные продукты, БД, люди, различного рода технические и программные средства связи и т.д. Хотя сама идея ИС и некоторые принципы их организации возникли задолго до появления компьютеров, компьютеризация в десятки и сотни раз повысила эффективность ИС и расширила сферы их применения. Такое понимание ИС становится в литературе синонимом термина «Автоматизированные информационные системы» (АИС). Для общности можно заметить, что средствами автоматизации могли бы использоваться не только информационные технологии, но в настоящее время, такие случаи крайне редки.

Поскольку Вы будущие специалисты в области информационных технологий, Вас будут интересовать информационные системы, использующие компьютерную обработку информации.

Другой термин, широко используемый в литературе - «Автоматизированная система» (АС). В серии стандартов 34 /9/ его определяют, как «система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций». Если учесть, что «функция автоматизированной системы - совокупность действий АС, направленной на достижение определенной цели» /9/, то подмножеством АС являются информационные системы, использующие автоматизированный способ обработки информации

(АИС). Обратите внимание, что термин АС шире АИС, так как включает понятия систем с неограниченным набором функциональных средств, в то время, как функции АИС обязательно включают сбор, хранение и обработку информации.

В данном учебном пособии рассматриваются вопросы проектирования автоматизированных информационных систем.

Что же будет результатом проектирования информационной системы? Совокупность технических средств, программных средств, регламент выполнения бизнес-процесса? Что же должно быть создано в результате проекта? Из определения информационных систем следует, что должно быть разработано и/или адаптировано программное обеспечение, развернутое на соответствующих компьютерах, использующих необходимые технические средства для реализации заданных функций. Работа программно- аппаратного комплекса должна быть согласована с бизнес – процессами, выполняемыми сотрудниками организации. Для реализации этого многообразия задач в комплексе стандартов 34, АС разделяется на функциональные и обеспечивающие части.

Мы рассмотрели термин «проектирование информационных систем». Приблизило ли нас это к созданию самой информационной системы? Наверное, нет, хотя мы теперь можем определить проект и информационную систему, которая будет создана в результате этого проекта. В тоже время видно, что проектирование информационных систем использует материал других областей: описание и моделирование бизнес - процессов; свод знаний по управлению проекта (PM), в частности, по управлению программными проектами (SWEBOK); программную инженерию; аппаратно-технические средства телекоммуникационных систем и т. д. Собственно, эти знания и составляют предмет проектирования информационных систем. Необходимо помнить, что в российских стандартах, посвященных информационным технологиям, не определено понятие информационных систем, поэтому используют-

ся стандарты, посвященные программному обеспечению (ЕСПД) и автоматизированным системам.

Что касается проектирования программного обеспечения, то эта область знаний в настоящее время динамично развивается, поэтому, терминология в этой области далеко не устоявшаяся. Существует большое количество методологий в области разработки программного обеспечения. В данном учебном пособии не ставится задача исследования всего многообразия материалов в области разработки программного обеспечения.

Прежде *чем рассматривать дальше*, обратим внимание на цели построения информационных систем, т.е. цели проектирования информационных систем. Согласно PMBOK /3/ – «Цель (Objective) - это, на что направлена работа; стратегическая позиция - которую следует занять; задача - которую следует решить; результат - которого следует достичь; продукт - который следует произвести или услуга - которую следует оказать». Следовательно, целью проекта является построение информационной системы. Проект «стартует», когда целесообразность построения информационной системы (в нашем случае) обоснована. С другой стороны, в стандарте ГОСТ 34.601-90 на этапе 1.1. "Обследование объекта и обоснование необходимости создания АС" регламентируются следующие мероприятия:

а) сбор данных об объекте автоматизации и осуществляемых видах деятельности;

б) оценка качества функционирования объекта и осуществляемых видов деятельности, выявление проблем, решение которых возможно средствами автоматизации;

в) оценка (технико-экономическая, социальная и т.д.) целесообразности создания АС.

Согласно ГОСТу, определения цели создания информационной системы являются частью создания АС. Трудно представить , что современный проект по созданию информационной системы будет финансироваться без

заранее определенных целей, как следует из ГОСТа. Более распространенной ситуацией является проведение консалтингового проекта, результатом которого и будет обоснование построения информационной системы. В этом случае, консалтинговый проект включает, согласно терминологии стандарта 34, стадии формирования требований к АС, разработку концепции АС и возможность технического задания АС. Проект по создания информационной системы в этом случае определяется результатами консалтингового проекта.

Несколько другой подход рекомендует «Унифицированный процесс» (RUP). В первой фазе проекта (фаза «Начало») требуется определить границы проекта и его цели, получить экономическое обоснование проекта, т.е. создания, в нашем случае, информационной системы. Следовательно, определение целей проекта входит в первую фазу проекта, что согласуется с ГОСТом. Логика при таком подходе довольно очевидная. В самом начале необходимо определить возможный путь решения проблемы и оценить это решение экономически целесообразно или нет. Трудно представить возможность выполнения проекта, который должен будет использовать ресурсы больше, чем выделено.

Считается, что в рамках проекта информационной системы Вы обязаны определить цели создания Вашей системы и обосновать их. Следует помнить, что неправильно понимаемые цели проекта приводят, как правило, к разрушению проекта.

Очевидной целью создания информационной системы является автоматизация операций и действий бизнес - процесса. Поскольку основными функциями информационной системы являются сбор, хранение и обработка информации, то очевидна автоматизация именно этих функций бизнес-процесса. Наиболее органично использовать компьютеры для обработки информации: создание различных видов отчетов, поиск информации, поиск решения и т.д. Реже, автоматизация затрагивает ввод информации. Например: использование технических средств ввода информации (сканеры, штрих ко-

ды), устранение повторного ввода одних и тех же данных и т.д. Следует обратить внимание, что использование электронных носителей информации вместо бумаги часто приводит к удорожанию хранения информации.

Контрольные вопросы

1. Что такое информация?
2. Как Вы понимаете термин «система»?
3. С какими системами Вы сталкивается при проектировании информационных систем?
4. Что такое АС?
5. Что такое АСУ?
6. В чем отличие информационной системы от АС?
7. Как Вы думаете, что общее у информационной системы и АСУ?
8. Что Вы считаете результатом проектирования здания и результатом проектирования ИС?
9. Как Вы понимаете термин «Проект»?
10. Что Вы понимаете под «системным свойством»?
11. Приведите пример систем и покажите, что они удовлетворяют системному свойству.

2. Стандарты

Стандартизация — это деятельность, направленная на разработку и установление требований, норм, правил, характеристик, как обязательных для выполнения, так и рекомендуемых; обеспечивающая право потребителя на приобретение товаров надлежащего качества, а также, право на безопасность и комфортность труда. Цель стандартизации — достижение оптимальной степени упорядочения в той или иной области посредством широкого и многократного использования установленных положений, требований, норм для решения реально существующих, планируемых или потенциальных задач. Основными результатами деятельности по стандартизации должны быть повышение степени соответствия продукта (услуги), процессов их функционального назначения, устранение технических барьеров в международном товарообмене, содействие научно-техническому прогрессу и сотрудничеству в различных областях.

Стандарты удешевляют совокупную стоимость владения системами, облегчают возможность расширения, модификации и масштабирования систем. Следование стандартам позволяет производителям техники наладить не мелкосерийное, а массовое производство продукции, повысить ее качество. Использование стандартов помогает снизить квалификационные требования к персоналу, сформировать четкие программы обучения, лучше подготовить персонал к решению практических задач.

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 ис-

пользуемых решений, для снижения трудоемкости и себестоимости работ, повышения их качества.

Кроме того, стандарты - это знания и опыт профессионалов, которые помогают начинающим специалистам выбрать пути решений насущных задач. С другой стороны, стандарты содержат правила и рекомендации, исполнение которых возможно только, когда эти правила и рекомендации знаешь и понимаешь. Выполнение стандартов требует определенной культуры организации, а также дисциплины ее сотрудников. Дивиденды, которые может принести применение стандартов, требует довольно кропотливой работы, направленной на поддержание их использования.

Довольно странно было бы услышать споры о применимости стандартов, например, среди инженеров-строителей о применимости стандартов в строительстве. Как Вы думаете - почему? Чтобы ответить на этот вопрос, необходимо упомянуть о сертификации.

Сертификация — процедура подтверждения соответствия, посредством которой независимая от изготовителя (продавца, исполнителя) и потребителя (покупателя) организация удостоверяет в письменной форме, что продукция соответствует установленным требованиям. Часто отождествляют сертификацию и лицензирование. Лицензия — это право (разрешение) на осуществление какой-либо деятельности, сертификат на услуги — документ, подтверждающий, что качество услуг соответствует определённым требованиям. В России существует больше 100 систем сертификации, из них около 20 — обязательных, остальные — добровольные. Одной из основных систем сертификации является, например, ГОСТ Р, поднадзорная ГосСтандарту и ГосСтрою, переименованные в Федеральное агентство по техническому регулированию и метрологии и в Федеральное агентство по строительству и жилищно-коммунальному хозяйству, соответственно.

Теперь понятно, что строительство без соблюдения соответствующих стандартов и нормативных актов невозможно, как и большинство видов ин-

женерной деятельности в различных отраслях. В отличие от строительных норм и правил, все стандарты в области разработки программного обеспечения имеют рекомендательный характер. Это одна из причин того, что можно услышать ряд критических замечаний по поводу применения стандартов. Дискуссии могут быть очень горячие. Сейчас их вести довольно бессмысленно. Жизнь заставляет знать и применять стандарты. Во-первых, конкуренция между разработчиками становится настолько серьезной, что малейшая некомпетентность приводит к проигрышу. Во-вторых, современный заказчик хорошо знает стандарты и умело ими пользуется и при постановке задач, и при приеме результатов выполнения.

Среди всего многообразия стандартов принято выделять следующие основные типы стандартов:

Корпоративные стандарты разрабатываются крупными фирмами (корпорациями) с целью повышения качества своей продукции. Такие стандарты разрабатываются на основе собственного опыта и с учетом требований мировых стандартов. Корпоративные стандарты не сертифицируются, но являются обязательными для применения внутри корпорации. В условиях рыночной конкуренции могут иметь закрытый характер. В ИТ сфере известны стандарты, разработанные Microsoft, Intel, IBM.

Отраслевые стандарты действуют в пределах организаций некоторой отрасли. Например, строительные нормы и правила разрабатываются с учетом требований мирового опыта и специфики отрасли. Подлежат сертификации.

Государственные стандарты (ГОСТы) принимаются государственными органами, имеют силу закона. Разрабатываются с учетом мирового опыта или на основе отраслевых стандартов. Могут иметь как рекомендательный, так и обязательный характер (стандарты безопасности). Для сертификации создаются государственные или лицензированные органы сертификации.

Международные стандарты. Разрабатываются, как правило, специальными международными организациями на основе мирового опыта и лучших корпоративных стандартов. Имеют сугубо рекомендательный характер. Право сертификации получают организации (государственные и частные), прошедшие лицензирование в международных организациях.

Существует большое количество стандартов в области информационных технологий, которые могут быть применимы при проектировании информационных систем. Материал, изложенный практически в каждой главе данного учебного пособия, основан на тех или иных стандартах.

Основными разработчиками международных стандартов являются следующие организации:

ISO - International Organization for Standardization – Международная организация по стандартизации, добровольная некоммерческая организация со штаб-квартирой в Женеве (web-site: http://www.iso.ch/), занимающаяся разработкой международных стандартов во многих областях, включая ИТ. Основана в 1946 г. как всемирная федерация органов стандартизации. Членами ISO являются более 130 национальных институтов, занимающихся стандартизацией. Название ISO не является аббревиатурой - оно происходит от древнегреческого слова isos, означавшего "равный, равносильный".

ACM - Association for Computing Machinery – Ассоциация по вычислительной технике, международная научно-образовательная организация со штаб-квартирой в Нью-Йорке (web-site: http://www.acm.org/). Основана в 1947 г., является головной организацией для SIGGRAPH и четырех десятков других групп по интересам. Занимается вопросами повышения технической компетентности специалистов в области компьютерных технологий, организует и проводит конференции, издает журналы и бюллетени по компьютерным технологиям, разрабатывает и продвигает стандарты. Известна, также, разработкой образовательных стандартов. Под эгидой ACM проводятся ежегодные международные студенческие олимпиады по программированию.

SEI - Software Engineering Institute - Институт Программной Инженерии находится при университете Карнеги-Меллона. Разработал модели SW-CMM и CMMI (web-site: http://www.sei.cmu.edu). Проводит исследования в области программной инженерии с упором на разработку методов оценки и повышения качества ПО. Стандарты по качеству ПО и зрелости организаций, разрабатывающих ПО.

PMI - Project Management Institute - Международный Институт Проектного Менеджмента (Управления Проектами). Штаб-квартира в Филадельфии (Пенсильвания). Международная общественная организация (web-site: http://www.pmi.org и http://www.pmi.ru), объединяющая профессионалов в области проектного менеджмента. PMI объединяет от 100000 до 135000 членов (данные различных источников расходятся) в 125 странах мира. PMI - некоммерческая организация, целью которой является продвижение, пропаганда, развитие проектного менеджмента в разных странах. PMI разрабатывает стандарты проектного менеджмента, занимается повышением квалификации специалистов.

IEEE - Институт инженеров по электротехнике и радиоэлектронике (произносится "ай-трипл-и"), ИИЭР (США) крупнейшая в мире организация (web-site: http://www.ieee.org/), объединяющая более 300 тыс. технических специалистов из 147 стран, ведущая организация по стандартизации, отвечающая также за сетевые стандарты. Образована в 1963 г. в результате слияния американских обществ IAEE, основанного в 1884 г., и IRE, основанного в 1912 г. ИИЭР проводит и спонсирует технические конференции, симпозиумы и семинары, ведет большую издательскую и образовательную деятельность.

IEC - International Electrotechnical Commission - Международная электротехническая комиссия (МЭК), расположенная в Женеве. Международная организация, занимающаяся стандартами в области электроники и электротехники, в том числе в области информационных технологий. Состоит из национальных комиссий 40 стран мира. Основана в 1906 г. В ИТ области наи-

более известны стандарты, разработанные объединенным техническим комитетом по информационным технологиям (JTC1) ИСО/МЭК, в частности, его 7-ым подкомитетом «Системная и программная инженерия».

ГОСТы в области информационных технологий

Первая группа стандартов, рассматриваемая нами, посвящена программной документации. Группа стандартов ГОСТ 19.х называется Единая система программной документации (ЕСПД). Прежде, чем перейти к рассмотрению этих стандартов, необходимо заметить, что ЕСПД и другие стандарты по информационным технологиям активно ссылаются на стандарты Единой системы конструкторской документации (ЕСКД). Наряду с разработкой программ и оформлением различных моделей, нам постоянно придется создавать текстовые документы, которые в учебном процессе оформляются так же, как и на производстве, в виде отчетов. Мы рекомендуем оформлять отчеты в соответствие с ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе. Структура и правила оформления». Данный стандарт распространяется на отчеты фундаментальных, поисковых, прикладных научно-исследовательских работ. Стандарт не согласуется с ЕСПД, но в тоже время, поддерживает новые концепции оформления документов, принятые в ЕСКД. Возможно, в Вашем учебном заведении используется свой корпоративный стандарт оформления курсовых и дипломных работ. Требования стандарта кратко приведены в Приложении А.

Единая система программной документации (ЕСПД)

В состав ЕСПД входят: основополагающие и организационно-методические стандарты; стандарты, определяющие формы и содержание программных документов, применяемых при обработке данных; стандарты, обеспечивающие автоматизацию разработки программных документов. Стандарты ЕСПД подразделяют на группы, приведенные в таблице 1.

Обозначение стандарта ЕСПД должно состоять из: цифры - 19, присвоенных классу стандартов ЕСПД; одной цифры (после точки), обозначающей код классификационной группы стандартов, указанной в таблице 1; двузначного числа, определяющего порядковый номер стандарта в группе; двузначного числа (после тире), указывающего год регистрации стандарта.

Таблица 1 Группы стандарта ЕСПД /7/

Код группы	Наименование группы
0	Общие положения
1	Основополагающие стандарты
2	Правила выполнения документации разработки
3	Правила выполнения документации изготовления
4	Правила выполнения документации сопровождения
5	Правила выполнения эксплуатационной документации
6	Правила обращения программной документации
7	Резервные группы
8	
9	Прочие стандарты

Перечень основных документов ЕСПД.

1. ГОСТ 19.001-77 ЕСПД. Общие положения.
2. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
3. ГОСТ 19.102-77 ЕСПД. Стадии разработки.
4. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.
5. ГОСТ 19.104-78 ЕСПД. Основные надписи.
6. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.
7. ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.
8. ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.

9. ГОСТ 19.202-78 ЕСПД. Спецификация. Требования к содержанию и оформлению.
10. ГОСТ 19.301-79 ЕСПД. Порядок и методика испытаний.
11. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.
12. ГОСТ 19.402-78 ЕСПД. Описание программы.
13. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
14. ГОСТ 19.501-78 ЕСПД. Формуляр. Требования к содержанию и оформлению.
15. ГОСТ 19.502-78 ЕСПД. Описание применения. Требования к содержанию и оформлению.
16. ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.
17. ГОСТ 19.504-79 ЕСПД. Руководство программиста.
18. ГОСТ 19.505-79 ЕСПД. Руководство оператора.
19. ГОСТ 19.506-79 ЕСПД. Описание языка.
20. ГОСТ 19.508-79 ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению.
21. ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные документы, выполняемые печатным способом.
22. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
23. ГОСТ 19.781-90. Обеспечение систем обработки информации программное.

Довольно большой блок стандартов (ГОСТ 19.402-78, ГОСТ 19.404-79, ГОСТ 19.502-78, ГОСТ 19.503-79, ГОСТ 19.504-79, ГОСТ 19.505-79) ориентирован на документирование результирующего продукта разработки. Некоторые стандарты впоследствии были заменены. ГОСТ 19781-90 Обеспечение

систем обработки информации программное. Термины и определения. Разработан взамен ГОСТ 19.781-83 и ГОСТ 19.004-80 и устанавливает термины и определения понятий в области программного обеспечения (ПО) систем обработки данных (СОД), применяемые во всех видах документации и литературы, входящих в сферу работ по стандартизации или использующих результаты этих работ.

Есть также группа стандартов, определяющая требования к фиксации всего набора программ и программной документации, которые оформляются для передачи программных средств. Так, например, ГОСТ 19.301-79 ЕСПД. «Программа и методика испытаний» - может использоваться для разработки документов планирования и проведения испытательных работ по оценке готовности и качества ПС.

Большие затруднения для школьников и студентов первого курса доставляет ГОСТ 19.701-90 ЕСПД. «Схемы алгоритмов, программ, данных и систем. Обозначения условные графические и правила выполнения». Он устанавливает правила выполнения схем, используемых для отображения различных видов задач обработки данных и средств их решения, и полностью соответствует стандарту ИСО 5807:1985. Стандарт не применим для отображения концепций объектно-ориентированного программирования.

Разберем некоторые из этих стандартов более подробно.

ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов /8/

Программы подразделяют на два вида:

- компонент - программа, рассматриваемая как единое целое, выполняющая законченную функцию и применяемая самостоятельно или в составе комплекса;

– комплекс - программа, состоящая из двух или более компонентов и (или) комплексов, выполняющих взаимосвязанные функции, и применяемая самостоятельно или в составе другого комплекса

К программным относят документы, содержащие сведения, необходимые для разработки, изготовления, сопровождения и эксплуатации программ. Определяются следующие виды программной документации:

– спецификация - состав программы и документации на нее;
– ведомость держателей подлинников - перечень предприятий, на которых хранят подлинники программных документов;
– текст программы - запись программы с необходимыми комментариями;
– описание программы - сведения о логической структуре и функционировании программы;
– программа и методика испытаний - требования, подлежащие проверке при испытании программы, а также порядок и методы их контроля;
– техническое задание - назначение и область применения программы, технические, технико-экономические и специальные требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний;
– пояснительная записка - схема алгоритма, общее описание алгоритма и (или) функционирования программы, а также обоснование принятых технических и технико-экономических решений;
– эксплуатационные документы - сведения для обеспечения функционирования и эксплуатации программы.

Кроме того, определены виды эксплуатационных документов:

– ведомость эксплуатационных документов - перечень эксплуатационных документов на программу;

- формуляр - основные характеристики программы, комплектность и сведения об эксплуатации программы;
- описание применения - сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств;
- руководство системного программиста - сведения для проверки, обеспечения функционирования и настройки программы на условиях конкретного применения;
- руководство программиста - сведения для эксплуатации программы;
- руководство оператора - сведения для обеспечения процедуры общения оператора с вычислительной системой в процессе выполнения программы;
- описание языка - описание синтаксиса и семантики языка;
- руководство по техническому обслуживанию - сведения для применения тестовых и диагностических программ при обслуживании технических средств.

Стадии проекта, в соответствии с ГОСТом 19.102-77, регламентируют содержание работ, которые в свою очередь, определяют виды создаваемых документов.

ГОСТ 19.102-77 ЕСПД. Стадии разработки /9/

В данном стандарте определены стадии, этапы и содержания работ, представленные в таблице 2.1.

Необходимо отметить, что стадии и этапы определяют временную организацию проекта. Следовательно, в стандарте определено последовательное во времени выполнение стадий и этапов.

Таблица 2.1 Стадии разработки, этапы и содержание работ

Стадии разработки	Этапы работ	Содержание работ
1. Техническое задание	Обоснование необходимости разработки программы	Постановка задачи. Сбор исходных материалов. Выбор и обоснование критериев эффективности и качества разрабатываемой программы. Обоснование необходимости проведения научно-исследовательских работ.
	Научно-исследовательские работы	Определение структуры входных и выходных данных. Предварительный выбор методов решения задач. Обоснование целесообразности применения ранее разработанных программ. Определение требований к техническим средствам. Обоснование принципиальной возможности решения поставленной задачи.
	Разработка и утверждение технического задания	Определение требований к программе. Разработка технико-экономического обоснования разработки программы. Определение стадий, этапов и сроков разработки программы и документации на неё. Выбор языков программирования. Определение необходимости проведения научно-исследовательских работ на последующих стадиях. Согласование и утверждение технического задания.
2. Эскизный проект	Разработка эскизного проекта	Предварительная разработка структуры входных и выходных данных. Уточнение методов решения задачи. Разработка общего описания алгоритма решения задачи Разработка технико-экономического обоснования.
	Утверждение эскизного проекта	Разработка пояснительной записки. Согласование и утверждение эскизного проекта.

Продолжение таблицы 2.1

Стадии разработки	Этапы работ	Содержание работ
3. Технический проект	Разработка технического проекта	Уточнение структуры входных и выходных данных. Разработка алгоритма решения задачи. Определение формы представления входных и выходных данных. Определение семантики и синтаксиса языка. Разработка структуры программы. Окончательное определение конфигурации технических средств.
	Утверждение технического проекта	Разработка плана мероприятий по разработке и внедрению программ. Разработка пояснительной записки. Согласование и утверждение технического проекта.
4. Рабочий проект	Разработка программы	Программирование и отладка программы.
	Разработка программной документации	Разработка программных документов в соответствии с требованиями ГОСТ 19.101-77.
	Испытания программы	Разработка, согласование и утверждение порядка и методики испытаний. Проведение предварительных государственных, межведомственных, приёмо-сдаточных и других видов испытаний. Корректировка программы и программной документации по результатам испытаний.
5. Внедрение	Подготовка и передача программы.	Подготовка и передача программы и программной документации для сопровождения и (или) изготовления. Оформление и утверждение акта о передаче программы на сопровождение и (или) изготовление. Передача программы в фонд алгоритмов и программ.

Для каждой стадии определен соответствующий набор документов и продуктов, следовательно, их выполнение должно быть выполнено во времени последовательно. Забегая вперед, можно отметить, что в этих стандартах определена каскадная модель жизненного цикла проекта. Это один из основных недостатков данной группы стандартов. В стандартах, разрабатываемых в 70-80 годы прошлого столетия, по-другому и не могло быть - тогда другой

модели еще не знали. Хуже то, что этот стандарт является действующим и сейчас, но Вы помните, что плохие правила это все же лучше их отсутствия.

ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению /10/

Стандарт устанавливает порядок построения и оформления технического задания на разработку программы или программного изделия для вычислительных машин, комплексов и систем независимо от их назначения и области применения. Техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки;
- в техническое задание допускается включать приложения.

В разделе «Введение» указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие. В разделе «Основания для разработки» должны быть указаны:

- документ (документы), на основании которых ведется разработка;
- организация, утвердившая этот документ, и дата его утверждения;
- наименование и (или) условное обозначение темы разработки.

Раздел «Требования к программе или программному изделию» должен содержать следующие подразделы:

- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости;
- требования к маркировке и упаковке;
- требования к транспортированию и хранению;
- специальные требования.

В разделе «Технико-экономические показатели» должны быть указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

В разделе «Стадии и этапы разработки» устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а так же, сроки разработки и определяют исполнителей.

В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

Допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

Недостатки комплекса стандартов ЕСПД

Стандарты ЕСПД создавались более 25 лет назад, это средние века развития информационных технологий, поэтому большая часть стандартов ЕСПД морально устарела.

К числу основных недостатков ЕСПД можно отнести:

- ориентацию на единственную, "каскадную" модель жизненного цикла (ЖЦ) ПС;

- отсутствие четких рекомендаций по документированию характеристик качества программных средств;
- отсутствие системной увязки с современными стандартами поддержки жизненного цикла продуктов и услуг;
- нечетко выраженный подход к документированию ПС как товарной продукции;
- отсутствие рекомендаций по самодокументированию ПС, например, в виде экранных меню и средств оперативной помощи пользователю ("хелпов");
- отсутствие рекомендаций по составу, содержанию и оформлению перспективных документов на программные средства, согласованных с рекомендациями международных и корпоративных стандартов;
- средства документирования и специфицирования программной продукции, декларированные в стандарте, не дают возможность отобразить современные концепции объектно-ориентированного программирования;
- документация, регламентированная стандартом, предназначена для того, чтобы фиксировать конечное состояние разработки на каждой стадии, в то время как разработчику требуется специфицировать сам процесс разработки;

За последующие десятилетия было принято довольно много стандартов в области информационных технологий. Особенно бурно этот процесс протекал за последнее десятилетие, в связи с принятием российских стандартов в соответствии со стандартами, разработанными международными организациями ISO, IEEE, IEK, ANSI и др. В качестве примера приведем стандарты и

руководящие документы, охватывающие документирование программного обеспечения:

ГОСТ Р ИСО 9127—94 Документация пользователя и информация на упаковке для потребительских программных пакетов;

ГОСТ Р ИСО /МЭК 9294—93 Руководство по управлению документированием программного обеспечения;

Р 50.1.029—2001 Информационные технологии поддержки жизненного цикла продукции. Интерактивные электронные технические руководства. Общие требования к содержанию, стилю и оформлению;

Р 50.1.030—2001 Информационные технологии поддержки жизненного цикла продукции. Интерактивные электронные технические руководства. Требования к логической структуре базы данных.

При этом необходимо не упускать из вида, что довольно большое количество международных стандартов пока не нашли отражения в отечественных стандартах, но они известны и доступны для того, чтобы использовать в профессиональной деятельности. В данном пособии не ставится задача выполнить полный обзор стандартов в области информационных технологий.

ЕСПД обладает еще одним существенным недостатком с точки зрения проектирования информационных систем. Данный стандарт рассматривает программное обеспечение в отрыве от среды, в которой это программное обеспечение будет работать. Собственно говоря, стандарт и не рассматривает систему, в которой работает программное обеспечение.

Только в стандартах комплекса ГОСТ 34 используется системный подход, позволяющий рассматривать взаимодействие программные средства со средой их функционирования.

Стандарты в области автоматизированных систем

История стандартов автоматизированных систем в нашей стране проходит по созданию двух комплексов: ГОСТ 24 ГОСТ 34. В 80-е годы начали

выходить стандарты (ГОСТ 24), посвященные автоматизированным системам управления (АСУ). Позже, данный комплекс стандартов заменяется ГОСТ 34, посвященным различным автоматизированным системам (АС). В ГОСТ 34 АСУ рассматривается как подмножество АС. Вот некоторый перечень стандартов в области АС и АСУ:

ГОСТ 24.101-80 Виды и комплектность документов (Заменен на ГОСТ 34.201-89)

ГОСТ 24.102-80 Обозначение документов (Заменен на ГОСТ 34.201-89)

ГОСТ 24.103-84 Автоматизированные системы управления. Общие положения

ГОСТ 24.104-85 Автоматизированные системы управления. Общие требования. (Раздел 3 заменен на ГОСТ 34.603-92).

ГОСТ 24.201-79 Требования к содержанию документа Техническое задание (Заменен ГОСТ 34.602-89)

ГОСТ 24.202-80 Требования к содержанию документа Технико-экономическое обоснование

ГОСТ 24.203-80 Требования к содержанию общесистемных документов

ГОСТ 24.204-80 Требования к содержанию документа Описание постановки задачи

ГОСТ 24.205-80 Требования к содержанию документов по информационному обеспечению

ГОСТ 24.206-80 Требования к содержанию документов по техническому обеспечению

ГОСТ 24.207-80 Требования к содержанию документов по программному обеспечению

ГОСТ 24.208-80 Требования к содержанию документов стадии Ввод в эксплуатацию

ГОСТ 24.209-80 Требования к содержанию документов по организационному обеспечению

ГОСТ 24.210-82 Требования к содержанию документов по функциональной части

ГОСТ 24.602-86 Состав и содержание работ по стадиям (Заменен на ГОСТ 34.601-90)

ГОСТ 24.703-85 Типовые проектные решения. Основные положения

ГОСТ 34.003-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения (Взамен ГОСТ 24.003-84, ГОСТ 22487-77)

ГОСТ 34.201-89 Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначения документов при создании автоматизированных систем (Взамен ГОСТ 24.101-80, ГОСТ 24.102-80)

ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. (Взамен ГОСТ 24.601-86, ГОСТ 24.602-86)

ГОСТ 34.602-89 Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы (Взамен ГОСТ 24.201-85)

ГОСТ 34.603-92 Информационная технология. Виды испытаний автоматизированных систем (Взамен ГОСТ 24.104-85 в части разд. 3.)

Кроме ГОСТов публикуются руководящие материалы. Довольно часто используется документ РД 50 - 34.698 – 90 «Методические указания. Информационная технология. Автоматизированные системы требования к содержанию документов».

История создания стандарта связана с тем положением, которое сложилось в 80-х годах. В различных отраслях использовалась плохо согласованная или несогласованная нормативно-техническая документация для автома-

тизированных систем управления (АСУ), автоматизированные системы управления предприятием (АСУП), автоматизированные системы управления технологическим производством (АСУТП), системы автоматизированного проектирования (САПР) и др. Это затрудняло интеграцию систем, обеспечение их эффективного совместного функционирования. Практика применения этих стандартов показала, что в них применяется единая система понятий, есть много общих объектов стандартизации, однако требования стандартов не согласованы между собой; имеются различия по составу и содержанию работ; различия по обозначению, составу, содержанию и оформлению документов и пр. Определено несколько важных положений, отражающих особенности АС как объекта стандартизации, например: "в общем случае АС состоит из программно-технических (ПТК), программно-методических (ПМК) комплексов и отдельных компонентов организационного, технического, программного и информационного обеспечений". Разделение понятий ПТК и АС закрепляло принцип, по которому АС есть не "ИС с БД", но:

- "организационно-техническая система, обеспечивающая выработку решений на основе автоматизации информационных процессов в различных сферах деятельности (управление, проектирование, производство и т. д.) или их сочетаниях" (по РД 50-680-88), что особенно актуально в аспектах бизнес-реинжиниринга;
- "система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций" (по ГОСТ 34.003-90).

Эти определения указывают на то, что АС - это, в первую очередь, персонал, принимающий решения и выполняющий другие управляющие действия, поддержанный организационно-техническими средствами. ГОСТ 34 выделяет в качестве компонента АС пользователя автоматизированной системой - «лицо, участвующее в функционировании АС или использующее результаты ее функционирования». Как основные компоненты можно выде-

лить программно-технический комплекс автоматизированной системы и ее обеспечение. ПТК представляет собой совокупность средств вычислительной техники, программного обеспечения и средств создания и заполнения машинной информационной базы при вводе системы в действие достаточных для выполнения одной или более задач АС. В литературе часто определяют ПТК, как компонент, реализующий функциональную часть автоматизированной системы. Для реализации работы системы определены различные виды ее обеспечения /12/: организационное, методическое, техническое, математическое, лингвистическое, информационное, правовое, программное, эргономическое, что не ограничивает всех возможных вариантов обеспечения системы, так, например, в ГОСТ 34.602-89 определяется метрологическое обеспечение.

Организационное обеспечение АС: Совокупность документов, устанавливающих организационную структуру, права и обязанности пользователей и эксплуатационного персонала АС в условиях функционирования, проверки и обеспечения работоспособности АС.

Методическое обеспечение АС: Совокупность документов, описывающих технологию функционирования АС, методы выбора и применения пользователями технологических приемов для получения конкретных результатов при функционировании АС.

Техническое обеспечение АС: Совокупность всех технических средств, используемых при функционировании АС.

Математическое обеспечение АС: Совокупность математических методов, моделей и алгоритмов, примененных в АС.

Программное обеспечение АС: Совокупность программ на носителях данных и программных документов, предназначенная для отладки, функционирования и проверки работоспособности АС.

Информационное обеспечение АС: Совокупность форм документов, классификаторов, нормативной базы и реализованных решений по объемам,

размещению и формам существования информации, применяемой в АС при ее функционировании.

Лингвистическое обеспечение АС: Совокупность средств и правил для формализации естественного языка, используемых при общении пользователей и эксплуатационного персонала АС с комплексом средств автоматизации при функционировании АС.

Правовое обеспечение АС: Совокупность правовых норм, регламентирующих правовые отношения при функционировании АС и юридический статус результатов ее функционирования.

Эргономическое обеспечение АС: Совокупность реализованных решений в АС по согласованию психологических, психофизиологических, антропометрических, физиологических характеристик и возможностей пользователей АС с техническими характеристиками комплекса средств автоматизации АС и параметрами рабочей среды на рабочих местах персонала АС.

Разработчики комплекса ГОСТ34пошли по пути, более близкому к схемам конкретных методик, чем к стандартам типа ISO12207, тем не менее, в данном стандарте определяется понятие жизненного цикла АС. В отличие от стандарта ISO12207 регламентируется фактически модель жизненного цикла – каскадная. Определены стадии и этапы, выполняемые организациями - участниками работ по созданию АС, устанавливаются в договорах и техническом задании. С другой стороны, определения перечня задач выполняемых при разработке, близко к подходу ISO.

Введение единой, достаточно качественно - определенной терминологии (ГОСТ34), наличие достаточно разумной классификации работ, документов, видов обеспечения и др. безусловно, полезно. ГОСТ34 способствует более полной и качественной стыковке разных систем, что особенно важно в условиях, когда разрабатывается все больше сложных комплексных АС, например, типа CAD-CAM, которые включают в свой состав АСУТП, АСУП, САПР-конструктора, САПР-технолога, АСНИ и др. системы /11/.

Стадии создания автоматизированных систем (ГОСТ 34.601-90) /13/

В стандарте не указана модель жизненного цикла, но процесс создания АС определяется как совокупность упорядоченных во времени, взаимосвязанных, объединённых в стадии и этапы работ. Если принять во внимания, что каждая стадия заканчивается завершающим комплектом документов, то понятно, что представлена каскадная модель жизненного цикла. Выделяется восемь стадий выполнения проекта по созданию автоматизированной системы, приведенных в таблице 2.2.

Таблица 2.2 Стадии и этапы создания АС

Стадии	Этапы работ
1. Формирование требований к АС	1.1. Обследование объекта и обоснование необходимости создания АС. 1.2. Формирование требований пользователя к АС. 1.3. Оформление отчёта о выполненной работе и заявки на разработку АС (тактико-технического задания).
2. Разработка концепции АС.	2.1. Изучение объекта. 2.2. Проведение необходимых научно-исследовательских работ. 2.3. Разработка вариантов концепции АС, удовлетворяющего требованиям пользователя. 2.4. Оформление отчёта о выполненной работе.
3. Техническое задание.	Разработка и утверждение технического задания на создание АС.
4. Эскизный проект.	4.1. Разработка предварительных проектных решений по системе и её частям. 4.2. Разработка документации на АС и её части.
5. Технический проект.	5.1. Разработка проектных решений по системе и её частям. 5.2. Разработка документации на АС и её части. 5.3. Разработка и оформление документации на поставку изделий для комплектования АС и (или) технических требований (технических заданий) на их разработку. 5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации.
6. Рабочая документация.	6.1. Разработка рабочей документации на систему и её части.

Стадии	Этапы работ
	6.2. Разработка или адаптация программ.
7. Ввод в действие.	7.1. Подготовка объекта автоматизации к вводу АС в действие. 7.2. Подготовка персонала. 7.3. Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями). 7.4. Строительно-монтажные работы. 7.5. Пусконаладочные работы. 7.6. Проведение предварительных испытаний. 7.7. Проведение опытной эксплуатации. 7.8. Проведение приёмочных испытаний.
8. Сопровождение АС	8.1. Выполнение работ в соответствии с гарантийными обязательствами. 8.2. Послегарантийное обслуживание.

Как видно, определена стадия сопровождения. Это не встречалось в ЕСПД, что поддерживает концепцию стандарта ISO 12207, в котором определен процесс сопровождения. Стандарт 34 не заменяет ЕСПД, более того, он ссылается на ЕСПД в части разработки программного обеспечения и программной документации, но в тоже время, создание и сопровождение программного обеспечения должно подчиняться задачам создания АС.

Необходимо принимать во внимания, что в состав АС могут входить технические средства, требующие проектирование и строительства. В состав информационной системы обычно входят уже созданные технические средства (компьютеры, телекоммуникационные устройства и т.д.), но это не устраняет задачу развертывания технических средств и программного обеспечения, следовательно, и комплекс мероприятия, связанных с этим. При создании информационных систем используется: компьютерная сеть, что требует создания рабочей документации на нее (чертежи и схемы соединения), создание серверной с соответствующим проектом энергообеспечения и возможным строительством, а также проектом организации пожарной безопасности и т.д.

Содержание работ, определенных на каждом этапе, не конкретизирует детали реализации функциональной и обеспечивающих частей системы.

Допускается объединять и опускать некоторые стадии или этапы, их перечень и задачи в каждом конкретном проекте. Перечень работ и задач в этом случае определяется договором, а не стандартом. Ключевым документом взаимодействия сторон является ТЗ - техническое задание на создание АС.

Техническое задание на создание автоматизированной системы (ГОСТ 34.602-89) /14/

ТЗ на АС является основным документом, определяющим требования и порядок создания (развития или модернизации) автоматизированной системы, в соответствии с которым проводится разработка АС и ее приемка при вводе в действие. ТЗ разрабатывает организация-разработчик (по ГОСТ 34.602-89), но формально выдает ТЗ разработчику заказчик (по РД 50-680-88).

Стандарт определяет все работы, предшествующие ТЗ и направленные на создания ТЗ, как предпроектные работы. Это соответствует первым трем стадиям создания АС. Такое построение работ соответствует каскадной модели жизненного цикла. В тоже время в стандарте определен инструмент, дающий возможность использовать другие модели жизненного цикла. Во-первых, есть возможность изменять требования с помощью документов «Дополнения». Во-вторых, если конкретные значения требований не могут быть установлены в процессе разработки ТЗ на АС, в нем можно сделать запись о порядке установления и согласования этих требований. При этом в текст ТЗ на АС изменений не вносят.

ТЗ на АС содержит следующие разделы, которые могут быть разделены на подразделы:

1) общие сведения;

2) назначение и цели создания (развития) системы;

3) характеристика объектов автоматизации;

4) требования к системе;

5) состав и содержание работ по созданию системы;

6) порядок контроля и приемки системы;

7) требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;

8) требования к документированию;

9) источники разработки;

В ТЗ на АС могут включаться приложения.

Раздел «Назначение и цели создания (развития) системы» состоит из подразделов:

1) назначение системы, где указывают вид автоматизируемой деятельности (управление, проектирование и т.п.) и перечень объектов автоматизации (объектов), на которых предполагается ее использовать, кроме того для АСУ дополнительно указывают перечень автоматизируемых органов (пунктов) управления и управляемых объектов;

2) цели создания системы, где приводят наименования и требуемые значения технических, технологических, производственно-экономических или других показателей объекта автоматизации, а также указывают критерии оценки достижения целей создания системы.

Раздел «Требования к системе» состоит из следующих подразделов:

1) требования к системе в целом;

2) требования к функциям (задачам), выполняемым системой;

3) требования к видам обеспечения.

Требования к функциям, выполняемым системой, приводятся по каждой подсистеме. Для подсистемы указываются: перечень функций, задач или их комплексов, подлежащих автоматизации; временной регламент реализации каждой функции, задачи; требования к качеству реализации каждой

функции; требования к форме представления выходной информации и достоверности выдачи результатов; перечень и критерии отказов для каждой функции, по которой задаются требования по надежности.

Требования к обеспечению системы определены в стандарте для каждого вида.

«Требования к документированию» приводят согласованный разработчиком и заказчиком системы перечень подлежащих разработке комплектов и видов документов. Требования к содержанию документов, разрабатываемых при создании АС, установлены указаниями РД 50-34.698-90, соответствующими государственными стандартами Единой системы программной документации (ЕСПД), Единой системы конструкторской документации (ЕСКД), Системы проектной документации для строительства (СПДС) и ГОСТ 34.602. Перечень документов приведен ниже.

В разделе «Состав и содержание работ по созданию системы» определяются стадии и этапы создания АС, перечень документов, предъявляемых по окончанию соответствующих стадий и этапов работ.

По согласованию между разработчиком и заказчиком системы в состав ТЗ на АС включают приложения, содержащие расчет ожидаемой эффективности системы и оценку научно-технического уровня системы.

Основные международные стандарты, затрагивающие вопросы проектирования информационных систем

Наиболее известными стандартами программной инженерии являются:

- ISO/IEC 12207 - Information Technology - Software Life Cycle Processes - Процессы жизненного цикла программных средств. Стандарт содержит определения основных понятий программного обеспечения информационных систем (в частности программного продукта и жизненного цикла программного продукта и систем), структуры жизненного цикла как совокупности процессов, детальное описание процессов жизненного цикла.

- SEI CMM - Capability Maturity Model (for Software) - модель зрелости процессов разработки программного обеспечения. Стандарт отвечает на вопрос: «Какими признаками должна обладать профессиональная организация по разработке ПО?». Профессионализм организации определяется через зрелость процесса, применяемого этой организацией. Выделяются пять уровней зрелости процесса. В последующие годы была создана модель CMMI (http://www.sei.cmu.edu/cmmi/models/), как результат интеграции моделей CMM для продуктов и процессов.
- ISO/IEC 15504 - Software Process Assessment - Оценка и аттестация зрелости процессов создания и сопровождения ПО. Является развитием и уточнением ISO 12207 и SEI CMM. Содержит расширенное по отношению ISO 12207 количество процессов жизненного цикла и 6 уровней зрелости процессов. Дается подробное описание схемы аттестации процессов, на основе результатов которой может быть выполнена оценка зрелости процессов и даны рекомендации по их усовершенствованию.
- PMI PMBOK - Project Management Body of Knowledge - Свод знаний по управлению проектами Международного института проектного менеджмента . Содержит описания состава знаний по следующим 9 разделам (областям знаний) управления проектами. Стандарт в области управления разработкой программных проектов SWBOK основан на положениях данного свода знаний.
- SWBOK - Software Engineering Body of Knowledge - Свод знаний по программной инженерии - содержит описания состава знаний по 10 разделам (областям знаний) программной инженерии.
- ACM/IEEE CC2001 - Computing Curricula 2001 – Академический образовательный стандарт в области компьютерных наук. Выделено 4 основных раздела компьютерных наук: Computer science, Computer engineering, Software engineering и Information systems, по каждому из которых описаны

области знаний соответствующего раздела, состав и планы рекомендуемых курсов.

В последующих главах мы будем подробно разбирать некоторые из этих стандартов в контексте проектирования информационных систем.

Заключение

На первый взгляд, может показаться, что описанные в этой главе отечественные стандарты стали историей развития программного обеспечения и автоматизированных систем. На эту мысль наводит и терминология, используемая в них: АС, АСУ, АСУТП, САПР. В современной литературе чаще можно встретить англоязычные аббревиатуры: MRP, ERP, CAM, CAD и т.д. Но это обманчивое впечатление. Во-первых, данные стандарты, несмотря на их преклонный возраст, действующие, т.е. большинство проектов информационных систем используют эти стандарты, как регламентирующие материалы. Во-вторых, в них отражен опыт разработчиков, позволяющий выполнять и современные проекты на достаточно высоком уровне.

Российские разработчик довольно часто сталкиваются с проблемой использования современных методологий в рамках регламента ЕСПД и ГОСТ 34. Большинство заказчиков требует, и закрепляет это требование договором, использовать документацию, регламентированную этими стандартами. При этом необходимо принимать во внимание, что ЕСКД (Единая система конструкторской документации), СНиП (Строительные нормы и правила) и СПДС (Системы проектной документации для строительства) являются законом для разработчиков в соответствующих отраслях. Существует довольно большое количество публикаций, которые показывают, что использование современных методологий, например, таких как RUP, при соответствующей организации может не противоречить использованию ГОСТ 19 и ГОСТ 34. Более того, возможна автоматическая генерация документов, регламентированных ГОСТами из артефактов RUP.

Владения знаниями современных методологий разработки информационных систем позволяет применять рекомендации ГОСТов достаточно эффективно.

В данной главе описаны подробно только две группы стандартов. На протяжении всего методического пособия мы будем постоянно обращаться к отечественным и международным стандартам, так как в них отражаются проверенные многолетним опытом современные знания.

Контрольные вопросы

1. Чем отличается сертификация от стандартизации?
2. Что такое лицензирование?
3. Когда Вы разрабатываете ПО, Ваша деятельность стандартизована, сертифицирована или лицензирована?
4. Можно ли разработать ПО без знания стандартов?
5. Что Вы выберете, как пользователь, сертифицированный программный продукт или нет, и почему?
6. В чем отличие ГОСТ 34 и ГОСТ 19?
7. В чем отличие стандартов группы ГОСТ34 и ГОСТ19?
8. Вы разрабатываете программный продукт. Какими стандартами при этом Вы будете руководствоваться?
9. Какие стандарты являются обязательными при выполнении программного проекта?
10. Какие существуют стадии жизненного цикла АС?

3. Жизненный цикл информационной системы

Процессный подход в менеджменте

Первоначальное использование элементов процессного подхода относится к началу XX века. Однако широкое распространение он получил лишь в конце этого века, когда доминировавший до этого времени структурный подход (иногда его называют административный, иерархический и т.д.) полностью утратил свое прогрессивное значение.

Теоретические основы структурного подхода и базирующиеся на ней системы массового производства были заложены представителями классической теории менеджмента – Ф.Тейлором, А.Файолем, М.Вебером и их последователями.

В соответствии с данной теории, организация представлялась как механизм, обладающий рядом функций, распределяемых между подразделениями. Управление такой организацией осуществлялось именно по структурным подразделениям.

Принципиальное отличие процессного подхода от структурного заключается в том, что основное внимание менеджмента концентрируется не на самостоятельных функциях, выполняемых различными подразделениями и должностными лицами, а на межфункциональных (кроссфункциональных) процессах, объединяющих отдельные функции в общие потоки и нацеленные на конечные результаты деятельности организации. Процессный подход обладает рядом преимуществ перед структурным подходом:

• непрерывность менеджмента организации на стыках между отдельными процессами, подразделениями и должностными лицами при их взаимодействии;

• сглаживание противоречий, связанных с иерархическим (вертикальным) построением менеджмента и горизонтальными бизнес-процессами производства и поставок продукции (услуг);

• большая нацеленность деятельности организации на интересы потребителей;

• расширение возможности реинжиниринга бизнеса организации.

Важнейшим аспектом в реализации процессного подхода выступают технологии его внедрения: методология TQM (Total Quality Management), ИСО серии 9000 и реинжиниринг бизнес-процессов.

Если первая методология направлена на улучшение уже имеющихся процессов без изменения их в организации, то суть реинжиниринга состоит в отказе от существующих процессов и замене их новыми.

Версия стандартов ИСО серии 9000:2000 полностью ориентирована на процессный подход к управлению организацией и предполагает переход с качества продукции на качество процессов. В связи с вышесказанным, в настоящее время процессный подход становится основополагающей базой современных подходов к управлению. Развитие процессного подхода связано с тем, что он является основой системы менеджмента качества организации регламентированной стандартами ИСО серии 9000:2000.

Что же такое процесс? В стандарте ГОСТ Р ИСО 9000-2001 «Системы менеджмента качества. Основные положения и словарь» дано определение: «совокупность взаимосвязанных или взаимодействующих видов деятельности, преобразующих входы в выходы».

На рисунке 3.1 приведена основанная на процессном подходе система менеджмента качества, описанная в семействе стандартов ИСО 9000. Он показывает, что заинтересованные стороны играют существенную роль в предоставлении организации входных данных. Наблюдение за удовлетворенностью заинтересованных сторон требует оценки информации, касающейся восприятия заинтересованными сторонами степени выполнения их потребностей и ожиданий, т.е. качества выпускаемой продукции. Эта информация определяет стратегию менеджмента.

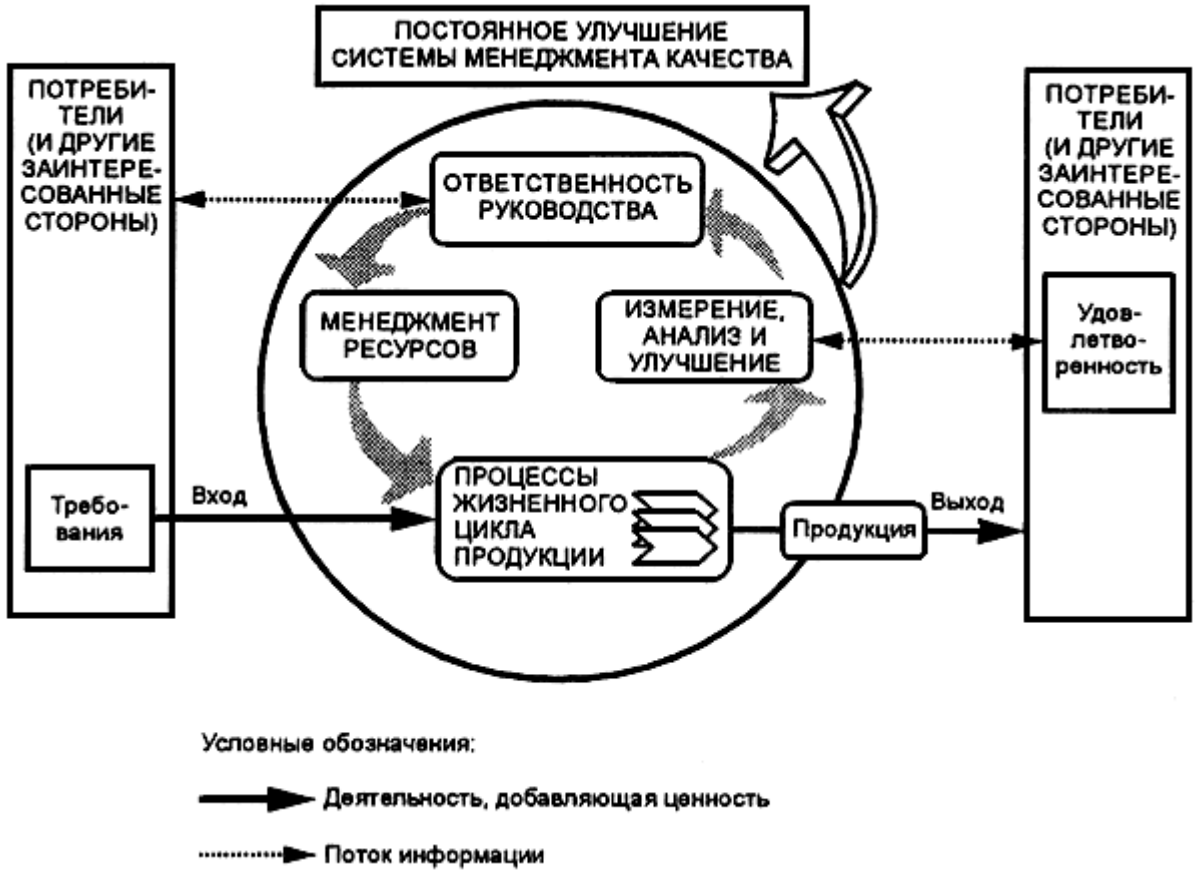


Рисунок 3.1 – Модель системы менеджмента качества, основанная на процессном подходе

В данной модели фактически отражены два основных понятия, которые нас будут интересовать дальше: качество и жизненный цикл продукции. Качество: степень соответствия присущих характеристик потребностям или ожиданиям, которые могут быть установлены или предполагаются, или являются обязательными (потребность или ожидание могут обозначаться единым термином - требование). Стандарт ИСО 9001:2000 выделяет четыре группы процессов, связанных с системой менеджмента качества: процессы управленческой деятельности руководства; процессы обеспечения ресурсами; процессы жизненного цикла продукции; процессы измерения, анализа и улучшения.

Имеются четыре общие категории продукции: услуги; программные средства; технические средства; перерабатываемые материалы. Многие виды

продукции содержат элементы, относящиеся к различным общим категориям продукции. Разработка, поставка и сопровождение информационной системы может включать как материальные виды продукции (компьютеры, сетевые устройства), так и нематериальные (программное обеспечение, информационная база). Специфика жизненного цикла информационной системы определяется тем, что она содержит все виды продукции и услуги. Они изучаются и описываются в различных областях знаний, в которых даже терминология может быть не согласованной.

Жизненный цикл автоматизированной системы упоминается в стандарте 34, но детально процессы не выделяются ни в одном из стандартов этой серии. Можно выделить следующие группы стандартов и области знаний, в которых рассматриваются процессы жизненного цикла продукции, являющиеся составной частью информационной системы.

Во-первых, комплекс стандартов ISO 9000, которые определяют концепции процессного менеджмента, позволяет выделить бизнес - процессы, описать их цели, задачи, входную и выходную информацию. Некоторые стандарты из этого комплекса посвящены специфике процессов при разработке, поставке, монтаже и обслуживании программного обеспечения (ISO 9000-3:1997).

Во-вторых, это стандарты и своды знаний в области управления проектами. Среди них можно отметить:

International Competence Baseline (ICB) — официальный международный Свод знаний в области PM, который поддерживается и развивается IPMA. Для 32 стран-членов IPMA он является основой для разработки национальных Сводов знаний. Ассоциация «СОВНЕТ» в России представляет данный свод знаний;

Американский национальный стандарт ANSI/PMI 99-001-2004, разработанный Институтом управления проектами (Project Management Institute) и

и известный, как «Руководства к Своду знаний по управлению проектами» (PMI PMBOK);

ГОСТ Р ИСО 10006-2005 «Руководство по менеджменту качества при проектировании», который выделяет принципы и методы управления качеством, применение которых важно для достижения целей менеджмента качества и дополняет руководящие указания ИСО 9004.

В-третьих, стандарты на системы управления IT - сервисами. Здесь можно отметить свод знаний ITIL (IT Infrastructure Library) — Библиотека передового опыта организации ИТ агентства CCTA/OGC (Великобритания) и построенные на его основе стандарты: ISO/IEC 20000-1:2005 IT Service Management – Спецификации; ISO/IEC 20000-2:2005 IT Service Management - Набор лучших практик. ISO/IEC 20000-1 содержит требования к управлению IT Сервисами и, прежде всего, ориентирован на сотрудников, занимающихся разработкой, внедрением и поддержанием Системы Управления IT Сервисами. ISO/IEC 20000-2 содержит руководящие указания и рекомендации, учитывающие лучшие мировые практики управления IT Сервисами в рамках Системы, соответствующей требованиям ISO/IEC 20000-1. Эта часть чаще всего используется компаниями, которые готовятся к сертификационному аудиту по ISO/IEC 20000-1 или планируют системные улучшения.

В-четвертых, стандарты, посвященные программной инженерии:

SEI CMM SW- Capability Maturity Model (for Software) - модель зрелости процессов разработки программного обеспечения [3] и CMMI - Capability Maturity Model Integration – интегрированная модель зрелости функциональных возможностей. Стандарты разработаны - Институтом программной инженерии при университете Карнеги-Меллона, США (Software Engineering

[3] в некоторых источниках переводится Capability Maturity Model, как модель зрелости функциональных требований (ABBY Lingvo 12), более уместный перевод, по нашему мнению, - модель зрелости разработки программного обеспечения, а для CMMI – интегрированная модель зрелости производства, но мы с Вами будем использовать перевод, используемый в литературе.

Institute - SEI). Стандарт CMM SW (в дальнейшем CMM) отвечает на вопрос: «Какими признаками должна обладать профессиональная организация по разработке ПО?». Профессионализм организации определяется через зрелость процесса, применяемого этой организацией. Выделяются пять уровней зрелости процесса. В последующие годы была создана модель CMMI (http://www.sei.cmu.edu/cmmi/models/), как результат интеграции моделей CMM для продуктов и процессов. Модель зрелости CMMI – 1.1 уточняет и совершенствует предшествовавшие модели CMM, а также учитывает основные требования существующих международных стандартов в области менеджмента программных средств. Значительное внимание в CMMI уделяется процессам разработки и учету итераций при изменении требований заказчиков, их прослеживанию к функциям, компонентам, тестам и документам проекта. В настоящее время опубликован документ CMMI for Development (Version 1.2). Совершенствование моделей зрелости продолжается.

ISO/IEC 15504 - Software Process Assessment - Оценка и аттестация зрелости процессов создания и сопровождения ПО. Является развитием и уточнением ISO 12207 и SEI CMM. Содержит расширенное по отношению ISO 12207 количество процессов жизненного цикла и 6 уровней зрелости процессов. Дается подробное описание схемы аттестации процессов, на основе результатов которой может быть выполнена оценка зрелости процессов и даны рекомендации по их усовершенствованию.

В-пятых, стандарты, посвященные жизненному циклу ПО:

IEEE Std 1074-1997 IEEE «Standard for DevelopingSoftware Life Cycle Processes», определяющий методологию создания процессов жизненного цикла программного обеспечения (SLCP), модели жизненного цикла программного обеспечения (SLCM) , жизненный цикл программного обеспечения (SLC), основанный на SLCM.

ГОСТ Р ИСО/МЭК 12207-99 «Информационная технология. Процессы жизненного цикла программных средств» устанавливает общую структуру

процессов жизненного цикла программных средств, на которую можно ориентироваться в программной индустрии;

ГОСТ Р ИСО/МЭК ТО 15271-2002 "Информационная технология. Руководство по применению ГОСТ Р ИСО/МЭК 12207 (Процессы жизненного цикла программных средств)";

ISO/IEC 15288:2002 "Системотехника. Процессы жизненного цикла систем", предусматривает проведение необходимых оценок качества и функционирования ИС;

ISO/IEC TR 19760:2003 «Проектирование систем - Руководство по применению ISO/IEC 15288 (Процессы жизненного цикла системы)».

Такое деление стандартов по областям знаний, как Вы понимаете, весьма вольное. Возникновение и развитие науки (лучше говорить - область знания) software engineering (программная инженерия) связано с кризисом в области программных проектов. Применение опыта в области процессного менеджмента дало возможность найти пути выхода из него. Именно тогда была предложена концепция жизненного цикла ПО. В 1970 г. У.У. Ройс (W.W. Royce) произвел идентификацию нескольких стадий жизненного цикла и им было высказано предположение, что контроль выполнения стадий приведет к повышению качества ПО и сокращению стоимости разработки.

В 1985 г. был разработан первый формализованный и утвержденный стандарт жизненного цикла для проектирования ПС систем военного назначения по заказам Министерства обороны США DOD-STD-2167 A «Разработка программных средств для систем военного назначения» (уточнен в 1988 г.). Этим документом регламентированы 8 фаз (этапов) при создании сложных критических ПС и около 250 типовых обязательных требований к процессам и объектам проектирования на этих этапах. В 1994г. принят Министерством обороны США для замены DOD-STD-2167 A и ряда других стандартов MIL-STD-498 «Разработка и документирование программного обеспечения». Он предназначен для применения всеми организациями и

предприятиями, получающими заказы Министерства обороны США. В 1996 г. утверждено руководство «Применение и рекомендации к стандарту MIL-STD-498». Основную часть составляют 75 подразделов — рекомендаций по обеспечению и реализации процессов ЖЦ сложных критических ПС высокого качества и надежности, функционирующих в реальном времени.

Данные стандарты можно считать историей развития концепции жизненного цикла программных средств. Последующие стандарты мы с Вами разберем более детально.

Стандарты, описывающие жизненный цикл ПО, можно сгруппировать по организациям – авторам:

- Группа стандартов ISO:
  - ISO/IEC 12207 Standard for Information Technology — Software Life Cycle Processes (есть российский аналог ГОСТ Р ИСО/МЭК 12207-99 «Информационная технология. Процессы жизненного цикла программных средств»). Стандарт определяет архитектуру жизненного цикла ПО и систем (автоматизированных систем), включающих три группы процессов: основные процессы, вспомогательные процессы и организационные процессы.
  - ISO/IEC TR 15271:1998 Information technology -- Guide for ISO/IEC 12207 (есть российский аналог ГОСТ Р ИСО/МЭК ТО 15271-2002 «Информационная технология. Руководство по применению ГОСТ Р ИСО/МЭК 12207»). В стандарте приведены рекомендации по практическому применению ГОСТ Р ИСО/МЭК 12207 в условиях реализации конкретных проектов создания программных средств.
  - ISO/IEC 15288 Standard for Systemы Engineering — System Life Cycle Processes (ГОСТ Р ИСО/МЭК 15288-2005 «Системная инженерия. Процессы жизненного цикла систем»). Отличается

от ГОСТ Р ИСО/МЭК 12207-99 нацеленностью на рассмотрение программно-аппаратных систем в целом.

- ISO/IEC 15504 (SPICE) Standard for Information Technology — Software Process Assessment. Существует технический отчет 1998 г. в 9 частях (ISO/IEC TR 15504) и пять стандартов (ISO/IEC 15504-1:2004…ISO/IEC 15504-5:2006 Information technology -- Process Assessment). Группа стандартов определяет правила оценки процессов жизненного цикла ПО и их возможностей; опирается на модель CMMI или ранее CMM, и ориентирована на оценку процессов и возможность их улучшения.

- Группа стандартов IEEE:
  - IEEE Std 1074-1997 IEEE Standard for Developing Software Life Cycle Processes. Описывает структуру процессов разработки и сопровождения, а также других процессов, связанных с ними. Определяет основные виды деятельностей, выполняемых в рамках этих процессов, и опирается на документы на входе и выходе этих деятельностей.
  - IEEE/EIA Std 12207-1997 IEEE/EIA Standard: Industry Implementation of International Standard ISO/IEC 12207:1995 Guide for Information Technology — Software Life Cycle Processes. Аналог ISO/IEC 12207, сменил ранее использовавшиеся J-Std-016-1995 EIA/IEEE Interim Standard for Information Technology--Software Life Cycle Processes--Software Development Acquirer-Supplier Agreement и стандарт министерства обороны США MIL-STD-498

- Группа стандартов, связанная с моделью зрелости возможностей: CMM (Capability Maturity Model) и CMMI (http://www.sei.cmu.edu/cmmi/models/)

В целом, эти стандарты связаны , как показано на рисунке 3.1.

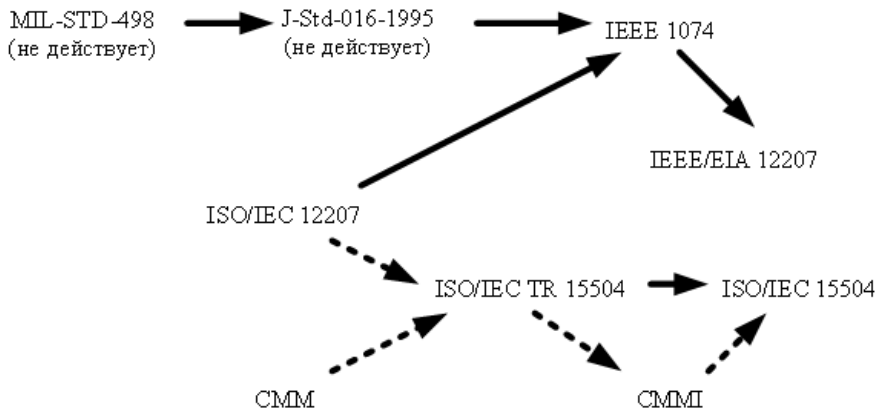


Рисунок 3.1 – Связь стандартов в области жизненного цикла программного обеспечения и систем (стрелки указывают направления исторического развития).

Основные термины, используемые в данных стандартах:

Процесс (process): Совокупность взаимосвязанных и взаимодействующих видов деятельности, преобразующих входы в выходы.

Работа синоним **деятельность** (activity): Совокупность действий, в результате которых расходуются время и ресурсы , выполнение которых необходимо для достижения или содействия достижению одного или нескольких результатов.

Система (system): Комбинация взаимодействующих элементов, организованных для достижения одной или нескольких поставленных целей (Система обладает системным свойством, т.е. свойством, которым не обладает ни один ее элемент).

Жизненный цикл системы (system life cycle): Развитие рассматриваемой системы во времени, начиная от замысла и заканчивая списанием.

Модель жизненного цикла (life cycle model): Структурная основа процессов и действий, относящихся к жизненному циклу, которая служит в качестве общей ссылки для установления связей и взаимопонимания сторон.

Обеспечивающая система (enabling system): Система, которая служит дополнением к рассматриваемой системе на протяжении стадий ее жизненного цикла, но необязательно вносит непосредственный вклад в ее функционирование.

Базовая линия (baseline): Спецификация или продукт, которые были официально рассмотрены и согласованы, чтобы впоследствии служить основой для дальнейшего развития, могут быть изменены только посредством официальных и контролируемых процедур изменения.

Проект (project): Попытка действий с определенными начальными и конечными датами, предпринимаемая для создания продукта или услуги в соответствии с заданными ресурсами и требованиями.

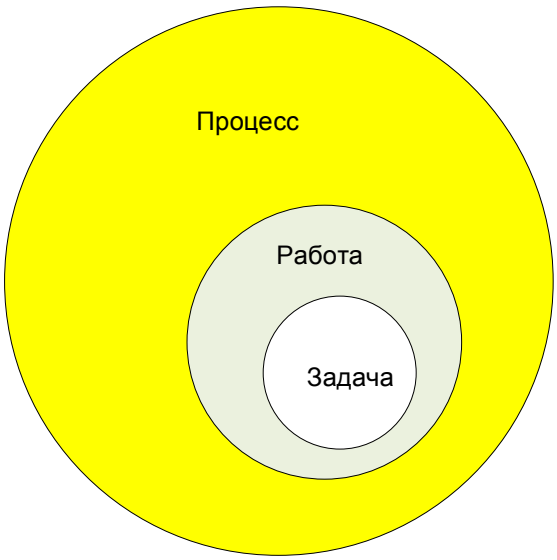
Ресурс (resource): Активы (организации), которые используются или потребляются в ходе выполнения процесса.

Процессы жизненного цикла программных средств в ГОСТ Р ИСО/МЭК 12207-99 /15, 16/

ГОСТ Р ИСО/МЭК 12207-99 - один из основополагающих стандартов в области жизненного цикла программного обеспечения. Данный стандарт воплощает концепции управления качеством ISO 9000 в области использования программных средств. Все последующие стандарты в этой области, так или иначе, ссылаются на данный стандарт.

Стандарт устанавливает, используя четко определенную терминологию, общую структуру процессов жизненного цикла программных средств, на которую можно ориентироваться в программной индустрии. Для следующих видов деятельности: приобретении системы, содержащей программные средства; оказании программной услуги; поставка, разработка, эксплуа-

тация и сопровождение программных продуктов определены: процессы, работы, как составляющие процесса, и задачи, как составляющие работы (рисунок 3.2). В стандарте определяется процесс, который может быть использован при определении, контроле и модернизации процессов жизненного цикла программных средств. Стандарт может быть использован для понимания сути программных продуктов и услуг.



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

Рисунок 3.2 – Соотношение между различными видами деятельности

Существует вид программного продукта, на который не распространяется данный стандарт. Это готовый продукт, который не входит в поставляемый. Если Вы покупаете, так называемую, «коробочную версию» программного продукта, например «1С. Бухгалтерия 7.7» в магазине, то Вы знаете, что жизненный цикл данного программного продукта не подпадает под действие этого стандарта. В тоже время, если Вы заключаете договор со второй стороной об адаптации этого же самого продукта, то данная деятельность подпадает под действие ГОСТ Р 12207. Кроме того, данный стандарт, как и все стандарты в области информационных технологий, носит рекомендательный характер. Это значит, что при оформлении договора об адаптации «1С. Бухгалтерия 7.7» необходимо сослаться на данный стандарт либо в до-

говоре, либо в техническом задании для того, чтобы обязать поставщика услуг выполнять требования данного стандарта.

Стандарт дает описание верхней архитектуры жизненного цикла программного продукта, но в тоже время не регламентирует ни модель жизненного цикла, ни методы разработки программных средств. Пользователи должны сами выбирать модель жизненного цикла применительно к своему программному проекту; сами распределять процессы (работы, задачи), выбранные из стандарта, на данной модели; выбирать и применять методы разработки программных средств.

Стандарт не регламентирует документацию, создаваемую в ходе цикла программного обеспечения. Регламент документации на разработанное программное средство или услугу определен стандартами ГОСТ 19 и ГОСТ 34, описанными в предыдущей главе, а также стандартами ГОСТ Р ИСО/МЭК ТО 9294-93 «Руководство по управлению документированием программного обеспечения», ГОСТ Р ИСО 9127—94 «Документация пользователя и информация на упаковке для потребительских программных пакетов», ISO TR 9127:1988 «Системы обработки информации - Документация пользователя и сопроводительная информация для пакетов программ потребителя», ANSI/IEEE 1063:1993. «Пользовательская документация на программные средства» и др.

Все процессы в архитектуре жизненного цикла, представленные на рисунке 3.3, делятся на три группы: основные процессы, вспомогательные процессы, организационные процессы.

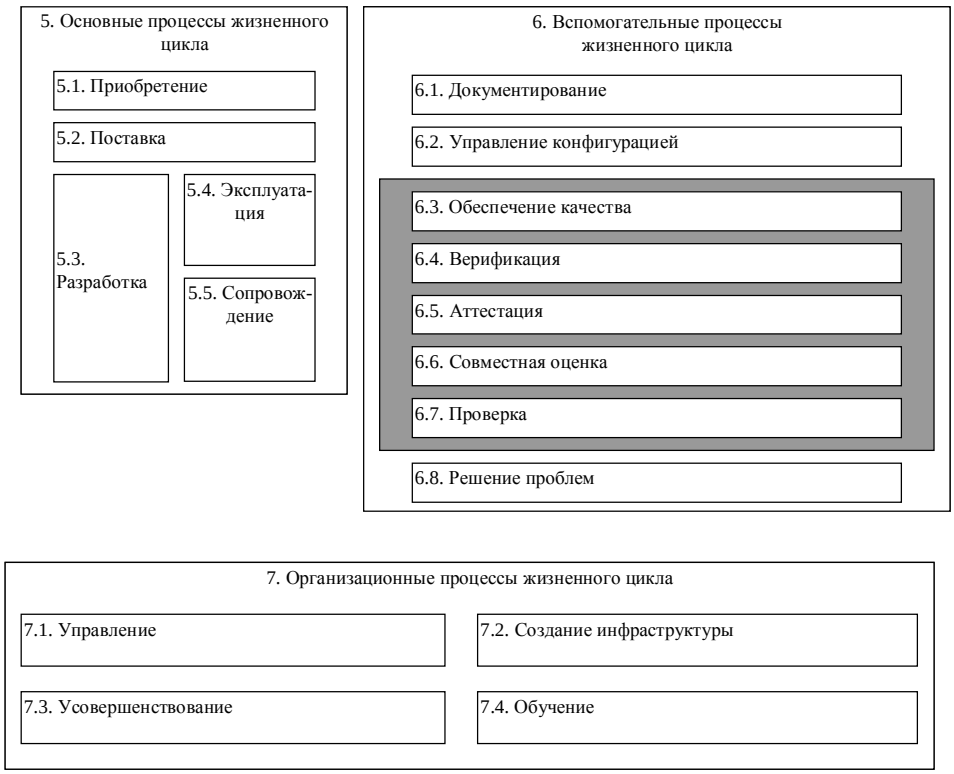


Рисунок 3.3- Архитектурное представление жизненного цикла.

Каждый процесс в ГОСТ Р ИСО/МЭК 12207 рассмотрен с точки зрения ответственности (обязанностей) сторон. Сторона, выполняющая процесс, несет ответственность за весь данный процесс, даже если выполнение отдельных задач поручено другим людям.

Основные процессы жизненного цикла (раздел 5) 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. Выделено пять основных сторон, и определено пять основных процессов.

1) Процесс заказа. Основная сторона - заказчик.

2) Процесс поставка. Основная сторона - поставщик.

3) Процесс разработки. Основная сторона - разработчик.

4) Процесс эксплуатации. Основная сторона - оператор.

5) Процесс сопровождения. Основная сторона - персонал сопровождения.

Вспомогательный процесс является целенаправленной составной частью другого процесса, обеспечивающий успешную реализацию и качество выполнения программного проекта. Вспомогательный процесс инициируется и используется другим процессом. Определено восемь вспомогательных процессов, пять из которых, направлены на повышение качества.

Организационные процессы применяются в какой-либо организации для создания и реализации основной структуры, охватывающей взаимосвязанные процессы жизненного цикла и соответствующий персонал, а также для постоянного совершенствования данной структуры и процессов. Эти процессы являются типовыми, независимо от области реализации конкретных проектов и договоров. Определено 4 организационных процесса.

Особое место занимает процесс адаптации. Определены основные работы, которые должны быть выполнены при адаптации ГОСТ Р 12207 к условиям конкретного программного проекта, где описывается краткое руководство по организации этого процесса.

В рамках одного проекта ГОСТ Р ИСО/МЭК 12207 может быть использован многократно и выборочно. Например, в конкретном проекте создания ПС заказчик может заключить договор с поставщиком на адаптацию и конфигурацию ПС. Поставщик может заключить договор с субподрядчиком на выполнение всей разработки. Поставщик (в режиме заказчика) и его субподрядчик (в режиме разработчика) могут использовать конкретный метод реализации ГОСТ Р ИСО/МЭК 12207. В обеих ситуациях необходимо прикладное применение ГОСТ Р ИСО/МЭК 12207 для отражения достигнутых соглашений.

Использование принципов общей системной инженерии, использованных в стандарте, дает возможность использовать его в процессе системной инженерии. Когда программное средство является частью общей системы, его выделяют из системы, создают и включают в систему.

Понятие «Система», данного стандарта, идентично понятию «Автоматизированная система». Как мы разобрали в первой главе, понятие «Информационная система» является одним из частных случаев понятия «Автоматизированная система», следовательно, основной процесс проектирования информационной системы описан в стандарте ГОСТ Р ИСО/МЭК 12207 в разделе 5.3 «Разработка». Именно эти работы и задачи нас интересуют в данном учебном пособии.

Процесс разработки программных средств, согласно ГОСТ Р ИСО/МЭК 12207, должен проводиться на двух уровнях: системном и программном (рисунок 3.4).

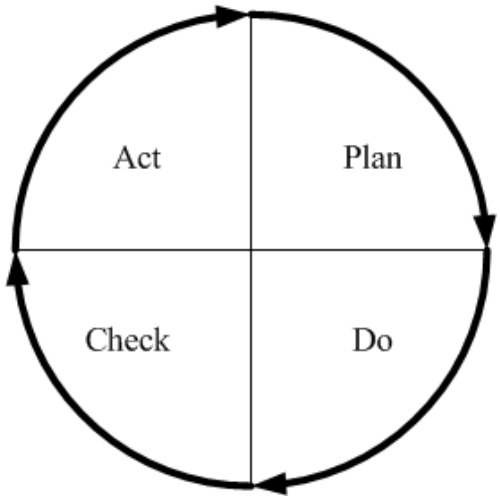
Рисунок 3.4 Классификация работ

Три вида работ, не представленных на рисунке (процесс разработки включает 13 видов работ): подготовка процесса (5.3.1), ввод в действие программных средств (5.3.12), обеспечение приемки программных средств(5.3.13), относятся к работам уровня проекта.

В процессе реализован Цикл качества Деминга (Deming) (рисунок 3.5), представляющий собой простую и наглядную модель управления качеством.

Согласно данной модели, для предоставления соответствующего уровня качества нужно непрерывно повторять следующие этапы:

- Планирование (Plan): что нужно сделать, когда это нужно сделать, кто должен это сделать, как это следует сделать и с помощью чего.
- Выполнение (Do): выполнение запланированных работ.
- Проверка (Check): определяется, есть ли ожидаемый результат после выполнения работ.
- Действие (Act): производится корректировка планов с учетом информации, полученной на этапе проверки.



Планирование (Plan) **à** Выполнение (Do) **à** Проверка (Check) **à** Действие (Act)

Рисунок 3.5 – Цикл Деминга (PDCA).

Краткая характеристика процессов разработки

Процесс разработки, как было указано выше, можно разбить на три типа работ: работы по организации проекта, системные и программные. Порядок перечисления работ в процессе не имеет особенного значения с точки зрения выполнения их в жизненном цикле программного обеспечения.

Работы по организации проекта включают: подготовку процесса, ввод в действие программных средств и обеспечение приемки программных средств. Первый вид работы будет выполняться на начальной стадии проек-

та, а последующие два - завершают работы проекта. Все виды работ относятся ко всему проекту.

Одной из первых задач, которую нужно решить при подготовке процесса, является определение модели жизненного цикла программного средства. Подготовка процесса включает задачи по настройке и выполнению вспомогательных процессов, в частности, процесса документирования. Важно, что при этом разработчик должен выбрать стандарты, методики и инструментарий, который он должен адаптировать и настроить.

Выполнение работ, ввод в действие программных средств осуществляется согласно принципу PDCA. Работы должны быть спланированы (план готовится документально), выполнены, проверены разработчиком, а результаты проверки должны быть отражены в соответствующих документах. Приемка программных средств должна осуществляться согласно договору.

К **работам, связанным с системой,** относятся:

- анализ требований к системе;
- проектирование системной архитектуры, сборка системы;
- квалификационные испытания системы.

Дальше мы будем разбирать ГОСТ Р ИСО/МЭК 15288 «Информационные технологии. Системная инженерия. Процессы жизненного цикла систем», который устанавливает общие основы жизненного цикла систем. Оба стандарта согласованы, но нужно помнить, что в ГОСТ Р ИСО/МЭК 12207 рассматривается процесс разработки на более низком уровне абстракции.

Все задачи, рассматриваемые в данных видах работы, относятся ко всей системе. Напомним, что информационная система - система людей, хранимых данных и процессов обработки данных и информации в контексте конкретной организации. Задачи, выполняемые на этом уровни абстракции, относятся не только к информационным технологиям. Именно на этом уровни изучаются бизнес - процессы с целью выявления требований к функциональному содержанию программного обеспечения. Именно на этом уровне

проверяется качество реализованных задач. В силу сказанного, на данном уровне разработка и описание системной архитектуры скорее будут производиться в терминах бизнеса, а не в ИТ терминах.

Примерный перечень характеристик системного уровня (относящихся к программному средству и подлежащих учету) включает в себя:

- межсистемные и внутрисистемные интерфейсы;
- интерфейсы пользователя;
- влияние ошибок программного средства на защиту и безопасность системы;
- оценку вычислительных мощностей и временных ограничений;
- наличие программ, реализованных техническими средствами;
- наличие соответствующих компьютеров.

Если в систему входит много подсистем или элементов конфигурации, для них должны быть полностью проведены работы системного уровня из процесса разработки. Должны быть учтены все требования к интерфейсам и сборке (интеграции) системы.

Последний тип работ относится к **программному уровню**:

- анализ требований к программным средствам;
- проектирование программной архитектуры;
- техническое проектирование программных средств;
- программирование и тестирование программных средств;
- сборка программных средств;
- квалификационные испытания программных средств.

Эти работы составляют сердцевину разработки.

Описание задач в данных работах напоминает этапы работ ГОСТ 34 и ГОСТ 19, но при этом существенно отличается объект, к которому применяется понятие эскизный проект, технический проект и реализация. В ГОСТ Р ИСО/МЭК 12207 все задачи, связанные с разработкой ПС, относятся применительно к каждому программному объекту архитектуры (или объекту про-

граммной конфигурации, если он определен). Это дает возможность реализовать любую модель жизненного цикла в рамках этого стандарта. Принимая во внимание что, разработчики ГОСТ Р ИСО/МЭК 12207 максимально приблизили терминологию отечественных стандартов, можно довольно легко адаптировать ГОСТ 34, ГОСТ 19 к современным моделям жизненного цикла ПС. Однако это не снимает проблем документирования ПО, разработанного с применением объектно-ориентированных методологий.

Должны быть определены характеристики программного уровня, например:

- потенциальное число элементов программной конфигурации;
- типы, объемы и критичность программных средств;
- технический риск;
- типы, комплектность и носители документов;
- необходимость новой разработки, изменения или повторного применения программного средства;
- аспекты из ГОСТ Р ИСО/МЭК 9126, такие как надежность.

Процесс адаптации

Процесс адаптации является процессом применения положений ГОСТ Р ИСО/МЭК 12207 к условиям реализации конкретного программного проекта. Данный процесс состоит из следующих работ:

- определение условий выполнения проекта;
- запрос исходных данных;
- выбор процессов, работ и задач;
- документирование решений по адаптации и их обоснование.

Пример использования ГОСТ Р ИСО/МЭК 12207 в общей модели жизненного цикла системы приведен на рисунке 3.5.

Процесс жизненного цикла в IEEE Std 1074-1997 IEEE Standard for Developing Software Life Cycle Processes /17/

Документ IEEE 1074 обеспечивает поддержку процесса жизненного цикла разработки ПО (Software life cycle process, SLCP). Процесс SLCP определен как характерное описание процесса, который основывается на жизненном цикле ПО проекта (Software life cycle, SLC), а также на интегральных процессах и процессах менеджмента, используемых организацией. Интегральные процессы включают менеджмент конфигурации, метрические показатели, обеспечение качества, уменьшение степени риска, действия по оценке, планированию и обучению. Разработка процесса SLCP для данного программного проекта входит в обязанности менеджера и архитектора проекта. Реализация подобной методологии начинается с выбора соответствующей модели жизненного цикла разработки ПО (Software life cycle model, SLCM) с целью ее применения в рамках определенного проекта.

Стандарт 1074 не обращается к действиям непрограммного характера, например: заключения контракта, приобретения или разработки аппаратных средств. Он не дает полномочий для использования определенной модели SLCM. В стандарте 1074 предполагается, что менеджер проекта и архитектор процесса уже знакомы с разнообразием процессов модели SLCM, с критериями их выбора, с критериями определения признаков и ограничений конечной системы и среды разработки, которые влияют на этот выбор.

Продукт 1074 — это цикл SLCP, который требуется для выполнения определенного программного проекта. Хотя стандарт 1074 описывает разработку отдельного, полного процесса SLCP, который должен применяться для проекта, пользователь этого стандарта должен признать, что SLCP может сам по себе включать низкоуровневые жизненные циклы разработки ПО. Здесь используется та же концепция, что и при менеджменте конфигурации, когда отдельный элемент конфигурации может включать зависимые ее элементы.

Этот стандарт одинаково пригоден к разработке процесса SLCP на любом уровне.

Рисунок 3.5 - Использование ГОСТ Р ИСО/МЭК 12207 для обеспечения модели жизненного цикла системы

Последовательность реализации стандарта, являющаяся эталонной моделью его применения, состоит из трех шагов.

Шаг 1. Выбор SLCM: архитектор процесса должен идентифицировать SLCM, действия которой будут отображены в модели. Этот шаг охва-

тывает классификацию, оценку, отбор и приобретение SLCM. Организация должна иметь множество моделей жизненного цикла с целью их повторного использования, но в данном проекте может использоваться только одна из этих моделей. Архитектор процесса при этом должен:

- идентифицировать все циклы модели SLCM, которые являются доступными при разработке проекта;
- идентифицировать признаки, которые применяются для желаемой окончательной системы и среды разработки;
- идентифицировать любые ограничения, которые могли бы быть наложены на выбор;
- оцепить различные циклы модели SLCM на основе предыдущего опыта и организаторских способностей;
- выбрать цикл SLCM, который лучше всего удовлетворяет атрибутам и ограничениям проекта.

Шаг 2. Создание жизненного цикла ПО (SLC): действия, идентифицированные по Приложению А стандарта, отображаются на модель SLCM. Порядок расположения действий определяется:

- начальным порядком выбранной модели SLCM;
- ограничениями графика работ - могут потребоваться перекрытия различных действий в SLCM или же их выполнение параллельно;
- согласованием входов и выходов, а также критериями входа связных действий.

Архитектор процесса создает и обосновывает лист неиспользуемых действий, а затем проверяет правильность отображения действий на модель. Он должен гарантировать, что действия полностью отображены на выбранную модель SLCM и что полученный SLC содержит весь набор действий, необходимых для успешного завершения проекта ПО. Вместе с этим, архи-

тектор процесса должен проверить согласованность информационных потоков порядку выполнения работ, отображенных в SLC.

Шаг 3. Утверждение SLCP. Предшествующие шаги создают SLC. На данном шаге доступные ресурсы организационных процессов должны применяться к действиям жизненного цикла, при этом ограничения должны быть выверены. Выходная информация каждого действия должна быть связана с соответствующим выходным документом.

Стандарт 1074 включает шесть основных категорий процесса:

1. процесс модели жизненного цикла разработки ПО;
2. процессы менеджмента проектов;
3. процессы предварительной разработки;
4. процессы разработки;
5. процессы, выполняемые после разработки;
6. интегральные процессы.

Существует 17 подпроцессов и 65 действий, входящих в состав подпроцессов. Все эти объекты приводятся в таблице. Ниже приведена карта категорий, процессов и действий, в соответствии с Приложением А стандарта.

I. Процессы менеджмента проекта

А. Начало выполнения проекта

1. Создание процесса жизненного цикла ПО
2. Выполнение оценки
3. Распределение проектных ресурсов
4. Определение метрик

Б. Действия по планированию проекта

1. План оценок
2. План управления конфигурацией плана
3. План развития системы (если признан пригодным)
4. План инициализации
5. План документации

6. План обучения
7. План руководства проектом

В. Мониторинг и управление проектом

1. Анализ рисков
2. Планирование непредвиденных обстоятельств
3. Управление проектом
4. Сохранение записей
5. Метод отчета о реализации проблемы

II. Этапы предварительной разработки процесса

А. Исследование концепции

1. Идентификация идей или потребностей
2. Формулирование потенциальных подходов
3. Изучение реализуемости поддержки
4. Уточнение и придание окончательной формы идее или потребности

Б. Распределение системы

1. Анализ функций
2. Разработка структуры системы
3. Разложение требований системы на составные части

В. Использование стороннего ПО

1. Идентификация требований к импортируемому ПО
2. Оценка источников импортируемого ПО
3. Определение импорта стороннего программного обеспечения
4. Импорт ПО

III Процессы разработки

А. Требования

1. Определение и разработка требований к ПО
2. Определение требований к интерфейсу

3. Распределение по приоритетам и интеграция требований к ПО

Б. Разработка проекта

1. Проектирование исполняемой архитектуры
2. Проектирование баз данных
3. Разработка проекта интерфейсов
4. Выбор или разработка алгоритмов
5. Выполнение детализованной разработки проекта

В. Внедрение

1. Создание исполняемого кода
2. Создание рабочей документации
3. Выполнение интеграции

IV. Процессы сопровождения

А. Установка

1. Размещение программного обеспечения
2. Инсталляция ПО
3. Приемка программного обеспечения в рабочую эксплуатацию

Б. Эксплуатация и поддержка

1. Использование системы
2. Обеспечение технической помощи и консультаций
3. Поддержка журнала запроса поддержки

В. Сопровождение

1. Идентификация потребности усовершенствования ПО
2. Внедрение методов получения отчетов о проблемах системы
3. Повторное обращение к жизненному циклу разработки ПО

Г. Вывод из эксплуатации

1. Уведомление пользователя
2. Выполнение параллельных действий
3. Вывод системы из эксплуатации

V. Интегрированные процессы

А. Оценка

1. Обзоры поведения
2. Создание матриц трассируемости
3. Аудит поведения
4. Разработка тестовых процедур
5. Создание тестовых данных
6. Тестирование
7. Отчет результатов оценки
8. Аттестация и верификация

Б. Менеджмент конфигурации программных средств

1. Разработка методов идентификации конфигурации
2. Осуществление контроля конфигурации
3. Выполнение учета состояния

В. Разработка документации

1. Внедрение документации
2. Создание и распространение документации

Г. Обучение

1. Разработка обучающих материалов
2. Утверждение программы обучения
3. Внедрение учебной программы

Для каждого действия определено описание, входная и выходная информация. Так, например, для действия «проектирование исполняемой архитектуры» входная информация: плановая информация о конфигурационном

менеджменте, системная архитектура, требования к импортируемому ПО, требования к ПО. Выходная информация – проект архитектуры ПО. В данном действии преобразовываются требования к ПО и архитектуры системы в концептуальный проект высокого уровня. Идентифицируются компоненты ПО, а также определяют структуру этих компонентов. Может использоваться моделирование и прототипирование для оценки альтернатив. К концу этой деятельности описание проекта каждого компонента программного обеспечения должно быть закончено и определены их отношения и ограничения. Все внутренние интерфейсы взаимодействия компонентов должны быть определены. До проектирования исполняемой архитектуры должны быть вызваны действия: обзор поведения, обзор представленной конфигурации и внедрение документации.

Стандарт 1074 не отождествляется с определенным процессом модели SLCM, не может существовать без определенных циклов SLC организации. Стандарт 1074 не предполагает использование определенной методологии разработки ПО и даже не рекомендует ее. Он не является самодостаточным, т.е. более строгие требования могут быть добавлены в случае необходимости.

Жизненный цикл системы по ГОСТ Р ИСО/МЭК 15288 -2005 /18/

Стандарт ISO/IEC 15288:2002 «Системная инженерия. Процессы жизненного цикла систем» является практически первым международным стандартом, в котором всесторонне, с точки зрения организации процессов жизненного цикла (ЖЦ) рассматриваются методологические принципы проектирования систем. Стандарт разработан 7-ым подкомитетом «Системная и программная инженерия» объединенного технического комитета по информационным технологиям (JTC1) ИСО/МЭК. Документ создан на основе опыта проектирования сложных систем, сложившегося в оборонно-промышленных комплексах и крупных коммерческих корпорациях ведущих промышленных

держав и в 2003-2004 гг. введен в действие в США, Австралии, Великобритании, Канаде, Швеции и ряде других стран. Проект идентичного национального стандарта ГОСТ Р ИСО/МЭК 15288 разработан при участии технического комитета по стандартизации ТК 22 "Информационные технологии" Федерального агентства по техническому регулированию.

Стандарт ГОСТ Р ИСО/МЭК 15288 задает единую структуру для установления и развития связей и кооперации между сторонами, создающими и использующими современные системы и управляющими ими в целях совместной согласованной работы. Он обеспечивает основы для моделирования и реализации общих процессов, составляющих ЖЦ систем, предоставляя возможность для их оценки и совершенствования, охватывая все концепции и идеи, имеющие отношение к этим системам, начиная от замысла и до момента снятия с эксплуатации. Процессы ЖЦ, задаваемые стандартом, могут использоваться однократно, многократно или рекурсивно, как по отношению к системе в целом, так и к любым ее элементам. Применяется для систем единичного и массового производства, а также адаптируемых к требованиям заказчика.

Стандарт касается систем, созданных человеком и включает один или несколько элементов: аппаратное обеспечение, программное обеспечение, людей, процессы (процесс оценки), процедуры (инструкции оператора), основные средства и природные ресурсы (вода, организмы, минералы).

Стандартом устанавливается (рисунок 3.6) четыре группы процессов ЖЦ систем, а именно: процессы соглашения, процессы предприятия, процессы проекта и технические процессы.

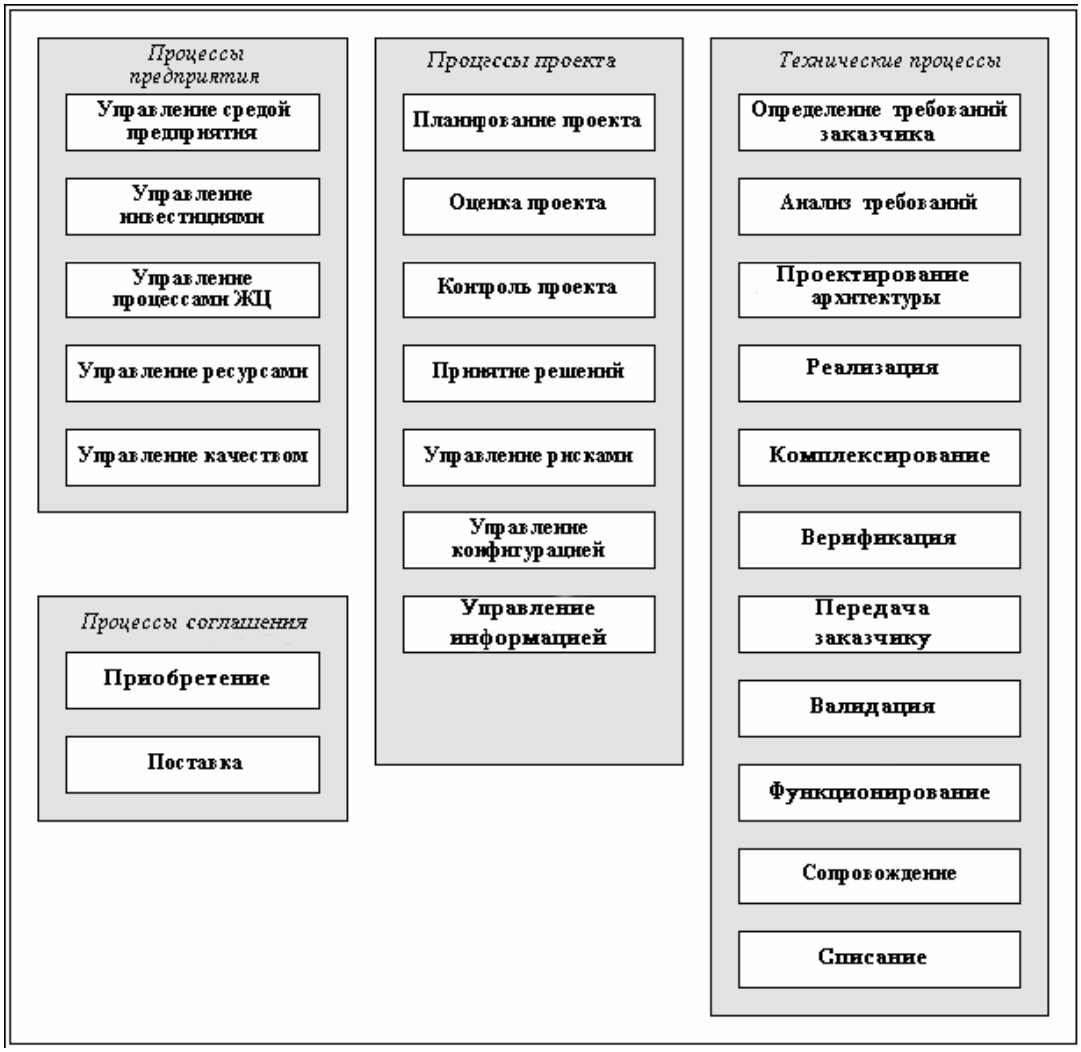


Рисунок 3.6 – Процессы жизненного цикла системы.

Процессы соглашения состоят из: процесса приобретения, используемого организациями для приобретения продукции или получения услуг; процесса поставки, используемого организациями для поставок продукции или оказания услуг.

Процессы предприятия управляют способность организации приобретать и поставлять продукцию или услуги посредством запуска проектов, их поддержки и контроля. Процессы предприятия обеспечивают ресурсы и инфраструктуру, необходимые для осуществления проектов.

Процессы проекта используются для установления и выполнения планов, оценки фактических достижений и продвижений проекта в соответствии с планами и для контроля выполнения проекта, вплоть до его завершения.

Технические процессы применяются для определения требований к системе, преобразования этих требований в эффективный продукт или использование его для обеспечения требуемых услуг, для поддержки обеспечения услуг, для удаления продукта, когда он изымается из обращения.

По отношению к каждому из процессов, в документе рассматриваются цели, результаты реализации и виды деятельности, обеспечивающие достижения указанных результатов. Всего в стандарте специфицировано 25 процессов, 123 результата реализации и 208 видов деятельности.

Данный стандарт интересен в предмете проектирования информационных систем с двух сторон. Во-первых, процессы уровня проекта и технические процессы могут быть для организации и управления программными проектами. Во-вторых, использованное описание процессов на разном уровне абстракции системы дает возможность получить наиболее полную модель системы. Особенно такой подход важен при внедрении корпоративных информационных систем, которые изменяют не только некоторое множество бизнес-процессов, но и все предприятие, как систему.

Модель зрелости процессов разработки программного обеспечения (Capability maturity model for software- SW CMM) /19/

Принципы обеспечения качества продукта были развиты институтом SEI в виде концептуальной структуры зрелости процессов, формирующей управленческий и инженерный фундамент для количественного контроля над производственным процессом.

Незрелая организация-разработчик:

- противодействует любым изменениям, а управляющее звено обычно сфокусировано на решении неотложных проблем (деятельность, известная как «пожаротушение»);
- обычно превышает графики работ и бюджет, так как они не основаны на реальных оценках;

- по мере приближения к критическим срокам сдачи проекта, идет на компромисс между сроками выполнения, функциональностью и качеством продукта;
- не имеет объективной основы для вынесения решения о качестве продукта или решения проблем, связанных с процессами и разрабатываемым продуктом;
- разрабатывает программный продукт, качество которого непредсказуемо.

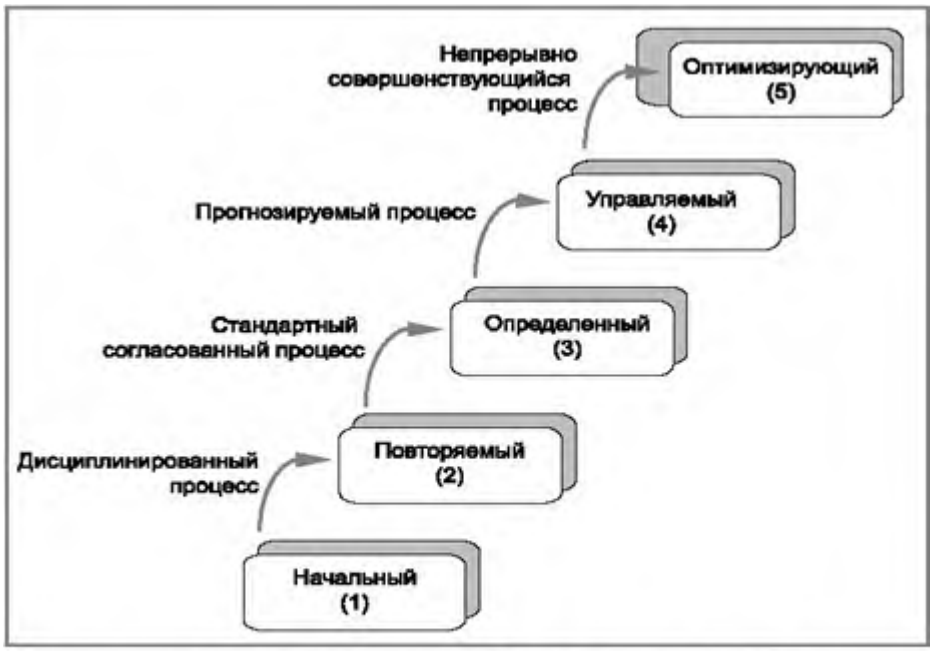
Зрелая организация - разработчик:

- четко определяют распределение ролей и сфер ответственности в пределах определенного процесса на протяжении всего проекта и в рамках всей организации;
- определяет одну из важнейших задач менеджмента, как постоянный мониторинг удовлетворенности заказчика созданным решениям;
- применяют объективную, количественную оценку качества продукта, а также анализируют проблемы, возникающие с продуктом или процессом;
- реалистично определяют план-графики и бюджеты, основываясь на показателях производительности предыдущих проектов; что дает возможность достигать ожидаемые результаты по затратам, срокам разработки, функциональности и качеству продукта

CMM предоставляет концептуальную структуру, эволюционно организующую совершенствование предприятия пяти уровней зрелости. Эти пять уровней зрелости, с одной стороны, определяют порядковую шкалу для измерения зрелости и оценки продуктивности производственного процесса, а с другой стороны, помогают организации расставить приоритеты среди своих мероприятий по улучшению процесса разработки. Модель зрелости процес-

сов в последующем развитии будет иметь два представления, что закрепит две группы функции, заложенные в ней.

СММ это одна модель, но в ней заложено две различные цели ее использования. Во-первых, она предназначена для совершенствования процессов разработки программной продукции внутри организации (стрелки на рисунке 3.7). Во-вторых, она предназначена для внешней оценки предприятия-разработчика (сертифицирование по СММ), с целью определения возможности производства им качественной продукции, что, в конечном итоге, стимулирует предприятие проводить работы по совершенствованию своей организации (характеристики уровней зрелости, показанных на рисунке 3.7).



Помеченные стрелки указывают на тип продуктивности процесса, устанавливаемый организацией на каждом шаге его структуры.

Рисунок 3.7 - Пять уровней зрелости производственного процесса

1) Начальный. Производственный процесс характеризуется как создаваемый каждый раз под конкретный проект, а иногда как хаотический. Определены лишь некоторые процессы и успех проекта зависит от усилий индивидуумов.

2) Повторяемый. Установлены основные процессы управления проектом, позволяющие отслеживать затраты, следить за графиком работ и функ-

циональностью создаваемого программного решения. Установлена дисциплина процесса, необходимая для повторения достигнутых ранее успехов в проектах разработки подобных приложений.

3) Определенный. Производственный процесс документирован и стандартизован как для управленческих работ, так и для проектирования. Этот процесс интегрирован в стандартный производственный процесс организации. Во всех проектах используется утвержденная адаптированная версия стандартного производственного процесса организации.

4) Управляемый. Собираются подробные количественные показатели производственного процесса и качества создаваемого продукта. Как производственный процесс, так и продукты оцениваются и контролируются с количественной точки зрения.

5) Оптимизирующий. Постоянное совершенствование процесса достигается благодаря количественной обратной связи с процессом и реализации передовых идей и технологий.

Организация CMM - иерархическая (рисунок 3.8).

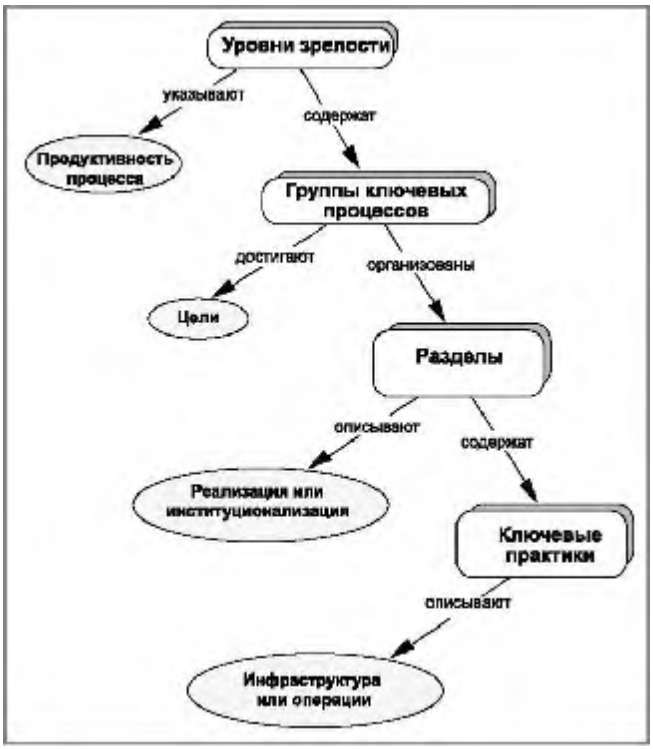


Рисунок 3.8- Структура CMM.

В таблице 3.1 приводятся ключевые процессы CMM, сгруппированные по уровням зрелости.

Таблица 3.1 Ключевые процессы CMM и их цели

УЗ	Название процессов	Цели процессов
Повторяемый	Управление требованиями	согласование требований, выдвигаемых к проекту разработки ПО заказчиком и разработчиком
	Планирование проекта	создание обоснованных планов разработки ПО и управления выполнением проекта разработки
	Отслеживание хода проекта и контроль над ним	обеспечение адекватного обзора фактического выполнения проекта, позволяя руководству предпринимать эффективные меры при значительном отклонении хода проекта от планов разработки
	Управление производственным субподрядом	выбор квалифицированных производственных субподрядчиков и эффективное управление ими
	Обеспечение качества ПО	дать руководству адекватное представление о ходе процесса разработки и о создаваемых продуктах
	Управление конфигурацией ПО	обеспечение целостности продуктов проекта разработки ПО в течение всего жизненного цикла проекта
Определенный	Координация производственного процесса организации	установление организационной сферы ответственности за выполнение мероприятий, совершенствующих общие производственные возможности организации
	Определение производственного процесса организации	разработка и поддержка практичного набора основных средств производственных процессов
	Программа обучения	рост квалификации и знаний сотрудников
	Интегрированное управление разработкой ПО	интеграция операций разработки и управления в последовательный и определенный производственный процесс

Продолжение таблицы 3.1

УЗ	Название процессов	Цели процессов
Определенный	Инженерия разработки программного продукта	последовательное выполнение четко определенного процесса, интегрирующего все операции разработки в целях рационального и эффективного создания качественных и надежных программных продуктов (в этой группе описываются технические работы проекта, такие как анализ требований, проектирование, кодирование и тестирование)
	Межгрупповая координация	установление средств активного взаимодействия разработчиков с другими инженерными группами
	Экспертные оценки	эффективное устранение дефектов в промежуточных программных продуктах на ранних стадиях разработки
Управляемый	Количественное управление процессом	установление количественного контроля над производительностью процесса проекта разработки
	Управление качеством ПО	развитие количественного понимания качества программных продуктов проекта и достижение определенных показателей качества
Оптимизирующий	Предотвращение дефектов	выявление причин дефектов и предотвращение их повторного появления
	Управление технологическими изменениями	выявление новых полезных технологий и их организованного внедрения в организации
	Управление изменениями процесса	непрерывное усовершенствование производственных процессов, используемых в организации, с целью улучшения качества производимого ПО, повышения производительности и сокращения цикла разработки продукта

Каждый уровень, кроме первого, описан в представленной структуре. Описание каждого уровня зрелости состоит из нескольких групп ключевых

процессов. Каждая группа ключевых процессов разбита на пять разделов. В разделах приводятся ключевые практики, при совместном выполнении которых достигаются цели группы ключевых процессов.

Важной особенностью данного подхода к рассмотрению жизненного цикла программного обеспечения (SLC) является то, что определены ключевые процессы, обеспечивающие соответствующий уровень качества производства организации - разработчика.

Жизненный цикл программного обеспечения согласно ИСО/ МЭК ТО 15504

В июне 1991 г. техническим комитетом ISO/IEC JTC1/SC7 было начато изучение потребностей в стандартизации методов оценки качества и аттестации зрелости процессов создания программных средств. В рамках проекта по усовершенствованию процессов создания программных средств и определению возможностей (Software Process Improvement and Capability dEtermination (SPICE), была начата стандартизация уже существующих методологий оценки качества программных средств. Был выпущен технический отчет из девяти частей ISO/IEC TR 15504 Technical Report -1998 «Information technology Software process assessment. Part 1…9» (далее ИСО/ МЭК ТО 15504) /20-28/. За основу разработки были выбраны концепции, изложенные в стандарте ISO/IEC 12207 и модели зрелости процессов CMM (Capability Maturity Model for Software). В 2004..2006 гг. издается переработанный стандарт ISO/IEC 15504 в 5 частях /29-33/. В нашей стране издан перевод ISO/IEC TR 15504 Technical Report -1998 /34/.

ИСО/МЭК ТО 15504 предоставляет основу для аттестации процесса жизненного цикла программных средств. Основным определяющим фактором при использовании ИСО/МЭК ТО 15504 является цель, с которой проводится аттестация. Такой целью может быть:

- улучшение понимания процессов жизненного цикла программных средств;
- поддержка усовершенствования процессов;
- поддержка определения зрелости процессов.

Фактически ставилась задача использования концепции CMM в рамках, заданных стандартом ISO 12207. В силу этого, впервые появляется двойное представление эталонной модели процессов и их зрелости, или, в терминологии стандарта, двухмерная архитектура. Первым измерением является измерение- «процесс», которое характеризуется набором утверждений о назначении процессов. Утверждения о назначении процессов описывают в измеримых терминах, что должно быть достигнуто для того, чтобы выполнить назначение процесса. Процессы определены в соответствии с ИСО/МЭК 12207: 1995. Вторым измерением является «зрелость», характеризующая уровень зрелости, достигнутой организационной единицей, в конкретном процессе, или который может использоваться организацией в качестве целевой точки.

Эталонная модель определяет высокоуровневые, фундаментальные цели, обязательные для хорошо организованной программной инженерии. Эти высокоуровневые цели описывают то, что должно быть достигнуто, а не то, как этого достичь. Эталонная модель применима к любой организации, работающей с программными средствами и желающей организовать процессы приобретения, поставки, разработки, использования, развития и поддержки программных средств и последовательно повышать их зрелость. Модель не предполагает определенных организационных структур, моделей административного управления, моделей жизненного цикла программных средств, технологий программирования или методологий разработки.

В представлении «процесс» рассматриваются три группы процессов жизненного цикла:

- группа основных процессов жизненного цикла состоит из категорий процессов потребитель-поставщик и инженерной;
- группа вспомогательных процессов жизненного цикла состоит из вспомогательной категории процессов;
- группа организационных процессов жизненного цикла состоит из управленческой и организационной категорий процессов.

. Категории процессов обозначаются:

- CUS Потребитель - поставщик (CustomerSupplier);
- ENG Инженерная (Engineering);
- SUP Вспомогательная (Suipport);
- MAN Управленческая (Management);
- ORG Организационная (Organization).

Для каждого процесса определяется степень соответствия процессам стандарта ISO 12207. Степень соответствия определяет тип процесса:

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

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

Инженерная категория состоит из процессов, которые непосредственно определяют, реализуют или поддерживают программный продукт, его связь с системой и пользовательскую документацию.

Вспомогательная категория состоит из процессов, которые могут быть использованы любым другим процессом (включая другие вспомогательные процессы) жизненного цикла программных средств.

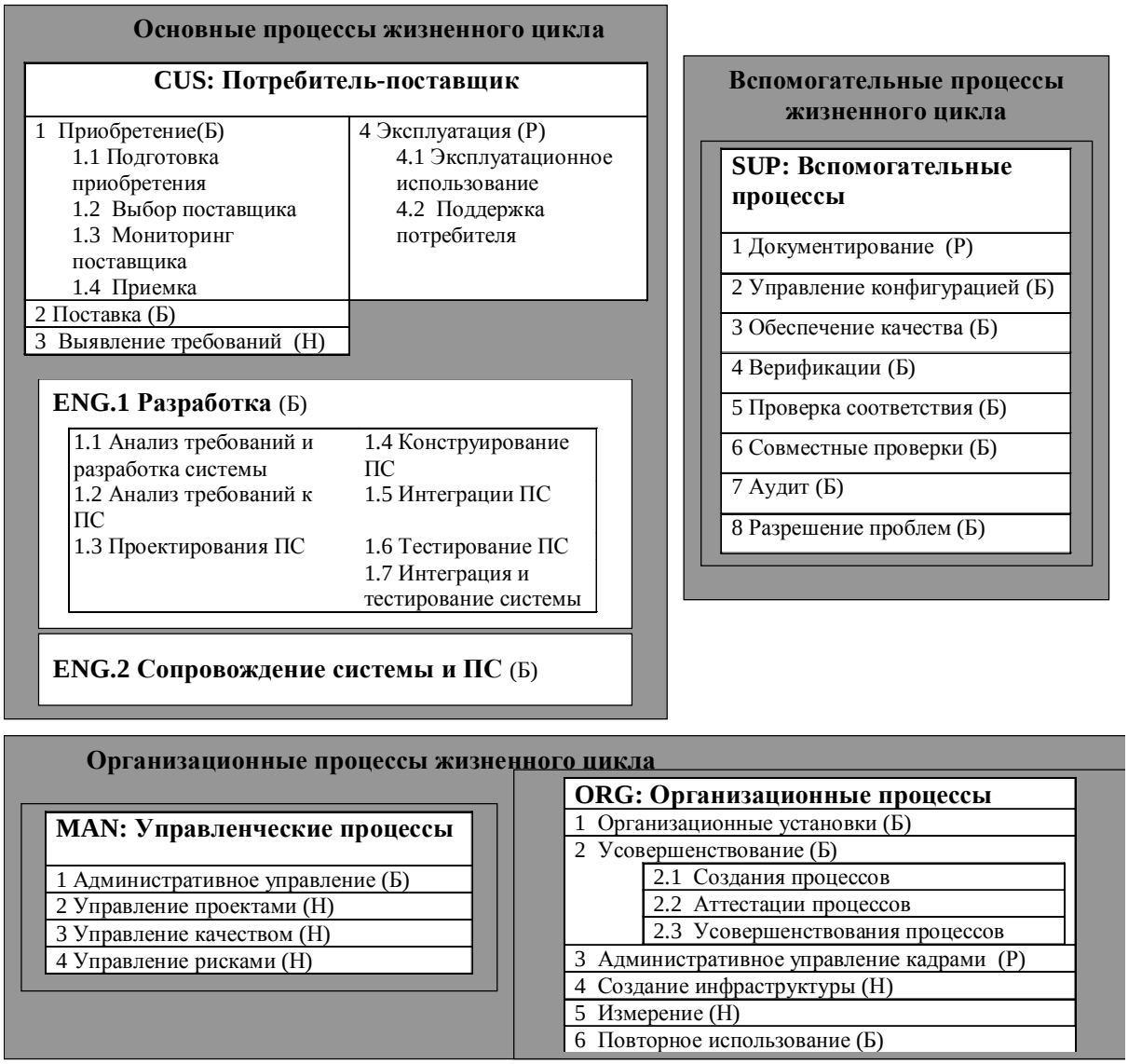
Управленческая категория состоит из процессов, которые содержат мероприятия общего характера, а также могут быть использованы всяким, кто управляет любым видом проектов или процессов жизненного цикла программных средств. ·

Организационная категория состоит из процессов, которые определяют цели организации и создают активы процессов, продуктов и ресурсов, которые, при использовании в имеющихся в организации проектах, помогут организации достичь ее цели.

Категории процессов и процессы представляют собой группировку по виду деятельности. Каждый процесс в эталонной модели описывается в терминах утверждения о назначении. Эти утверждения представляют собой уникальные функциональные цели процесса при воспроизведении в конкретной среде. Утверждение о назначении включает дополнительный материал, определяющий итоги успешного осуществления процесса.

На рисунке 3.9 приведено измерение «процесс» эталонной модели.

Измерение «зрелость» определяет развитие зрелости процесса в терминах атрибутов процесса, сгруппированных в уровни зрелости. Атрибуты процессов - это оценка меры зрелости процесса по соответствующей шкале. Каждый атрибут, с одной стороны, отражает зрелость руководства, а, с другой стороны, дает возможность оценить эффективность процесса и его вклад в достижение целей организации. Уровень зрелости характеризуется набором атрибутов, совместный анализ которых дает возможность существенно улучшить реализацию процесса. Измерение «зрелость» предоставляет рациональный путь улучшения процессов.



Типы – (Б)- базовый, (Н)- новый, (Р) – расширяемый

Рисунок 3.9- Измерение «процесс» эталонной модели

В модели имеется шесть уровней зрелости (рисунок 3.10):

Уровень 0 - Неполный (Incomplete). Процесс не соответствует своему назначению. Рабочие продукты и итог процесса отсутствуют или их невозможно выявить.

Уровень 1 - Выполняемый (Performed). Процесс в целом соответствует своему назначению. Его выполнение не может быть строго спланировано и отслежено. Сотрудники организации знают, что мероприятие должно быть осуществлено. Существует общее соглашение , как и когда его выполнить.

Рабочие продукты процесса определены. Достижение результата процесса определяется рабочими продуктами.

Уровень 2 - Управляемый (Managed). Процесс планируется и отслеживается. В соответствии с его процедурами выпускаются рабочие продукты, которые соответствуют определенным требованиям и стандартам. Рабочие продукты соответствуют определенным требованиям к качеству при заданных ограничениях по времени и ресурсам (это основное отличие от Уровня1).

Уровень 3 -Устоявшийся (Established). Выполнение и управление процесса определяется последовательностью действий, построенных на основе надлежащих принципов разработки ПС. Индивидуальная реализация процесса используются на основе адаптации стандартного, документированного процесса к конкретным условиям. Выделяются необходимые ресурсы для адаптации процессов. Основное отличие от уровня 2 состоит в том, что процесс уровня 3 использует стандартный, документированный процесс с заданным результатом.

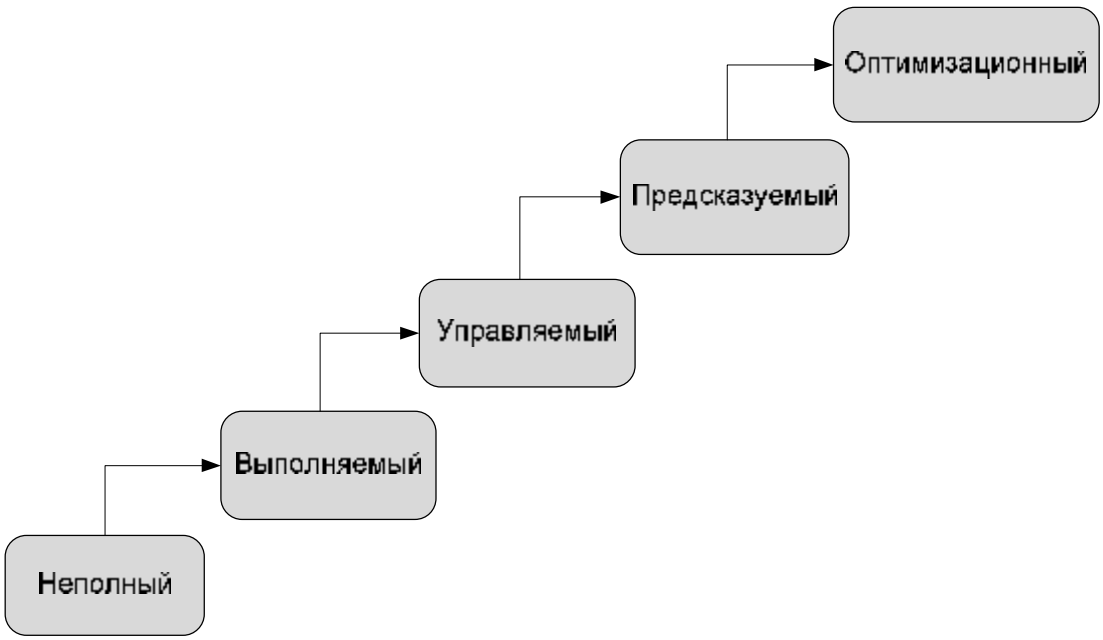


Рисунок 3.10 - Уровни зрелость процесса

Уровень 4 - Предсказуемый (Predictable). На практике процесс осуществляется в заданных рамках, устанавливающих определенные цели процес-

са. Собираются и анализируются измеряемые показатели производительности. Это дает количественное понимание возможностей процесса с предсказуемой и управляемой производительностью. Качество продукции определено количественными показателями. Основное отличие от уровня 3 состоит в том, что заданный процесс осуществляется в предписанных рамках, определенных результатами процесса.

Уровень 5 - Оптимизируемый (Optimizing). Бизнес – цели процесса достигаются всегда. Оптимизируется производительность процесса на основе текущих и будущих потребностей бизнеса. В соответствии с бизнес - целями в организации устанавливаются количественные ориентиры (цели) процесса, определяющие его эффективность. Мониторинг выполнения процесса и анализ данных, полученных при этом, используется в качестве обратной связи для его совершенствования. Оптимизация процесса включает апробирование инновационных идей и технологий, проводя при этом изменение неэффективных процессов с точки зрения заданных целей и задач. Основное отличие от уровня 4 состоит в том, что определенные и стандартные процессы претерпевают постоянные уточнения и улучшения текущими и будущими бизнес - целями.

Принципиальное отличие измерения «зрелости» ИСО/МЭК ТО 15504 от «Модели зрелости процессов разработки ПС» (SW CMM) состоит в объекте рассмотрения. В ИСО/МЭК ТО 15504 объектом измерения зрелости является процесс жизненного цикла ИСО 12207. Уровень зрелости его оценивается значениями соответствующих атрибутов. В «Модели зрелости процессов разработки ПС» (SW CMM) объектом измерения зрелости является жизненный цикл разработки программного обеспечения. Уровни зрелости его оцениваются присутствием в нем заданных на каждом уровне процессов. Нас опять вводит в заблуждение слишком вольная трактовка англоязычных терминов. «Capability maturity model for software» - буквально «модель зрелости функциональных возможностей (точнее возможности исполнять функции)

разработки программного обеспечения». Из перевода понятно, что «зрелость» относится не к процессам, из которых состоит разработка, а ко всей разработке, или как мы теперь можем сказать, к жизненному циклу программного обеспечения. В отечественной литературе закрепился термин «Модель зрелости процессов разработки программного обеспечения» /19/, звучит красивее, чем буквальный перевод, но перевирает смысл.

В измерении «зрелость» эталонной модели мера зрелости основывается на наборе атрибутов процессов (process attribute, PA). Атрибуты процессов используются для того, чтобы определить , достиг ли процесс заданного уровня. Каждый атрибут является мерой конкретного аспекта зрелости процесса. Атрибуты, в свою очередь, оцениваются в процентах, что дает дополнительное понимание конкретных аспектов зрелости процессов, необходимое для усовершенствования процессов и определения их зрелости. На рисунке 3.11 приведены атрибуты для каждого уровня зрелости. Для каждого уровня зрелости предполагается определенное значение рейтинга каждого атрибута. Так, например, рейтинг F - Fully achieved (обладает полностью) - соответствует наивысшему рейтингу атрибута. Для каждого последующего уровня зрелости предполагается, что рейтинг атрибутов процесса, перечисленных на всех предыдущих уровнях, равен F (обладает полностью). Например, для процесса, аттестованного на 3 уровне зрелости («управляемый»), рейтинг атрибутов: выполнение процесса, управление выполнением и управление рабочими продуктами должен быть, обязательно, равен F.

Подводя итог описания эталонной модели необходимо отметить, что ИСО/МЭК ТО 15504 дает возможность выбрать и адаптировать процессы разработки программного обеспечения в соответствии с целями и задачами бизнеса, а также в соответствии с уровнем зрелости разворачиваемых процессов.

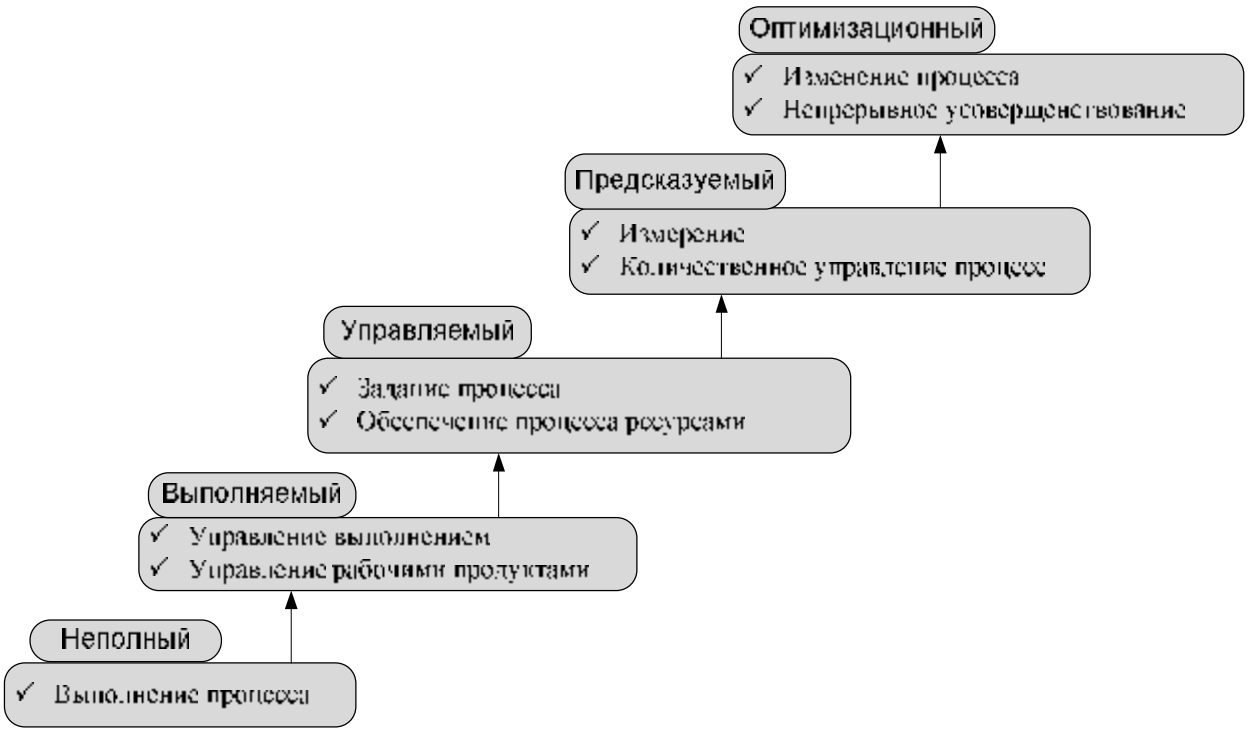


Рисунок 3.10- Атрибуты процессов (PA) для каждого уровни зрелости

Жизненный цикл разработки программного обеспечения в модели CMMI

Модели развития функциональных возможностей (CMMs) содержат описание эффективных процессов с точки зрения различных областей знаний. Элементы моделей основаны на концепциях, развитых Кросби, Демингом, Джураном и Хэмфри (Crosby, Deming, Juran, and Humph). Подобно выше описанной модели CMM, «Интегрированные модели развития функциональных возможностей (CMMI) » представляет руководство для использования в процессах разработки. CMMI модели - не процессы или описание процессов организации. Выполнение реальных процессов в организации зависят от многих факторов, включая особенности прикладной области, структуру и размер организации. Области процесса модели CMMI не могут быть отображены один к одному в процессы, используемые в организации.

В моделях CMMI процесс рассматривается с точки зрения приложения усилий, с целью достижения непрерывного совершенствования органи-

зации. Цель «Интегрированной СММ» состоит в том, чтобы представить пользователю руководство для улучшения процессов в организации, а также способствовать управлению развитием, приобретением и обслуживанием изделий или услуг. При этом используются проверенные мировой практикой структурные решения, помогающие предприятию оценить ее организационную зрелость, установить приоритеты совершенствования процессов и реализовывать их.

Разработчики определяют CMMI, как комплекс продуктов CMMI (CMMI 1.1 включает 8 технических отчетов /35-42/). Контент комплекса дает возможность: генерировать сложные модели, обучить их использованию, определить материалы для оценки.

Используется два различных представления: непрерывное (Continuous) и поэтапное (Staged).

Непрерывное представление рекомендуется использовать:

- при выборе очередности модернизации в соответствии с бизнес – целями, что позволяет снизить организационные риски;
- при сравнении организаций на основе сопоставления пространства процессов, или же сопоставлением результатов выполнения эквивалентных фаз;
- для обеспечения легкого перехода от стандарта EIA/IS 731 к CMMI;
- при применении стандарта ISO/IEC 15504 для оценки улучшения процесса, так как организация пространства процессов в CMMI и ISO/IEC 15504 подобны.

Поэтапное представление рекомендуется использовать в случаях, когда :

- необходимо выбрать проверенные опытом мероприятия по модернизации предприятия, как последовательное движение от

уровня к уровню, при этом предыдущий уровень является базой развития для последующего развития;

- оценка предприятий на основе оценки их уровней зрелости производства;
- переход от SW-CMM к CMMI;
- необходимо провести оценку предприятия, результат которого будет использоваться при сравнение его с другим.

Модели отображают содержание различных сводов знаний (например, системная инженерия, программная инженерия, интегрированная разработка продукта и процесса) или в их комбинациях (например, CMMI-SE/SW, CMMI-SE/SW/IPPD). Кроме описанных представлений, выбор модели будет зависеть от области знаний, в контексте которой необходимо рассмотреть модель. Доступно 4 области знания, которые называются «Дисциплина»:

- системная инженерия (SE);
- программная инженерия (SW);
- интегрированная разработка продукта и процесса (IPPD);
- поиск поставщиков (SS).

Дисциплина «Системная инженерия» выбирается при проектировании систем, вне зависимости от того, используется ли ПО в данных системах или не используется. Системная инженерия фокусирует преобразование потребностей клиента, его ожиданий и ограничений в решения по выпуску продукции, а также в решения по поддержке этой продукции на протяжении ее жизни.

Дисциплина «Программная инженерия» распространяется на разработку программного обеспечения. В соответствии с этим, методы процессов специфицируются с точки зрения поставленных задач.

Интегрированная разработка продукта и процесса (IPPD) - систематический подход, в котором используется сотрудничество с заинтересованными сторонами на протяжении всей жизни продукции, с целью качественного

удовлетворения потребностей, ожиданий и требований клиента. Процессы, поддерживающие методы IPPD, должны быть интегрированы с другими процессами в организации. Модели CMMI мы рассмотрим на примере методов и практик этой дисциплины.

Дисциплина и представление определяют соответствующее описание CMMI модели /35-42/. При выборе дисциплины модель будет содержать: менеджмент процессов, менеджмент проекта, процессы области поддержки и инжиниринга, где определенные методы интерпретируются с точки зрения выбранной дисциплины.

Выбор модели - это выбор соответствующего документа из перечисленной группы. Как предполагают разработчики, основные дисциплины и их комбинация будут расширяться. Так, например, дисциплина «Поиск поставщиков» (Supplier Sourcing) появилась несколько позднее. Она отражена в двух технических отчетах, отражающих непрерывную и поэтапную модели для набора дисциплин «Системная инженерия», « Инженерия ПО», «Интегрированная разработка продукта и процесса» и «Поиск поставщиков» (CMMI-SE/SW/IPPD/SS, V1.1).

Тем самым, единая модель рассматривает процессы разработки и выпуска продукции в трех измерениях: дисциплина (или группа дисциплин), непрерывное представление, поэтапное представление.

Все технические отчеты построены по единой схеме, которая включает общие разделы:

Предисловие;

1 раздел — Введение;

2 раздел — Модель компонентов;

3 раздел — Терминологическая модель;

4 раздел — Функциональные возможности уровней и обобщенная модель компонентов (для непрерывной модели). Общие функциональные воз-

можности, обобщенные цели и обобщенные практика (для поэтапной модели);

5 раздел — Структура взаимодействия процессов; аннотированы четыре категории процессов раздела 7, их общий обзор и схемы взаимодействия CMMI процессов;

6 раздел — Использование модели CMMI. Краткие рекомендации для пользователей по применению модели и обучению. Отмечается совместимость и соответствие процессов модели с регламентированными процессами предыдущей модели CMM , стандартами ISO 15504 и EIA/IS 731;

7 раздел — Область процесса. Последний, самый большой в каждом стандарте, он занимает около 500 страниц из полного объема документа, который составляет свыше 700 страниц. В этом разделе представлены подробные рекомендации для реализации каждого из перечисленных в нем процессов, которые учитывают особенности конкретной модели.

В пяти приложениях приводятся:

A — состав использованных литературных источников и документов;

B — сокращения;

C — глоссарий на основе терминологии ISO, применяемой только в четырех стандартах ISO 9000, ISO 12207, ISO 15504:1-9, ISO 15288;

D — описания требований и предложений для формирования компонентов модели по уровням зрелости;

E — список участников разработки CMMI — проекта.

Прежде, чем мы с Вами начнем разбирать структуру стандарта, необходимо обратить внимание на ту сложную задачу, которую поставили перед собой разработчики модели CMMI. С одной стороны, стандарт должен быть согласован с моделью CMM (принцип обратной совместимости), а с другой стороны, он должен быть согласован со стандартом ISO 15504. Напомню, в модели CMM разработка программного обеспечения подразделяется на 5 уровней зрелости в зависимости от того, какие процессы используются при

разработке. В ISO 15504 каждый процесс, участвующий в жизненном цикле, может быть отнесен к одному из пяти уровней зрелости в зависимости от того, какое значение имеют его атрибуты, т.е. в зависимости от его реализации. Понятно, что уровни зрелости CMM и ISO 15504 должны быть связаны и, эта связь должна быть установлена моделью CMMI. Разработчики CMMI решили эту задачу довольно оригинально. Они построили два представления: непрерывное и поэтапное, о чем говорилось выше. В непрерывном представлении процессы жизненного цикла продукции делятся на 6 уровней зрелости (Таблица 3.4).

Таблица 3.4 Уровни возможностей процессов в непрерывном представлении

Уровни возможностей	Название
0	Неполный (Incomplete)
1	Исполняемый (Performed)
2	Управляемый (Managed)
3	Определенный (Defined)
4	Количественно управляемый (Quantitatively Managed)
5	Оптимизационный (Optimizing)

Количество уровней, если сравнивать с ISO 15504, формально увеличилось с 5 до 6, добавлен 0 уровень. В тоже время, нулевой уровень отделяет процессы, которые не могут быть рассмотрены в модели ISO.

Все процессы объединены в группы. Для дисциплины IPPD (интегрированная разработка продукта и процесса) используют следующую группировку процессов:

Менеджмент процессов:

- Содержание организационных процессов;
- Определение организационных процессов;
- Организация обучения;

- Организация преобразования (изменений) процессов;
- Организация инноваций и расширений;

Управление проектом:

- Планирование проекта;
- Мониторинг и контроль процессов проекта;
- Управление соглашениями с поставщиками;
- Интегрированное управление процессами и продуктами проекта;
- Управление рисками;
- Интеграция команды разработчиков;
- Интегрированное управление поставщиками;
- Количественное управление проектом;

Инженерия (технология):

- Управление требованиями;
- Разработка требований;
- Технические решения;
- Интеграция продукта;
- Верификация;
- Валидация (аттестация, утверждение);

Поддержка:

- Управление конфигурацией;
- Обеспечение качества процессов и продуктов;
- Измерение и анализ процессов и продуктов;
- Анализ и принятие решений на изменения;
- Организация окружения для интеграции;
- Анализ причин и разрешение проблем (устранение дефектов).

Как видно, данная группировка процессов не согласуется с группировкой процессов в ISO 12207.

Для каждого процесса определена область описания этого процесса (Area Process) или просто область процесса. В связи с тем, что представление в моделях разное, структурные единицы в каждом конкретном описании тоже разные. В непрерывной модели структурной единицей является область процесса. На рисунке 3.11 показана структура описания областей процессов.

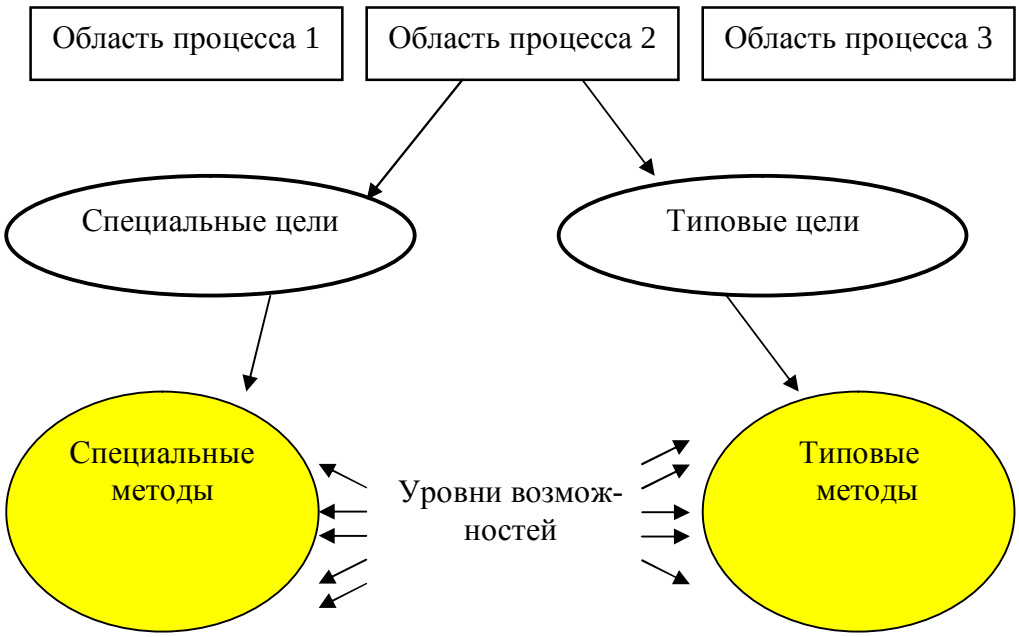


Рисунок 3.11- Структура описания непрерывного представления.

В зависимости от поставленных общих и специфических целей и реализованных типовых и специфических методов, процесс квалифицируется по уровням возможностей. В качестве примера, в таблице 3.5, приведены типовые цели и методы.

Из выше приведенного описания становится понятным, почему данная модель называется непрерывной. В той организации, в которой предполагается использование этой модели, принимается решение использовать процессы, выбранные из соответствующих групп.

Таблица 3.5 Типовые цели и методы процесса в зависимости от функционального уровня

Название уровня	Типовые цели	Типовые методы
Исполняемый	GG 1 Достигнуть специальных целей	GP 1.1 Выполнение основных методов
Управляемый	GG 2 Институционализировать управление процессом	GP 2.1 Устанавливать организационную политику
		GP 2.2 Планировать процесс
		GP 2.3 Обеспечить ресурсы
		GP 2.4 Распределить ответственность
		GP 2.5 Обучить людей
		GP 2.6 Управлять конфигурациями
		GP 2.7 Идентифицировать и вовлечь важные заинтересованные стороны
		GP 2.8 Контролировать и управлять процессом
		GP 2.9 Объективно оценивать соблюдения правил
		GP 2.10 Проверять статус с более высоким уровнем менеджмента
Определенный	GG 3 Институционализировать определенный процесс	GP 3.1 Устанавливает определенный процесс
		GP 3.2 Собирает информацию, способствующую усовершенствованию
Количественное управление	GG 4 Институционализировать процесс «количественное управление»	GP 4.1 Устанавливает количественные цели для процесса
		GP 4.2 Стабилизирует выполнение подпроцесса
Оптимизационный	GG 5 Институционализировать процессы оптимизации	GP 5.1 Гарантировать непрерывное усовершенствование процесса
		GP 5.2 Исправлять корневые причины проблем

При реализации области выбранных процессов ставятся соответствующие специальные цели, которые исполняются специальными методами, а также типовые цели, исполняемые типовыми методами. Реализация специальных методов и набор типовых методов определяет уровень возможностей выбранного процесса. Последующее совершенствование процесса и переход

его на другой уровень может проводиться в организации непрерывно по мере накопления опыта. При этом меняются реализация специальных методов и добавляются методы типовые. Так, например, на первом (Исполняемом) уровне требуется выполнить один типовой метод, а на втором (Управляемом) уровне требуется выполнить еще дополнительно десять типовых методов. Эта работа может проводиться в организации непрерывно, эволюционным путем.

В поэтапной модели CMMI, в соответствии с моделью CMM, устанавливаются 5 уровней зрелости жизненного цикла продукции (Таблица 3.6.).

Таблица 3.6 Уровни зрелости жизненного цикла продукции в поэтапном представлении

Уровни зрелости	**Название**
1	Начальный (Initial)
2	Управляемый (Managed)
3	Определенный (Defined)
4	Количественно управляемый (Quantitatively Managed)
5	Оптимизационный (Optimizing)

Первый уровень отличается значительной неопределенностью состава и содержанием процессов в различных относительно простых проектах. В документе он не комментируется. Поэтому, при уточнении и детализации содержания процессов в поэтапном варианте CMMI рекомендуется ограничиваться четырьмя основными уровнями:

Второй уровень — формализует базовое управление проектами и включает следующие процессы:

- управление требованиями;
- планирование проекта;
- мониторинг и контроль проекта;
- управление соглашениями с поставщиками;

– измерение и анализ процессов и продуктов;
– обеспечение качества процессов и продуктов;
– управление конфигурацией.

Третий уровень — содержит стандартизацию основных процессов:

– разработка требований;
– технические решения;
– интеграция продукта;
– верификация;
– валидация (аттестация);
– содержание организационных процессов;
– определение организационных процессов;
– организация обучения;
– интегрированное управление процессами и продуктами проекта;
– управление рисками;
– интеграция команды разработчиков;
– интегрированное управление поставщиками;
– анализ и разрешение проблем (устранение дефектов);
– организация окружения для интеграции.

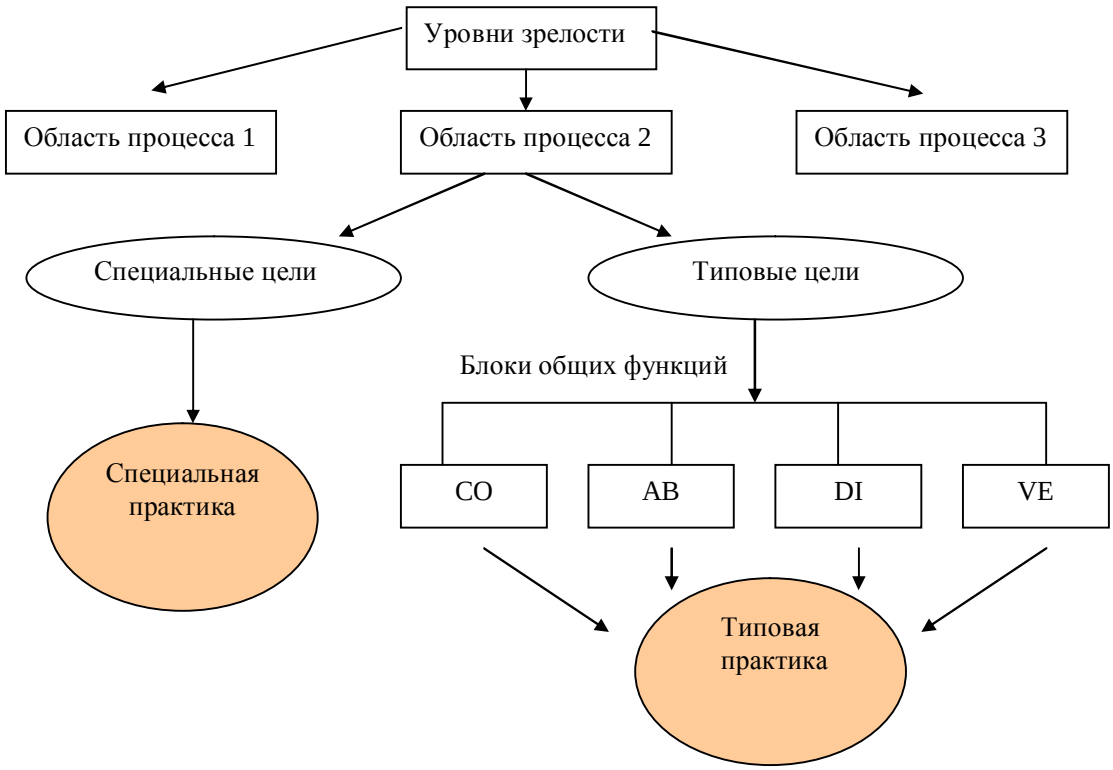
Четвертый уровень — определяет количественное управление и включает следующие процессы:

– организация представления качества процессов;
– количественное управление всем проектом и ресурсами.

Пятый уровень — оптимизационный, непрерывное совершенствование, включает следующие процессы:

– организация, инновация, количественное управление процессами и обеспечение ресурсами;
– анализ причин дефектов, совершенствование качества и управления процессами и продуктами;

Единицей структурного описания поэтапного представления является уровень зрелости, который определяет область процесса (рисунок 3.12). В области процесса определены специальные и типовые (общие, с точки зрения уровня функциональных возможностей) цели.



CO - обязательства выполнения;
AB - возможности выполнения;
DI - реализация управления;
VE - реализация верификации.

Рисунок 3.12 – Организация описания структурной единицы поэтапного представления.

Типовые цели определяют типовые функции, которые объединены для каждой области процесса в четыре блока. Блоки общих функций - компоненты модели, используемые для логической группировки типовых методов, а не для определения порядка их следования. Блоки общих функций подразделяются :

- обязательства выполнения - Commitment to Perform (CO);
- возможности выполнения - Ability to Perform (AB);

- реализация управления - Directing Implementation (DI);
- реализация верификации - Verifying Implementation (VE).

Из представленного описания становится понятным определение данного представления, как «поэтапное». Действительно, для каждого уровня зрелости определяется набор процессов. Организация, планирующая разработку программного продукта или же производства любой продукции, исходя из выбранного уровня зрелости, определяет набор процессов, которые она будет реализовывать. При переходе с одного уровня зрелости на другой, она должна снова спланировать новый набор процессов, реализация которых будет проводиться на следующем ЭТАПЕ ее развития.

С точки зрения поддержки и регламентирования полного жизненного цикла крупных проектов программных средств, к недостаткам моделей CMMI, относительно профиля существующих стандартов ISO, можно отнести следующие /44/:

- не все процессы предусмотрены в составе процессов моделей CMMI — 1.1, которые развиваются и детально комментируются для их реализации в стандартах ISO 9004:2000 и ISO 90003:2004;
- не отражены особенности системной инженерии и международные стандарты, регламентирующие процессы жизненного цикла сложных систем ISO 15288:2002 и ISO 19760:2003;
- при анализе процессов обеспечения качества используется ряд традиционных характеристик систем и программных продуктов, которые применяются в сложных проектах, но не описаны и не комментируются базовые международные стандарты, систематизирующие и регламентирующие качество программных средств — ISO 9126:1-4, ISO 14598:1-6, ISO 15939;
- отсутствуют описания характеристик и конкретных процессов обеспечения информационной и функциональной безопасности

программных продуктов и ссылки на многочисленные стандарты в этой области;

- не отражены регламентированные интерфейсы открытых систем на взаимодействие программных компонентов, а также с операционной и внешней средой, в соответствии со стандартами ISO 9945:1-4;
- документирование процессов и продуктов ЖЦ ПС комментируется только по мере их реализации и не представлены обобщенные требования к технологической и эксплуатационной документации на программный продукт в соответствии со стандартами ISO 9294, ISO 15910, ISO 18019.

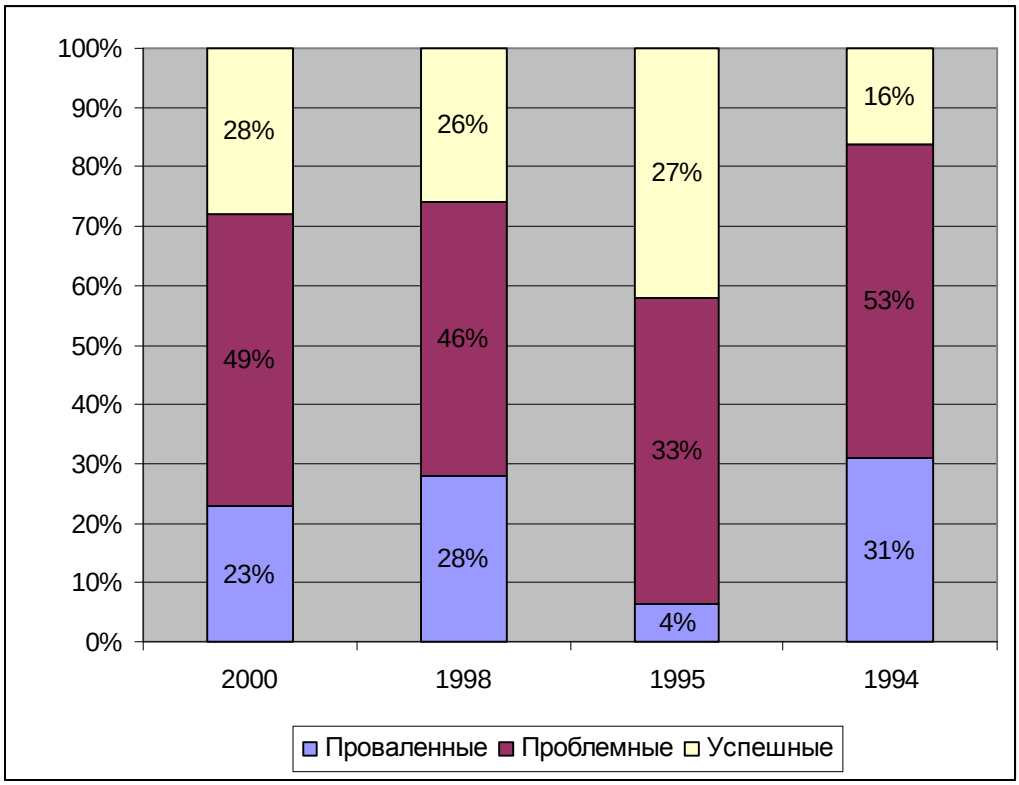
Заключение

В данной главе представлены различные взгляды на жизненный цикл программного обеспечения и систем. Выбор процессов, действий и задач зависит от сложности проекта, зрелости организации, существования корпоративных стандартов этой организации, зрелости заказчика и многих других факторов. Интеграционные тенденции, которые прослеживаются в международных стандартах и мировой практике, определяют жизненный цикл программного продукта и информационных систем в едином информационном пространстве, в котором объединяются различные области знаний и лучший опыт за счет унификации терминологий и целей. Незнание жизненного цикла систем и ПО или неумение применять эти знания, может привести к серьезным потерям ресурсов любого проекта, любого состава исполнителей.

Может показаться, что процессы менеджмента качества, которые и определяют вопросы организации жизненного цикла продуктов, не должны затрагивать профессиональную работу программистов. Возражение можно найти в статистике выполнения программных проектов /45/:

– ежегодно более $200 млрд США тратит на более, чем 170 тыс. проектов разработки ПО в сфере IT;
– 31,1% из них закрываются, так и не завершившись;
– 52,7% проектов завершаются с превышением первоначальных оценок бюджета/сроков и ограниченной функциональностью;
– потери от недополученного эффекта внедрения ПО измеряются триллионами.

Статистика по 30000 проектам по разработке ПО в американских компаниях представлена на рисунке 3.13.



:

- Успешные – своевременно и в рамках бюджета был выполнен весь намеченный фронт работ
- Проблемные – нарушение сроков, перерасход бюджета и/или сделали не все, что требовалось
- Проваленные – не доведенные до конца из-за перерасхода средств, бюджета, качества.

Рисунок 3.13 – Статистика выполнения программных проектов в 1994-2000гг.

Контрольные вопросы

1. Что такое процессный подход?
2. Чем процессный подход отличается от проектного?
3. Чем процессный подход отличается от операционного?
4. Для чего используется цикл Деминга?
5. Что такое жизненного цикл продукта, услуги?
6. Почему понятие жизненного цикла актуально в современном производстве?
7. Чем отличаются подходы в определении жизненного цикла в различных стандартах?
8. Какова цель ГОСТ Р ИСО/МЭК 12207-99?
9. Что такое модель CMM и чем она отличается от модели CMMI?
10. Что общего в подходах CMMI и ИСО/ МЭК ТО 15504?
11. Что такое SLCM, SLC, SLCP?

4. Модели жизненного цикла

Прежде, чем мы с Вами рассмотрим различные модели, давайте разберем понятие «Модель жизненного цикла» (life cycle model - LCM).

Рассмотрим несколько определений:

ГОСТ Р ИСО/МЭК 12207-99 - «Модель жизненного цикла: структура, состоящая из процессов, работ и задач, включающих в себя разработку, эксплуатацию и сопровождение программного продукта, охватывающая жизнь системы от установления требований к ней до прекращения ее использования». В стандарте IEEE 1074 определяется модель жизненного цикла программного обеспечения.

ГОСТ Р ИСО/МЭК 15288—2005 - 4.8 – «Модель жизненного цикла: структурная основа процессов и действий, относящихся к жизненному циклу, которая также служит в качестве общей ссылки для установления связей и взаимопонимания сторон».

IEEE 1074 - «Модель жизненного цикла ПО (SLCM или МЖЦ ПО): выбранная организацией структура, на которую наносится последовательность действий стандарта IEEE 1074, с целью создания жизненного цикл ПО (SLC или ЖЦ ПО).

Исходя из рассмотренных определений, следует, что модель жизненного цикла задает некоторую структуру системы, которая используется для определения последовательности выполнения процессов, работ и задач. Если определена последовательность выполнения процессов, действий и задач для данной системы, то определен жизненный цикл конкретной системы. Модель жизненного цикла системы - это обобщенное описание жизненного цикла многих систем, в которых последовательность процессов, действий и задач подобна.

В ГОСТ Р ИСО/МЭК 12207-99 первой задачей в первом действии процесса «разработка» разработчик должен определить или выбрать модель

жизненного цикла программных средств, соответствующую области реализации, величине и сложности проекта. Правда, «первая задача» в «первом действие» не определяет, что эта задача будет выполняться обязательно первой во времени, но, тем не менее, эту задачу разработчик обязан выполнить в начале проекта. Где же разработчику брать эти модели? Ответ простой - в литературе.

Но зададимся вопросом: для чего требуется определить жизненный цикл системы. Ответ, следующий из ГОСТ Р ИСО/МЭК 12207-99, вполне логичный: для выполнения разработки этой системы. Но тогда возникает вопрос, а разве нельзя разрабатывать ПО без определения модели жизненного цикла? Каждый студент к четвертому курсу разрабатывал или пробовал разрабатывать программное обеспечение, при этом, скорее всего, он не имел представление о ЖЦ ПО. В Р. Фаттрел /47/ очень остроумно иллюстрирует модель «Делать, пока не будет сделано», приведенную на рисунке 4.1.

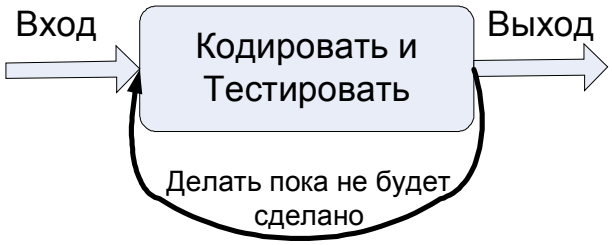


Рисунок 4.1 – Модель процесса «Делать, пока не будет сделано».

Что в этой модели не устраивает? Не устраивает мелочь – этот процесс не управляем. Следствие этого - в истории развития программирования, так называемый, «Кризис программирования», начавшийся в 60 -70гг. Результатом кризиса стало изобретение многих прогрессивных методологий в области разработки ПО. Нас интересует область знания software engineering (программная инженерия). Термин «программная инженерия» - впервые был озвучен в октябре 1968 года на конференции подкомитета НАТО по науке и технике (г.Гармиш, Германия), на которой присутствовало 50 профессиональных разработчиков ПО из 11 стран. Рассматривались проблемы проекти-

рования, разработки, распространения и поддержки программ. На встрече анализировались проблемы и перспективы развития ПО. Отмечалась возрастающее воздействие ПО на жизнь людей. Серьезно ставился вопрос об усовершенствовании применяющихся принципов и методов разработки ПО. Вскоре была предложена концепция жизненного цикла ПО (SLC – Software Lifetime Cycle), как последовательность шагов-стадий, которые необходимо выполнить в процессе создания и эксплуатации ПО. В 1970 г. У.У. Ройс (W.W. Royce) произвел идентификацию нескольких стадий в типичном цикле и высказал предположение, что контроль выполнения стадий приведет к повышению качества ПО и сокращению стоимости разработки. В этом же году каскадная модель была предложена как альтернатива принципа кодирования – устранение ошибок. Это была первая модель, которая формализовала структуру этапов разработки ПО.

В настоящее время существует множество моделей жизненного цикла. Из них можно выделить три фундаментальные /16, 18/. Этими фундаментальными моделями жизненного цикла являются:

- каскадная;
- инкрементная;
- эволюционная.

Каждая из указанных моделей может быть использована самостоятельно или скомбинирована с другими для создания гибридной модели жизненного цикла. Конкретную модель жизненного цикла следует выбирать так, чтобы процессы, работы и задачи были связаны между собой и определены их взаимосвязи с предшествующими процессами, работами (видами деятельности) и задачами (заданиями).

В третьей главе мы подробно разобрали понятие жизненного цикла. Те стандарты, которые мы разбирали, описывают жизненный цикл различных объектов. Нас будет в дальнейшем интересовать жизненный цикл проекта. В тоже время, соотношение понятий жизненного цикла проекта, продукта и

операционной деятельности[4] предприятия довольно интересно, так как затрагивает модель жизненного цикла проекта.

Без привязки к конкретной модели жизненный цикл проекта разделяют на фазы, которые связывают начало проекта с его завершением. Под фазой проекта /3/ (Project Phase) понимают ряд логически связанных операций[5] проекта, обычно завершающихся достижением одного из основных результатов поставки. Фаза проекта является элементом жизненного цикла проекта. Она не является группой процессов управления проектами. Фаза проекта может разделяться на подфазы и далее - на элементы, то есть фаза проекта может иметь иерархическую структуру. Обычно для обозначения подфазы используют специальный термин, например: этап, стадия, итерация и др.

В ГОСТ 34 и ГОСТ 19 используется понятие «стадия», как синоним фазы, и «этап», как элемент стадии. Стадии и этапы создания АС выделяются как части процесса создания по соображениям рационального планирования и организации работ, заканчивающихся заданным результатом /14/.

Какой бы термин не использовался в том или ином стандарте, Вы должны представлять иерархическую структуру организации работ проекта. Путаница в терминологии, связанная с моделью жизненного цикла, может привести к различному пониманию целей и задач, на различных временных участках выполнения проекта. Мы часто будем использовать термин «фаза», в силу того, что он дает большую свободу в описании жизненного цикл проекта (стандартами ГОСТ 34 и ГОСТ 19 понятия стадия и этап привязаны к конкретным работам).

В «Руководстве к своду знаний по управлению проектами» /46/ отмечается, что фаза проекта характеризуется завершением и одобрением одного

---

[4] Операционная деятельность (Operations) Организационная функция, осуществляющая непрерывное выполнение операций, которые производят один и тот же продукт или предоставляют одну и ту же услугу. В качестве примеров можно привести: производственные операции, бухгалтерские операции./3/

[5] Синонимы - работа (ИСО/МЭК 12207), действия (ИСО/МЭК 15288), activity

или нескольких результатов поставки. Под результатом поставки понимается некоторый продукт (иногда его называют артефакт), например, спецификация, отчет по анализу осуществимости, детальный план или опытный образец. Основная задача, которая должна быть решена с помощью этих продуктов - измерить и проверить прогресс проекта. Результаты поставки, а значит и фазы, являются частью общего последовательного процесса, предназначенного для обеспечения необходимого контроля над проектом и получения нужного продукта или услуги с заданным качеством. На рисунке 4.2 показано соотношение между фазами проекта, входами и выходами проекта.

Рисунок 4.2 – Соотношение между фазами проекта и результатами их выполнения (результатами поставки) /46/

Очень часто трудно определить - с чего начинается проект? В различных методологиях на этот вопрос отвечают по-разному. Как мы видим на приведенном рисунке, проект начинается с идеи, при этом проработка целесообразности проекта выполняется на начальной фазе. Подобный подход мы можем найти в ГОСТ 34 - стадии создания: «Формирование требований к АС» и «Разработка концепции АС» ГОСТ 34, а также в методологии RUP (Rational Unified Process) - фаза «Начало». В тоже время в «Своде знаний PMI PMBOK» указывается на то, что организация, обнаруживая благоприятную возможность, которую она хотела бы использовать, часто авторизует анализ осуществимости чтобы решить, следует ли браться за выполнение

проекта. Когда результат этого предварительного анализа не очевиден, лучше выделять его в отдельный проект.

Еще раз важно отметить, что особенность организации фаз жизненного цикла проекта, разобранная выше, не привязана к конкретному жизненному циклу. В любой модели жизненного цикла проект во времени делится или на фазы, или стадии, или этапы и т.д.(название определяется терминологией данной модели). Фазы - это способ рассмотрения прогресса проекта во времени, поэтому, фазы жизненного цикла проекта не совпадают с группами процессов управления проектом, подробно описанными в главе 3.

Соотношение жизненного цикла продукта, проекта и операционной деятельности предприятия

Выявление стадий, этапов, фаз и операций и т.д. требуется для определения целей, задач и результатов каждого элемента этой иерархической структуры. Как мы выяснили выше, создание этой иерархии требуется процессам управления проектом, выполнимых с целью повышения качества продукта. Важно понять, что термин «Качество» относится не только к конечному продукту, но и к организации работ по выпуску этого продукта, более того, концепция качества группы стандартов ИСО 9000 заключается в том, что качество продукции определяется качеством производства этой продукции, то есть задачи организации производства ставятся на первое место.

Организации выполняют работы для достижения ряда целей. Работы можно представлять как проекты или как операции, хотя они иногда могут пересекаться.

Обычно деятельность предприятия представляют как операционную или процессную деятельность, которая протекает довольно долгий период. Если задачи процесса или операции выполнены, то ставятся новые задачи и процесс продолжается далее.

Проектная деятельность по своему определению это временное предприятие. То есть, проект имеет определенные границы начала и завершения проекта. Это главное отличие проекта от операционной деятельности: проект - временное предприятие, а операционная деятельность – это продолжающийся и повторяющийся процесс во времени. Конечные цели проекта и операционной деятельности также существенно отличаются. Задача проекта - достижение поставленной цели, после чего проект завершается. Операционная деятельность, напротив, обычно служит для обеспечения нормального течения бизнеса. Проект завершается после выполнения поставленных конкретных задач, в то время, как операции получают новые цели и продолжают выполняться.

Понятие «проектная деятельность» и «операционная деятельность» весьма сильно переплетается. Любой проект состоит из процессов. Основу любого проекта составляет структура пооперационного перечня работ (wогк breakdown structure, WBS). С другой стороны, проекты часто используются в качестве средства выполнения стратегического плана организации, следовательно, являются частью ее операционной деятельностью.

Для того, чтобы увязать проект как временное предприятие с продолжительной операционной деятельностью, необходимо в проекте иметь, по крайне мере, две фазы: инициализации и финальная. В определении жизненного цикла проекта указывается, какие переходные операции при завершении проекта включаются в него, чтобы связать проект с текущими операциями исполняющей организации.

Следует различать жизненный цикл проекта и жизненный цикл продукта. Например, проект, предпринимаемый с целью разработки системы поддержки документооборота, является лишь одним из аспектов жизненного цикла продукта (системы документооборота). На рисунок 4.2 показан жизненный цикл продукта, начиная с бизнес-плана, идеи, до продукта, текущих операций и реализации продукта.

Жизненный цикл проекта состоит из серии фаз создания продукта. Могут быть инициированные проекты по модификации продукта (в нашем примере, системы документооборота). В некоторых областях приложения, например, в разработке новой продукции или разработке программного обеспечения, организации считают жизненный цикл проекта частью жизненного цикла продукта. Стандарты ГОСТ 34 и ГОСТ 19 включают составление бизнес – плана в жизненный цикл проекта, за счет включения работы «оценку (технико-экономической, социальной и т.д.) целесообразности создания АС» в этап «Обследование объекта и обоснование необходимости создания в АС» (ГОСТ 34.601-90).

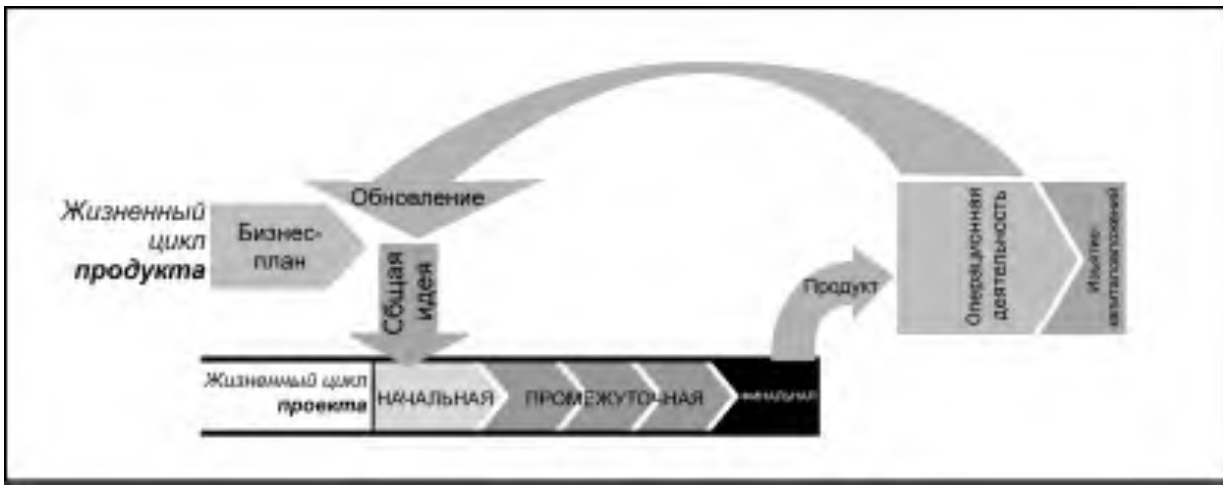


Рисунок 4.2- Соотношения жизненного цикла продукта и проекта

Каскадная модель

Концепция каскадной модели.

Каскадная модель жизненного цикла по существу реализует принцип однократного выполнения каждого из следующих видов деятельности в их естественных границах:

- установление потребностей пользователя;
- определение требований;
- проектирование системы;
- изготовление системы;

– испытание;

– корректировка;

– поставка или использование.

Количество стадий может меняться весьма в широких пределах в зависимости от задач проекта. Обычно ее изображают в виде водопада (рисунок 4.3)

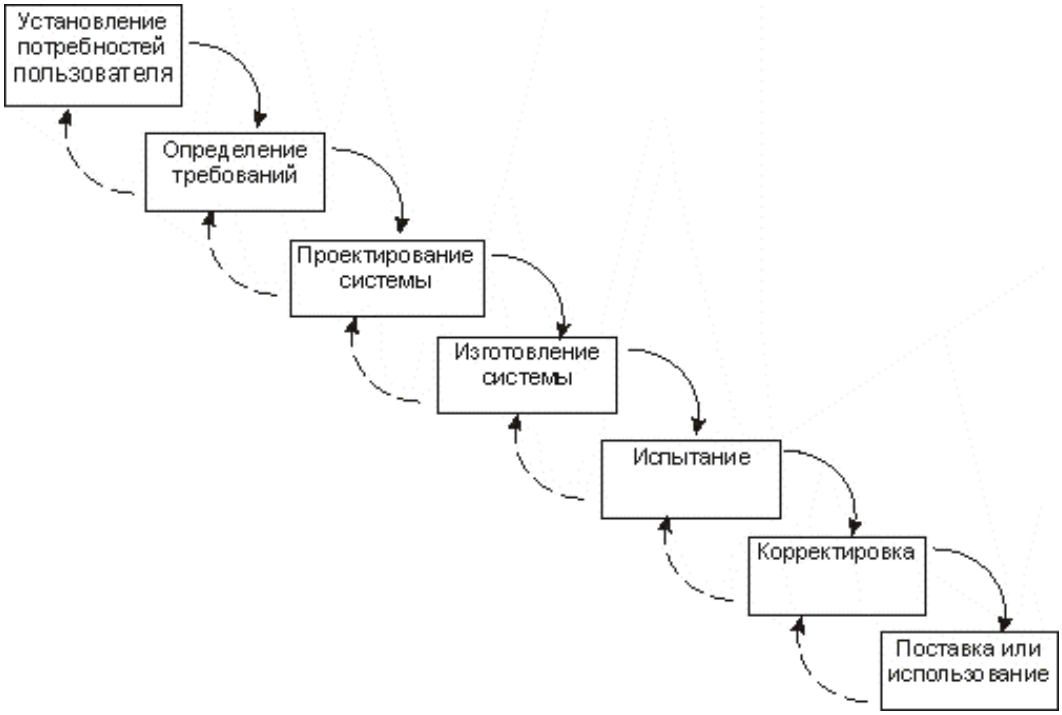
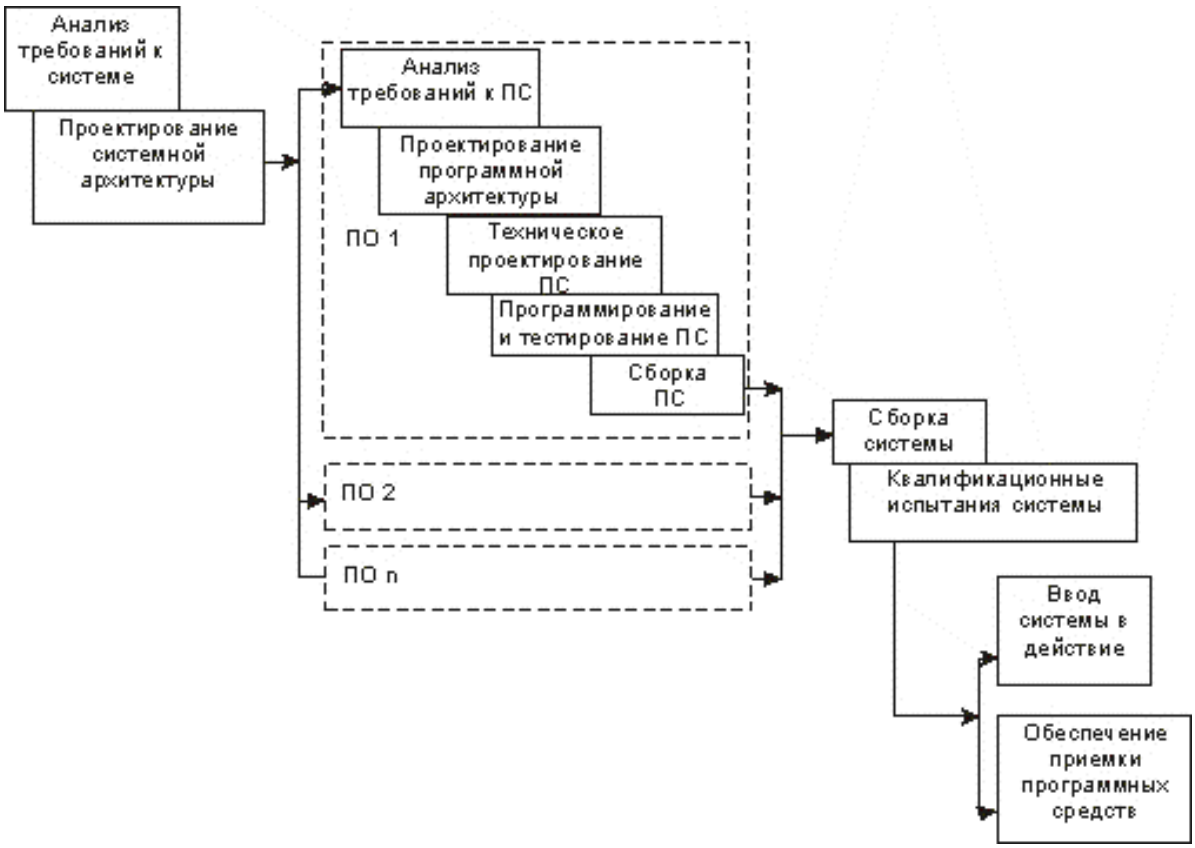


Рисунок 4.3 – Схематическое представление каскадной модели.

При этом предполагается, что процесс, который стоит левее и выше, доминирует над процессом правее и ниже. При применении такого принципа разработки каждого программного объекта соответствующие работы и задачи процесса разработки обычно выполняют последовательно. Они могут быть частично выполнены параллельно в случаях перекрытия последовательных работ. В результате завершения определенной стадии каскадной модели создается тот или иной промежуточный продукт (выход поставки) и формируется базовая линия, являющаяся исходной точкой для последующей стадии. Так, например, в результате выполнения стадии проекта по ГОСТ 34 «Техническое задание» создается и утверждается техническое задание на разработку автоматизированной системы. Выходом поставки в данном слу-

чае является техническое задание, оно и является отражением базовой линии проекта. На стадии «Ввод в действие» реализуются выходы поставки: автоматизированная система и ее комплектация (информационная и техническая документация, программно- технические средства и т.д.), акты испытания и ввода в эксплуатацию. Из них базовой линией являются автоматизированная система и ее комплектация.

Когда несколько программных объектов разрабатывают одновременно, для всех этих объектов работы и задачи процесса разработки могут быть выполнены параллельно (рисунок 4.4).



ПО – программный объект

Рисунок 4.4 Представление каскадной модели в руководстве ГОСТ Р ИСО/МЭК ТО 15271-2002 /16/

Процессы сопровождения и эксплуатации обычно реализуют после процесса разработки. Процессы заказа и поставки, а также вспомогательные

и организационные процессы, обычно выполняют параллельно с процессом разработки.

Обычно считается, что классическая каскадная модель не предполагает возвращения на предыдущие стадии. В тоже время, каскадная модель с обратной связью дает возможность вернуться на предыдущий этап. Существование обратных связей не усиливают модели, а наоборот , ее разрушают, так как они реализуют циклы разработки, для которых в данной модели не существует методик управления. Возникает очень интересный вопрос: каскадная модель с обратной связью может ли быть представлением модели итеративной? Сейчас нам трудно ответить на этот вопрос, но к его рассмотрению мы вернемся в параграфах «Спиральная модель жизненного цикла» и «Методология RUP».

Преимущества модели:

- проста в понимании и удобна в применении, так как процесс разработки выполняется поэтапно;
- она хорошо срабатывает, когда требования к качеству доминируют над требованиями к затратам и графику выполнения проекта;
- она определяет процедуры по контролю за качеством. Каждые полученные данные подвергаются обзору. Такая процедура используется командой разработчиков для определения качества системы;
- ход выполнения проекта легко отображается на линейную временную шкалу или диаграммы Гантта;
- в модели используется однократное представление всех возможностей (характеристик) системы;
- существует необходимость только единственной фазы перехода от старой системы к новой.

Недостатки модели:

- требования к объектам определены недостаточно четко;
- система обычно слишком велика, чтобы все работы по ее созданию выполнять однократно;
- не могут быть учтены предполагаемые скорые изменения в технологиях работ;
- возможные текущие изменения требований к системе;
- ограниченность ресурсов, например средств или персонала;
- промежуточный продукт может быть непригоден для использования.

Область применение каскадной модели:

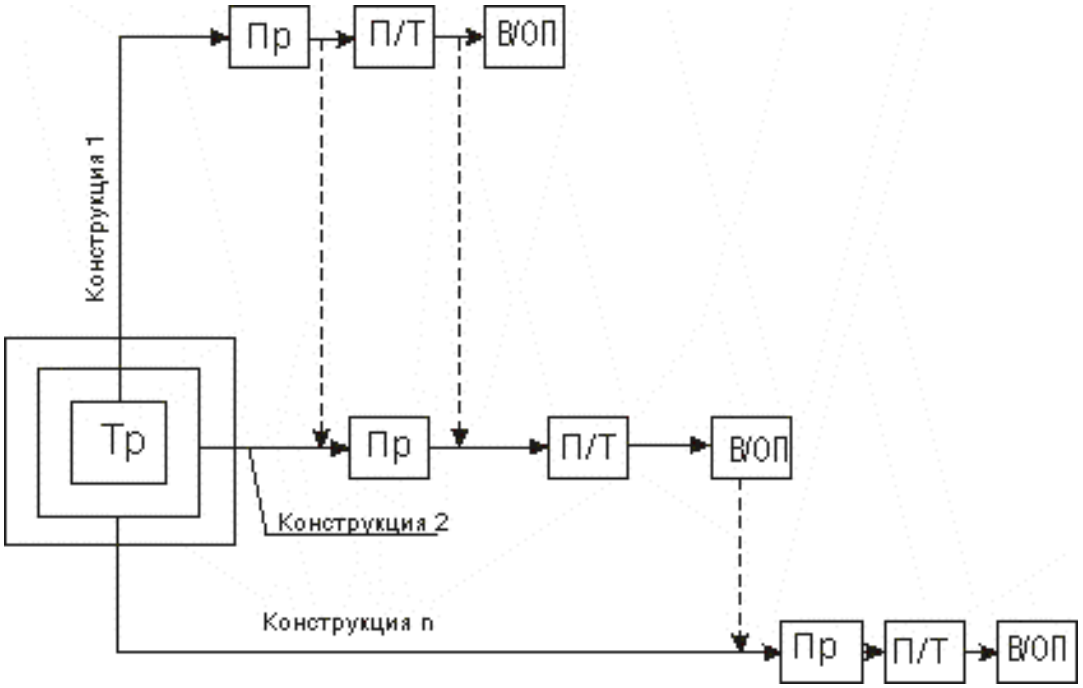
- неизменяемое определение продукта и понятные технические методики;
- компания имеет опыт построения такого типа системы;
- создание и выпуск новой версии уже существующего продукта, если вносимые изменения вполне определены и управляемы;
- перенос уже существующего продукта на новую платформу.

Инкрементная модель

Инкремент (increment) – в переводе с английского языка означает «приращение, расширение, увеличение». Инкрементная модель жизненного цикла, называемая запланированным усовершенствованием продукта, начинается с выдачи набора требований и реализует разработку последовательности конструкций (рисунок 4.5). Первая конструкция содержит часть требований, в последующую конструкцию добавляют дополнительные требования и так далее до тех пор, пока не будет закончено создание системы. Для каждой конструкции выполняют необходимые процессы, работы и задачи, например, анализ требований и создание архитектуры могут быть выполнены сразу, в то время, как разработку технического проекта программного средства, его программирование и тестирование, сборку программных средств и их квали-

фикационные испытания, выполняют при создании каждой из последующих конструкций.

В данной модели каждая конструкция, включающая работы и задачи процесса разработки, выполняется также, как и в каскадной модели, последовательно или частично параллельно с перекрытием. При частично одновременной разработке последовательных конструкций работы и задачи процесса разработки могут быть выполнены параллельно для ряда конструкций.



информационные потоки

Тр- требовании;

Пр- проектирование;

П/Т-программирование и тестирование;

В/ПП- ввод в действие и обеспечение приемки

Рисунок 4.5. - Пример инкрементной модели /16/

Для каждой конструкции обычно выполняются одни и те же работы и задачи процесса разработки в той же последовательности. Процессы сопровождения и эксплуатации могут быть реализованы параллельно с процессом разработки. Процессы заказа и поставки, а также вспомогательные и органи-

зационные процессы, обычно выполняют параллельно с процессом разработки.

Преимущества модели:

- не требуется заранее тратить средства, необходимые для разработки всего проекта (поскольку сначала выполняется разработка и реализация основной функции или функции из группы высокого риска);
- в результате выполнения каждого инкремента получается функциональный, пригодный для использования промежуточный продукт;
- существует возможность поддерживать постоянный прогресс в ходе выполнения проекта;
- используется естественное разделение системы на наращиваемые компоненты (инкременты);
- снижаются затраты на первоначальную поставку программного продукта;
- риск распределяется на несколько меньших по размеру инкрементов (не сосредоточен в одном большом проекте разработки);
- инкременты функциональных возможностей проще тестировать;
- улучшается понимание требований для более поздних инкрементов (что обеспечивается благодаря возможности пользователя получить представление о раннее полученных инкрементов на практическом уровне);
- появляются возможности наращивания привлекаемого персонала и средств, или же одной и той же командой выполнять последовательно все инкременты;
- в конце каждой инкрементной поставки существует возможность пересмотреть риски;

– у заказчика появляется возможность осваивать продукт постепенно.

Недостатки модели:

– определение полной функциональной системы должно осуществляться в начале жизненного цикла, чтобы обеспечить определение инкрементов, в то время, как требования к объектам могут быть определены недостаточно четко;
– в начале разработки должны быть предусмотрены сразу все возможности системы и предполагаемые изменения в технологических работах;
– трудно учесть возможные текущие изменения требований к системе;
– привлечение ресурсов (средств или персонала) на длительный период ограничено;
– формальный критический анализ и проверку намного труднее выполнить для инкрементов, чем для системы в целом;
– поскольку создание некоторых модулей будет завершено значительно раньше других, возникает необходимость в четко определенных интерфейсах;
– может возникнуть тенденция к оттягиванию решений трудных проблем на будущее, чтобы скорее продемонстрировать успех.

Область применения модели:

– большинство требований к продукту можно сформулировать заранее, но, в то же время, их появление ожидается через определенный период времени;
– требуется быстрая поставка продукта с неполным (базовым) функционалом;
– при разработке программ, связанных с низкой или средней степенью риска;/45/

– при выполнении проекта с применением новой технологии, что позволяет пользователю адаптироваться к системе.

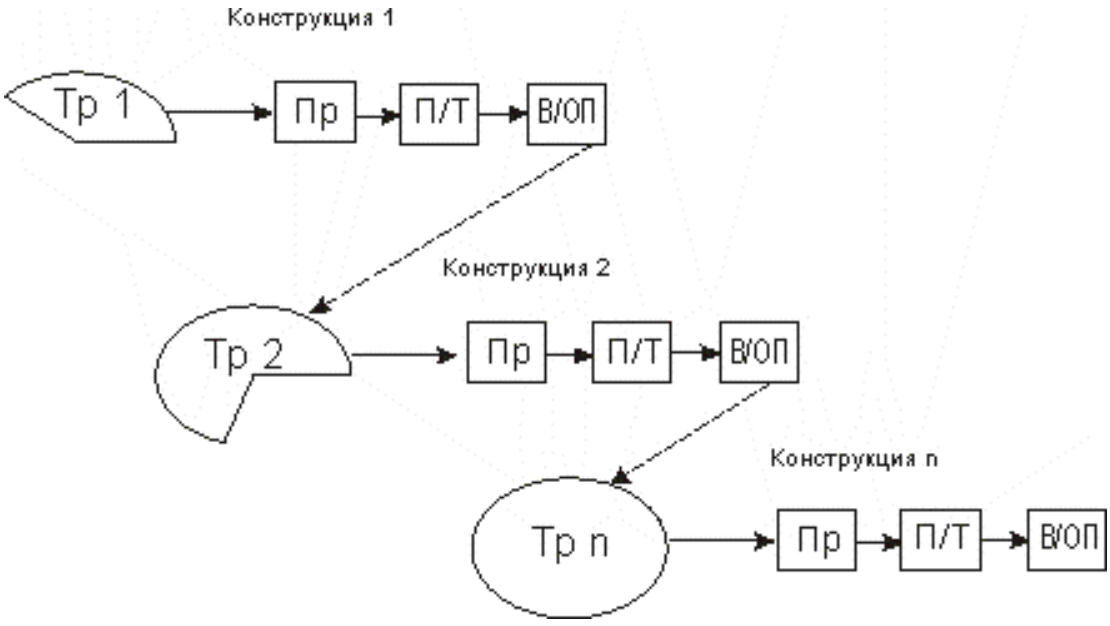
Эволюционная модель

Модель предполагает разработку продукта (прототипа), который на последующих стадиях разработки будет усовершенствован. В эволюционной модели жизненного цикла систему также разрабатывают в виде отдельных конструкций, но в отличие от инкрементной модели, требования изначально не могут быть полностью осознаны и установлены. В данной модели требования устанавливают частично и уточняют в каждой последующей конструкции (рисунок 4.6).

При эволюционном методе для каждой конструкции работы и задачи процесса разработки выполняют последовательно или параллельно с частичным перекрытием.

Работы и задачи процесса разработки обычно выполняют многократно в той же последовательности для всех конструкций. Процессы сопровождения и эксплуатации могут быть реализованы параллельно с процессом разработки. Процессы заказа и поставки, а также вспомогательные и организационные процессы, обычно выполняют параллельно с процессом разработки.

И в инкрементной, и в эволюционной модели используется, так называемый, итерационный подход, который проявляется в многократном повторении практически одной последовательности процессов, работ и задач. Последовательное уточнений требований к системе на каждой последующей фазе и есть тот краеугольный камень, который отличает две итерационные модели. В эволюционной модели разрабатывается промежуточный прототип, для того, что бы определить или уточнить требования к системе; в модели инкрементной разрабатывается промежуточный продукт, который является частью будущей системы.



Тр- требовании;

Пр- проектирование;

П/Т-программирование и тестирование;

В/ПП- ввод в действие и обеспечение приемки

Рисунок 4.6 - Пример эволюционной модели /16/

Необходимо обратить внимание на возможное сходство результатов работы моделей. В инкрементной модели разработчик общается с заказчиком в момент представления промежуточного продукта, результаты анализируются и могут быть использованы в последующем инкременте. Следовательно, требования могут уточняться, как и в эволюционной модели. В эволюционной модели промежуточный продукт может удовлетворять заказчика по базовому набору функциональных возможностей, что не потребует изменение этой части проекта в будущем, как и в инкрементной модели. Сходство реальных жизненных циклов не должно смущать. Цели каждой фазы, стратегия управления проекта в инкрементной и эволюционной модели, как мы увидели выше, разные.

Преимущества модели:

- модель дает возможность пользователям опробовать систему на ранних этапах ее разработки, принять активное участие при планировании, анализе, разработке и оценке продукта;
- реализует преимущества инкрементной модели:
    - обеспечивает разбиение большого потенциального объема работы по разработке продукта на небольшие части, при этом не требуется определить весь функционал системы;
    - сокращение графика посредством перекрывания версий и неизменяемость ресурсов при постепенном росте системы;
    - выпуск пригодного для использования промежуточного продукта;
    - привлечение персонала и ресурсов по мере их необходимости;
- повышается вероятность предсказуемого поведения системы с помощью уточнения поставленных целей;
- можно выполнять частую оценку совокупных затрат, а уменьшение рисков связано с затратами;
- упрощается надзор за изменением технологии.

Недостатки модели:

- разработка может продолжаться до бесконечности, так как заказчик может выдвигать новые требования после завершения каждой стадии;
- большое количество промежуточных стадий может привести к необходимости в обработке внутренней дополнительной и внешней документации;
- использование модели может привести к удорожанию проекта, так как увеличивается время, затраченное на планирование, повторное определение целей, выполнение анализа и реализация прототипа;

- требуются методы и средства построения прототипа;
- в производстве использование спиральной модели еще не получило такого широкого масштаба, как применение других моделей;
- заказчик может предпочесть получить прототип, вместо того, чтобы ждать появления полной, хорошо продуманной версии.

Область применения модели:

- требования заранее не известны или же могут существенно меняться во время выполнения проекта;
- нужна проверка концепции проекта;
- осуществляются временные демонстрации;
- выполняется новая, не имеющая аналогов разработка (в отличие от эксплуатации продукта на уже существующей системе);
- в проекте используются архитектурные решения или технологии, с которыми разработчик имеет малый опыт работы;
- в проекте требуется применить сложные алгоритмы или интерфейсы;
- в случае больших проектов;
- для организаций, которые не имеют ресурсов для выполнения всего проекта сразу;
- с целью демонстрации прогресса проекта и качества решений за короткий период времени.

Модификация фундаментальных моделей жизненного цикла

Необходимо отметить, что в литературе нет общей точки зрения на классификацию моделей жизненного цикла. Можно выделить два основных источника, классифицирующие модели жизненного цикла: ГОСТ Р ИСО/МЭК ТО 15271-2002 /16/ и книга «Управление программными проек-

тами: достижение оптимального качества при минимуме затрат» (Фатрел Р., Шафер Д., Шафер Л.) /45/.

В данном учебном пособии используется классификация, которая заложена, фактически ГОСТ Р ИСО/МЭК 12207-99. В данном стандарте, как мы отмечали, не рассматриваются модели жизненного цикла. В тоже время, в Приложении С приводится иллюстрация процесса разработки, которая организует данный процесс на четырех уровнях (рисунок 4.6): инициализации и завершения проекта, проектирования системы, проектирования ПС, кодирования (получение программного кода и его тестирование).

В каскадной модели определяются программные объекты, как независимые объекты, выявленные на уровне анализа системы. Жизненный цикл программного объекта не может повлиять на жизненный цикл других объектов, поэтому их разработка осуществляется независимо. Работы и задачи по каждому объекту выполняются последовательно, тем самым обеспечивается каскадная (линейная) структура.

Инкрементная модель определяет конструкции, как составляющие программных объектов, тем самым допускает итеративную разработку этих объектов. Ограничение модели заключается в том, что требования к объекту должны быть выявлены на стадии выявления требований к системе. Следовательно, разрешаются итерации на уровне проектирования ПС.

Эволюционная модель определяет конструкции, как прототип системы, тем самым допускает итерации, улучшающие систему. Следовательно, итерации допускаются на уровне проектирования системы.

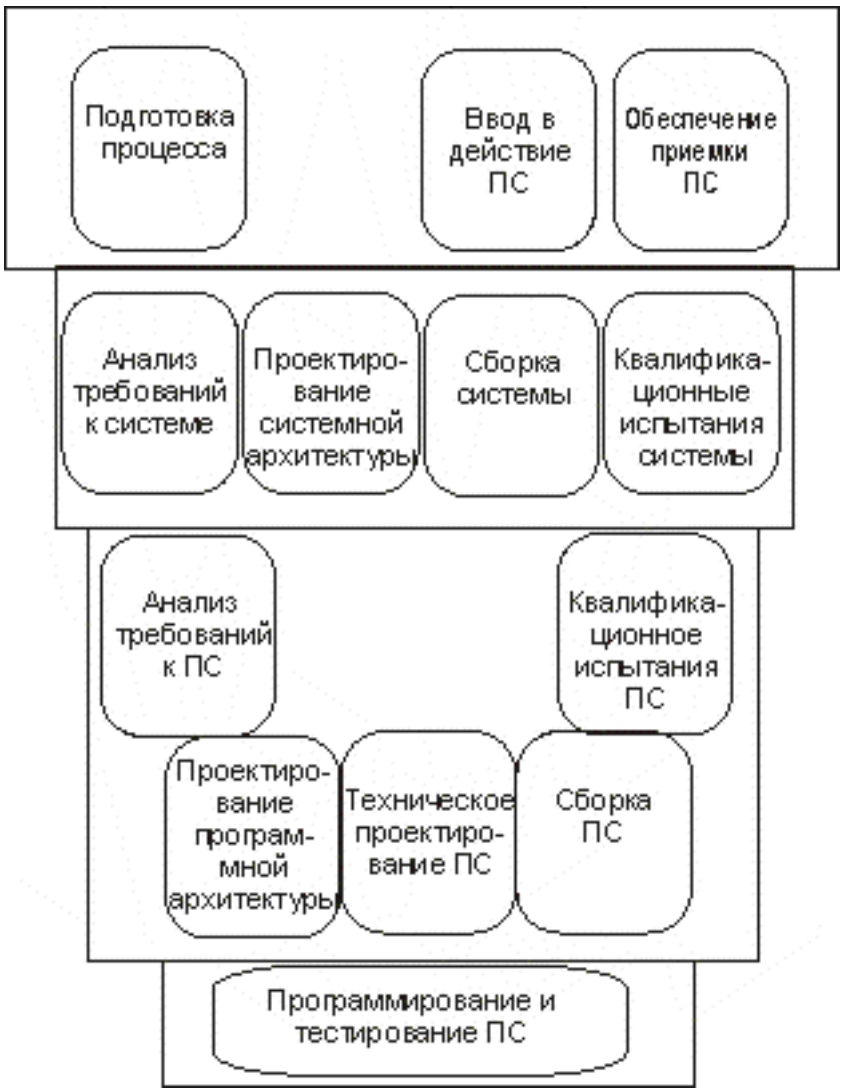


Рисунок 4.6 – Процесс разработки согласно ГОСТ Р ИСО/МЭК 12207-99.

Как мы видим, классификация моделей жизненного цикла, которую мы разобрали, определяется контекстом системного подхода, использованном в стандартах ИСО/МЭК.

В книге /45/ используется несколько другой подход к классификации моделей жизненного цикла. Можно его назвать «исторический», так как модели жизненного цикла рассмотрены с точки зрения истории их создания и развития. В данном параграфе мы рассмотрим некоторые из наиболее распространенных моделей жизненного цикла. Все эти модели являются реализацией фундаментальных моделей или их модификацией.

В тоже время, мы не согласны с авторами /45/ в том, что методология RAD (Rapid Application Development) рассматривается в качестве модели жизненного цикла.

Собственно говоря, мы снова наступаем всё на те же «грабли» - неустоявшаяся и противоречивая терминология. Нас будет ожидать еще один «подводный», лучше сказать - терминологический, «камень» в области моделей жизненного цикла. Этот камень уже заложен классификацией моделей жизненного цикла, разобранной выше. Постараемся разобраться с ним сейчас. Мы разобрали каскадную, инкрементную и эволюционную модели. Инкрементная и эволюционная модель предполагает итерационный процесс разработки.

Инкрементная модель предполагает выпуск в каждом инкременте продукции, которая может реализовать не весь требующийся функционал системы. Такой продукт обычно не называют прототипом, так как в нем реализована часть законченной системы, которая в дальнейшем будет наращена дополнительной функциональностью в других инкрементах, за счет добавления новых компонентов системы.

Эволюционная модель предполагает на каждой итерации выпуск продукции, которая так же реализует только часть функций системы. В отличие от инкрементной модели этот продукт называется прототипом, так как предполагается его изменение (а не наращивание) на будущих итерациях.

Познакомимся с термином «итеративная инкрементная разработка» (Iterative and Incremental Development - IID). Трудно понять, как может быть инкрементная разработка неитеративной. Тем не менее, такой термин существует, более того, активно используется во всех современных методологиях разработки: Rational Unified Process, Dynamic Systems Development Method, Extreme Programming и модификации agile software development (глава «Современные методологии разработки программного обеспечения»).

Термин обозначает, что жизненный цикл программного обеспечения использует итерации, в которых процессы, работы и задачи выполняются по каскадной модели жизненного цикла. Результаты итерации могут быть самые разнообразные: планы, требования, программный код, модели и т.д. Эти объекты обычно называют артефактами[6]. При итеративной инкрементной разработке (IID) артефакты каждой итерации будут улучшаться в последующих. Тем самым определяется не что иное, как эволюционная модель, но данную модель принято называть IID (итеративная и инкрементная разработка). Данную модель мы опишем в последующих главах.

V –образная модель жизненного цикла

Является разновидностью каскадной модели. В этой модели планируется верификация и аттестация элементов системы в момент их проектирования. Так, например, на этапе опытного функционирования используется документ «Программа и методика испытаний комплекса средств автоматизации», которая предназначена для установления технических данных, подлежащих проверке при испытании компонентов АС и комплекса средств автоматизации проектирования, а также порядок испытаний и методы их контроля. В данной модели «Программа и методика испытаний» начинает разрабатываться на фазах анализа и проектирования. В модели подчеркивается взаимодействие фаз аналитических с фазами верификации и тестирования программного кода, подсистем и системы, что отображается на рисунке 4.7 пунктирными линиями между прямоугольниками.

[6] Артефакт (artifact) – спецификация физического элемента информации, используемого или порождаемого в процессе разработки программного обеспечения (например, им может быть внешний документ или рабочей продукт). Артефактом может быть модель, описание или программный продукт /47/.

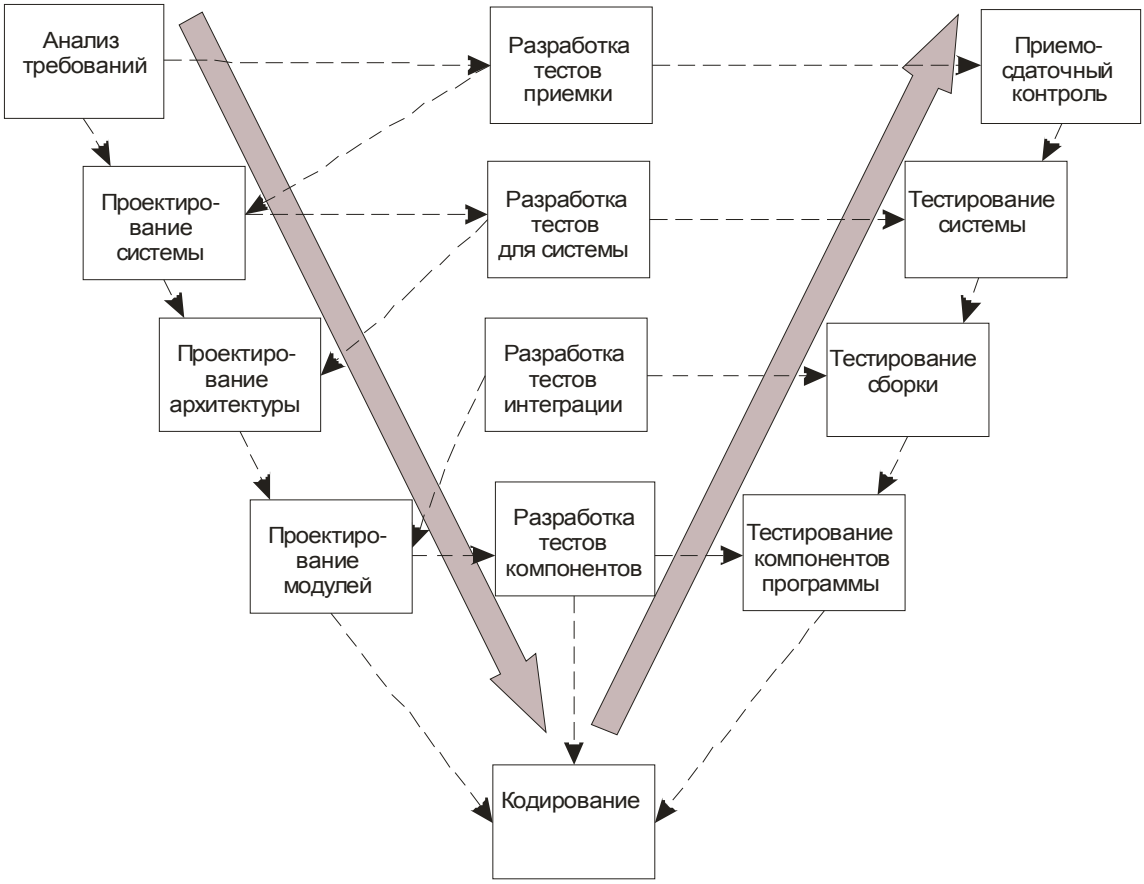


Рисунок 4.7- V- образная модель жизненного цикла программного обеспечения /48/.

Горизонтальное взаимодействие между процессами, показанное на рисунке 4.7, дает возможность планировать и разрабатывать тесты уже на начальных стадиях проекта, что невозможно по каскадной модели. В этой схеме, разработка и планирование тестов отделена от разработки, боле того, требования к тестам хорошо согласуются с требованиями к системе, подсистемам и ПО, так как, они вырабатываются на этапах их анализа и проектирования. Подобный подход используется во всех современных методологиях разработки информационных систем и ПО.

Во всех других аспектах, V- образная модель жизненного цикла ПО аналогична каскадной модели, со всеми ее преимуществами и недостатками. Данную модель целесообразно использовать , когда требования к надежности разрабатываемой системы повышены. Например, при разработке систем жизнеобеспечения человека.

Структурная модель эволюционного прототипирования (СМЭП)

Модель описана в книге /47/. Это одна из модификаций фундаментальной эволюционной модели. В данной модели можно выделить четыре фазы: планирование проекта, разработка быстрого прототипа, итеративное прототипирование и производственная разработка проекта (рисунок 4.8).

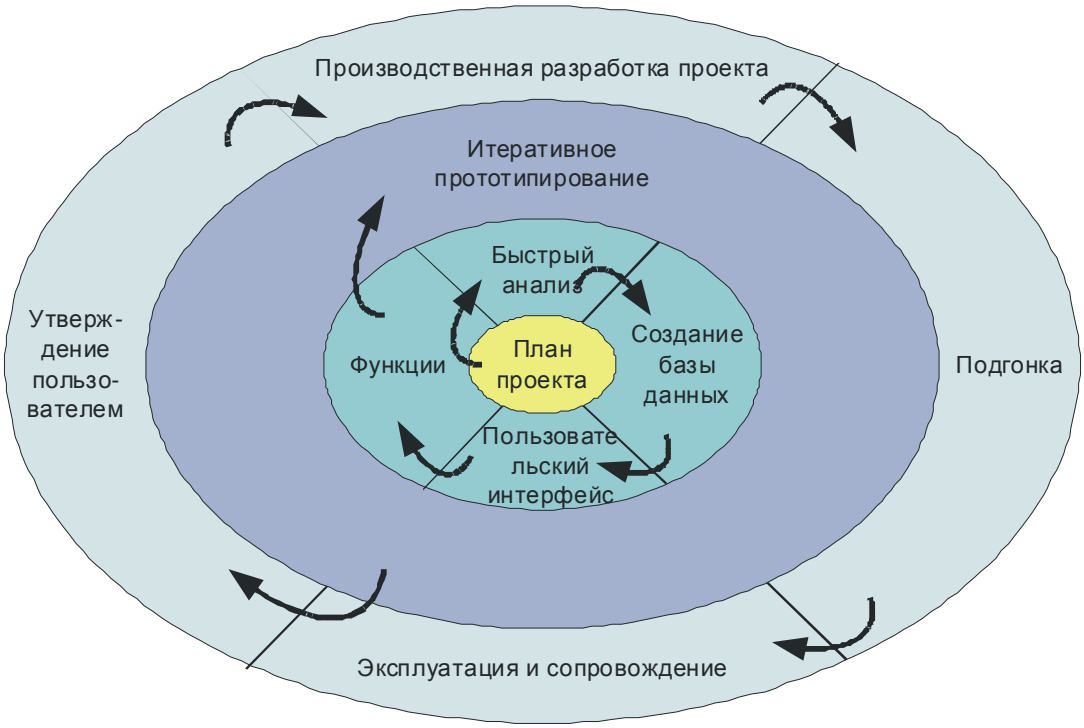


Рисунок 4.8 – Структурная эволюционная модель быстрого прототипирования /47/

Планирование проекта можно считать первым этапом фазы разработки быстрого прототипа. На этом этапе разрабатывается план проекта на основе предварительных требований к системе. На этапе «Быстрый анализ» используются методы, дающие возможность ускорить анализ, определить и специфицировать требования. На этом этапе не ставится задача создания полной модели системы. Три последующих этапа: создание базы данных, разработка пользовательского интерфейса и реализация основного функционала системы, дают возможность создать быстрый прототип, готовый для демонстрации пользователю. Во время выполнения фазы «Итерационное прототипирование» пользователь определяет проблемы прототипа, а разработчик про-

должает работу по его усовершенствованию. Данная фаза заканчивается в тот момент, когда пользователь полностью одобрил функциональные возможности системы. Создание документа, официально подтверждающего функциональные возможности системы, начинает заключительную фазу разработки. Прототип дорабатывается в рабочий проект, разрабатывается рабочая документация, производится подгонка системы. На этом этапе производится замена прототипа на полнофункциональную систему.

Данная модель реализует концепцию эволюционной модели. Следовательно, СМЭП наследует от эволюционной модели все достоинства и недостатки. СМЭП предполагает активное участие пользователей на всех стадиях выполнения проекта.

Спиральная модель жизненного цикла

Спиральная модель жизненного цикла (спиральная модель) является методом разработки систем (systems development method- SDM), широко используемая в информационных технологиях. Модель комбинирует особенности модели прототипирования (эвалюционную) и модели водопада. Спиральная модель предназначена для больших, сложных и дорогостоящих проектов/49/.

Спиральная модель была определена Барри Боэмом в его статье «A Spiral Model of Software Development and Enhancemen» /50/.Это не первой модель, которая использует итерационное развитие, но это была первая модель, которая объясняет, для чего нужны итерации. Первоначально предполагалось, что длина итерации должна составлять от 6 месяцев до 2 лет. Каждая итерация начинается с формирования целей проектирования и заканчивается оценкой результата, полученного клиентом. Оценка результатов, выполняемых на каждой итерации, производится с точки зрения конечных целей проекта.

Итерации в спиральной модели, представленные на рисунке 4.9, выполняются по следующему алгоритму.

Предварительный проект, первый прототип (рисунок 4.9), создается по схеме: определяются укрупненные требования к системе, проводятся интервьюирование внешних или внутренних пользователей системы, изучаются аспекты существующей системы, разрабатывается план управления требованиями, план проекта и т.д. Первый прототип обычно предоставляет общее и далеко не полное приближение функционала конечного продукта.

Второй прототип получен в соответствии с четырехстадийной процедурой разработки: (1) оценка эффективности, недостатков и рисков первого опытного образца; (2) определение требования ко второму прототипу; (3) планирование и проектирование второго прототипа; (4) построение и тестирование второго прототипа.

По выбору клиента, выполнение проекта может быть прервано, если риски велики. В качестве факторов риска могут использоваться ограничения в стоимости разработки, стоимости эксплуатации или любой другой фактор, который, по мнению клиента, подтверждает недостижимость функционала конечного продукта.

Полученный опытный образец оценивается по схеме, описанной выше, и если возникает потребность в другом прототипе, то четырехступенчатая процедура повторяется.

Описанные шаги повторяются до тех пор, пока клиент не определит, что прототип удовлетворяет требованиям конечного продукта.

Конечная система построена и основана на рабочем прототипе, она полностью оценивается и тестируется.

Установленная практика обслуживания выполняется с целью предотвращения крупномасштабных отказов и минимизации времени простоя.

Модель отличается тем, что в нее включены анализ рисков, управление ими, а также процессы поддержки и менеджмента. Здесь предусмотрена раз-

работка программного продукта при использовании метода прототипирования или быстрой разработки приложений посредством применения языков программирования и средств разработки четвертого поколения (и выше).

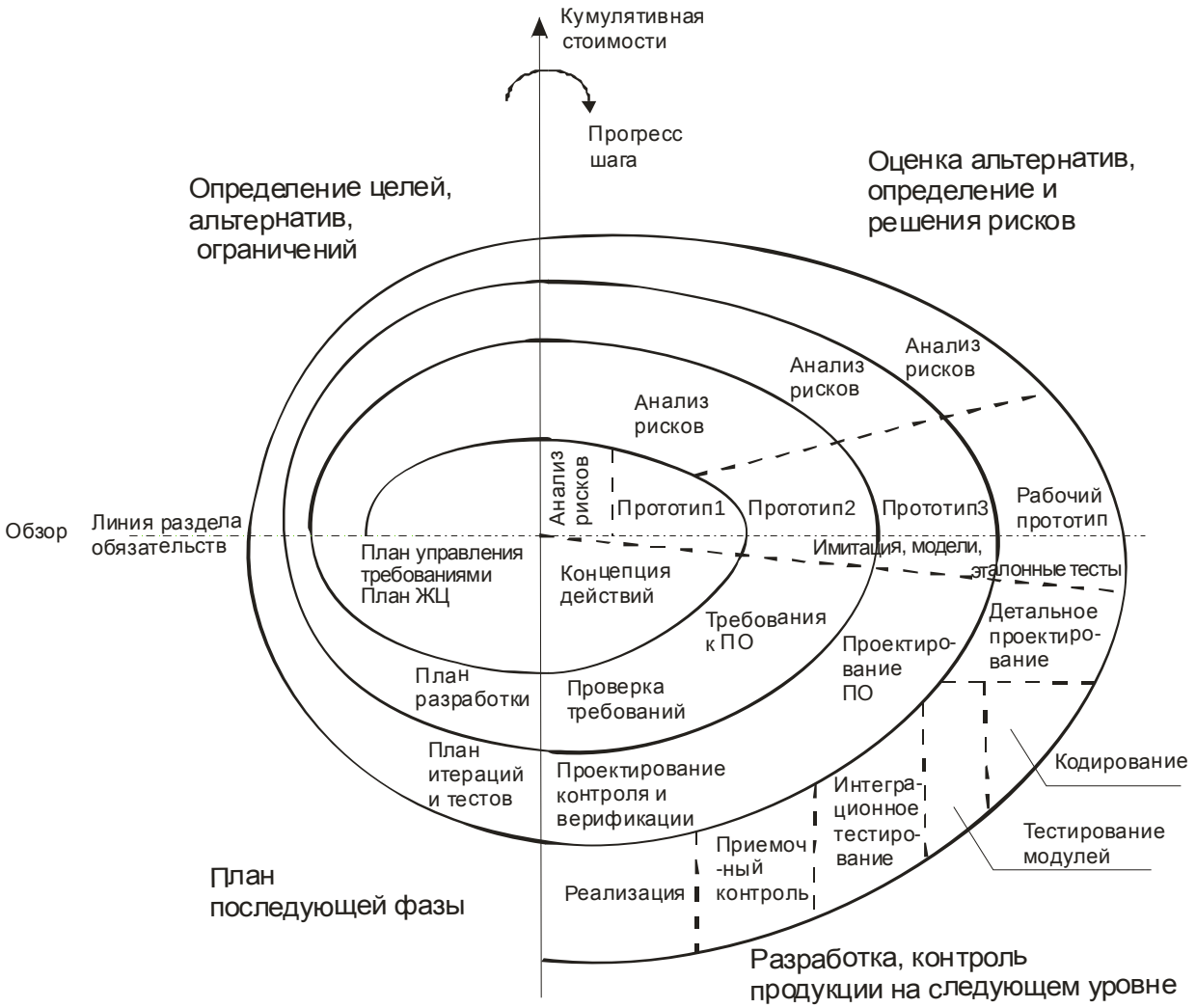


Рисунок 4.9 – Спиральная модель жизненного цикла /50/.

Модель отображает базовую концепцию, которая заключается в том, что каждый цикл представляет собой набор операций, соответствующий такому же количество стадий, как и в модели каскадного процесса. Каждый уровень сложности начинается с общей формулировки потребностей и заканчивая кодированием каждой отдельной программы.

Спиральная модель относится к эволюционным моделям. К сожалению, терминологическая путаница присутствует и здесь: с одной стороны «спиральная модель» - это эволюционная модель, в терминах принятой клас-

сификации «моделей жизненного цикла», а с другой стороны - «спиральная модель», устоявшийся термин, который широко известен в литературе и связан с вышеизложенными принципами построения модели жизненного цикла.

Контрольные вопросы

1. Как соотносятся понятия модель ЖЦ и ЖЦ?
2. Кто определяет модель жизненного цикла и на каком этапе разработки информационной системы?
3. Предполагается ли в каскадной модели жизненного цикла параллельное выполнение работ?
4. Можно ли в каскадной модели разбивать работу на части?
5. Обязательно делить на части проект в инкрементной модели?
6. Если мы выполняем проект по частям в инкрементной и в каскадной модели, то чем же отличается разработка в этих моделях?
7. Что общего в эволюционной, спиралевидной моделях и СМЭП?
8. Если в эволюционной модели и инкрементной модели производится наращивание функциональности по мере выполнении проекта, то в чем отличие моделей?
9. Что общего в спиралевидной модели и каскадной модели?
10. Что общего и в чем отличие спиралевидной и V – образной модели?

5. Моделирования систем. Основные понятия, история и инструмент.

МОДЕЛЬ (лат . modulus - мера, образец), в широком смысле - любой образ, аналог (мысленный или условный: изображение, описание, схема, чертеж, график, план, карта и т. п.) какого-либо объекта, процесса или явления («оригинала» данной модели), используемый в качестве его «заместителя», «представителя» /1/.

МОДЕЛИРОВАНИЕ - исследование каких-либо явлений, процессов или систем объектов, путем построения и изучения их моделей; использование моделей для определения или уточнения характеристик и рационализации способов построения вновь конструируемых объектов. Моделирование - одна из основных категорий теории познания: на идее моделирования базируется любой метод научного исследования - как теоретический (при котором используются различного рода знаковые, абстрактные модели), так и экспериментальный (использующий предметные модели)/1/.

Более короткое определение, которое вполне подходит для дальнейшего использования, дается в /47/. Модель- это полное описание системы, сделанное с некоторой точки зрения на определенном уровне абстракции.

В данной главе приводится описание некоторых моделей и технологий их построения. В учебном пособии не ставится задача описать все известные виды моделей и познакомить со всеми методологиями моделирования, используемыми в проектировании информационных систем.

Прежде, чем мы начнем разбирать различные модели, необходимо выяснить местоположение этой области знаний в предмете проектирование информационных систем.

В предыдущих главах были рассмотрены жизненные циклы программного обеспечения, в которых определялись процессы, действия и задачи, выполняемые при разработке информационной системы. В главе, посвященной

модели жизненных циклов рассматриваются вопросы временной последовательности выполнения тех же самых процессов, действий и задач. Теоретически мы знаем, какие задачи решать и в какой последовательности, но при этом мы не знаем, что и как делать? Мы этого не знаем, потому что не разобрали методику проектирования. Все кажется просто. Давайте вернемся к главе «Терминология» и вспомним, что является результатом проектирования информационных систем? Правильно, сама информационная система, т.е. и программный код.

Напомним, что в данном учебном пособии ставятся следующие задачи: познакомить с методологиями разработки информационной системы; обучить методикам моделирования систем, как бизнес - систем, так и информационных систем; познакомить с инструментами моделирования. В учебном пособии невозможно охватить вопросы программирования и кодирования, хотя инструменты, позволяющие реализовать кодогенерацию, будут разобраны. В связи с вышесказанным, технология проектирования будет нас интересовать до того самого момента, когда требуется осуществлять кодирование. При этом не должно создаваться ложное впечатление, что проектирование информационных систем ограничено моделированием. Создание моделей органически связано с созданием всей информационной системы, в том числе, с кодированием и тестированием.

В данной главе будут разобраны языки моделирования и методики построения моделей с использованием этих языков. В последующих главах будет показано, в каких процессах, действиях и задах следует применять моделирование. Материал данной главы тесно связан с методологиями проектирования информационных систем, поэтому некоторые методологии будут упомянуты в этом материале.

При построении модели системы могут ставиться различные цели:

- изучение поведения системы;

- визуализация системы;

- документирование принимаемых решений;

- согласование понимания систем;

- изменение объекта моделирования, например, его оптимизация;

- специфицирование[7] и документирование системы и ее поведения;

- и др.

Процесс разработки информационной системы требует выполнения большого количества действий и задач, при решении которых требуется проводить моделирование системы. Разработчики сталкиваются с тремя различными системами: бизнес-система, проектируемая информационная система и программно-техническая система, как составляющая информационной системы. Каждая из этих систем имеет различные аспекты своего представления. С одной стороны, информационная система должна быть согласована с технологическими процессами предметной области (с бизнесом); с другой стороны - она становится частью ИТ - структуры предприятия, в которой соответствующие службы обеспечивают ее нормальную работу; с третьей стороны - изменение ИТ - технологий и бизнес - процессов требует адаптации информационной системы в новых условиях, что в свою очередь потребует от разработчиков модифицировать ее. Можно продолжить перечисление различных аспектов отображения системы.

Различные методы моделирования возникали в истории инжиниринга ПО в соответствии с теми задачами, которые перед ними ставили. Многие методики постепенно уходят в историю. Не существует единого взгляда на моделирование информационных систем, позволяющего объединить разнородные модели в единое представление. Мы будем рассматривать методы моделирования в контексте их развития, в связи с этим, последующее рассмотрение моделей может показаться мозаичным.

---

[7] СПЕЦИФИКАЦИЯ (англ, specification) - документ, содержащий подробное перечисление узлов и деталей какого-либо изделия, конструкции, установки, и т. п., входящих в состав сборочного или монтажного чертежа; документ с перечислением условий, которым должен удовлетворять производственный заказ.

Немного истории

Проект ICAM

В лаборатории Военно-воздушных сил США (USAF) Wright-Patterson AFB Materials Laboratory (1976г.) началась работа по программе «Интегрированное производство с управлением от ЭВМ» (Integrated Computer-Aided Manufacturing -ICAM). Цель программы - изучить и развить промышленные технологии с использованием автоматизированных систем управления производством (АСУП, Computer Integrated Manufacturing - CIM). В целом на эту программу было выделено порядка $100 млн.

Главная задача, которая стояла перед разработчиками (Dennis E. Wisnosky и Dan L. Shunk) , объединить функции всего предприятия между собой. Они нашли выход, «отобразив предприятие перевернутой пирамидой, вершина которой была направлена на процессы» /51/. Процесс стал центром исследования предприятия. Такая концепция далеко опередила время. Как мы видели, стандарт ISO 9000 так же поддерживает ее. ICAM программа также определила необходимость проведения анализа и документирования основных процессов. Таким образом, из ICAM вышли стандарты моделирования и анализа бизнес - процессов IDEF. IDEF означает Icam DEFinition (описание ICAM). Мы разберем в данном разделе только три стандарта IDEF0, IDEF3, IDEF1X.

Проект SSADM

Метод структурированного анализа и проектирования систем (Structured Systems Analysis and Design Method - SSADM) - подход системного анализа к проектированию информационных систем. SSADM был произ-

веден для ССТА[8]. Названия «Structured Systems Analysis and Design Method» и «SSADM» - теперь зарегистрированные торговые марки Министерства торговли Великобритании (OGC).

Применение методов системного проектирования дает возможность упорядочить разработку ПО, предоставить структуру действий сбора, хранения, преобразования и распространения информации с тем, чтобы минимизировать средства на разработку компьютерных систем с заданными требованиями.

SSADM – реализует каскадную модель жизненного цикла проектирования информационных систем; SSADM символизирует в области проектирования систем кульминацию скрупулезного подхода, основанного на движении документов. Обычно SSADM противопоставляют методы RAD[9] и DSDM[10].

SSADM - основывается на методах разработки различных школ:

– Питер Чекланд (Peter Checkland), разработчик Методологии программируемых систем (Soft Systems Methodology), как частный случай интеллектуальных систем;

---

[8] Central Computer отвечающее Telecommunications Agency - Центральное агентство по вычислительной технике и телекоммуникациям (Central Computer and Telecommunications Agency — ССТА), в настоящее время именуемое Office of Government Commerce — OGC, великобританское правительственное учреждение, с 1980г. отвечающее за использованием ИТ технологий в правительстве.

[9] Быстрая разработка приложений (Rapid application development -RAD) - процесс разработки ПО. Джеймс Мартин (James Martin) сформулировал основные идеи RAD в 1980г.

[10] Метод динамического анализа и проектирования систем (Dynamic Systems Development Method - DSDM) - концепция, основанная на RAD. DSDM использует идею непрерывного пользовательского участия в итеративном и инкрементном подходе, что дает возможность гибко реагировать на изменяющиеся требования, развивать систему ПО, удовлетворяя требованиям бизнеса и ограничениям ресурсов. DSDM - одно из множества agile (проворных) методов разработки ПО, которое формирует Agile Alliance (www.agilealliance.org).

- Лэрри Константин (Larry Constantine), разработчик структурного проектирования (структурные карты) и изобретатель диаграмм потоков данных(data flow diagrams- DFD);
- Вэйни Стеиунс (Wayne Stevens), соавтор Лэрри Константина и Гленфорда Миерсома в разработке структурного проектирования;
- Крис Генье и Триш Сарсон (Chris Gane & Trish Sarson)- авторы методов и инструментов структурного анализа систем;
- Эд Йордон (Ed Yourdon), разработчик метода Йордона в структурном программировании;
- Майкл А. Джэксон (Michael A. Jackson), разработчик метода структурного программирования Джэксона (структурные карты).

В данном учебном пособии будут рассмотрены несколько подходов, использованных в структурном программировании: DFD диаграммы в нотации Генье- Сарсона и Йордона-Де Марко; метод Йордона в структурном программировании.

Необходимо отметить, что метод Йордона базируется на трех моделях: DFD, STD[11] и ERD[12]. Позднее, концепция диаграмм состояний была развита в UML[13]. ERD диаграммы были развиты в работах Чена, в контексте этих работ данные диаграммы описаны в настоящем пособии.

Моделирование данных /52/

Семантическое моделирование данных изначально возникло в целях повышения эффективности и точности проектирования баз данных. Методы

---

[11] Диаграмма переходов(State transition diagram) или диаграмма состояний (State diagrams) - графическое представление конечных автоматов

[12] Диаграмма сущность – связь (entity-relationship diagram –ERD) – графическое представление моделирования данных (структуры данных).

[13] UML (Unified Modeling Language обычно не переводится) – язык моделирования общего назначения, включает графическую нотацию для создания абстрактных системных моделей (UML модели).

семантического моделирования оказались применимы ко многим пользовательским проблемам, они дают легкое преобразование в различные модели данных, основанные на записях: сетевые, иерархические, реляционные. Эбриал представил двоичную семантическую модель данных в 1974г., затем в течение нескольких лет появилось множество моделей данных. Наиболее известной и популярной из них явилась модель Чена (P.Chen) – модель сущность-связь (Entity –Relationship Model - ERM)/53/.

Разработчики семантического моделирования данных в первую очередь уделяли внимание структуре данных, разработчики языков ООП (объектно-ориентированных программирование) главным образом интересовались поведением объектов. Их интересовала манипуляция данными. Семантическая ERM модель была расширена (иногда упоминают EER – Enhanced Entity-Relationship модель). На стыке этих дисциплин преобладает терминология объектно-ориентированных языков (объект, а не категория; наследование, генерализация, родительский объект, дочерний объект и т.д.). Полученная методология затрагивает концептуальный уровень проектирования базы данных. В методическом руководстве, по примеру книги /54/, мы будем называть данную модель – моделью Чена.

Один из упомянутых выше стандартов IDEF1X так же является разновидностью ERM. В отличие от модели Чена, данный стандарт затрагивает вопросы не только концептуальной разработки, но и логический уровень моделирования данными. В пособии IDEF1X будет использоваться для моделирования логического уровня.

Стандарт UML

Развитие объектно-ориентированных языков и сложности требований к ПО заставило разрабатывать новые методы моделирования. Объектно-ориентированные языки моделирования появились в середине 70-х г. В начале 90-х годов количество языков исчислялось десятками. Наиболее интерес-

ные из них: Booch (Grady Booch Грейди Буч), OOSE (Ivar Jacobson-Айваром Якобсоном),OMT (James Rumbaugh - Джеймс Рамбо), Shlaer-Mellor, Coad-Yordan. В 1994 году Г.Буч и Д. Рамбо, работавшие в компании Rational Software, объединили свои усилия для создания нового языка объектно-ориентированного моделирования. За основу языка ими были взяты методы моделирования, разработанные Бучем (Booch) и Рамбо (Object Modeling Technique — OMT). OMT был ориентирован на анализ, а Booch — дизайн программных систем. В октябре 1995 года была выпущена предварительная версия 0.8 унифицированного метода (англ. Unified Method). В этом же году к проекту присоединился Айвар Якобсон (автор OOSE). Достоинством OOSE являлись возможности спецификации бизнес-процессов и анализа требований при помощи сценариев использования. OOSE был также интегрирован в унифицированный метод.

Данный проект выиграл конкурс по созданию стандартного средства объектно-ориентированного моделирования, объявленный консорциумом OMG (Object Management Group).

К разработке новых версий языка в рамках консорциума UML Partners присоединились такие компании, как Digital Equipment Corporation, Hewlett-Packard, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle Corporation, Rational Software, Texas Instruments и Unisys. Результатом совместной работы стала спецификация UML 1.0, вышедшая в январе 1997 года. В ноябре того же года за ней последовала версия 1.1, содержавшая улучшения нотации, а также некоторые расширения семантики. Данную версию OMG объявила стандартом.

Формальная спецификация последней версии UML 2.0 опубликована в августе 2005 года. Семантика языка была значительно уточнена и расширена для поддержки методологии Model Driven Development (MDD).

UML 1.4.2 принят в качестве международного стандарта ISO/IEC 19501:2005.

Инструментарий моделирования

Было бы очень странно, если бы создатели моделей разработки программного обеспечения или же разработчики программного обеспечения, использующие эти модели, не поставили задачу автоматизации процесса разработки и моделирования. Можно задачу поставить более широко - автоматизация жизненного цикла разработки программного обеспечения или автоматизация процессов жизненного цикла программного обеспечения. В литературе популярным является термин CASE (Computer-aided software engineering), который отражает общее название группы технологий, методов и средств проектирования программного обеспечения, поддерживаемых соответствующими средствами автоматизации этапов анализа, проектирования, разработки и сопровождения систем (Lingvo 12). Вот некоторые типичные инструменты CASE:

- кодогенерация (Code generation tools);
- моделирование данных (Data modeling tools);
- UML;
- рефакторинг (Refactoring tools);
- конфигурационный менеджмент (Configuration management tools), включающий версионный контроль.

История CASE берет начало с создания компанией Nastec Corporation of Southfield, Mich в 1982 г. на базе микрокомпьютерной системы программы GraphiTex, объединяющей возможности графического и текстового редактора. Наибольший пик развития CASE системы получили в 90-е годы.

Большое количество инструментов появилось для обеспечения методологии SSADM. Например: схемы баз данных, диаграммы потоков данных, ER диаграммы, программные спецификации, пользовательская документация.

Условно инструменты CASE можно разделить на три группы:

- **Upper CASE**: инструменты, работающие на стадии анализа (иногда проектирования) жизненного цикла разработки ПО, например, инструменты для построения диаграмм, генераторы отчетов и форм, инструменты анализа;
- **Lower CASE**: инструменты поддержки генераторов схем баз данных, генераторов программ, реализации, тестирования, управление конфигурацией;
- **I-CASE**: инструменты, объединяющие предыдущие CASE - инструменты позволяют синхронизировать работу по определению информационных требований и построению баз данных (генерация SQL скриптов).

В учебном пособии не ставится задача описать интерфейс ПО, относящийся к инструментам CASE. Как правило, интерфейс этих средств интуитивно понятен и хорошо документирован. Тем не менее, мы будем ссылаться на следующие программы:

- BPWin (AllFusion Process Modeler) и ERWin (AllFusion ERWin Data modeler) компании Computer Associates International, Inc.
- MS Vision компании Microsoft
- Rational Rose Enterprise Edition (Rose), компании IBM

Мы не ставим задачей выявить достоинства и недостатки данных программ. Каждая программа прошла довольно долгий путь конкурентной борьбы и осталась на рынке. BPWin и MS Visio могут использоваться при работе методами структурного анализа (IDEF0, DFD, IDEF3, моделирование бизнес - процессов.). MS Visio, ERWin и Rose могут использоваться на этапе моделирования данных (ER моделирование, генерация SQL скриптов). MS Visio и Rose предоставляют возможность работать с UML практически на всех этапах разработки программного обеспечения (Rose - вплоть до генерации кода).

Контрольные вопросы

1. Что такое проект ICAM?
2. Какие задачи пытались решить в проекте SSDAM?
3. Что общего в подходах решения задач в проектах ICAM и SSDAM?
4. Что общего и что отличает объектно-ориентированное и структурное моделирование?
5. Какой подход более предпочтителен: объектно-ориентированный или структурный?
6. Что такое моделирование?
7. Что требуется моделировать при проектировании информационных систем?
8. Что такое CASE?
9. Какой функционал предполагается у CASE?
10. Как можно подразделить CASE?

6. Моделирования систем. Структурный анализ и проектирование

В данном разделе будут разобраны два, широко применяемые, подхода структурного анализа и проектирования:

- технология[14] структурного анализа и проектирования SADT (Structured Analysis and Design Technique) и методология[15] IDEF0;
- методы[16], ориентированные на потоки данных: структурное проектирование Е.Иордана, структурный системный анализ С. Гейна и Т. Сарсона, построения диаграммы потоков данных.

Методология IDEF0

Структурный анализ и проектирование (Structured Systems Analysis and Design Method - SSADM) – подход, разработанный в проекте Центрального агентства по вычислительной технике и телекоммуникационным системам (Великобритания). В то время это был не единственный проект. Дуглас Т. Росс сначала в Массачусетском университете, а затем в компании SofTech, разработал технологию структурного анализа и проектирования SADT/58/. Данную методологию разработчики проекта ICAM взяли за основу при разработке методологии IDEF0. Методы IDEF0 является подмножеством методов SADT, некоторые методы, например, специфицирования диаграмм и работы заказчик- разработчик, не вошли в IDEF0.

---

[14] Технология – совокупность производственных методов и процессов в определенной отрасли промышленности, а так же научное описание способов производства /56/.

[15] Методология – совокупность методов, применяемых в какой-либо науке /56/.

[16] Метод – в науке способ и порядок исследования предмета для получения наиболее полного и соответствующего истине результата /56/, не путайте с методикой. Методика – совокупность методов обучения чему-нибудь, практического выполнения чего-нибудь /56/. В педагогике план и система изложения предмета для достижения более легкого, полного и прочного усвоения его учеником. Учение о способах для педагогических целей изложения данной науки /57/.

В 2001 г. приняты рекомендации по стандартизации ГОСТ Р 50.1.028—2001 «Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования» /59/. Данные рекомендации являются адаптированным переводом стандарта IDEF0 /60/.

Область применения стандарта весьма широка: специфицирование и документирование бизнес - процессов, цель - получить комплект документации описывающий процессную организацию производства, например при сертификации по ISO 9000; исследование бизнес процессов разработчиками, цель – добиться единого понимания в группе разработчиков или/и с заказчиками автоматизируемых бизнес - процессов; документирование предложений по реинжинирингу бизнес процессов, цель - добиться понимания руководства предлагаемых решений и т.д. Как вы видите, основная область применения стандарта - моделирование, документирование и специфицирование бизнес - процессов.

Концепция IDEF0

Основные концепции стандарта отражены в понимании блочного моделирования и его строгого и формализованного графического представления, который дает лаконичность и точность при передаче информации, отраженной в модели при ее построении. Разработка модели должна выполняться итеративно с соблюдением основного правила процессного подхода-процесс не должен быть привязан к организационной структуре

Модель - искусственный объект, представляющий собой отображение(образ) системы и ее компонентов

***M** моделирует **A**, если **M** отвечает на вопросы относительно **A**.*

Система представляет собой совокупность взаимосвязанных и взаимодействующих частей, выполняющих некоторую полезную работу.

Модель описывает, что происходит в системе, как ею управляют, какие сущности она преобразует, какие средства использует для выполнения своих функций и что производит.

Блочное моделирование и его графическое представление. IDEF – диаграмма - это графическое представление (нотация) любой изучаемой системы в виде набора взаимодействующих и взаимосвязанных блоков (*функции*), отображающих процессы, операции, действия, происходящие в изучаемой системе. Стрелки представляют *интерфейс*, посредством которых блок взаимодействует с другими блоками или с внешней средой. Графический язык дает возможность **лаконично и точно** описать системы.

Средства IDEF0 облегчают **передачу информации** от различных участников проекта, т.к. предоставляют: простой инструмент изображения (блоки и стрелки), текстовый материал на естественном языке, иерархическую структуру организации диаграмм от общего к частному.

IDEF0 - это стандарт, с присущим ему **строгим и формализованным** описанием всех деталей, что его выгодно отличает от большинства методик и методологий структурного анализа и проектирования.

В стандарте заложена **итеративная** процедура разработки модели. Модель определяет функцию (это бизнес - процесс, действие, задача и т.д.) системы, при этом система не должна быть привязана к организационной структуре предприятия, а наоборот, функциональная модель должна определять организационную структуру[17].

[17] Важно не запутаться в нагромождении термина «структура» в данном материале. IDEF0 - методология структурного анализа и проектирования. Диаграмма IDEF0 моделирует предприятие, с точки зрения деятельности (функции). Единицей структуры является функция, которая изображается блоком. Организационная структура также является моделью предприятия, только в ней единица структуры – подразделение предприятия.

Синтаксис[18] графического языка

В стандарте определены три синтаксических элемента: блоки (рисунок 6.1), стрелки и синтаксические правила. В то же время, в приложении А, которое является обязательным, описаны форма бланка диаграммы и правила его заполнения. Тем самым, приложение А также определяет синтаксис.

Блок описывает функцию. Внутри каждого блока помещается его имя и номер. Имя должно быть активным глаголом или глагольным оборотом, описывающим функцию.

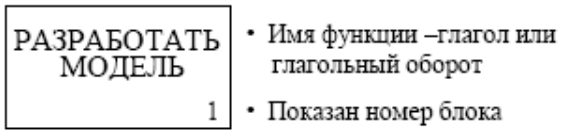


Рисунок 6.1 - Синтаксический элемент «Блок»

Стрелки не представляют последовательность событий, они лишь показывают, какие данные или материальные объекты должны поступить на вход функции для того, чтобы эта функция могла выполняться. Синтаксические правила определяют, как должны рисоваться стрелки и блоки. Например, блоки должны быть прямоугольниками.

Стандартный бланк методологии приведен на рисунке 6.2.

Интересно, что Visio предоставляет бланк только с нижним заголовком, а бланк BPWin отличается от бланка, установленного ГОСТом, размером нижнего заголовка и некоторыми незначительными деталями. Пример содержательной части диаграммы приведен на рисунке 6.3 (без бланка)

[18] Синтаксис - набор правил построения фраз языка (в данном случае объекты нотации), позволяющий определить осмысленные предложения в этом языке. греч.Syntaxis - порядок

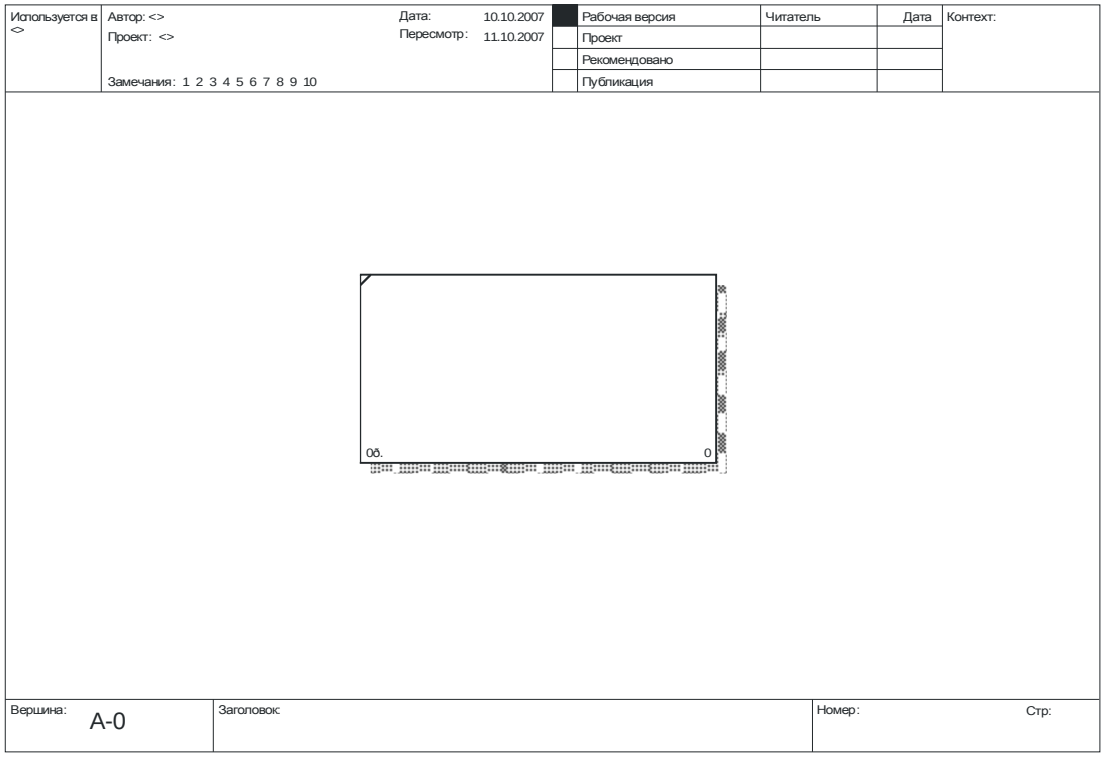


Рисунок 6.2 - Форма бланка IDEF0.

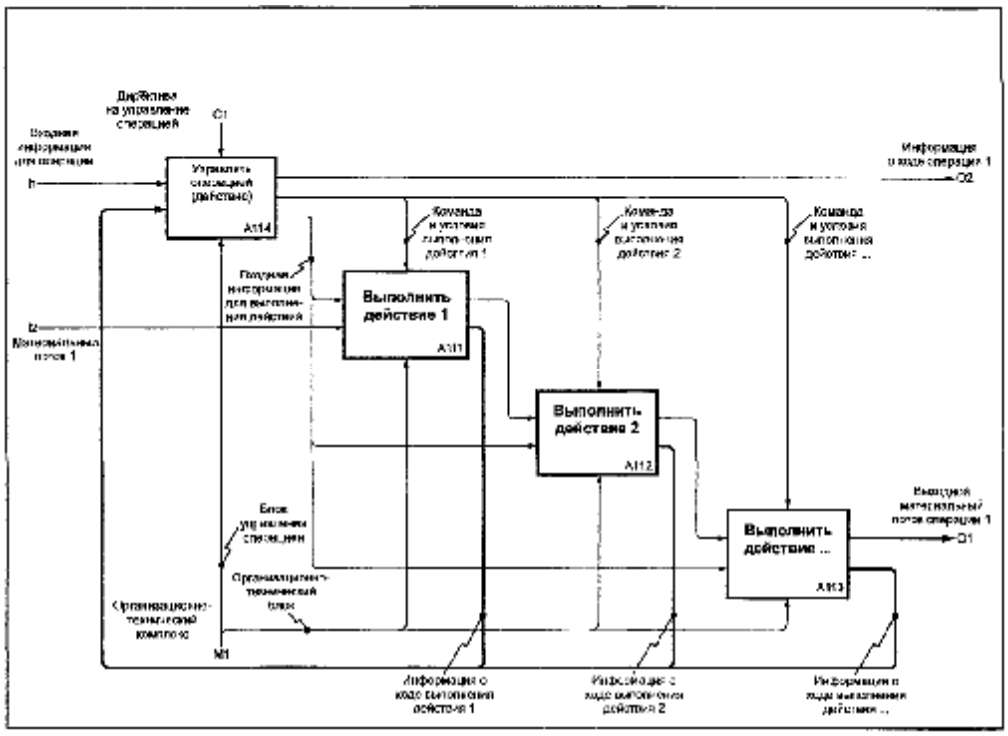


Рисунок 6.3 – Пример диаграммы IDEF0.

Семантика[19] языка

Семантика нотации IDEF0 определяется:

- семантикой блоков и стрелок,
- именами блоков и метками стрелок,
- составом модели IDEF0 (диаграмма, глоссарий и текст),
- элементами контекстной диаграммы (цель и точка зрения),
- иерархией диаграмм в модели (родительские и дочерние диаграммы)
- диаграммами иллюстраций FEO.

Блоки именуются глаголами или глагольными оборотами и эти имена сохраняются при декомпозиции. Стрелки и их сегменты, как отдельные, так и связанные в «пучок», помечаются[20] существительными или оборотами существительного.

Каждая сторона блока имеет стандартное назначение, с точки зрения связи стрелка/блок. К левой стороне блока примыкают стрелки входящие(вход); к правой стороне – стрелки выходящие (выход); к верхней – стрелки управления (управление), разновидность входа; к нижней – стрелки входящие (механизмы), обозначают что или кто выполняет данную функцию, стрелки выходящие (вызов), обозначают вызов функции, находящейся на другой диаграмме данной модели или же в другой модели (рисунок 6.4)

[19] Семантика определяет содержание (значение) синтаксических компонентов языка и способствует правильности их интерпретации. Интерпретация устанавливает соответствие между блоками и стрелками с одной стороны и функциями и их интерфейсами – с другой.

[20] Помечаются – ставятся метки. Метки являются «именами» стрелок. Метки сегментов позволяют конкретизировать данные или материальные объекты, передаваемые этими сегментами, с соблюдением синтаксиса ветвлений и слияний.

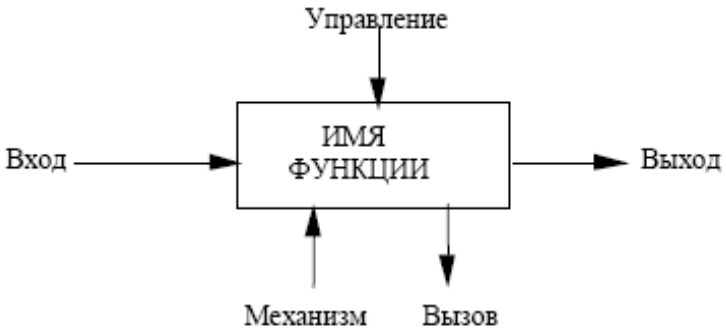


Рисунок 6.4 – Расположение стрелок относительно функционального блока.

Функция идентифицируется именем и номером, а стрелка – меткой, которая может присоединяться к функции линией выноски. Стрелки идентифицируют данные или материальные объекты. Они должны быть помеченными существительным или оборотом существительного.

Модель IDEF0 состоит из трех типов документов: графические диаграммы (диаграммы IDEF0 и диаграммы – иллюстрации FEO- For Exposition Only), текст и глоссарий. Основа модели - диаграммы IDEF0. Текст используется для объяснений и уточнений характеристик, потоков, внутриблочных соединений и т.д. Глоссарий предназначен для определения аббревиатур (акронимов), ключевых слов и фраз, используемых в качестве имен и меток на диаграммах. Диаграмма FEO предназначена для включения в модель иллюстраций, она не подчиняется синтаксису IDEF0. Инструмент «BPWin» включает диаграммы DFD и IDEF3 в качестве диаграмм FEO.

Наиболее важным элементом концепции модели IDEF0 является ее иерархическая структура. Эта структура воплощается в следующих элементах модели: иерархическое отношение различных диаграмм в модели, иерархия в понятии функция, декомпозиция стрелок и блоков, ссылочный код, дерево узлов.

Иерархическое отношение различных диаграмм в модели закреплено следующим образом. Каждая модель должна иметь контекстную диаграмму.

Объект моделирования контекстной диаграммы единственный функциональный блок и граничные стрелки. На контекстной диаграмме должна быть отражена цель[21] моделирования и точка зрения[22] моделирования (рисунок 6.5).

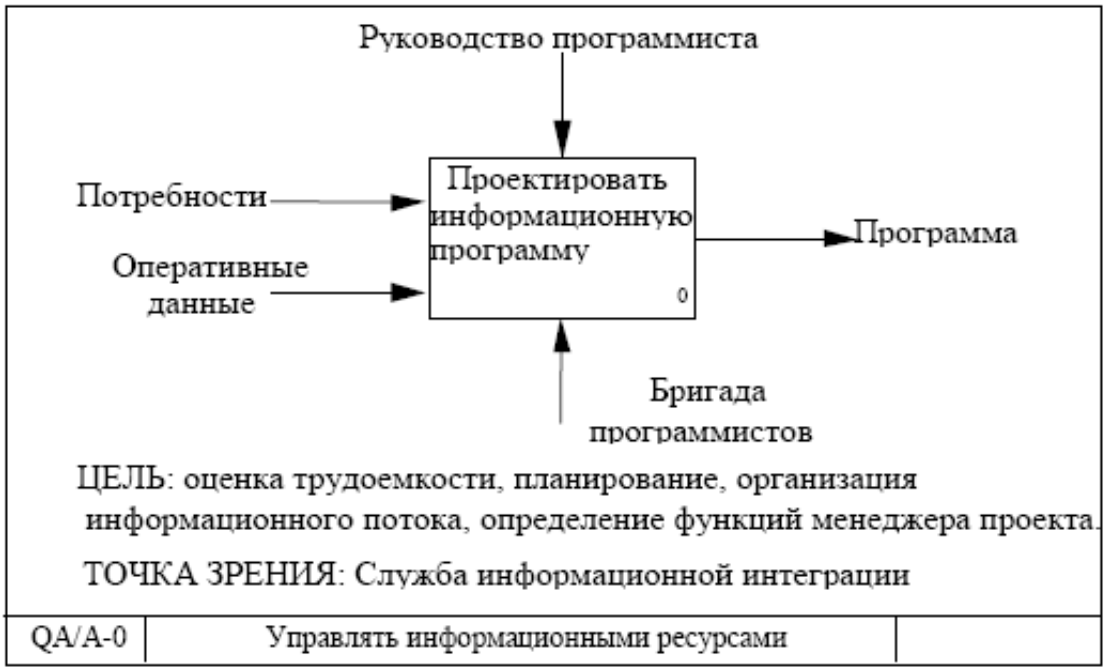


Рисунок 6.5- Пример контекстной диаграммы.

Каждая функция и, соответственно, ассоциированный с ней блок, изображенный на диаграмме, могут декомпозироваться, т. е. разбиваться на подфункции, которые будут изображаться блоками на другой диаграмме. Первая диаграмма будет называться родительская, а вторая дочерняя (рисунок 6.6.).

[21] Формулировка «Цели» выражает причину создания модели, т.е. содержит перечень вопросов, на которые должна отвечать модель.

[22] Точка зрения должностного лица или подразделения, с позиции которых создается модель.

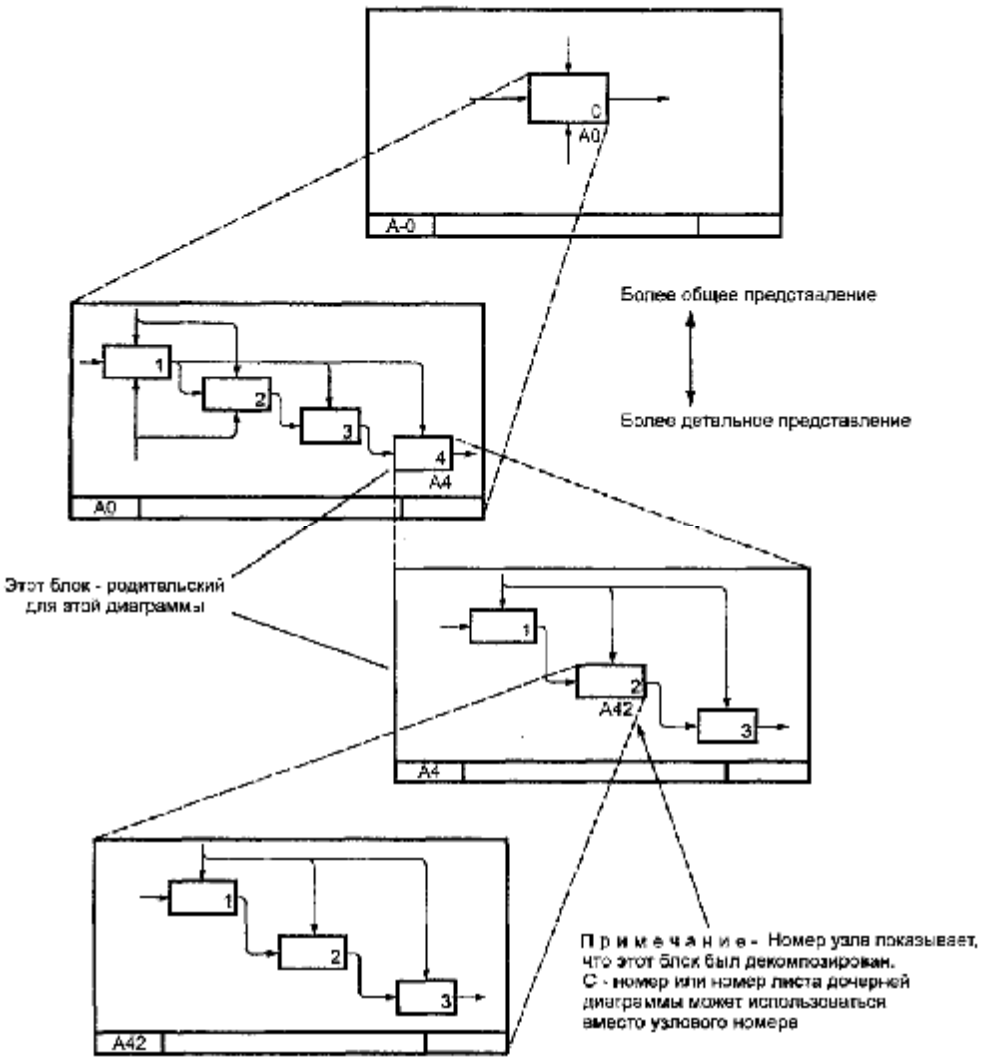


Рисунок 6.6 – Соотношение родительских и дочерних диаграмм.

Каждый блок диаграммы ассоциирован с узлом[23] и имеет соответствующий узловой номер[24]. Каждая диаграмма, кроме контекстной, является декомпозицией соответствующего блока. Она имеет узловой номер диаграммы[25] (номер в левом нижнем углу диаграммы). Узловой номер блока определяется как конкатенация (сцепление) узлового номера диаграммы с номером блока на этой диаграмме. Номер блока на контекстной диаграмме – 0,

[23] Узел: блок, порождающий дочерние блоки; родительский блок.

[24] Узловой номер: код, присвоенный блоку и определяющий его положение в иерархии модели; может быть использован в качестве подробного ссылочного выражения

[25] Узловой номер диаграммы: часть узловой ссылки диаграммы, которая соответствует номеру родительского блока

на всех остальных от 1 до 6. Узловой номер диаграммы соответствует узловому номеру блока, декомпозицию которого она отображает. Для контекстной диаграммы узловой номер – «A-0», а узловой номер блока контекстной диаграммы «A0». В некоторых случаях требуется отметить номер модели, тогда перед узловым номером диаграммы пишется номер модели и ставится слеш «/».

То, что блок является дочерним и раскрывает содержание родительского блока на диаграмме предшествующего уровня, указывается специальным ссылочным кодом, написанным ниже правого нижнего угла блока (рисунок 6.6). Ссылочный код (С-номер) начинается с буквы А (по имени диаграммы А—0), содержит цифры, определяемые номерами родительских блоков (рисунок 6.7)

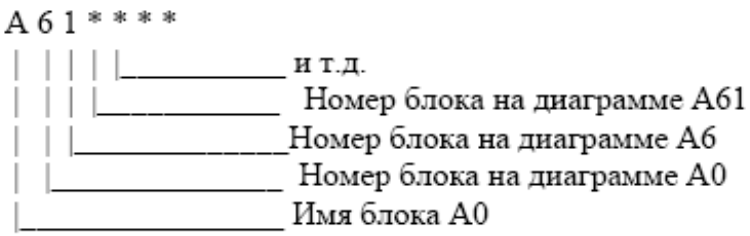


Рисунок 6.7 – Формирование ссылочного кода

Повторим еще раз.

Блок ассоциирован со следующими номерами:

- номер блока на диаграмме (от1 до 6 или 0 для контекстной диаграммы);
- узловой номер блока (<узловой номер диаграммы>&<номер блока>);
- ссылочный код (тот же , что и узловой номер блока, только проставляется в правом нижнем углу блока, вне прямоугольника).

Диаграмма ассоциирована с узловым номером диаграммы (это узловой номер блока, который она декомпозирует, для контекстной диаграммы «<имя диаграммы> - 0», например «A-0» или «G1/A-0» где G1 – номер модели).

Такая, запутанная, на первый взгляд, структура дает нам замечательное древовидное представление модели IDEF0. Причем графа един в своем представлении, как для блоков, так и для диаграмм. Номер узла можно понимать, как узловой номер диаграммы, так и узловой номер блока. Пример древовидной структуры приведен на рисунке 6.8.

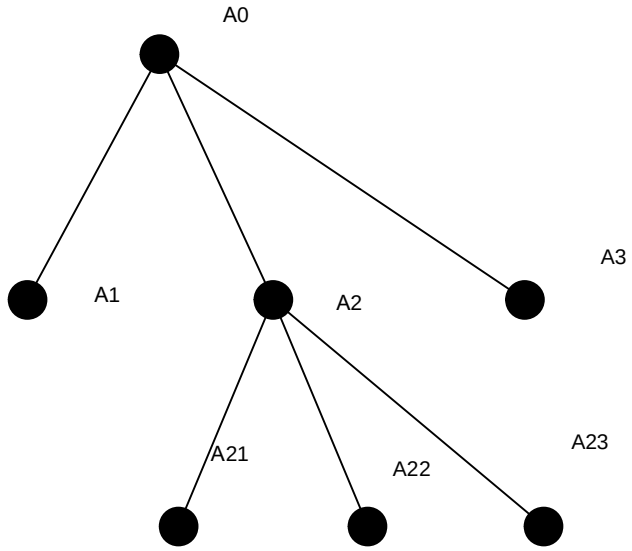


Рисунок 6.8 – Дерево узлов (названия узлов не приводятся).

Любой блок IDEF0 ассоциируется с выполняемой функцией, поэтому модель называется функциональной. В стандарте закреплено рассмотрение функции на различных уровнях абстракции. Функции подразделяют в порядке детализации на: деятельность (синоним дело, имеет цель и потребляет ресурсы, а также преобразует множество потоков входных в выходные), процесс (синоним бизнес процесс, преобразует входные потоки в выходные), операция (преобразует объекты входные в выходные), действие (преобразует свойства объектов, для процесса управления действие называется командой).

Каждый вид функции определяется через нижележащую в иерархии функцию. Например, деятельность - это совокупность процессов и т.д. На одной диаграмме должны рассматриваться функции примерно одного уровня абстракции. Каждая декомпозиция - это изменение уровня абстракции при рассмотрении системы.

Свойства диаграмм

Входные и управляющие стрелки формируют ограничения для данного блока. Функция, ассоциированная с блоком, может выполняться только тогда, когда на вход и управление поступят объекты, указанные в метках диаграммы.

В методологии IDEF0 существует 6 типов отношений между блоками в пределах одной диаграммы: доминирование; управление; выход - вход; обратная связь по управлению; обратная связь по входу; выход – механизм.

Только отношение «Доминирование» отображается положением блоков на диаграмме. Блок, который находится левее и выше другого блока, - доминирующий [26], а второй блок – доминируемый. Все остальные отношения определяются положением стрелок соединяющих два блока. Например, управление - выход доминирующего блока является управлением доминируемого, обратная связь по управлению – выход доминируемого блока является управлением доминирующего и т.д.

Правило построение диаграмм

В стандарте сформулировано двадцать одно правило построения диаграмм. Некоторые правила, обеспечивающие читаемость диаграммы, мы не будем рассматривать.

1. В составе модели должна присутствовать контекстная диаграмма А-0

2. Блоки на диаграмме должны располагаться по диагонали – от левого верхнего угла диаграммы до правого нижнего в порядке присвоенных номеров.

3. Неконтекстные диаграммы должны содержать не менее трех и не более шести блоков.

[26] Доминирование понимается как влияние

4. Каждый блок неконтекстной диаграммы получает номер, помещаемый в правом нижнем углу; порядок нумерации - от верхнего левого к нижнему правому блоку.

5. Каждый блок, подвергнутый декомпозиции, должен иметь ссылку на дочернюю диаграмму.

6. Имена блоков (выполняемых функций) и метки стрелок должны быть уникальными.

7. При наличии стрелок со сложной топологией целесообразно повторить метку для удобства ее идентификации.

8. Следует обеспечить максимальное расстояние между блоками и поворотами стрелок.

9. Блоки всегда должны иметь хотя бы одну управляющую и одну выходную стрелку, но могут не иметь входных стрелок.

10. Если одни и те же данные служат и для управления, и для входа, вычерчивается только стрелка управления.

11. Циклические обратные связи для одного и того же блока изображаются только для того, чтобы их выделить.

12. Стрелки объединяются, если они имеют общий источник или приемник.

13. Если возможно, стрелки присоединяются к блокам в одной и той же позиции.

14. При соединении большого числа блоков необходимо избегать необязательных пересечений стрелок.

15. Блоки (функции) могут быть сопряженными через среду.

16. Две или более функций могут быть сопряженными через запись.

17. Каждая диаграмма должна быть выполнена на стандартном бланке (рисунок 6.2).

Рекомендации построения IDEF0 модели

IDEF0 - является одним из прикладных направлений системного анализа бизнес - процессов. Моделирование начинается с определения элементов контекстной диаграммы. Необходимо правильно выбрать объект моделирования (систему) – функцию верхнего уровня. Данная функция определяет границы системы. Границы системы задаются целью построения модели и точкой зрения. Все, что находится вне этой границы, является окружающей средой. Следующим шагом требуется определить взаимодействие окружающей среды с нашей системой. Объекты окружающей среды не персонифицированы, в данной модели не имеет значения, откуда пришли материальные или информационные потоки и куда ушли. На этом этапе мы находимся на самом верхнем уровне абстракции рассмотрения системы, нас не очень интересуют мелкие детали. Именно здесь отчетливо видно согласованность выбранного объекта моделирования, целям и точке зрения[27].

На этом этапе можно условно разделить цели моделирования на две категории: описание существующей функции, моделирование в таком контексте называют «as-is» (как есть); описание функции в будущем, после того как мы изменим нашу систему, моделирование в таком контексте называют «to-be» (как должно быть). На первой стадии выполнения проектов автоматизации разработчик сталкивается с двумя важными вопросами: что автоматизировать (лучше сказать - выбор объектов автоматизации) и в чем заинтересованность заказчика при выполнении автоматизации. В терминах ГОСТ 34:

- характеристика объекта автоматизации, описание требований к системе;
- эффект, ожидаемый от системы, условия создания и функционирования системы.

Моделирование as-is и to-be поможет ответить на эти вопросы.

---

[27] Этот этап моделирования самый трудный, здесь совершается до 70% ошибок

Рассмотрение функции на самом высоком уровне абстракции должно адекватно соответствовать уровню абстракции в рассмотрении материальных и информационных потоков[28].

Последующая декомпозиция функции потребует соответствующей декомпозиции материальных и информационных потоков.

Блок IDEF0 иногда называют преобразующим, так как любая функция обязана преобразовывать вход в выход. Различные уровни функции (деятельность, процесс, операция и действие) осуществляют преобразование объектов различных уровней (множество потоков, потоки, объекты, свойства объектов). Следовательно, не может вход и выход потока быть одним и тем же.[29].

Информационные потоки можно разделить на три группы: описательные, предписывающие и ограничительные. На рисунке 6.9 показано соотношение групп информации на входах и выходах блоков.

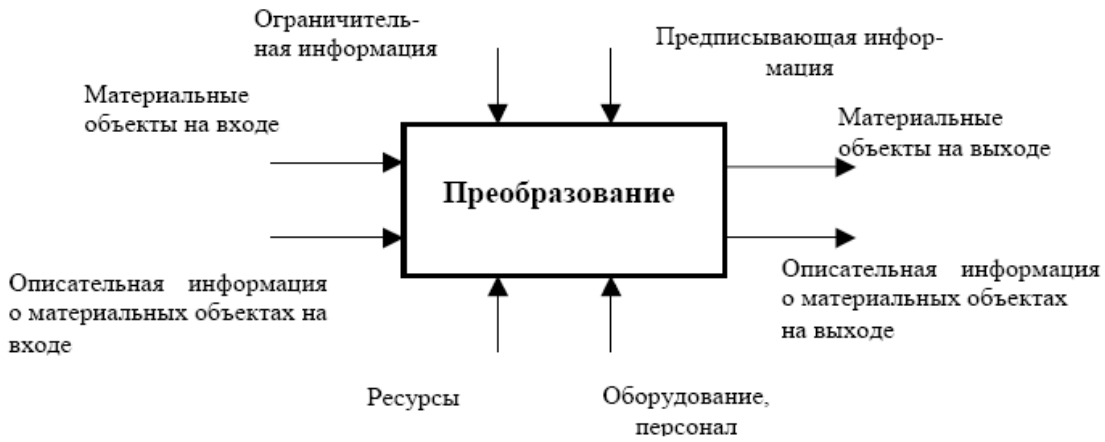


Рисунок 6.9 – Положение информационных потоков.

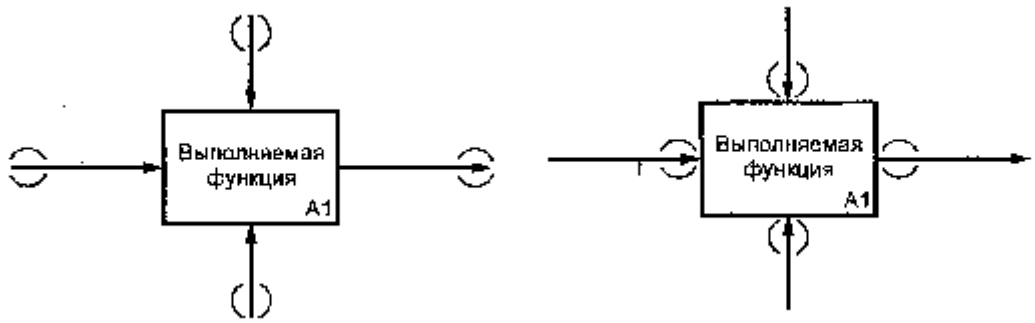
[28] Необходимо помнить, что для материальных потоков работает закон сохранения энергии. Если для данной функции на входе присутствуют материальные потоки, а на выходе их нет, то ваша функция их потребляет. К примеру, колбасный завод изготавливает колбасу и всю ее потребляет.

[29] При этом нужно помнить, что стрелки на диаграмме определяют ограничение функции.

Одним из основных инструментов описания модели является декомпозиция. При выполнении декомпозиции необходимо соблюдать правило согласованности родительской и дочерней диаграммы. Правило заключается в том, что все примыкающие стрелки к блоку должны быть повторены на дочерней диаграмме в виде граничных стрелок[30], более того, все эти объекты, ассоциированные с этими стрелками, должны быть идентичны. Если это правило не выполняется, то необходимо на диаграмме эту несогласованность показать туннелем.

Туннель первого типа (рисунок 6.11а): граничные стрелки присутствуют на декомпозирующей диаграмме и отсутствуют в числе примыкающих стрелок к блоку родительской диаграммы.

Туннель второго типа (рисунок 6.11б): примыкающие стрелки к блоку отсутствуют в числе граничных стрелок декомпозирующей диаграммы.



а б

Рисунок 6.11 – Туннель.

Туннель можно использовать и в том случае, если стрелки присутствуют, но существует несогласованность в объектах или свойствах блока и декомпозирующей диаграммы.

[30] Граничная стрелка – это стрелка, один из концов которой связан с источником или потребителем, а другой не присоединен ни к какому блоку на диаграмме. Отображает связь диаграммы с другими блоками системы и отличается от внутренней стрелки, которые связывают источник и потребителя, являющиеся блоками одной диаграммы.

Туннель дает возможность аналитику поддерживать итерационный принцип работы.

Стандарт не отвечает на вопрос, когда необходимо завершать детализацию. В /58/ указывается, что декомпозиция прекращается, когда диаграммы, образующие нижний уровень модели, достаточно детализированы для достижения цели модели. Другими словами, дальнейшая декомпозиция не требуется, если модель достаточно точна, чтобы отвечать на все вопросы, соответствующие ее цели.

Допустим, цель Вашего исследования бизнес - процесса - найти объект автоматизации. Как мы выяснили во второй главе, чаще всего объектом автоматизации является обработка информации, требующая большой затраты времени. Такая обработка может быть автоматизирована только в том случае, если информация будет вноситься в компьютер. Следовательно, при выборе объекта автоматизации требуется изучить не только процесс обработки информации, но и все процессы, где эта информация возникает и используется. Получив ответ на вопрос - как выполняется в рассматриваемых процессах работа с этой информацией, мы сможем закончить декомпозицию.

Стандарт особое внимание уделяет соотношению организационно-технической структуры предприятия и функциональной структуры модели. Довольно распространенная ошибка при построении модели - привязка выполняемых функций к организационной структуре подразделений.

Аналитик начинает описание бизнес - процесса с определения конкретных работ, выполняемых конкретными людьми, занимающими конкретные должности. В данном представлении организация работ подчинена организационной структуре предприятия. Результат работы оценивается с точки зрения целей и задач подразделения. Процессный подход, реализованный в функциональной модели, дает возможность посмотреть на процесс с точки зрения конечного результата. Аналитик должен описать процессы предприятия с точки зрения конечного результата.

Работа аналитика при описании бизнес - процесса сводится к преобразованию организационной структуры выполнения работ в функциональную структуру. Это положение не отражено в стандарте явно, но вытекает из контекста описания модели.

Аналитик должен определить цели и задачи процесса и, исходя из них, строить модель. ИСО 9000 определяет цели процесса довольно просто: преобразования вход в выход. Цели отображаются на диаграмме IDEF0 входными и выходными стрелками функции, а также названием процесса. Задачи процесса отражаются в декомпозиции функции.

Каждая функция выполняется посредством механизма, в который входят и люди с их должностными обязанностями. Модель IDEF0 дает возможность сгруппировать механизмы в организационно-технические структуры, уровни которой будут соответствовать уровням декомпозиции IDEF0. Так, например: деятельность **à** организационно-техническая система, процесс **à** организационно-техническая подсистема и т.д. Организационно-техническая структура становится результатом функционального моделирования.

При построении модели стандарт рекомендует особо выделить функцию «Управление». Не надо путать функцию «Управление» со стрелкой «Управление», входящий в блок сверху. Стрелка реализует концепцию ограничительной и предписывающей информации. Функция «Управление» должна генерировать такую информацию, которая будет отображаться на диаграмме отношением управления (рисунок 6.11).

Эффективность и производительность труда разработчиков функциональных моделей может быть повышена за счет применения типовых моделей и отдельных диаграмм, ориентированных на применение в конкретных предметных областях. Фрагмент типовой модели IDEF0 промышленного предприятия приведен в Приложении В стандарта.

В стандарте заложена очень важная концепция итерационной работы над моделью. Итерационная модель жизненного цикла IDEF0 согласуется с

принципами работы всех современных методологий разработки информационных систем.

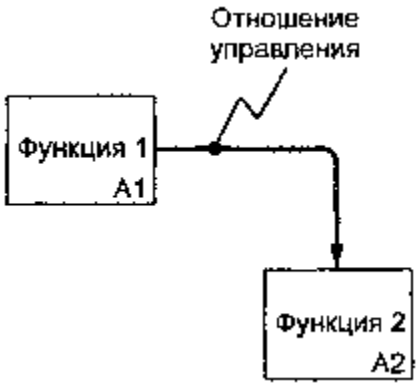


Рисунок 6.11 – Отношение управления.

Иерархическая модель IDEF0 начинается с контекстной диаграммы, т.е. по принципу сверху – вниз. Находясь на нижнем уровне декомпозиции, аналитик может обнаружить или новые правила выполнения функций, выражающиеся в необходимости получения материальных или информационных объектов от функций другого уровня, или новые функции, требующиеся для выполнения функции более высокого уровня. В том и другом случае он должен провести исследование по принципу снизу-вверх, а дойдя до той функции, с которой связан вновь обнаруженный объект, он снова должен проверить модель, двигаясь сверху - вниз.

При построении модели необходимо помнить, что диаграмма - главная составляющая модели, но не единственная. Могут присутствовать еще три компонента: текст, глоссарий и диаграммы иллюстрации. Текст и глоссарий, обязательные компоненты модели. Модель без этих компонентов не существует. Поясняющий материал к диаграмме аналитик располагает в тексте. Все документы, присутствующие на диаграмме, должны быть включены в текстовое описание или глоссарий в качестве приложения. Необходимо отличать глоссарий предметной области от глоссария модели. Глоссарий предметной области отражает термины и определения, принятые специалистами

области исследования. Глоссарий модели определяет аббревиатуры, ключевые слова фразы, используемые в именах меток и блоков.

Диаграмма потоков данных (Data Flow Diagrams DFD)

В DFD-методологиях вместо реальных объектов рассматриваются отношения, описывающие свойства объектов и правила их поведения. Они применимы к системам обработки информации, например, для разработки прикладного ПО, а не к системам с жесткими технологическими процессами.

В методологии Гейн - Сарсона и Де -Марка (T. De Marca) моделирование начинается с существующей системы и проводится до разработки новой (физической и логической моделей). Стратегия построения требований к новой системе включает:

- моделирование текущих операций;
- выявление причин выполнения именно этих операций;
- добавление новых требований;
- выбор границ автоматизации.

В методологии Йордана не рекомендуется моделировать текущую систему. В ней добавлена предварительная фаза разработки, названная созданием основной модели (essential model), а также определена техника «событийного разбиения» (event partitioning) для конструирования DFD-схемы. Большое внимание уделяется информационному моделированию (посредством ER- диаграммы) и моделированию поведения (через STD-диаграмм). В модели Йордана определено прототипирование в жизненном цикле разработки, а также имеется описание семантики потоков и правила преобразования входных данных в выходные.

В данном разделе будут описаны синтаксис и семантика диаграмм DFD в различных нотациях и рекомендации их построения, а также правила построения структурированного глоссария, предложенного Йорданом

В MS Visio присутствуют две нотации. При рассмотрении нотации Йордана /61/ необходимо принимать во внимание, что метод Йордана развивает DFD понимание DFD модели. В метод Йордана включена STD[31] диаграмма и ERD диаграмма. Включенные модели органически вписываются в концепцию разработки ПО. DFD диаграмма - статическая диаграмма, которая не рассматривает поведение элементов во времени. Данный пробел восполняется диаграммой состояний /62/. Органично вписывается применение в методологии диаграмм ERD. Потоки данных, направленные в хранилище данных, отображают свойства информационных объектов и их структуру. В методе Йордана эта информации используется при построении схемы данных. На рисунке 6.12 показан состав методологии Йордана /62/.

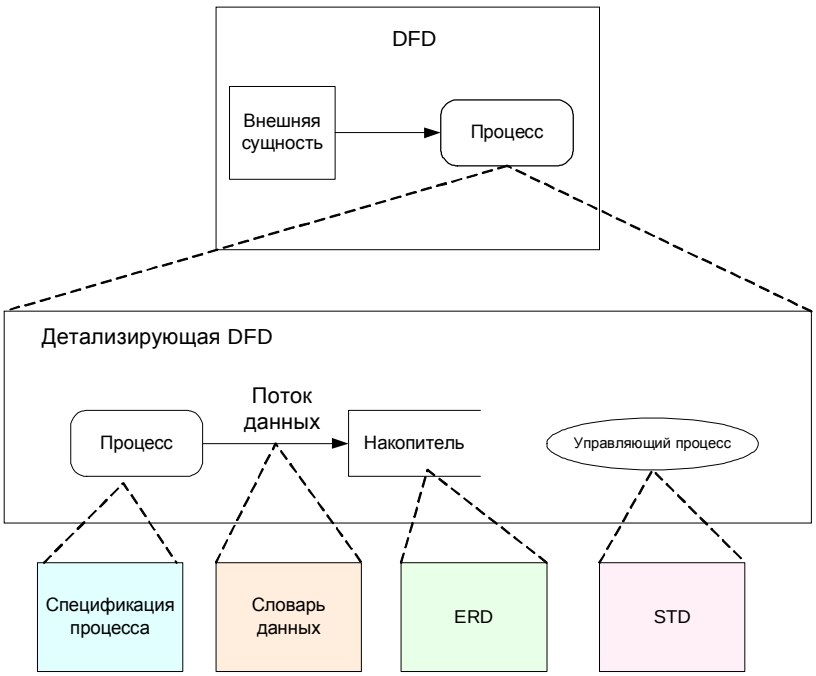


Рисунок 6.12 - Состав методологии Йордана по /62/

Состав модели Йордана, описанный в книге /61/

- контекстная диаграмма,
- лист событий,
- конечный отчет,

[31] State-Transition Diagram диаграмма переходов (состояний) методология моделирования последующего функционирования системы на основе её предыдущего и текущего функционирования

- диаграмм потоков данных,
- диаграмма сущность связь,
- диаграммы состояния,
- словарь данных,
- спецификаций процесса, для каждого процесса нижнего уровня.

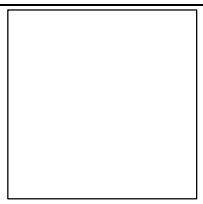
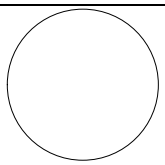
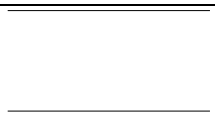
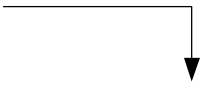
На примере этих небольших разночтений публикаций Йордана /61/ и русскоязычных публикаций, поясним основные сложности, которые, по нашему мнению, существуют в DFD модели. Во-первых, книги авторов DFD моделей не переведены на русский язык, что затрудняет знакомство с этой методологией широкого круга читателей. Во-вторых, статьи, книги Интернет публикации, многократно излагая DFD модели на русском языке, создали некоторое обобщенное представление, акцентирующее основное внимание на DFD диаграммах. В-третьих, развитие новых методологий проектирования и особенно развитие UML, уменьшило внимание к DFD. Поэтому в данном пособии DFD разобрано весьма скромно. Основная цель изложения материала - показать возможность использовать функциональное моделирование на последующих стадиях разработки программного обеспечения. Это будет выполнено в контексте работ /60-64/. Стоит отметить работы Калянова Г.Н., который внес свой вклад в развитии DFD модели.

Синтаксис и семантика объектов нотации

Основные синтаксические элементы в различных нотациях приведены в таблице 6.1. Как видно, можно считать не существенным различие между этими двумя нотациями. Основными структурными элементами диаграммы являются процессы и потоки данных, аналогично блокам и стрелкам в модели IDEF0. В нотации данной модели не показывается механизм или исполнитель процесса. В тоже время эта информации, как и другая, характеризующая процессы, может быть отражена в спецификацию процесса. Процесс DFD и функция IDEF0 полностью идентичны. Диаграмма DFD так же представляет

функциональную модель производства, которое является отображением процессного подхода.

Таблица 6.1 Изображение различных объектов DFD в нотациях Гейн-Сарсона и Йордана

Объект модели	Нотация Гейн- Сарсона	Нотация Йордана
Внешняя сущность		
Процесс		
Хранилище (накопитель данных)		
Поток данных		

Имя процесса должно содержать глагол в неопределенной форме с последующим дополнением (например, Проверить стаж работы) или отглагольное существительное (Проверка стажа работы). Цель процесса - преобразование входных потоков в выходные. Поэтому у процесса обязан быть входной и выходной поток. Процесс может иметь номер. Номер процесса во многих нотациях не регламентируется. Мы рекомендуем использовать узловой номер блока IDEF0 модели, со всеми правилами его построения.

Поток изображается в виде стрелки. Стрелка не может быть «висячая». Она начинается и заканчивается на объекте (объект может быть один и тот же). В качестве объекта могут быть все остальные элементы DFD диаграммы. Поток должен иметь уникальное имя, которое идентифицирует его в словаре данных. Название потока должно быть именем существительным и кратко характеризовать данные потока (например, паспортные данные клиента). По-

токи могут объединяться в другие потоки, в тоже время, данные в потоке могут быть объединены в пачки (адрес состоит из названия города, улицы и т.д.). Поток, исходя из названия диаграммы, не может отражать ничего иного кроме, как набор данных. DFD создавалась, как модель ,используемая в разработке информационных систем, и соответственно, в ней аналитика могут интересовать только информационные потоки. Мы будем считать за ошибку показ на диаграмме материальных потоков.

Вот тут бы въедливый студент мог бы заметить противоречие. Оно скрыто в различии терминов «информация»[32] и «данные»[33], а также в том, что данные и информация могут быть представлены как на материальных носителях, так и в электронном виде. Следовательно, на DFD диаграмме обязательно присутствуют материальные потоки – данные и/или информация на материальных носителях. При этом не обязательно прослеживать судьбу материального носителя информации.

Хранилище (накопитель данных) позволяет выявить данные, которые будут сохраняться вне процессов. Данные, которые оно содержит, может использоваться в любое время после ее определения (после поступления с входными потоками), при этом данные могут выбираться в любом порядке. Основное отличие накопителя от процесса заключается в том, что основная его цель хранить данные без изменения. Данные во входном и выходном потоке процесса обязаны быть отличными, накопителя – одинаковыми. При этом не уточняется способ помещения и извлечения данных в накопитель, нас не интересует, происходит ли извлечение данных для чтения (копирования) или для изъятия и другие подобные вопросы. Имя накопителя должно

---

[32] Информация, обработанная и представленная в формализованном виде для дальнейшей обработки (ГОСТ 7.0 -99). В точных науках есть ясное различие между данными и информацией, где данные - измерение, которое не имеет организацию, информация в отличие от данных имеет порядок (http://en.wikipedia.org/wiki/Data).

[33] Данные. Способ представления фактов, концепций или инструкций, подходящей для связи, интерпретации или обработки их людьми или автоматическими средствами /65/.

идентифицировать его содержимое и быть существительным во множественном числе. Накопитель должен иметь уникальный номер. Допускается использовать накопитель в разных диаграммах. Рекомендуется связь с накопителем показывать непосредственно на диаграмме, а не через родительскую диаграмму. В словаре данных специфицируются потоки, показанные на диаграмме.

Поскольку в системе используются материальные потоки (материальные носители данных или информации), то, соответственно, может допускаться хранилище этих носителей. Входные и выходные потоки хранилищ материальных носителей должны быть только материальные.

Внешняя сущность - это объект вне контекста системы, являющийся источником или приемником данных нашей системы (контекстного процесса), например: клиент, отдел кадров, Министерство образования и науки. Внешняя сущность, являясь объектом окружающей среды, определяет границу нашей системы. Нас интересует взаимодействие системы с окружающей средой, но не интересует взаимодействие объектов окружающей среды. Не допускаются потоки между внешними сущностями.

В DFD модели нет никакой информация о самой диаграмме. Нет элементов языка, позволяющих установить связь между диаграммой и процессом, который она детализирует. Этот, довольно существенный недостаток DFD диаграмм BPWin решает за счет использования рамок диаграмм от IDEF0 (точнее сказать DFD диаграммы выполняются как диаграммы FEO на бланке IDEF0). Нам кажется это оптимальным решением при DFD моделировании.

Модель DFD состоит из трех типов документов: графические диаграммы, спецификация процессов и глоссарий.

Спецификация процесса используется для описания функционирования процесса в случае отсутствия необходимости его детализации, т.е. она вы-

полняется для процессов нижнего уровня. Спецификация представляет собой: предусловие, постусловие и описание процесса.

Предусловие описывает условие, которое необходимо выполнить для выполнения процесса (в IDEF0 наличие объектов ассоциированных со всеми входными стрелками).

Постусловие описывает условие, которое должно быть выполнено по завершению процесса.

Описание процесса - это алгоритмы выполнения задач процесса, в ходе выполнения которых входной поток преобразуется в выходной. В описании не требуется определять метод реализации этого преобразования. Набор конструкций для построения спецификации должен быть простым и понятным. Йордан предлагает использовать структурированный язык, вид которого напоминает псевдокод. Применяются программные конструкции (цикл, условные переходы и т.д.), но при этом не используется синтаксис определенного языка программирования. Также Йордан предлагает применять различные таблицы, диаграммы и графики. В отечественной литературе понятие структурированного языка /62, 63/ весьма расплывчато. Применение в практике анализа конструкций (@ Вход, @Выход и т.д.), описанных в монографии Г.Н. Калянова, по нашему мнению не целесообразно. Нам также кажется надуманным использование /62/ термина мини-спецификация.

Множество всех спецификаций является полной спецификацией системы.

Словарь данных требует применения формализованных языковых конструкций в связи с тем, что данные используются в алгоритмах и в электронных хранилищах.

Словарь данных применяется:

- для описания содержания потоков и хранилищ, показанных в диаграммах DFD;

- для описания состава сложных пачек (адрес), которые могут содержать атомарные34 данные (города, государство и почтовый индекс);
- для описания состава пачек данных в хранилищах;
- для определения значений и единиц измерения атомарных данных.

Есть много общих языковых конструкций, используемых системными аналитиками для организации словаря данных. Один из них приведен ниже

:=	Составлен из
+	и
( )	дополнительный элемент(может присутствовать или отсутствовать)
{ }	повторение элементов
[ ]	альтернативные варианты
**	комментарии
@	ключевое поле в хранилище
|	отдельные альтернативные элементы выбора в конструкции [ ]

Так, например, определение адреса в потоке данных:

адрес = название страны + (название федерального округа) + (название типа субъекта федерации) + (название субъекта федерации) + (название района субъекта федерации)+ название типа населенного пункта + название населенного пункта + (название района населенного пункта) + название типа объекта населенного пункта + (название объекта населенного пункта) +номер дома + (номер корпуса)+ (номер квартиры)

название типа населенного пункта = [город | районный центр | поселок | село | рабочий поселок | и т.д]

[34] Атомарные данные не имеют собственной структуры.

имя субъекта федерации = {слова}

и т.д.

По примеру Калянова Г.Н./63 / можно было назвать это БНФ грамматикой, но это не совпадает с грамматикой ISO -14977.

При таком формализованном описании может потеряться наглядность отображения информации, а также конструкция описания пакета может быть достаточно сложной.

Мы рекомендуем, где это, возможно, иллюстрировать поток данных формой документа, с которой он ассоциирован. Так, например, при поступлении студента в университет он заполняет анкету. Поля этой анкеты образуют поток от внешней сущности «студент» к процессу, скажем «прием документов». В словаре данных вместе с формальным описанием потока данных на структурированном языке, описанном выше, требуется приложить (включить в приложение словаря и сослаться на него) форму анкеты студента.

Рекомендации построения DFD модели

Построение модели начинается с контекстной диаграммы. Задача контекстной диаграммы - определение границ системы (наша система - процесс). Для этого необходимо определить процесс (хотя бы его название) и идентифицировать все внешние объекты, с которыми он может взаимодействовать.

На этом этапе должно быть определено, как и в IDEF0, зачем строится модель, и с какой точки зрения она строится. Цель и точка зрения не фиксируется на диаграмме, но должны присутствовать в отчете. На контекстной диаграмме может присутствовать несколько функций (Йордан не запрещает присутствие хранилища данных, но мы не рекомендуем помещать их на контекстную диаграмму).

Далее необходимо идентифицировать информационные потоки между внешними сущностями и системой. Рекомендуется сразу идентифицировать потоки в глоссарии. Если количество потоков большое и мешает восприятию диаграммы, необходимо их объединить. Данные, сгруппированные в пакеты, не следует разъединять. В последующем, структура информации – это пакет, который может обеспечить выявление транзакций баз данных.

Следующий этап Йордан называет предварительным. На нем производится декомпозиция системы на процессы. При этом идентифицируются процессы, составляющие систему с их кратким описанием. На этом этапе не требуется создание спецификации процессов, а требуется только их выявить и согласовать со специалистом в предметной области. Не будем обращать внимания на то, что процессы могут находиться на разном уровне абстракции. На этой стадии преобладает в анализе движение сверху вниз.

На следующем шаге необходимо определить потоки данных между процессами. На этом этапе возможна группировка части процессов или включение их в другой процесс. Не требуется акцентировать свое внимание на хранилищах. Необходимо добиться согласования потоков родительских и дочерних диаграмм. На этом этапе начинается проектирование снизу вверх.

Далее осуществляется группировка процессов со сходными задачами, и определяются хранилища. Продолжается проектирование снизу вверх (Рисунок 6.13).

На всех этапах требуется вести глоссарий и спецификацию процессов. Как видите, такое проектирование постоянно контролирует количество процессов нижнего уровня, что препятствует лавинообразное размножение диаграмм, что может наблюдаться в IDEF0.

Декомпозиция и объединение процессов производится совместно с декомпозицией и объединением потоков.

При выявлении хранилищ необходимо соблюдать следующие правила. Хранилища системы появляются после декомпозиции контекстной диаграммы (Рисунок 6.14).

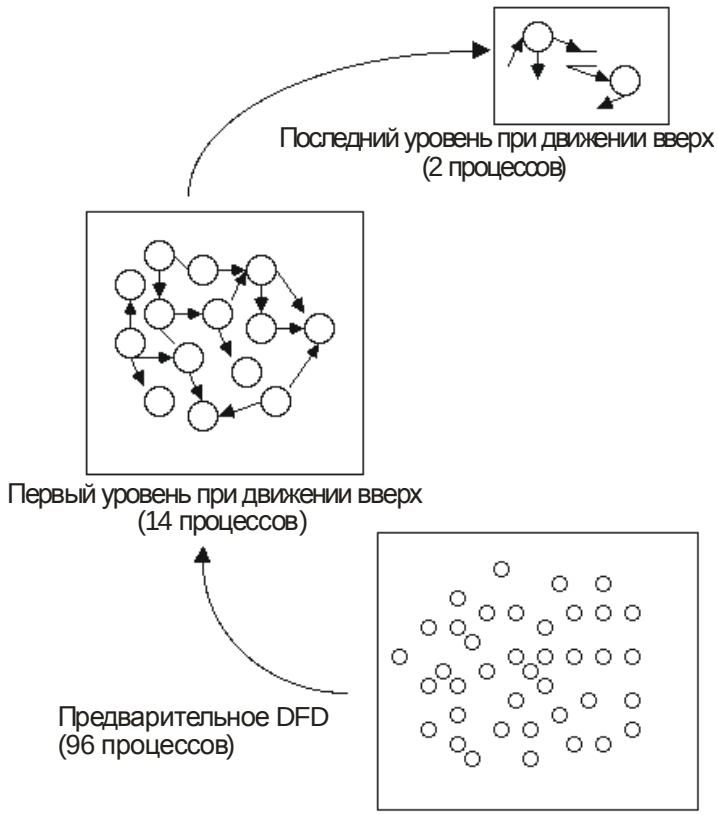


Рисунок 6.13 Проектирование снизу вверх по Йордану.

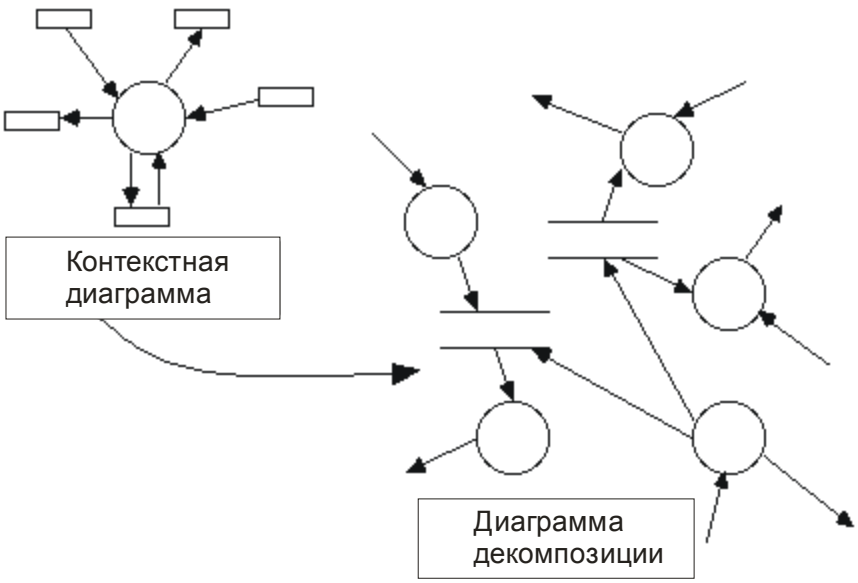


Рисунок 6.14 – Выявление хранилищ при движении сверху вниз

При группировке процессов по деятельностям в ситуации, показанной на рисунке 6.15.

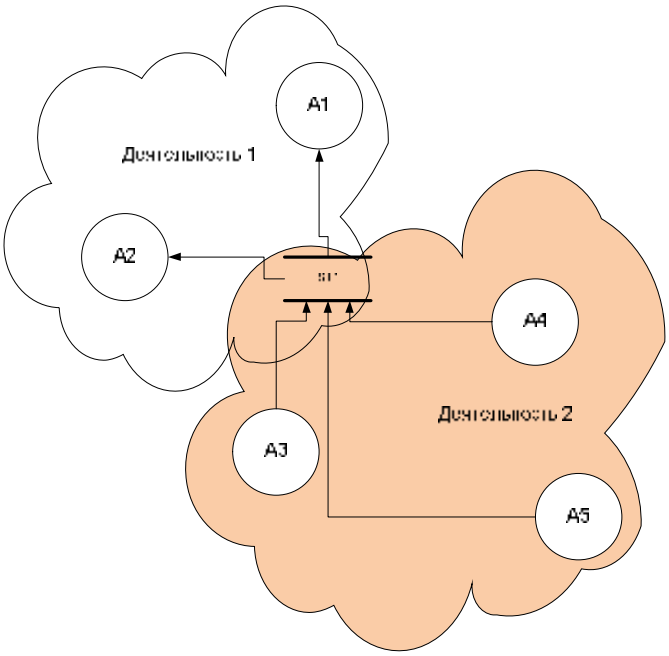


Рисунок 6.15 Группировка процессов.

Взаимодействие их через хранилище необходимо производить так, как показано на рисунке 6.16

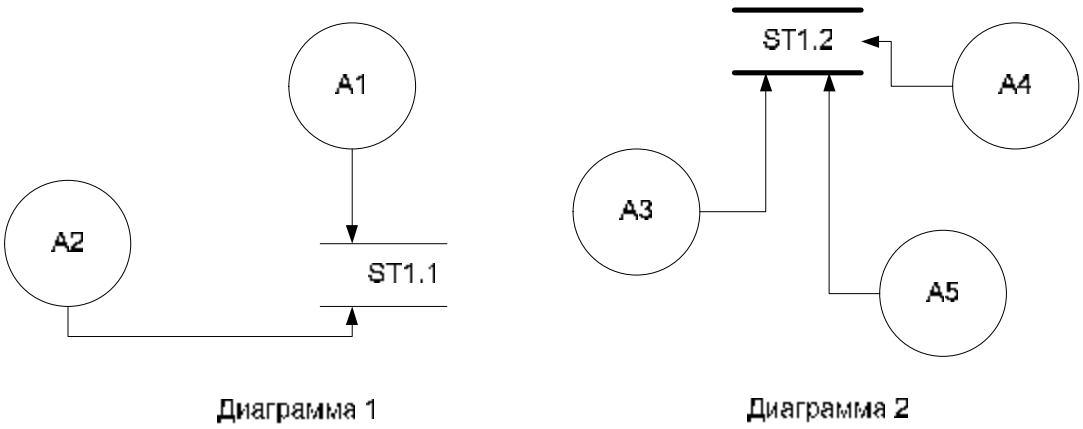


Рисунок 6.16 – Представление одного и того же хранилища на различных диаграммах после группировки процессов.

Одно и то же хранилище будет присутствовать на различных диаграммах. Такое взаимодействие предпочтительнее, чем связывание процессов через граничные потоки, показанное на рисунке 6.17.

Как видите, в DFD организован итерационный процесс, как и в IDEF0 моделировании, но организация движения сверху - вниз и снизу - вверх различается.

Сравнение IDEF0 и DFD моделирования

Рассмотрены две методологии структурного функционального анализа. Если не принимать во внимание включенные в модель Йордана ERD и STD диаграммы, которые мы не рассматривали, то модели очень похожи. Поэтому напрашивается законный вопрос, какая модель лучше? Мы попытались свести в таблице 6.2 основные отличия двух методологий.

Как видно из таблицы, отдать предпочтение было бы трудно какой - либо методологии. По нашему мнению, описание бизнес - системы лучше делать, используя IDEF0. Ее лучше понимают менеджеры и у нее есть несомненный плюс – существование российского стандарта.

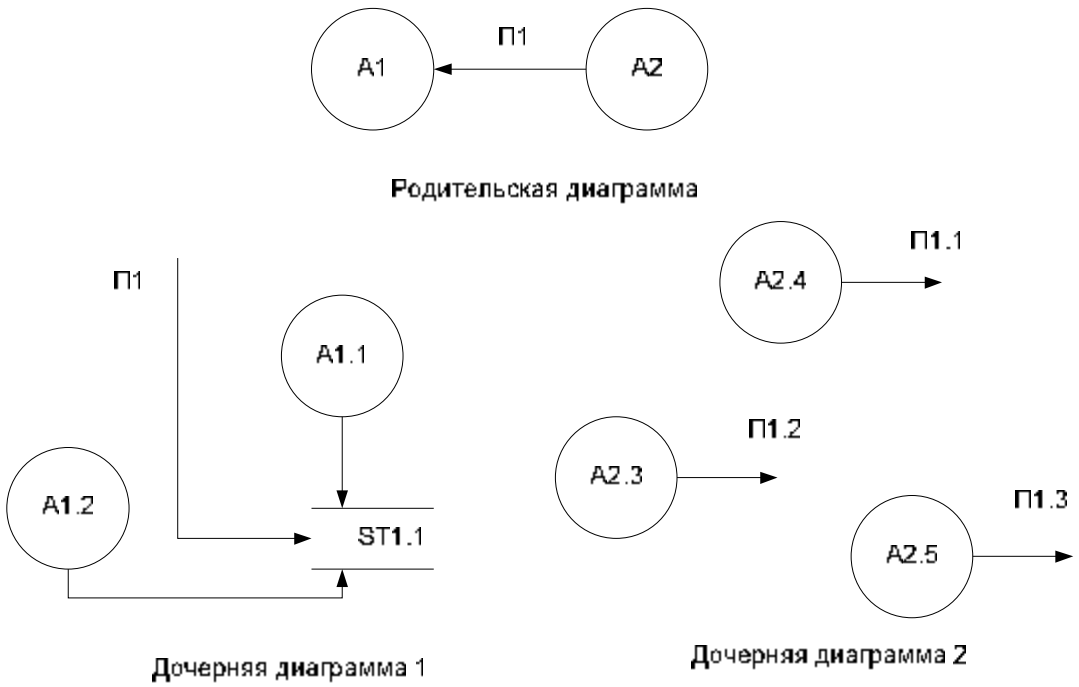


Рисунок 6.17- Взаимодействие процессов через родительскую диаграмму (не рекомендуется).

Таблица 6.2 Сравнение моделей IDEF0 и DFD/

Свойства	Модель IDEF0	Модель DFD
Графический язык	+	+
Строгость и формализм описания	+	-
Существование стандарта	+	-
Итеративность работы	+	+
Процессный подход	+	+
Возможность идентификации хранимых данных	-	+
Отличие в элементах модели, перечень элементов отсутствующих в другой модели	Бланк диаграммы с информационными полями диаграммы, дерево узлов, ссылочный код	Внешняя сущность, Хранилища данных
Наличие правил построение диаграмм	Правила лучше формализованы, их количество больше	+
Возможность процессов с материальными потоками	+	-
Согласованность с другими методологиями проектирования информационных систем	Слабая, необходимо специально ор-	Организована в методологии Йордана
Согласованность с имитационным моделированием систем	+	+

При проектировании информационных систем модель IDEF0 оправданна только при моделировании системы с точки зрения «as-is».

Использование аналитиком методологии DFD значительно усложняется тем, что полное описание ее существует только в англоязычном варианте. Описание методологии Йордана на русском языке можно считать только ознакомительным.

Когда весь проект выполняется методами структурированного анализа и проектирования систем, целесообразно применять DFD моделирование. Современные объектно-ориентированные методологии значительно превосходят методологию структурного моделирования по всем параметрам, кроме простоты. Если разработчик плохо знаком с парадигмой ООП и использова-

ние им основных концепций ООП маловероятно, применение DFD моделирования оправдано (услышь нас, студент - «неотличник»).

Контрольные вопросы

1. Что такое функциональное моделирование и чем оно отличается от структурного моделирования?
2. Когда Вы исследуете бизнес-процессы предприятия, какую модель, по Вашему мнению, следует использовать?
3. Что общего и в чем отличия DFD нотации и методологии Йордана?
4. В чем отличие функционального моделирования в методологиях Йордана и IDEF0?
5. Возможно ли моделировать материальные объекты потоками данных? Документы при этом следует отнести к материальным объектам или к информационным?
6. Могут ли свидетельствовать туннели на диаграмме IDEF0 о незавершенности процесса моделирования? Обоснуете свой ответ.
7. В чем преимущество модели IDEF0 по сравнению с DFD?
8. В каких случаях, по Вашему мнению, следует применять IDEF0 моделирование?
9. Целесообразно ли применять DFD моделирование для случая «as is»? Обоснуйте свое мнение.
10. Можно ли отобразить хранилища данных на диаграмме IDEF0 или в модели IDEF0?

7. Моделирование систем. Проектирование схемы данных

В данном разделе мы рассмотрим вопросы проектирования схемы данных. Будут рассмотрены две ER - модели (модели сущность-связь) или ERD (диаграммы сущность связь): ERD концептуального уровня (мы будем называть ее ERD в нотации Чена); ERD логического уровня - стандарт IDEF1X.

ER - модели не принадлежат множеству методологий, относящихся к структурному проектированию, но эти модели достаточно хорошо вписываются в концепцию структурного проектирования. В предыдущей главе мы разобрали DFD моделирование. DFD дает возможность фиксировать хранимые данные. Более того, хранилища данных дают возможность выявить первичную структуру этих данных, формирующую из них потоки и пакеты. Это дает возможность прейти к первому этапу моделирования данных. Йордан один из первых обратил внимание на органическую связь хранилищ DFD с ERD моделью, которая позволила объединить в своей методологии эти две модели. В данном пособии мы применим тот же подход.

Проектирование схемы данных является только частью жизненного цикла ПО. В книге Т. Конноли, Л. Бегг, А. Строчан /54/ приводится методология проектирования базы данных, состоящая из трех фаз: концептуального, логического и физического проектирования.

Концептуальное моделирование - процедура конструирования информационной модели предприятия, не зависящей от каких-либо физических условий реализации. Результатом фазы является концептуальная модель данных предприятия, полностью независимая от деталей реализации, таких как:

- выбранный тип СУБД,
- состав программ приложения,
- используемый язык программирования,

– конкретная вычислительная платформа и т.д.

Концептуальная фаза состоит из одного этапа создания локальной концептуальной схемы данных.

Фаза логического проектирования базы данных заключается в преобразовании концептуальной модели данных в логическую модель данных предприятия с учетом выбранного типа СУБД (например, реляционный тип СУБД), но не зависит от используемой СУБД и прочих физических условий реализации. Фаза логического проектирования состоит из двух этапов. Первый этап - преобразование локальной концептуальной модели данных в логическую. Второй этап – соединение всех локальных логических моделей данных в одну глобальную логическую модель.

Фаза физического проектирования процесс создания описания конкретной реализации базы данных, размещаемой во вторичной памяти. Предусматривает описание структуры хранения данных и методов доступа, предназначенных для осуществления наиболее эффективного доступа к информации. Фаза физического проектирования базы данных предусматривает принятие разработчиком окончательного решения о способах реализации создаваемой базы. Поэтому физическое проектирование обязательно производится с учетом всех особенностей используемой СУБД. Фаза физического проектирования состоит из четырех этапов: перенос глобальной логической модели данных в среду целевой СУБД; проектирование физического представления базы данных; разработка механизмов защиты; организация мониторинга и настройка функционирования системы.

ERD - модель в нотации Чена предоставляет инструмент для концептуального моделирования данных, а стандарт IDEF1X предоставляет инструмент для построении логической, реляционной схемы данных.

ERD модель (нотация Чена)

Данный подход основан П. Ченом /53/, в последствие модель была дополнена и развита другими исследователями. В /54/ разделяют ERD и EER модификации модели сущность связь. В данном разделе мы не будем рассматривать модель в разрезе истории ее развития, более естественный путь, выбранный нами, основан на работе /52/. Данная модель будет нами называться концептуальная модель данных, ER - модель Чена или ERD в нотации Чена, иногда, пусть простят нас за тавтологию, ERD - диаграмма в нотации Чена.

В своих работах П. Чен основывался на объектно-ориентированной (ООМ) и семантической[35] (СМ) моделях. ООМ – модель, представляющая категории реального мира в виде объектов, а не записей. СМ – модель, отражающая значения реальных категорий[36] и отношений между ними. Обе модели изображаются в виде графа, только в одной вершине графа - объекты, а в другой - категории.

ER - модель с одной стороны, отображает объекты реального мира фактически в виде диаграммы классов (правда несколько урезанную, по сравнению с UML), а с другой стороны, образует семантическую сеть[37], отражающую знания реального мира.

Терминология ER - модели больше напоминает объектно-ориентированный язык, но в тоже время, отталкиваться мы будем от языковых конструкций, отражающих понятия реального мира.

---

[35] Семантика – раздел языкознания , занимающийся значениями (смыслом) языковых единиц /56/.

[36] Под семантической категорией обычно понимается замкнутая система значений некоторого универсального семантического признака или же отдельное значение этого признака безотносительно к степени их грамматикализации и способу выражения в конкретном языке /66/.

[37] Семантическая сеть — один из способов представления знаний. В семантической сети роль вершин выполняют понятия базы знаний, а дуги (причем направленные) задают отношения между ними. Таким образом, семантическая сеть отражает семантику предметной области в виде понятий и отношений /2/.

Основы концептуальной модели

Сущность - отображение объектов реального мира, которую пользователь считает важной в моделируемой части реального мира. Сущность является объектом модели. Есть объект реального мира - Иванов Иван Иванович, а есть сущность, отображающая этого конкретного человека; есть предприятие - ООО «Непашемнесеем», а есть сущность, отображающая это конкретное предприятие.

Тип сущности (далее сущность) - это множество сущностей одного типа. Тип сущности отражает объектное множество[38] реального мира.

Экземпляр сущности является конкретным элементом типа сущности.

В модели тип сущности представляется прямоугольником (Рисунок 7.1). Название типа сущности - существительное. Экземпляр сущности не имеет графического представления.

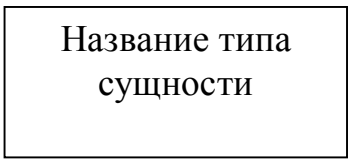


Рисунок 7.1 – Тип сущности.

Объектные множества подразделяются на лексические и на абстрактные. Элементы лексических объектных множеств можно напечатать, тогда, как элементы абстрактных объектных множеств напечатать нельзя. Так, например, имя будет лексическим объектным множеством, поскольку его элементами являются имена, то есть строки символов, которые можно напечатать.

Человек является абстрактным объектным множеством, поскольку его невозможно напечатать. С другой стороны, человека можно представить лексическими объектами, таким как имя, номер водительских прав и т.д. В концептуальной модели можно представить тип сущности «человек», отображающий объектное множество «человека», а также типы сущности «имя»,

[38] Объектное множество - это множество объектов одного типа.

«водительские права». Экземпляр типа сущности человек связан с экземплярами типов сущности, отражающих лексические объекты. Вот тут возникает некоторый пробел. Лексические объекты имеют названия (поэтому они и лексические). Мы понимаем, что «Иванов Иван Иванович» - экземпляр лексического объектного множества «ИМЯ». Объект «Иванов Иван Иванович» связан с конкретным объектом объектного множества «Человек», но назвать этот объект мы не можем (они же не лексические). Для того, чтобы идентифицировать нелексический объект, необходимо придумать этот идентификатор. Суррогатный ключ – идентификатор абстрактного объекта в какой-нибудь системе, вне этой системы смысла не имеет. Допустим, мы идентифицируем человека в системе налогообложения РФ, тогда суррогатный ключ является ИНН. Из контекста понятно, что суррогатный ключ - это лексическое объектное множество, каждый объект которого имеет изоморфное отображение в нелексическое объектное множество. Проще говоря, каждый человек в системе налогообложения РФ имеет один ИНН и каждый ИНН связан с конкретным человеком.

В реальности мы часто встречаемся с понятиями обобщение и конкретизация. Конкретное объектное множество может принадлежать другому объектному множеству или состоять из некоторого количества объектных множеств. Примеры: Человек ⊃ Мужчина, Человек ⊃ Женщина, Жители планеты ⊃ Человек. Представление в ER - модели приведено на рисунке 7.2.

Конкретизация - объектное множество является подмножеством другого объектного множества. Обобщение - объектное множество является надмножеством другого объектного множества.

В данном случае, объект - конкретный мужчина, которого зовут Иванов Иван Иванович, принадлежит к объектному множеству Человек и объектному множеству Мужчина. На ER - диаграмме (рисунок 7.2) это факт не отображается (нет отображения объекта).

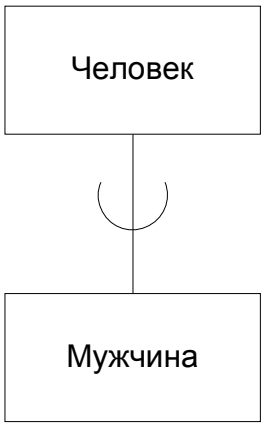


Рисунок 7.2 - Представление конкретизации и обобщения.

Отношением будем называть связь между объектами, принадлежащим разным объектным множествам. Так, например, объектные множества состоят[39]:

ЖЕНАТЫЙ МУЖЧИНА = {Николай, Александр, Иван}[40]

ЗАМУЖНЯЯ ЖЕНЩИНА = {Анна, Елена, Глафира}.

Тогда отношение СОСТОИТ В БРАКЕ, отражающее описание:

Николай	состоит в браке с	Анной
Александр	состоит в браке с	Еленой
Иван	состоит в браке с	Глафирой

Можно записать:

СОСТОИТ В БРАКЕ= {(Николай, Анна),(Александр, Елена),(Иван, Глафира)}

СОСТОИТ В БРАКЕ - будем называть составным объектным множеством.

Объектом составного множества является отношение между конкретными объектами. Каждый объект этого отношения отражает конкретную связь с конкретным объектом первого множества с объектом второго множества. Элемент множества (Николай, Анна) - это реальный объект мира, который можно описывать различными лексическими конструкциями, являю-

[39] Для простоты изложения объект мы характеризуем именем, это неправильно. Вы помните, что нелексические объекты должны характеризоваться суррогатным ключом, но если предположить, что имена у людей не повторяются, то в качестве суррогатного ключа может быть и имя.

[40] Фигурные скобки обозначают множество.

щиеся отражением этого реального объекта (например, брак счастливый). Все связи в данном примере обобщены в тип связи «состоит в браке», можно придумать другие типы связи с данными типами объектов.

Под **типом связи** мы будем понимать осмысленную ассоциацию между сущностями разных типов. В ER модели **тип связи** (далее связь) изображается ромбом (рисунок 7.3). Тип связи называется глаголом или отглагольной формой. Направление связи не отражается на ER диаграмме, хотя может и присутствовать в порядке слов описывающих эту связь (связь «находится замужем» в языковых конструкциях имеет направление). Экземпляр связи – это ассоциация между сущностями, включающая по одному экземпляру сущности из каждого участвующего в связи типе сущности.

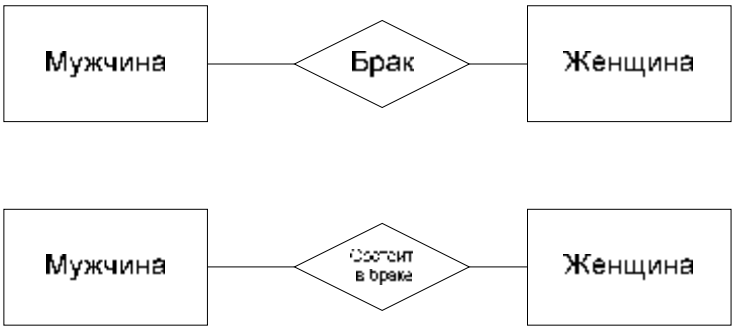


Рисунок 7.3 - Тип связи.

Основное отличие типа связи от типа сущности, в том, что тип связи не существует без типа сущности, а тип сущности может быть не связан с другими сущностями.

В нашем примере мы видим, что отношение «СОСТОИТ В БРАКЕ» само является объектным множеством, элементами которого будут семейные пары. Такое объектное множество, называется **составным объектным множеством**. Составное объектное множество не имеет графического отображения на ER диаграмме. В /52/ предлагается обозначать графически так, как показано на рисунке 7.4.

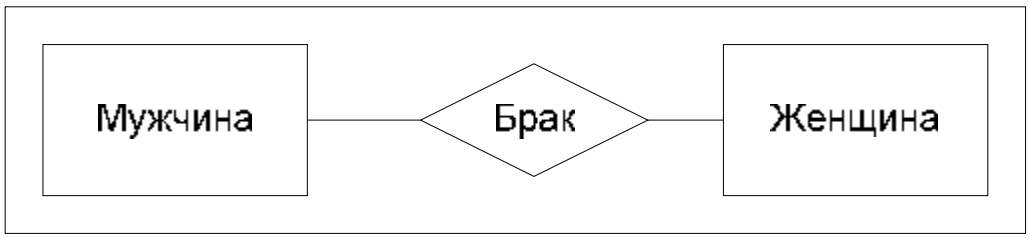


Рисунок 7.4 – Отображение составного объектного множества в ER модели /52/.

Можно ее интерпретировать как сущность, которая имеет внутреннюю структуру, схема внутри квадрата. Можно назвать эту сущность в нашем примере - «Семейная пара». Такое представление составного объектного множества дает нам возможность использовать его в других связях (рисунок 7.5). Чтобы не строить длинные конструкции, мы будем называть сущность, отображающую составное объектное множество, составной сущностью.

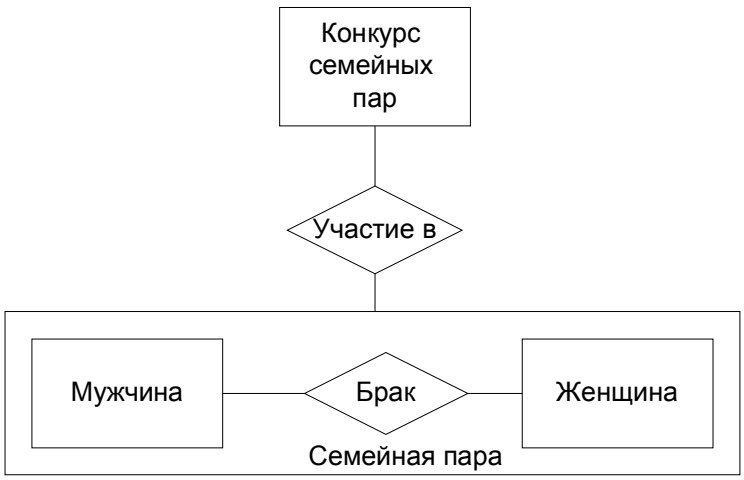


Рисунок 7.5- Участие составной сущности в связи /52/.

Такое отображение, несомненно, наглядно. В тоже время оно является причиной некоторых смысловых неточностей. Когда мы определили составное объектное множество, мы ничего не сказали о том, что в него могут входить не все объекты первого и второго множества. Рассуждения наши не изменятся, если такие объекты будут существовать. Способ показа составного множества, предлагаемый в книге /52/, не дает возможности отразить, что не все женщины и не все мужчины могут находиться в браке. Нам кажется уместнее считать отражение составного объектного множества типом свя-

зи. В этом случае мы считаем правомочным участием типа связи в другом типе связи в качестве составной сущности («связь со связью»). Для того, чтобы диаграмма была читаема, рекомендуется именовать составную сущность, как это показано на рисунке 7.6.

Часто обобщение/ конкретизацию относят к особым видам отношения. При этом нужно помнить, что в этом отношении экземпляр сущности присутствует сразу и в обобщающей сущности, и в конкретизирующей сущности.

Тип сущности, как отражение составного объектного множества, может иметь индивидуальные свойства. Одним их важных свойств типа связи является мощность связи (число кардинальности). Мощность связи обозначает максимальное количество элементов одного типа сущности, связанных с одним элементом другого типа сущности. Подобным образом можно сформулировать мощность связи в терминах объектного множества. В определении мощности заложено определение направления связи, хотя на диаграмме направление связи не указывается.

Рисунок 7.6 – Рекомендуемое отображение составной сущности и ее участие в связи

Наиболее распространенными являются бинарные связи с показателями кардинальности: «один к одному» (1:1), «один ко многим» (1:M) и «многие ко многим» (M:N) (Рисунок 7.7).

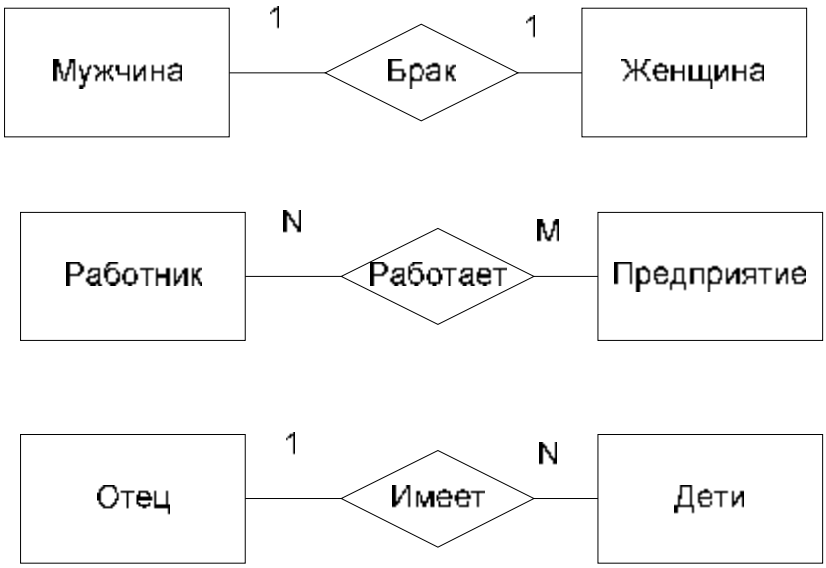


Рисунок 7.7 – Бинарные связи с различной мощностью

Алгоритм определение числа кардинальности показан на рисунке 7.8.

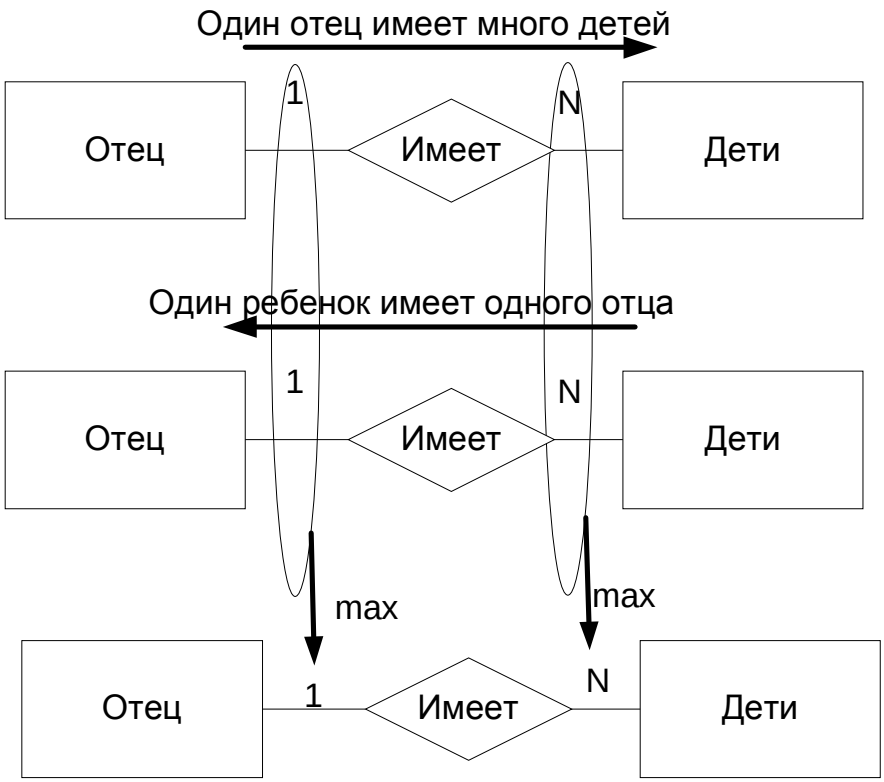


Рисунок 7.8 - Алгоритм определения числа кардинальности

Сущность является представлением объекта внешнего мира, а тип сущности – представлением объектного множества или множества объектов внешнего мира. В тоже время объект внешнего мира имеет множество характеристик. Под атрибутом мы будем понимать свойство типа сущности или

типа связи. Например, тип сущности человек может иметь атрибуты: имя, фамилия, отчество, возраст и т.д. Атрибуты изображаются в ER модели в виде овала (рисунок 7.9).

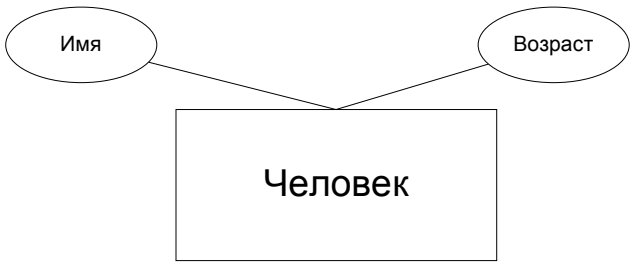


Рисунок 7.9 - Атрибуты

Такое определение рано или поздно приведет нас к вопросу примерно такого рода. Почему ИМЯ атрибут (пример выше), а не тип сущности (Рисунок 7.9)?

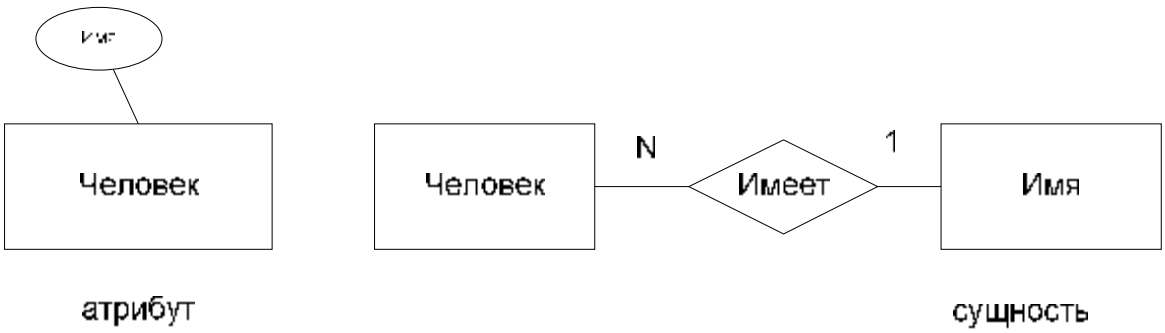


Рисунок 7.9 – Возможное использование «ИМЯ» как атрибута и как сущность.

Атрибут – функциональное[41] отношение объектного множества с другим объектным множеством. Отображение атрибута в виде овала просто сокращенная форма изображения этого функционального отношения. Целесообразно использовать атрибут в виде сущности (прямоугольник на диаграмме) в том случае, когда он используется в связи с другой сущностью. Если предположить, что человек имеет много имен, то в ER - модели практически ничего не меняется, только ИМЯ будет множественным атрибутом (см. ни-

[41] Функциональное отношение - это такое отношение, мощность которого в одну сторону равна единице.

же), а при изображении ИМЯ в качестве сущности возникнет мощность много ко многим. Рассмотрим различные виды атрибутов. Их отображение в нотации Чена приведено на рисунке 7.10.

Домен атрибута - набор значений, которые могут быть присвоены атрибуту. Домен значения в нотации ERD не нашел отображения.

Составной атрибут - атрибут, состоящий из нескольких компонентов, каждый из которых характеризуется независимым существованием. Составной атрибут представляет совокупность функциональных отношений объектного множества (сущность) с другими объектными множествами (составные части атрибута), имеющими общее имя.

Однозначный атрибут содержит одно значение для одной сущности.

Многозначный атрибут содержит несколько значений для одной сущности.

Производный атрибут представляет значение, производное от значения, связанного с ним атрибута или некоторого множества атрибутов, принадлежащих некоторому (не обязательно данному) типу сущности.

Рисунок 7.10 Изображение различных атрибутов на ERD нотации Чена.

В примерах рассмотрены атрибуты типов сущностей, но повторяя все те же рассуждения о том, что связь является отображением составного объ-

ектного множества, можно применить понятие атрибут и для типа связи (рисунок 7.11).

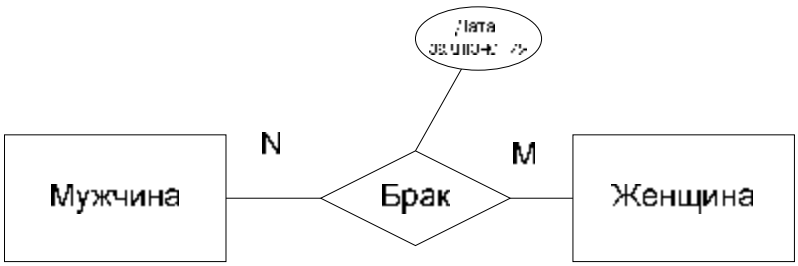


Рисунок 7. 11 – Атрибут связи

В некоторых источниках указывается, что наличие атрибута у связи показывает, что связь может быть скрытой сущностью и рекомендуют в этом случае пример на рисунке 7.11 представлять по иному (рисунок 7.12). На первый взгляд модели идентичны, но это не совсем так. Мы определили, что тип сущности - это отображение множества объектов реального мира. Но не существует объекта реального мира, представленного типом сущности «Брак». Этот тип сущности на самом деле отображает составное объектное множество. Мы уже отмечали принципиальное отличие между составным объектным множеством и объектным множеством. Оно заключается в том, что составное объектное множество не существует без составляющих их объектов. Нет типа сущности «Брак» без типов сущности «Мужчина» и «Женщина». Исходя из выше сказанного, мы не рекомендуем без необходимости заменять конструкцию 7.11 на конструкцию 7.12. Все рассуждения можно повторить и для связей составного объектного множества с объектным множеством (атрибут – это функциональная связь).

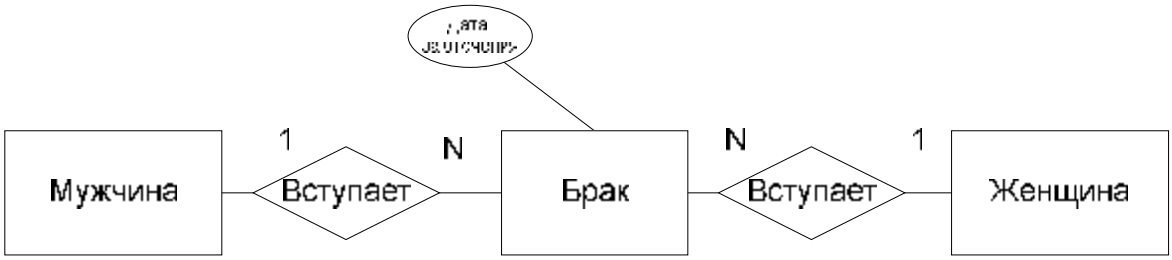


Рисунок 7.12 Представление связи в виде скрытой сущности.

Детали ER - модели

В предыдущем разделе мы рассмотрели основные концепции ERD - модели. В данном разделе рассмотрим оставшиеся аспекты концептуального моделирования. Фактически, мы рассмотри еще раз те же понятия,только более глубоко.

Мы будем разделять сущности на два вида: слабые и сильные. Слабый тип сущности - это отображение множества объектов реального мира, присутствие которых в данной модели зависит от объектов другой сущности. Сильный тип сущности отображает множества объектов реального мира, присутствие которых в данной модели не зависит от объектов другой сущности.

Здесь два ключевых слова: модель и зависит. Во-первых, сущность слабая или сильная определяется моделью, следовательно, задачами, решаемыми при моделировании. Во-вторых, именно в этой модели присутствие экземпляра слабой сущности диктуется присутствием экземпляра другой сущности. В качестве примера можно привести сущность «работник» и «иждивенец», при моделировании данных бизнес - процесса «начисление заработной платы». Присутствие экземпляра сущности «иждивенец» в данной модели может интересовать только в том случае, если он связан с конкретным экземпляром сущности «работник» (рисунке 7.13).

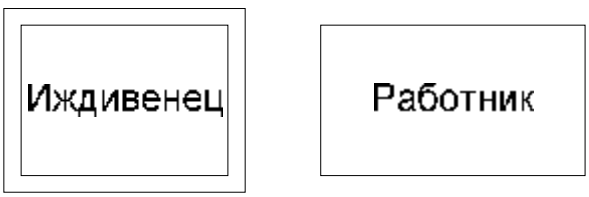


Рисунок 7.13- Сильные и слабые сущности

Необходимо отметить, что определение сильной и слабой сущности существует только в контексте отношения экземпляров данных типов сущности. Тип связи между слабой и сильной сущностью слабый (рисунок 7.14).

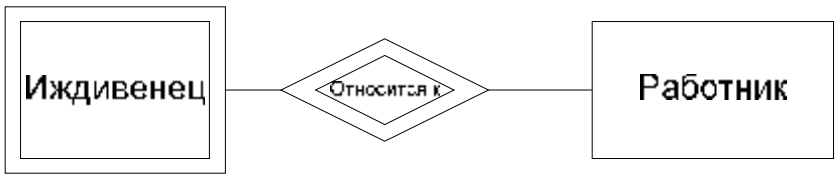


Рисунок 7.14 – Слабый тип связи

Учитывая сказанное, конструкцию рисунка 7.12 необходимо преобразовать, так как сущность «Брак» слабый тип сущности (не может существовать без сущности «мужчина» и «женщина») (рисунок 7.13).

Рисунок 7.13 – Пример слабой сущности

Слабые сущности иногда называют дочерними (child), зависимыми (dependent) или подчиненными (subordinate), а сильные — родительскими (parent) сущностями-владельцами (owner) или доминантными (dominant).

Может возникнуть путаница в терминах, так как часто дочерние и родительские сущности обозначают сущности, участвующие в связи один ко многим (так, например, в мастерах работающих с базами данных Delphi).

Существует несколько важных аспектов характеризующих «связь»: степень связи, полнота участия в связи и рекурсивные связи.

Степень связи - количество сущностей, которые охвачены данной связью (рисунок 7.14). Такие типы связи называют n-арными. Мощность n-арной связи определять бессмысленно, так как мощность – понятие, отнесенное к бинарной связи, поэтому в n-арных связях обычно проставляют мощность N.

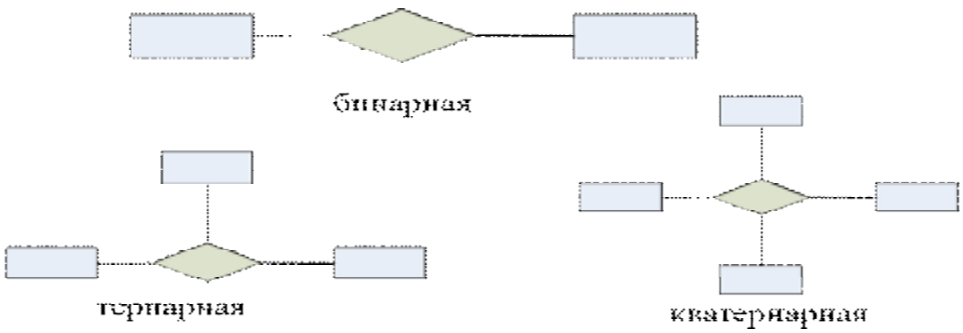


Рисунок 7.14 – Степень связи.

N-арные связи - достаточно редкое явление. Нужно внимательно относиться к их применению, настоящая n-арная связь не может быть преобразована в набор бинарных. Пример, приведенный на рисунке 7.15, показывает, что конкретный лектор читает конкретный предмет для данной специальности в данном семестре. В другом семестре другой преподаватель может читать точно также этот предмет, но для другой специальности.

Рисунок 7.15 Кватернарная связь

Если мы заменим связь сущностью, образуется четыре бинарные связи (рисунок 7.16), но при этом потеряется семантика отношения. N-арная связь требует, чтобы для каждой комбинации классов существовала только одна связь, а схема рисунка 7.16 допускает любое количество комбинаций для каждой связи.

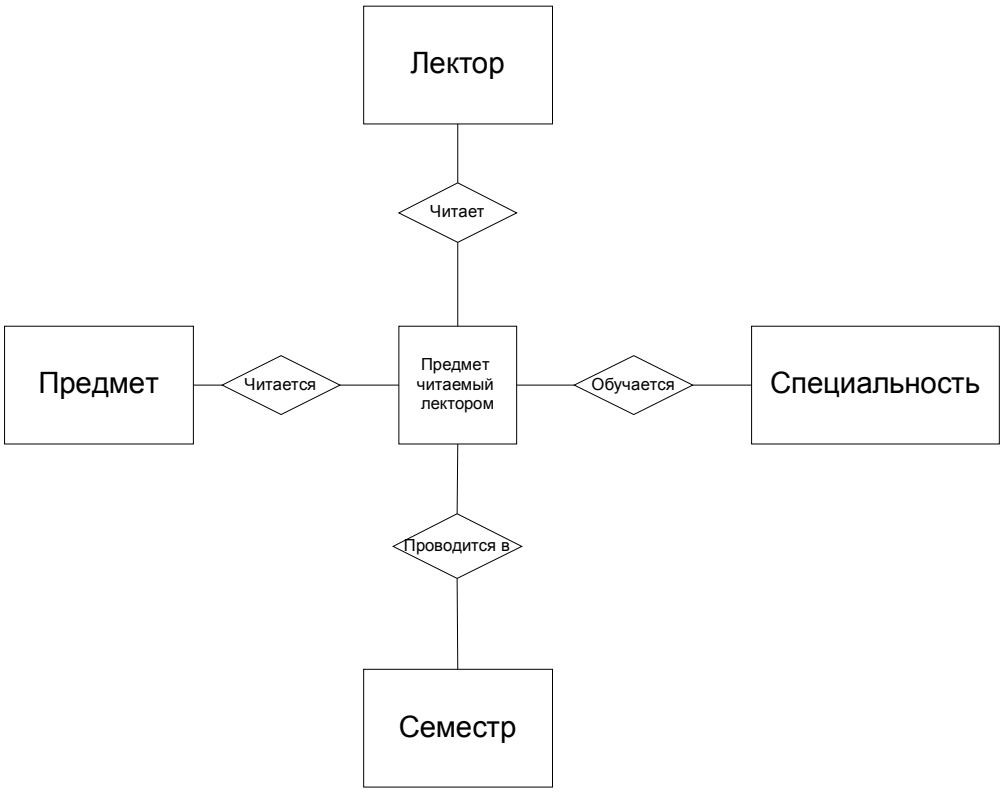


Рисунок 7.16 – Превращение n-арной связи в сущность

ER - модель допускает присутствие рекурсивной связи. Рекурсивная связь- это ассоциация объектов множества с другими объектами этого множества. Для лучшего понимания рекурсивной связи используются ролевые имена (рисунок 7.17).

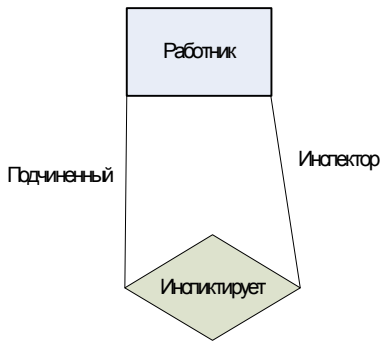


Рисунок 7.17 – Рекурсивная связь.

Ролевые имена могут быть использованы тогда, когда присутствует несколько связей между двумя сущностями.

В литературе /54/ используется понятие степень участия. С одной стороны, степень участия накладывает ограничение на связь. Та сущность, которая имеет полное участие в связи, представляет множество экземпляров сущ-

ности, каждое из которых находится в связи с каким-либо экземпляром другой сущности. С другой стороны, не может быть определен экземпляр сущности, для которого нет пары из числа экземпляра другой сущности. Частичное участие предполагает отрицание этого ограничения.

В стандарте IDEF1X полная степень участия называется обязательным участием, а частичное - необязательным. Мы будем использовать обе терминологии и 2 способа отображения степени участи (рисунок 7.18). В первом способе связь с полной степенью участия показана утолщенной линией (в /54/ такая связь показывается двойной линией). Во втором способе указывается минимальная и максимальная мощность связи (min,max).

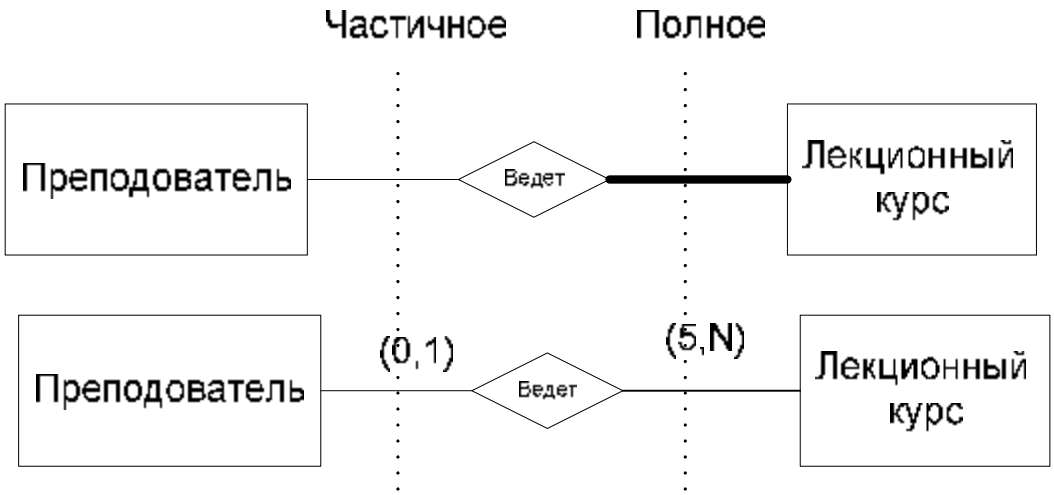


Рисунок 7.18 – Два способа отображения степени участия.

В предыдущем разделе мы разобрали понятия - генерализация/специализация. Развитие объектно-ориентированного программирования (ООП) оказало влияние на моделирования данных. Особый вид отношений генерализация/ специализация (для краткости будем называть генерализация) давал возможность воплотить парадигму ООП в ER модели. На первый взгляд генерализация дала возможность реализовать принцип наследования. Объект принадлежит генерализующему типу сущности (по аналогии с ООП его называют суперклассом), он обладает всеми атрибутами суперкласса, с другой стороны, он принадлежит специализированной сущности (этот тип называют подклассом) следовательно, он обладает всеми атрибутами

подкласса. В отличие от ООП здесь не может быть никакого перекрытия атрибутов (понятия метод в ERD не существует). В ERD один экземпляр присутствует в двух множества, в ООП объект может принадлежать либо суперклассу, либо подклассу.

В ERD генерализация дает возможность в суперклассе собрать все атрибуты, общие для нескольких подклассов, а в подклассах оставить только специальные атрибуты. Перекрытие атрибутов не допускается. Поскольку экземпляр принадлежит одновременно суперклассу и подклассу, мощность этой связи равна один к одному. Могут быть следующие варианты: один экземпляр суперкласса присутствует только в одном подклассе – специализации не пересекаются (в узле ставится буква d- disjoint), один экземпляр суперкласса присутствует только в разных подклассах – специализации пересекаются (в узле ставится буква o- non disjoint) (рисунок 7.19).

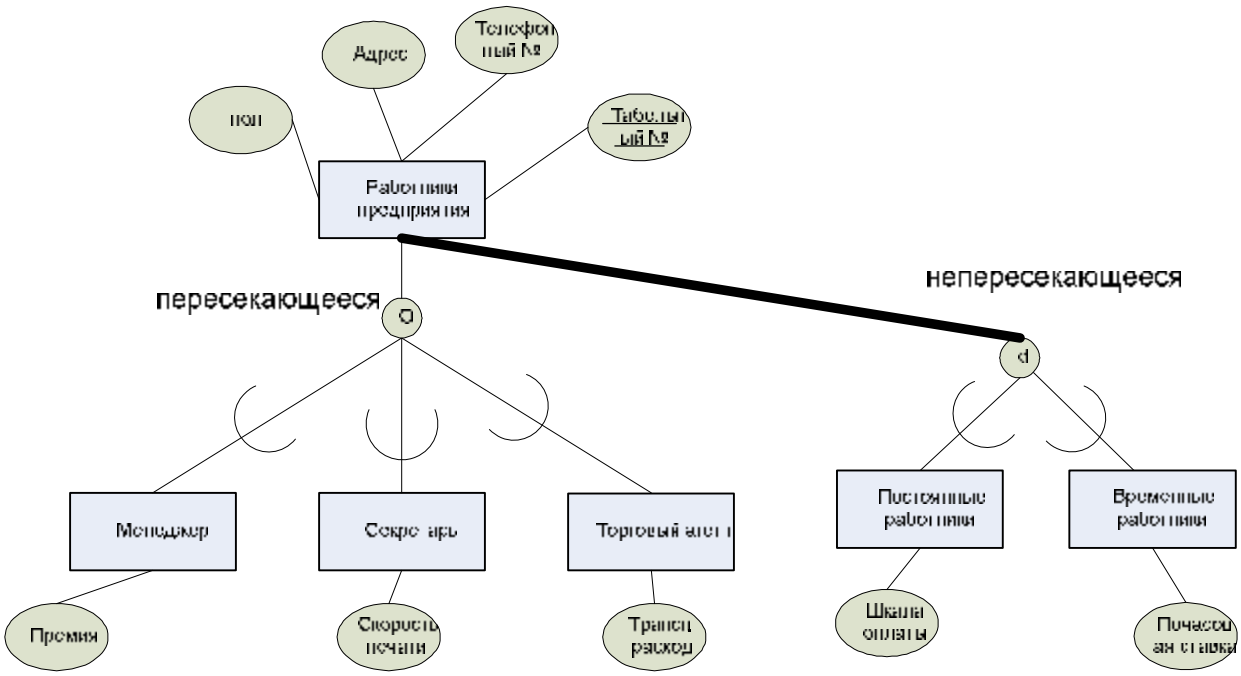


Рисунок 7.19 – Пример использование генерализации/специализации.

Легко представить, что экземпляр некоторого типа сущности имеет несколько суперклассов (подкласс в этом случае называется совместно используемый подкласс). Это означает, что он принадлежит сразу всем сущностям (рисунок 7.20).

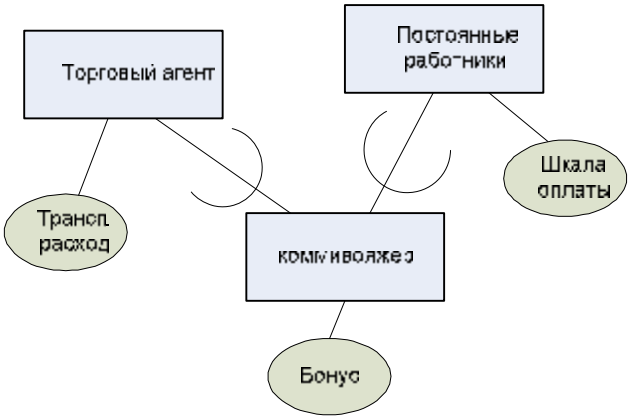
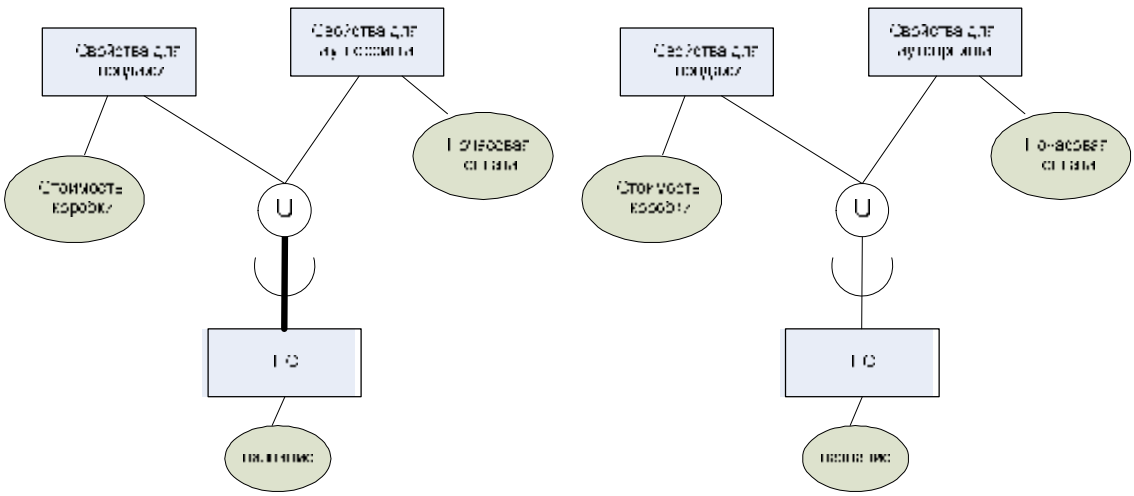


Рисунок 7.20- Совместно используемый подкласс.

Можно предположить, что возникнет ситуация, где экземпляр сущности «коммивояжер» может принадлежать либо к сущности «Постоянный работник», либо к сущности «Торговый агент». Такой подкласс называют категорией. Пример использования категории приведен на рисунке 7.21.



Степень участия полная Степень участия частичная

Рисунок 7.21 - Пример использование категории.

Полная степень участия может быть реализована с помощью генерализации/специализации (рисунок 7.22).

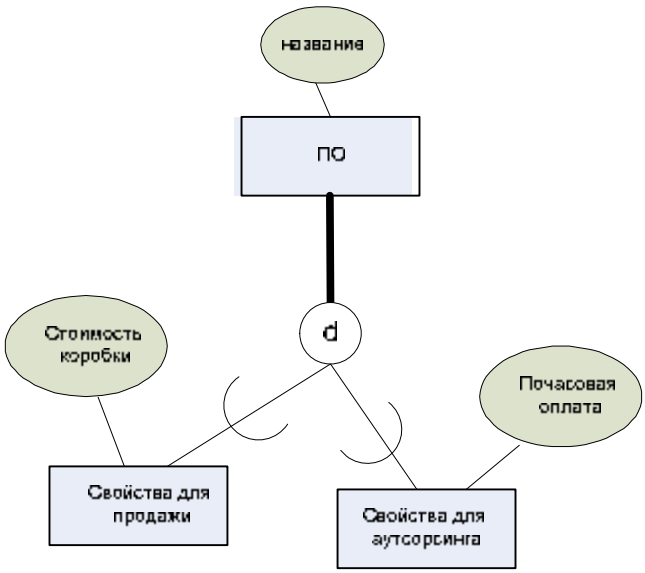


Рисунок 7.22 – Преобразование категоризации в генерализацию/специализацию.

Последнее, о чем мы упомянем в ERD - модели, это возможность документировать бизнес правила. Нотация ERD не предоставляет возможности графически изображать бизнес - правила. Тем не менее, аналитик может включить бизнес - правила в описание модели. Рекомендуется документировать транзакции. Это даст возможность осуществить тестирование модели на непротиворечивость.

Стандарт IDEF1X /67/

IDEF1X - семантическая техника моделирования данных. Синтаксис IDEF1X поддерживает семантические модели, что дает возможность использовать ее в концептуальном моделировании. Законченная модель IDEF1X совместима, расширяема, доступна для преобразования. IDEF1X имеет простую, прозрачную структуру, согласуемую с семантическими концепциями. Синтаксис и семантика IDEF1X относительно легкие для пользователей, но в тоже время достаточно мощные и логичные. Он широко известен и поэтому может использоваться различными участниками команды разработчика.

Существует большое количество коммерческих программных продуктов, поддерживающих IDEF1X. К достоинствам языка необходимо отнести

то синтаксис и семантику, которые могут использоваться не только на концептуальном уровне, но на логическом и физическом уровнях.

В стандарте изложена методика концептуального моделирования, но в пособии мы рассмотрим вопросы его использования на логическом уровне. Концептуальное моделирование в нотации Чена обладает большей гибкостью и выразительностью. Тем не менее, методика разработки концептуальной модели легко может быть адаптирована для нотации Чена.

Прежде, чем рассматривать нотацию IDEF1X, коротко опишем структуру стандарта IDEF1X, как документа.

После короткого введения и описания границ модели IDEF1 X, рассматривается синтаксис и семантика нотации IDEF1X, в которую входят:

- компоненты графического языка;
- описание жизненного цикла разработки IDEF1X, состоящего из трех уровней: спецификация ER представления без определения ключей, KB (key-based) представление, в котором определены все ключи, но не все атрибуты; FA (fully-attributed) представление, в котором модель полностью специфицирована;
- описание представления модели, где перечисляются возможные разделы представления;
- описание глоссария;
- модель отчета.

Стандарт содержит два приложения. Приложение А - «Концепции и процедуры работ (разработка проекта ICAM)». В данном приложении подробно расписаны этапы работы над моделью IDEF1X. Методология включает следующие фазы:

0 – инициализация проекта;

1- определение сущностей;

2- определение связей;

3 – определение ключевых полей;

4- определение атрибутов;

5 - документирование и проверка достоверности модели.

Приложение В - «Формализация». В нем описано теоретическое обоснование метамодель IDEF1X.

Мы не ставим перед собой задачу изучить стандарт. Основной нашей целью является знакомство с нотацией IDEF1X. Данный язык мы будем рассматривать в контексте конкретных инструментов, которые либо внесли в него что-то новое, либо не реализовали все его функциональные возможности. В стандарте не идет речь о конкретном типе баз данных, но возможности стандарта особенно проявляются при проектировании схемы данных для реляционных СУБД.

Работа в инструментарии MS Visio и ERwin дает возможность использовать синтаксические элементы стандарта для физического проектирования, правда в данных инструментах методология расширена. Многие инструменты дают возможность генерировать SQL скрипты, организующие базу данных в соответствующих СУБД. Как мы увидим позже , существует возможность получить схему данных объектно-ориентированными методами.

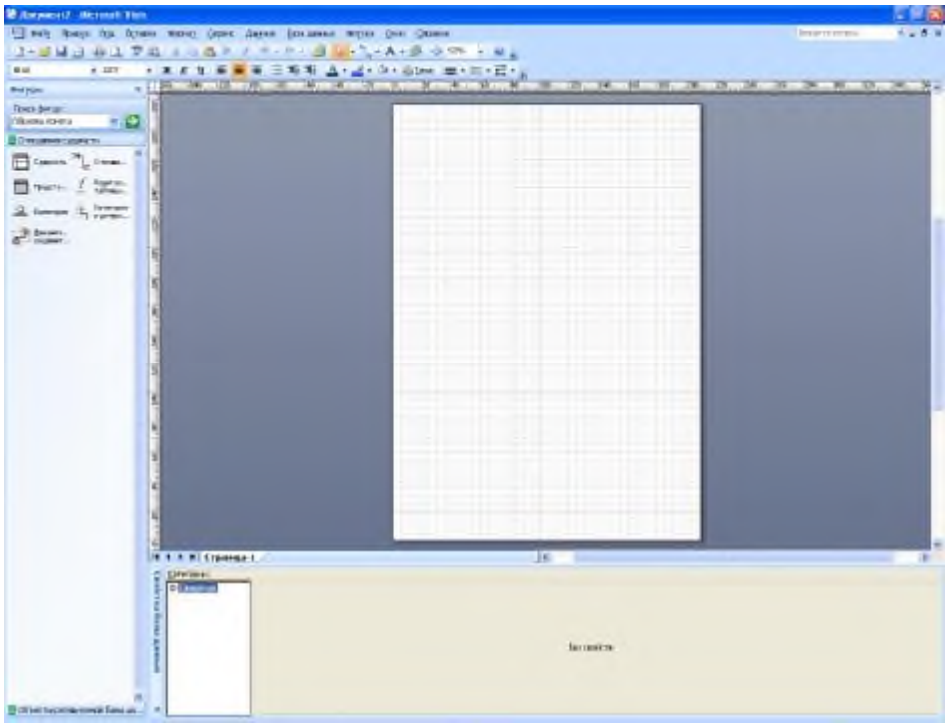
Логическое проектирование предполагает выбор типа СУБД. Мы будем рассматривать логическое проектирование только для реляционных баз данных.

В данной главе мы дадим короткий обзор всех синтаксических элементов. При логическом проектировании с использованием IDEF1X возникает задача преобразовать концептуальную модель в модель, совместимую с реляционной концепцией. Существует большое количество способов преобразования элементов модели ERD в реляционную схему. Например, IDEF1X поддерживает категоризацию, что совершенно не согласуется с реляционной моделью. В реляционной модели есть только отношения (специальный вид таблиц); определены ключи, это - первичный, потенциальный и внешний;

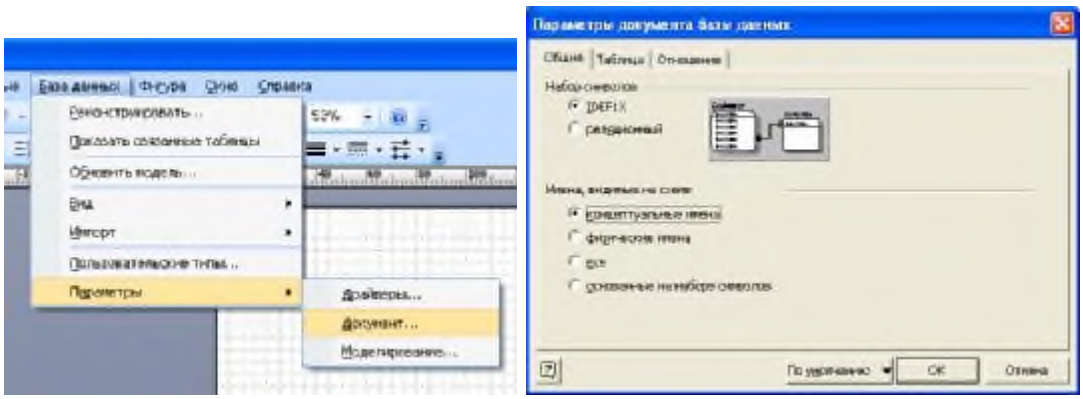
определены операции над этими таблицами. MS Visio и ERWin автоматически преобразуют категорию IDEF1X в реляционные отношения.

Для иллюстрации модели мы будем использовать приложение MS VISIO (MS VISIO 2007), которое Вы должны настроить.

Открыть Visio, выбрать категорию шаблона – программное обеспечение и базы данных, шаблон - схемы модели базы данных, создать. Затем выбрать Базы данных, Параметры, Документы, Общие, IDEF1X (рисунок 7.23)



а – Проект базы данных готов к работе.



б - Настройка модели IDEF1X.

Рисунок 7.23 – Настройка Visio для работы с моделью IDEF1X

Синтаксис и семантика нотации IDEF1X.

IDEF1X включат следующие компоненты в свою нотацию:

- Сущности:
  - Сильные сущности (Identifier-Independent Entities);
  - Слабые сущности (Identifier-Dependent Entities).
- Связи:
  - Идентифицирующие (Identifying Connection) связи;
  - Неидентифицирующие (Non-Identifying Connection) связи;
  - Категоризация;
  - Неспецифицированные связи.
- Атрибуты/ Ключи:
  - Атрибуты;
  - Первичный ключ;
  - Альтернативный ключ;
  - Внешний ключ.
- Отчет (описание).

Как мы уже говорили, стандарт согласован с концепциями семантической модели, поэтому мы будем пояснять термины только в том случае, если они расходятся с моделью Чена.

Сущность в IDEF1X изображается в виде прямоугольника (рисунок 7.24). На рисунке 7.25 раздел свойства базы данных Visio, в котором вы заполняете свойства каждого элемента модели. В данном методическом пособии не ставится задача обучению интерфейса какой-либо программы.

Рисунок 7.24 - Сильная сущность .

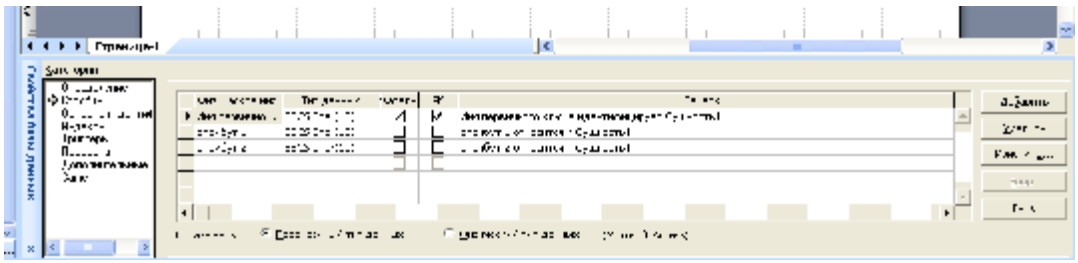


Рисунок 7.25 - Поле свойства базы данных

В данный момент мы работаем на уровне логического проектирования (в Visio это называется концептуальным уровнем), поэтому мы не выбираем конкретную базу данных, следовательно, типы атрибутов не указываются. Visio не предоставляет инструмент для работы с доменами (в ERWin data modeler такой инструмент присутствует). Уровень логического проектирование, как и уровень концептуального проектирования, предполагает работу с доменами.

В концептуальном проектировании были определены бизнес - правила (пример бизнес – правила: «преподаватель не может иметь почасовой нагрузки в одном университете более 300 ч/год»). На логическом уровне выбран тип базы данных (реляционный тип), поэтому инструмент предоставляет возможность формировать триггеры на SQL.Эту возможность предоставляет не стандарт IDEF1X, а конкретный CASE инструмент.

Буква «О» показывает, что для этого атрибута допускается значение «NULL». На рисунке 7.24 показана сильная сущность. Для того, что бы определить слабую сущность, необходимо создать связь и указать, что эта связь идентифицирующая (рисунок 7.25). Инструмент не только создает связь и определяет сильную и слабую сущность (рисунок 7.26), но также выполняет реализацию сильной и слабой сущности для реляционной базы данных. В первичном ключе слабой сущности появился внешний ключ сильной сущности.

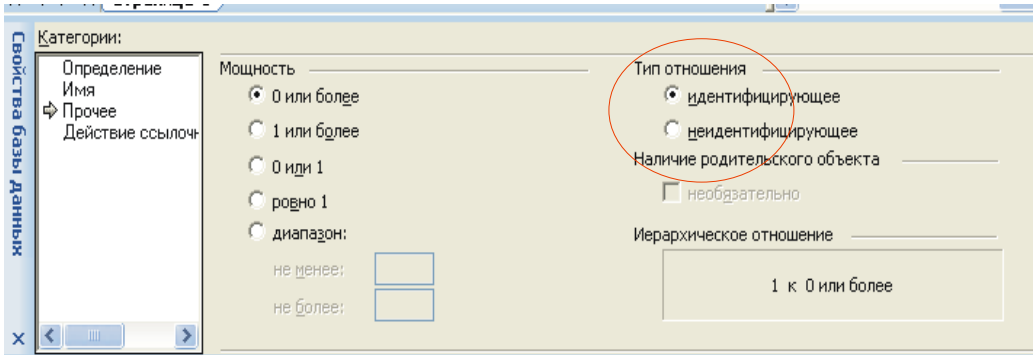


Рисунок 7.25. Определение идентифицирующей связи

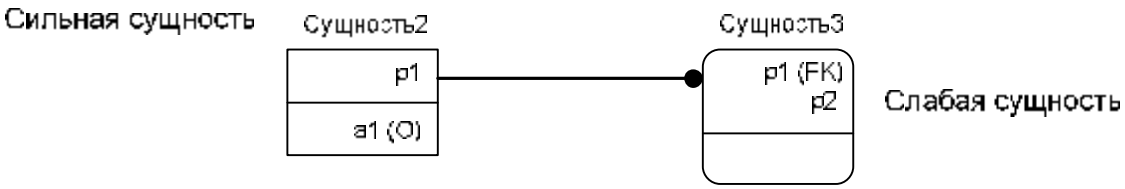


Рисунок 7.26 Сильная и слабая сущности соединены идентифицирующей связью.

На данном рисунке показана идентифицирующая связь - сплошная линия. Неидентифицирующая связь – пунктирная линия (рисунок 7.27).

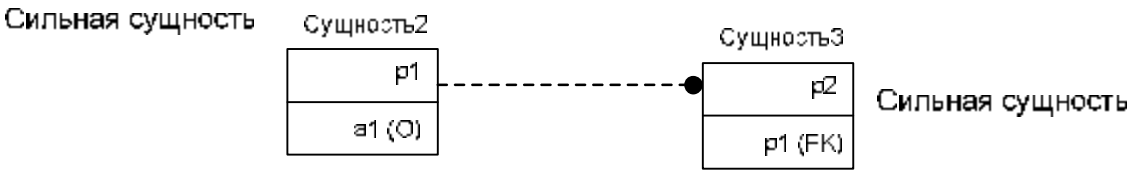


Рисунок 7.27 Неидентифицирующая связь.

Жирная точка показывает сторону связи «много». Сущность с примыкающей жирной точкой - дочерняя, другая – родительская. Дочерняя сущность должна иметь внешний ключ, ссылающийся на первичный ключ родительской сущности. Можно изменить мощность связи (0 или более, 1или более, 0 или равно 1, задать диапазон) , но при этом ничего не измениться в графическом представлении. Фактически, инструмент снова берет на себя функции преобразования модели в реляционную нотацию. Разработчик, помещая жирную точку на сущность, автоматически называет ее дочерней. Для

связи один к одному дочерней сущностью называется та сущность, участие которой в связи неполное.

Полнота участия или обязательность участия задается в свойствах дочерней сущности для атрибута внешнего ключа - галочка в свойстве «обязательное» (рисунок 7.28). В этом случае нельзя задать запись в дочерней сущности, не определив внешний ключ, как ссылку на существующий первичный ключ родительской сущности. Если задано не полное участие, то возле родительской сущности появляется ромб, а в дочерней таблице возле атрибута внешнего ключа появляется «О» (Рисунок 7.29).

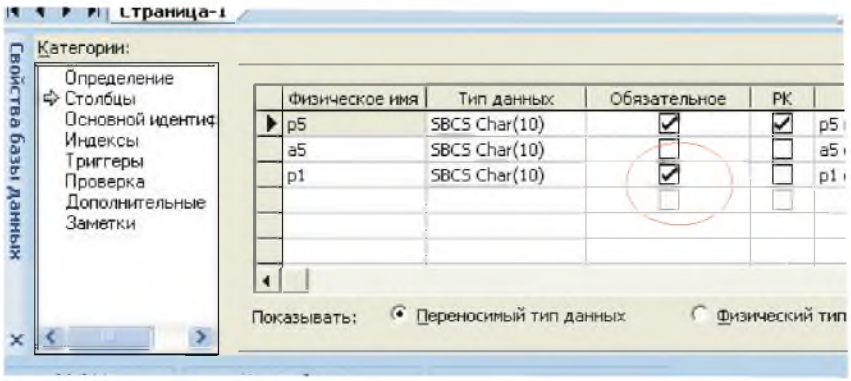


Рисунок 7.28 - Определение полного участия в связи

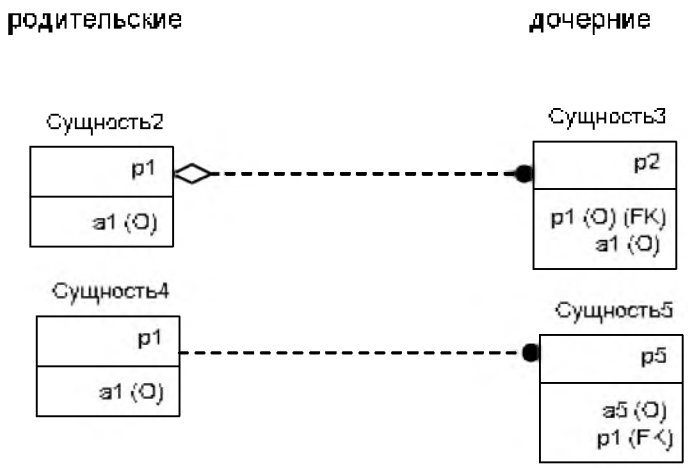


Рисунок 7.29 – Неполное (верхняя связь) и полное участие.

Указание в скобках FK (foreign key), говорит о том, что данный атрибут внешний ключ. В модели IDEF1X определено понятие категоризация.

Не вдаваясь в подробности анализа терминологической путаницы нотации Чена и IDEF1X, будем считать, что генерализация/специализация описанная в нотации Чена, почти соответствует понятию категоризация нотации IDEF1X.

Использование отношений категоризации подчиняется следующим правилам:

- каждая категория может иметь не более одной родовой сущности;
- категория может быть родовой сущностью для других категорий;
- никакая категория прямо или косвенно не может быть родовой сущностью для самой себя;
- родовая сущность может иметь несколько кластеров категорий. Для каждого кластера должен использоваться собственный дискриминатор;
- дискриминатор полного кластера категорий не может быть необязательным.

Если некоторая сущность разбивается на категории, то ее называют родовой сущностью, а сущность, соответствующую категории, называют сущностью-категорией или просто категорией. Очевидно, что каждый экземпляр категории одновременно является экземпляром родовой сущности.

Экземпляры сущности каждой категории образуют множество, объединение множеств всех категорий, принадлежит множеству экземпляров родовой сущности.

Допустим, мы выделили родовую сущность - сотрудник и две категории - лаборант (дополнительный атрибут - № лаборатории) и преподаватель (дополнительный атрибут - читаемый предмет) (рисунок 7.30)

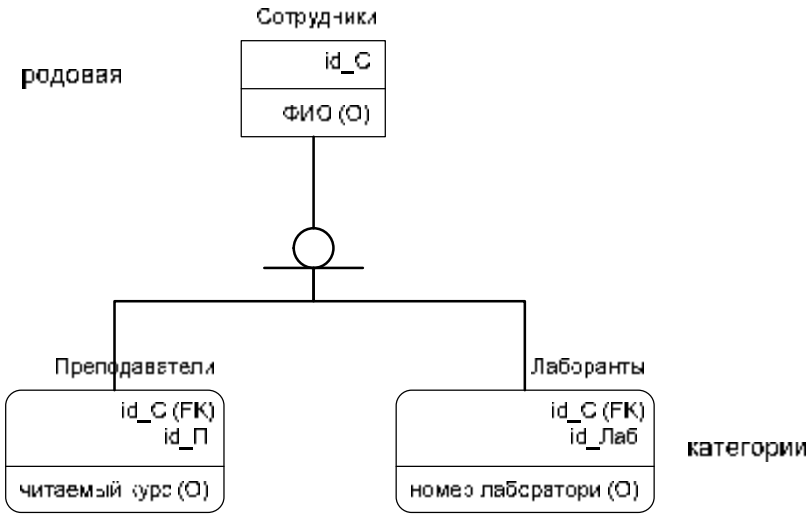


Рисунок 7.30 - Пример неполной категории

На данной диаграмме определено, что сотрудник не обязательно должен быть лаборантом или преподавателем (лучше сказать, что экземпляр сущности сотрудник не обязательно принадлежит экземплярам сущности преподаватель или лаборант). Обязательная принадлежность (рисунок 7.31) сущности родовой к сущности категории задается в свойствах категории.

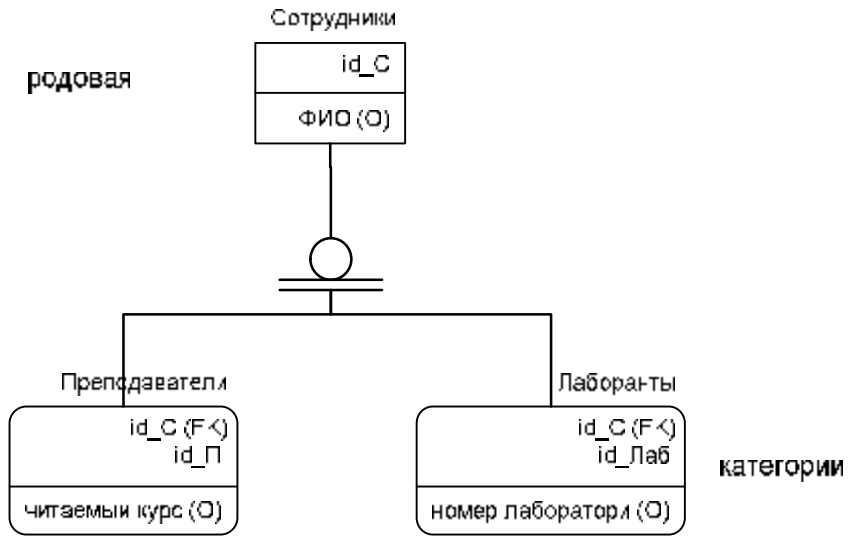


Рисунок 7.31 - Пример полной категории.

Кластер категории: набор одного или более взаимно исключающих отношений классификации для той же самой родовой сущности. Дискриминатор категории: признак в родовой сущности группы категории. Значения дискриминатора указывают, какой сущности категории принадлежит экземп-

ляр родовой сущности. Категории должны включать экземпляры родовой сущности с одним значением дискриминатора (рисунок 7.31).

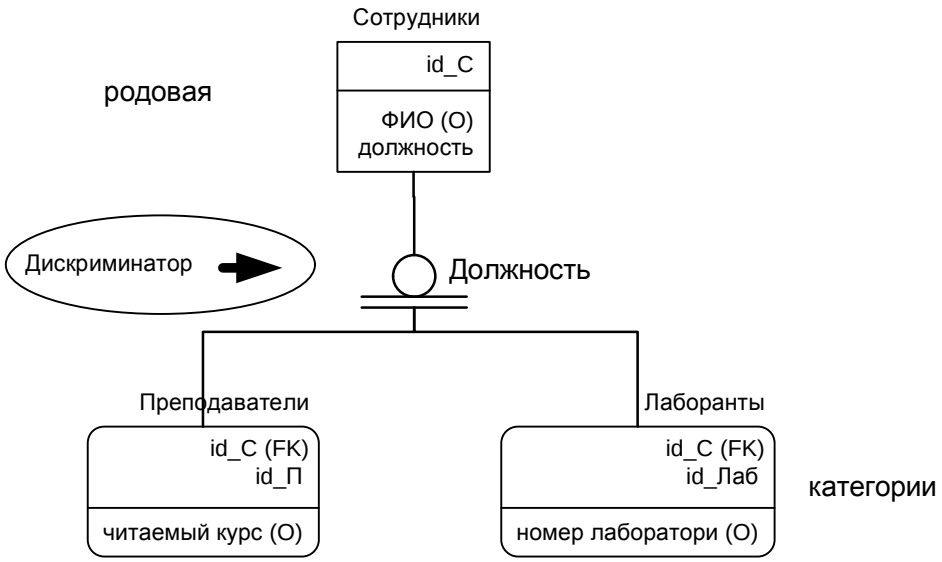


Рисунок 7.31 – Использование дискриминатора

В принципе сущность может иметь несколько потенциальных ключей. Каждый сотрудник, например, может идентифицироваться тройкой < имя, отчество, фамилия >, а так же значением атрибута ИНН. Один из потенциальных ключей всегда становится первичным ключом (каждая сущность должна иметь первичный ключ), а остальные потенциальные ключи называются альтернативными ключами. Каждому альтернативному ключу в рамках сущности присваивается номер, и напротив каждого атрибута, который принадлежит данному ключу, ставится метка (AK<номер ключа>).

Использование альтернативных ключей подчиняется следующим правилам:

- каждая сущность должна иметь один первичный ключ и любое количество альтернативных ключей;
- каждый ключ может состоять из одного или нескольких атрибутов;
- каждый атрибут может быть частью более чем одного ключа.

Контрольные вопросы

1. Что такое лексическое множество? Чем оно отличается от объектного множества?
2. Как понятие сущности соотносится с понятием объектного множества?
3. Что отличает связь от генерализации/специализации? Сформулируйте ответ в терминах объектного множества.
4. Что отличает концепции отношения генерализация/специализация в ER модели от ООП?
5. Чем отличается экземпляр суперкласса от экземпляра класса потомка?
6. Что отличает категорию в модели Чена и IDEF1X?
7. Модель IDEF1X является концептуальной моделью или логической? Обоснуйте свой ответ?
8. Что отличает концептуальное моделирование от семантического?
9. Можно ли применить понятие «наследование» к связи генерализация/специализация? Приведите пример для обоснования своего ответа.
10. Чем отличается альтернативный ключ от потенциального ключа?
11. Может ли использоваться атрибут с множественным значением в модели IDEF1X?
12. Что отличает модель IDEF1X от методологии IDEF1X?

8. Унифицированный процесс проектирования программного обеспечения

В современном мире информационных систем разработчику программного обеспечения тяжело конкурировать с гигантами ИТ - индустрии. При этом, сектор рынка, в котором есть возможность занять свою нишу - узкоспециализированные продукты для конкретного предприятия. Но так как делопроизводство развивается быстрыми темпами, необходима быстрая разработка программного обеспечения. Быструю разработку предлагают многие продукты (Eclipse, Delphi, Visual Studio), но нужно, чтобы приложения при этом максимально удовлетворяли потребностям заказчиков. Для обеспечения соответствия программного решения требованиям заказчика необходимо тщательно проектировать решение. Ускорению выработки такого решения способствует процесс разработки. Процесс разработки позволяет координировать действия каждого члена команды, обеспечивать быстрое получение результата путем распределения задач между участниками и постоянно улучшать качество разрабатываемого решения благодаря итеративному подходу.

Существует довольно много процессов разработки программного обеспечения -UP, XP, DSDM, SCRUM. Наиболее полно описывает получение конечного продукта Унифицированный процесс разработки, который можно применить к проекту любого типа. Среди таких методов разработки как Enterprise Unified Process, Rational Unified Process, Agile Unified Process, Essential Unified Process и других следует выделить унифицированный процесс компании Rational (RUP), представляющий контекст разработки для различных групп разработчиков и проектов, различных по времени разработки.

На основе библиотеки методов RUP разработан адаптированный учебный унифицированный процесс проектирования информационных систем,

рассчитанный на группы численностью 3-5 человек и позволяющий получать проектное решение в течении 1-2 месяцев.

Рисунок 8.1 - Унифицированный процесс

Рисунок 8.1 иллюстрирует структуру унифицированного процесса в двух направлениях:

- горизонтальная ось показывает время и жизненный цикл проекта;
- вертикальная ось представляет дисциплины – логически сгруппированные деятельности.

Граф показывает изменение усилий на протяжении времени. Например, на начальных итерациях следует уделять больше времени требованиям, на последующих итерациях – анализу и проектированию.

Дисциплина «Бизнес-моделирование»

Цель

Цели бизнес - моделирования:

- выявление проблем в организации и определение потенциальных улучшений;
- оценка воздействия организационного изменения;
- единое понимание организации у клиентов, разработчиков, пользователей и других сторон;
- получение требований к программному обеспечению, необходимых для поддержания организации;

- определить, как будущая система вписывается в организацию.

Организационной диаграммы недостаточно для понимания бизнеса. Необходимо иметь динамическое представление бизнес-процессов. Модели бизнес-процессов обеспечивают статическое представление структуры организации и динамическое представление процессов в пределах организации.

Бизнес-процесс должен изменяться в соответствии с факторами, которые управляют им и сохраняют его целостность. Такими факторами могут быть цели сокращения стоимости, улучшение качества или сокращения времени выпуска нового продукта. Необходимо моделировать бизнес так, чтобы локализовать проблемы или выявить возможные улучшения. Характеристика работоспособной и развивающейся организации – возможность изменения бизнес составляющих.

Много различных людей (заинтересованных сторон) должны понять бизнес. Поскольку у этих людей различные квалификации и интересы, им необходимы различные представления бизнеса. Необходимо моделировать бизнес просто и понятно, используя общую нотацию. Бизнес-модель должна иметь возможность быть описанной в различных представлениях и уровнях абстракции. Если не все могут понять бизнес-модель, то бизнес-моделирование стоит пропустить.

Бизнес имеет наиболее важное значение для заказчика при получении прибыли. Принятие решения в области бизнеса – наиболее важное решение в области качества. Информационные системы должны проектироваться таким образом, чтобы предоставленная информация была своевременна, точна, достаточна и релевантна. Информационная система поддерживает бизнес решения только тогда, когда существует точное понимание контекста, в котором принимаются эти решения.

Артефакты

Для достижения поставленных целей дисциплина «Бизнес-моделирование» описывает, как оценить текущую организацию и разработать видение новой организации. Используя это видение как основу нужно определить роли, процессы и функции организации в моделях бизнес-прецедентов и бизнес-анализа.

Дополнительно к этим моделям разрабатываются следующие артефакты:

- документ «Бизнес-Видение»;
- документ «Бизнес-архитектура»;
- дополнительные бизнес-спецификации;
- бизнес-правила (как документ и/или элементы модели бизнес-анализа);
- бизнес-глоссарий.

Процесс и нотация

Существует множество различных методик и нотаций для моделирования бизнеса, использующихся с различным успехом. Однако, процессов бизнес-моделирования меньше. Унифицированный процесс предоставляет метод бизнес-моделирования. Унифицированный язык моделирования (UML) может эффективно применяться как в моделировании программного обеспечения, так и бизнеса. Одно из самых важных преимуществ использования единой нотации при проектировании бизнеса и программного обеспечения – то, что бизнес-аналитики и разработчики используют один язык. Это позволяет напрямую эффективно транслировать модели бизнес-модели в модели программных систем, поддерживающих этот бизнес.

Моделирование, понимание и улучшение бизнеса необходимо для создания программной системы. Проведение начального исследования включает определение целей и границ. Это исследование также включает создание высокоуровневой схемы и заполнение ее по частям. Нельзя сосредотачиваться

на одной части, завершать ее и больше к ней не возвращаться. Зачастую необходимо повторно поверять уже смоделированные части, изменять их, основываясь на новом понимании и осмыслении. Не нужно ждать завершения всего бизнес-моделирования для проверки и усовершенствования результатов.

Целесообразнее всего бизнес-моделирование проводить итеративно, начиная с общего шаблона с постепенным его заполнением. На каждой итерации необходимо проверять эскиз и производить необходимые корректировки. Таким образом, происходит наполнение эскиза и проверка результатов работы. Эти шаги должны быть закончены до начала следующей итерации.

Связь с другими дисциплинами

Дисциплина «Бизнес-моделирование» связана со следующими дисциплинами:

- дисциплина «Управление требованиями» использует бизнес - модели как входную информацию для понимания требований к системе;
- дисциплина «Анализ и проектирование» использует бизнес - модели как входную информацию для определения внутриорганизационных программных систем.

Цели бизнес - моделирования

Бизнес-моделирование может применяться для различных целей, в зависимости от контекста и потребностей. Ниже приведены шесть таких сценариев.

Организационная структура

Построение простой карты организации и ее процессов для лучшего понимания требований к разрабатываемому программному средству. В этом случае бизнес-моделирование – часть проекта разработки программного

обеспечения, выполняющееся в начале фазы «Начало». Построение структурных диаграмм не подразумевает организационных изменений, однако, в действительности, разработка и внедрение нового приложения включает некоторое улучшение бизнеса.

Модель предметной области

Разработка систем для управления и представления информации, например, системы управления заказами или банковской системы, требует построения модели данных на бизнес-уровне, не рассматривая бизнес-процессы (потоки). Обычно моделирование предметной области, как часть проекта по созданию программного обеспечения, выполняется в течение фаз «Начало» и «Исследование».

Один бизнес для многих систем

При построении крупной системы или пакета программ возможно проведение одного бизнес-моделирования для нескольких проектов. Бизнес-модели помогают выявлять требования и создавать архитектуру для пакета программ. В таком случае, бизнес-моделирование позиционируется как отдельный проект.

Универсальная бизнес-модель

При разработке приложения, которое будет использоваться в нескольких организациях, например, система поддержки пользователей или биллинговая система, полезно определить общий для организаций бизнес для выявления всех требований для более сложной системы (улучшение бизнеса). Если не требуется вывод организаций на один уровень, бизнес-моделирование может помочь понять различия в использовании программных средств в организациях и расположить в порядке значимости функциональные возможности.

Новый бизнес

Бизнес-моделирование необходимо для разработки программного продукта, поддерживающего абсолютно новое направление бизнеса. В этом случае бизнес-моделирование служит для определения требований и пригодности (осуществимости) нового бизнеса. В этом сценарии бизнес-моделирование выделяется как отдельный проект.

Модернизация

Если организация решила полностью модернизировать (обновить) бизнес (бизнес - реинжиниринг), может потребоваться реализации одного или нескольких проектов бизнес-моделирования. Как правило, бизнес-реинжиниринг проходит в несколько этапов: предложение нового бизнеса, обратное проектирование существующего бизнеса, прямая разработка нового бизнеса и внедрение нового бизнеса.

Технологический процесс

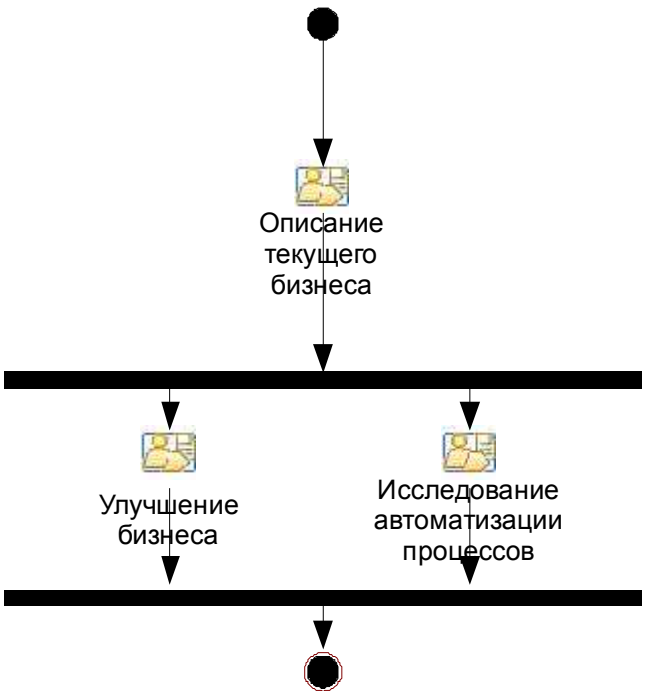


Рисунок 8.2 – Технологический процесс дисциплины «Бизнес-моделирование»

Описание текущего бизнеса

Оценка организации

Цели:

- Описать текущий статус организации заказчика, в которой должно быть развернуто приложение, в аспекте текущего процесса организации, инструментария, компетентностей людей, взаимодействия персонала, клиентов, конкурентов, технических тенденций, проблем и областей развития.
- Мотивировать целесообразность инжиниринга организации заказчика.
- Идентифицировать среди заинтересованных сторон участников процесса бизнес моделирования.

Исполнитель: Бизнес-аналитик

Входные документы: -

Выходные документы: Оценка организации

Рабочие операции:

- инициация оценки;
- определение заинтересованных сторон;
- определение ключевых лиц;
- оценка бизнес-идеи и стратегии организации;
- определение метрик организации;
- выявление причин, повлекших необходимость изменений;
- оценка способности заказчика к изменению бизнеса;
- выявление проблемных участков;
- запись рекомендаций.

Определение и согласование целей

Цели:

- определение целей бизнес-моделирования;
- разработка видения будущей организации;

- согласование целей бизнес-моделирования;
- определение реальных ожиданий заинтересованных сторон;

Исполнитель: Бизнес-аналитик

Входные документы: Оценка организации

Выходные документы: Документ «Бизнес-видение»

Рабочие операции:

- определение границ целевой организации;
- определение заинтересованных сторон;
- согласование целей;
- определение возможных ограничений;
- формулировка проблемы;
- расстановка приоритетов;
- разработка документа «Бизнес-видение»;
- оценка результатов.

Определение бизнес-целей

Только нисходящего, либо только восходящего подхода не достаточно для определения бизнес-целей. Необходимо исследовать и синхронизировать бизнес-цели от обеих перспектив. Объединенный подход облегчает понимание, помогает определить стратегические, тактические и оперативные уровни организации.

Цели:

- определение целей планирования и управления бизнесом;
- согласование долгосрочных стратегических целей и краткосрочных оперативных целей;
- транслирование бизнес-стратегии в действие;
- определение метрик для измерения и улучшения действий бизнеса.

Исполнитель: Бизнес-аналитик

Входные документы: Документ «Бизнес-видение»

Выходные документы: Бизнес-цели

Рабочие операции:

- анализ конкурентоспособного позиционирования;
- определение бизнес-целей;
- описание метрик;
- структурирование бизнес-целей;
- проверка результатов.

Определение бизнес-акторов и прецедентов

Цели:

- определение границ системы для моделирования;
- определение кто и что будет взаимодействовать с бизнесом;
- выделение бизнес-процессов;
- создание диаграмм модели бизнес-прецедентов;
- проверка модели бизнес-прецедентов.

Исполнитель: Бизнес-аналитик

Входные документы: Документ «Бизнес-видение»

Выходные документы: Бизнес-актор, Бизнес-прецедент, Модель бизнес-прецедентов, Дополнительные бизнес-спецификации.

Рабочие операции:

- определение бизнес-акторов;
- определение бизнес-прецедентов;
- обзор бизнес-целей;
- расположение по приоритетам бизнес-прецедентов;
- разработка предварительного потока работ бизнес-прецедентов;
- описание взаимодействия бизнес-акторов и прецедентов;
- группировка акторов и прецедентов в пакеты;
- представление модели бизнес прецедентов диаграммами прецедентов;
- проверка результатов.

Анализ бизнес-архитектуры

Выполнение этой задачи необходимо в том случае, если вы занимаетесь инжинирингом бизнеса, если же вы описываете существующий процесс, то ее выполнение опционально.

Цели:

- определение факторов, имеющих значительное влияния на бизнес;
- описание архитектуры бизнеса;
- определение ключевых механизмов, шаблонов и концепций для моделирования бизнеса.

Исполнитель: Бизнес-архитектор

Входные документы: Документ «Бизнес-видение»

Выходные документы: Документ «Бизнес архитектура», Модель развертывания, Бизнес сущности, Бизнес участники, Бизнес система, Модель бизнес анализа

Рабочие операции:

- разработка краткого обзора бизнес-архитектуры;
- описание сил, влияющих на бизнес-архитектуру;
- выделение высокоуровневых частей организации;
- выделение бизнес-систем;
- выделение ключевых абстракций бизнес-участников и бизнес-сущностей;
- выделение приоритетных бизнес-прецедентов;
- определение географически распределенных ресурсов;
- определение уровня корпоративной культуры участников;
- проверка результатов.

Составление общего бизнес-словаря

Целью задачи является определение общего словаря, который может использоваться во всех текстовых описаниях бизнеса, особенно в описаниях бизнес-прецедентов.

Исполнитель: Бизнес-аналитик

Входные документы: Документ «Бизнес-видение»

Выходные документы: Бизнес-словарь

Рабочие операции:

- определение терминов;
- проверка результатов.

Выделение бизнес-правил

Цели:

- определение бизнес-правил, рассматриваемых в проекте;
- подробное описание бизнес-правил.

Исполнитель: Бизнес-аналитик

Входные документы: Документ «Бизнес-видение»

Выходные документы: Бизнес-правила

Рабочие операции:

- сбор источников;
- выражение правил;
- проверка результатов.

Анализ бизнес-прецедентов

Цели:

- определение элементов (бизнес-систем, бизнес-исполнителей), выполняющих поток событий прецедента;
- распределение поведения прецедента в элементах, используя анализ реализации бизнес-прецедента;
- определение обязанностей, атрибутов и ассоциаций бизнес-систем и исполнителей;
- определение бизнес-сущностей и событий.

Исполнитель: Бизнес-проектировщик

Входные документы: Модель бизнес-прецедентов, Документ «Бизнес-архитектура», Дополнительные бизнес-спецификации

Выходные документы: Реализация бизнес-прецедента, Бизнес-система, Бизнес-исполнитель, Модель бизнес-анализа

Рабочие операции:

- определение бизнес-исполнителей;
- определение бизнес-сущностей;
- определение бизнес-событий;
- описание реализации бизнес-прецедентов;
- структуризация модели бизнес-анализа;
- проверка результатов.

Определение контекста бизнес-системы

Создание высокоуровневого представления системы, основанного на модели бизнес-прецедентов, отображающего интерфейсы, связи с акторами, включая входные и выходные бизнес-сущности, которыми обмениваются бизнес-акторы и бизнес-системы.

Исполнитель: Бизнес-аналитик

Входные документы: Модель бизнес-анализа, Модель бизнес-прецедентов, Дополнительные бизнес-спецификации.

Выходные документы: Модель бизнес-анализа, Бизнес-операция, Модель бизнес-прецедентов.

Рабочие операции:

- начало;
- создание начальной контекстной диаграммы;
- определение связей и интерфейсов;
- детализация бизнес-операций и других характеристик.

Анализ бизнес-операций

Цели:

- разработка текста потока событий «черного ящика» для каждой бизнес-операции в каждом важном бизнес-прецеденте в последовательности шагов, описанных в терминах действий и взаимодействий подсистем;
- дополнение описаний локальными решениями, процессными решениями и решениями исполнителей;
- описание взаимодействия подсистем с помощью диаграмм последовательности или взаимодействия (модель анализа), основанного на описании последовательности шагов.

Исполнитель: Бизнес-проектировщик.

Входные документы: Модель бизнес-анализа, Модель бизнес-прецедентов, Бизнес-операции.

Выходные документы: Модель бизнес-анализа, Модель бизнес-прецедентов, Бизнес-операции, Реализации бизнес-операций.

Рабочие операции:

- преобразование описания «черного ящика» в описания шагов подсистем;
- дополнение шагов локальными, процессными и исполнительскими решениями;
- распределение шагов по подсистемам;
- уточнение взаимодействий для каждой системной операции;
- оценка результатов.

Проектирование бизнес-операций

Цели:

- определение предварительного взаимодействия подсистем в реализациях операций Бизнес-модели;
- определение и усовершенствование операций подсистем.

Исполнитель: Бизнес-проектировщик.

Входные документы: Модель бизнес-прецедентов, Модель бизнес-анализа, Бизнес-операции, Реализации бизнес-операций.

Выходные документы: Бизнес-модель, Модель бизнес-развертывания, Бизнес-операции, Реализации бизнес-операций.

Рабочие операции:

- создание реализации операций;
- объединение подобных шагов подсистем и описание операций подсистем;
- фиксирование результатов.

Улучшение бизнеса

Структурирование модели бизнес-прецедентов

Цели:

- определение поведения бизнес-прецедентов, которые должны рассматриваться как абстрактные;
- определение новых абстрактных бизнес-акторов, которые описывают роли нескольких конкретных акторов.

Исполнитель: Бизнес-аналитик.

Входные документы: Бизнес-прецедент, Модель бизнес-прецедентов.

Выходные документы: Бизнес-прецедент, Бизнес-актор, Модель бизнес-прецедентов.

Рабочие операции:

- установка отношений включения между бизнес-прецедентами;
- установка отношений расширения между бизнес-прецедентами;
- установка отношений обобщения между бизнес-прецедентами;
- установка отношений обобщения между бизнес-акторами;
- проверка результатов.

Детализация бизнес-прецедентов

Цели:

- подробное описание рабочего потока бизнес-прецедентов;
- поддержка бизнес-стратегии бизнес-прецедентами;

- единое представление рабочего представления бизнес-прецедентов клиентами, пользователями и другими заинтересованными сторонами.

Исполнитель: Бизнес-проектировщик

Входные документы: Бизнес-прецедент

Выходные документы: Бизнес-прецедент, Дополнительные бизнес-спецификации

Рабочие операции:

- сбор информации о бизнес-прецеденте;
- детализация рабочего потока бизнес-прецедента;
- определение бизнес-целей, поддерживаемых бизнес-прецедентом;
- структуризация рабочего потока бизнес-прецедента;
- отображение связей между бизнес-акторами и бизнес-прецедентами;
- описание специальных требований бизнес-прецедента;
- описание целей бизнес-прецедента;
- описание точек расширения;
- проверка результатов.

Детализация бизнес-исполнителей

Цели:

- описание обязанностей бизнес-исполнителей;
- определение требований к бизнес-исполнителей.

Исполнитель: Бизнес-проектировщик

Входные документы: Реализация бизнес-прецедента, Бизнес-исполнитель

Выходные документы: Бизнес-исполнитель

Рабочие операции:

- определение области ответственности;
- определение операций;
- определение атрибутов;
- описание требований к бизнес-исполнителю;

- анализ связей;
- проверка результатов.

Проверка модели бизнес-прецедентов

Целью задачи является формальная проверка результатов бизнес-моделирования на соответствие представлению бизнеса заинтересованными сторонами.

Исполнитель: Преподаватель

Входные документы: Бизнес-прецедент, модель бизнес-прецедентов

Выходные документы: Результаты проверки

Рабочие операции:

- общие рекомендации;
- назначение даты для обсуждения;
- подготовка и составление отчета об обнаруженных дефектах.

Исследование автоматизации процессов

Определение и согласование целей

Цели:

- определение целей бизнес-моделирования;
- разработка видения будущей организации;
- согласование целей бизнес-моделирования;
- определение реальных ожиданий заинтересованных сторон;

Исполнитель: Бизнес-аналитик

Входные документы: Оценка организации

Выходные документы: Документ «Бизнес-видение»

Рабочие операции:

- определение границ целевой организации;
- определение заинтересованных сторон;
- согласование целей;
- определение возможных ограничений;

- формулировка проблемы;
- расстановка приоритетов;
- разработка документа «Бизнес-видение»;
- оценка результатов.

Определение требований к автоматизации

Команда должна сделать предварительную оценку требуемой поддержки для измененных бизнес-прецедентов. На начальной стадии важно указать, какие методики доступны для реализации бизнеса. Необходимо ответить на следующие вопросы: Возможно ли применение новых инструментальных средств для бизнес-прецедентов? Возможно ли использование существующих инструментальных средств поддержки бизнеса? Возможна ли покупка готовых продуктов? Имеются ли необходимые ресурсы для новой разработки? Какова конфигурация компьютерных систем, терминалов, рабочих станций и сетевого оборудования? Требуется ли совместимость с существующими продуктами?

Цели:

- понять, как новые технологии могут использоваться для более эффективной организации;
- определить уровень автоматизации;
- получить системные требования из артефактов бизнес-моделирования.

Исполнитель: Бизнес-аналитик

Входные документы: Оценка организации

Выходные документы: Дополнительные спецификации, Модель прецедентов, Модель анализа

Рабочие операции:

- обзор новых технологий;
- определение системных акторов и прецедентов;
- определение объектов модели анализа;
- определение других источников требований к системе;

- проверка результатов.

Дисциплина «Управление требованиями»

Цели дисциплины «Управление требованиями»:

- установление и поддержка соглашений с клиентами и другими заинтересованными сторонами о возможностях системы;
- предоставление лучшего понимания требований к системе;
- определение границ системы;
- обеспечение основ для планирования технического содержания итерации;
- обеспечение основ для оценки времени проектирования системы;
- определение пользовательского интерфейса системы, исходя из потребностей и целей пользователей.

Для достижения этих целей, важно, прежде всего, понять определение и границы проблемы, которую необходимо решить с помощью системы; определить заинтересованные стороны, выявить, собрать и проанализировать их требования.

Рабочие продукты дисциплины разрабатываются для полного описания системы – функционала системы – акцентированном на представлении всех заинтересованных сторон, включая заказчиков и потенциальных пользователей, как важный источник информации (в дополнение к системным требованиям).

Дисциплина «Управление требованиями» связана с другими дисциплинами процесса:

- дисциплина «Анализ и проектирование» получает первичную информацию из дисциплины «Управление требованиями»;
- дисциплина «Управление проектом» позволяет планировать проект и каждую итерацию. Рабочие продукты дисциплины «Управление требованиями» – важный вход для планирования итерации.

Дисциплина «Бизнес-моделирование» предоставляет организационный контекст для системы, представленный в виде:

- бизнес-правил;
- модели бизнес-прецедентов;
- модели бизнес-анализа.

Технологический процесс

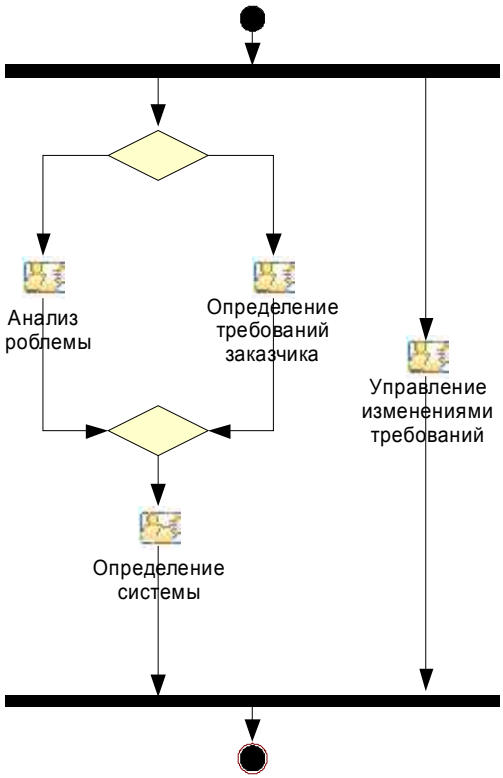


Рисунок 8.3 – Технологический процесс дисциплины «Управление требованиями»

Анализ проблемы

Создание общего словаря

Цель этой задачи состоит в том, чтобы составить общий словарь, который может использоваться во всех текстовых описаниях системы, особенно требованиях к программному обеспечению.

Исполнитель: Системный аналитик

Входные документы: -

Выходные документы: Глоссарий

Рабочие операции:

- определение терминов;
- проверка результатов.

Определение акторов и прецедентов

Цели этой задачи:

- определение границ системы – что будет обрабатываться системой, и что останется вне системы;
- определить, кто и что взаимодействует с системой;
- выделение функциональных возможностей системы.

Исполнитель: Системный аналитик

Входные документы: Требования заказчика, План итерации

Выходные документы: Модель прецедентов, Атрибуты требований

Рабочие операции:

- определение акторов;
- определение прецедентов;
- описание взаимодействия акторов и прецедентов;
- группировка акторов и прецедентов в пакеты;
- представление модели прецедентов в диаграммах;
- оценка результатов.

Разработка документа «Видение»

Целями задачи являются:

- определение соглашения о том, какие проблемы должны быть решены;
- определение заинтересованных сторон системы;
- определение границ системы;
- описание основных функциональных возможностей системы.
- Разрабатывая документ «Видение», нужно обращать внимание на:

- потенциальные (или фактические) пользователи системы отображаются на «человеческих» акторов разрабатываемой системы;
- обычно очень эффективно описывать границы системы с помощью акторов;
- ограничения, определенные в этой задаче будут служить начальным входом для проектирования ограничений, описываемых в дополнительных спецификациях.

Результаты бизнес-моделирования также представляют ценную информацию при определении заинтересованных сторон системы.

Исполнитель: Системный аналитик

Входные документы: Требования заказчика, План итерации

Выходные документы: документ «Видение», Атрибуты требований

Рабочие операции:

- определение соглашения по решаемой проблеме;
- определение заинтересованных сторон;
- определение границ системы;
- определение ограничений, накладываемых на систему;
- описание формулировки проблемы;
- определение функциональных возможностей системы;
- оценка результата.

Разработка плана управления требованиями

Цель задачи состоит в том, чтобы разработать план управления требованиями, который описывает информацию и механизмы управления, которые будут накапливаться и использоваться для измерения, сообщения и управления изменениями требований к продукту.

Перед началом описания требований проекта, нужно определить, каким образом регистрировать и организовывать их, а также как использовать атрибуты требований при управлении требованиями на протяжении жизненного цикла проекта.

Выбор соответствующих атрибутов и трассируемости для требований проекта поможет:

- оценивать воздействие на проект изменений требований;
- управлять объемом проекта;
- проверять, все ли требования к системе реализованы;
- управлять изменениями.

Необходимо фиксировать все решения, относящиеся к регистрации требований, элементам трассируемости, рекомендациям и стратегиям для атрибутов требований в плане управления требованиями.

Исполнитель: Системный аналитик

Входные документы: План разработки программного обеспечения, План итерации

Выходные документы: План управления требованиями

Рабочие операции:

- установка трассируемости;
- выбор атрибутов требований;
- написание плана.

Определение требований заказчика

Создание общего словаря

Цель этой задачи состоит в том, чтобы составить общий словарь, который может использоваться во всех текстовых описаниях системы, особенно требованиях к программному обеспечению.

Исполнитель: Системный аналитик

Входные документы: -

Выходные документы: Глоссарий

Рабочие операции:

- определение терминов;
- проверка результатов.

Выявление требований заказчикам

Цели задачи:

- понять, кто является заинтересованными сторонами проекта;
- собрать требования, о том, что должна выполнять система;
- отсортировать по приоритетам требования заинтересованных сторон;
- Выявление и сбор требований заказчика необходим для составления «списка желаний» различных заинтересованных сторон (клиентов, пользователей), которые бы включала система и которые были бы рассмотрены в проекте.

Исполнитель: Системный аналитик

Входные документы: План итерации

Выходные документы: Требования заказчика, Перечень атрибутов требований

Рабочие операции:

- определение источников требований;
- сбор информации;
- организовать обсуждение требований;
- проверка результатов.

Разработка документа «Видение»

Целями задачи являются:

- определение соглашения о том, какие проблемы должны быть решены;
- определение заинтересованных сторон системы;
- определение границ системы;
- описание основных функциональных возможностей системы.
- Разрабатывая документ «Видение», нужно обращать внимание на:
- потенциальные (или фактические) пользователи системы отображаются на «человеческих» акторов разрабатываемой системы;
- обычно очень эффективно описывать границы системы с помощью акторов;

- ограничения, определенные в этой задаче будут служить начальным входом для проектирования ограничений, описываемых в дополнительных спецификациях.

Результаты бизнес-моделирования также представляют ценную информацию при определении заинтересованных сторон системы.

Исполнитель: Системный аналитик

Входные документы: Требования заказчика, План итерации

Выходные документы: документ «Видение», Атрибуты требований

Рабочие операции:

- определение соглашения по решаемой проблеме;
- определение заинтересованных сторон;
- определение границ системы;
- определение ограничений, накладываемых на систему;
- описание формулировки проблемы;
- определение функциональных возможностей системы;
- оценка результата.

Определение акторов и прецедентов

Цели этой задачи:

- определение границ системы – что будет обрабатываться системой, и что останется вне системы;
- определить, кто и что взаимодействует с системой;
- выделение функциональных возможностей системы.

Исполнитель: Системный аналитик

Входные документы: Требования заказчика, План итерации

Выходные документы: Модель прецедентов, Атрибуты требований

Рабочие операции:

- определение акторов;
- определение прецедентов;
- описание взаимодействия акторов и прецедентов;

- группировка акторов и прецедентов в пакеты;
- представление модели прецедентов в диаграммах;
- оценка результатов.

Разработка дополнительных спецификаций

Целью задачи является документирование требований, которые не зафиксированы в прецедентах.

В течение дисциплины «Управление требованиями», основываясь на выявленных требованиях заказчика, требования, не применимые к прецедентам фиксируются в дополнительных спецификациях. Дополнительные спецификации могут включать как функциональные, так и не функциональные требования.

Выполняя эту задачу важно удостовериться, что все требования определены, зафиксированы и детализированы до уровня, понятного проектировщикам. Если требования трассируемы, или формально управляемы, необходимо убедиться, что каждое требование четко определено и именовано.

Исполнитель: Системный аналитик

Входные документы: Требования заказчика, План итерации

Выходные документы: Дополнительные спецификации, Атрибуты требований

Рабочие операции:

- сбор функциональных не прецедентных требований;
- определение возможностей системы;
- определение ограничений;
- определение требований к документации.

Управление зависимостями

Цель этой задачи состоит в том, чтобы использовать атрибуты и трассируемость требований проекта для управления границами проекта и изменениями требований.

Исполнитель: Системный аналитик

Входные документы: План управления требованиями

Выходные документы: План управления требованиями, Атрибуты требований, Документ «Видение»

Рабочие операции:

- назначение атрибутов;
- установка и проверка трассируемости;
- управление изменениями требований.

Определение системы

Разработка документа «Видение»

Целями задачи являются:

- определение соглашения о том, какие проблемы должны быть решены;
- определение заинтересованных сторон системы;
- определение границ системы;
- описание основных функциональных возможностей системы.
- Разрабатывая документ «Видение», нужно обращать внимание на:
- потенциальные (или фактические) пользователи системы отображаются на «человеческих» акторов разрабатываемой системы;
- обычно очень эффективно описывать границы системы с помощью акторов;
- ограничения, определенные в этой задаче будут служить начальным входом для проектирования ограничений, описываемых в дополнительных спецификациях.

Результаты бизнес-моделирования также представляют ценную информацию при определении заинтересованных сторон системы.

Исполнитель: Системный аналитик

Входные документы: Требования заказчика, План итерации

Выходные документы: документ «Видение», Атрибуты требований

Рабочие операции:

- определение соглашения по решаемой проблеме;
- определение заинтересованных сторон;
- определение границ системы;
- определение ограничений, накладываемых на систему;
- описание формулировки проблемы;
- определение функциональных возможностей системы;
- оценка результата.

Создание общего словаря

Цель этой задачи состоит в том, чтобы составить общий словарь, который может использоваться во всех текстовых описаниях системы, особенно требованиях к программному обеспечению.

Исполнитель: Системный аналитик

Входные документы: -

Выходные документы: Глоссарий

Рабочие операции:

- определение терминов;
- проверка результатов.

Определение акторов и прецедентов

Цели этой задачи:

- определение границ системы – что будет обрабатываться системой, и что останется вне системы;
- определить, кто и что взаимодействует с системой;
- выделение функциональных возможностей системы.

Исполнитель: Системный аналитик

Входные документы: Требования заказчика, План итерации

Выходные документы: Модель прецедентов, Атрибуты требований

Рабочие операции:

- определение акторов;
- определение прецедентов;
- описание взаимодействия акторов и прецедентов;
- группировка акторов и прецедентов в пакеты;
- представление модели прецедентов в диаграммах;
- оценка результатов.

Разработка дополнительных спецификаций

Целью задачи является документирование требований, которые не зафиксированы в прецедентах.

В течение дисциплины «Управление требованиями», основываясь на выявленных требованиях заказчика, требования, не применимые к прецедентам фиксируются в дополнительных спецификациях. Дополнительные спецификации могут включать как функциональные, так и не функциональные требования.

Выполняя эту задачу важно удостовериться, что все требования определены, зафиксированы и детализированы до уровня, понятного проектировщикам. Если требования трассируемы, или формально управляемы, необходимо убедиться, что каждое требование четко определено и именовано.

Исполнитель: Системный аналитик

Входные документы: Требования заказчика, План итерации

Выходные документы: Дополнительные спецификации, Атрибуты требований

Рабочие операции:

- сбор функциональных не прецедентных требований;
- определение возможностей системы;
- определение ограничений;
- определение требований к документации.

Управление зависимостями

Цель этой задачи состоит в том, чтобы использовать атрибуты и трассируемость требований проекта для управления границами проекта и изменениями требований.

Исполнитель: Системный аналитик

Входные документы: План управления требованиями

Выходные документы: План управления требованиями, Атрибуты требований, Документ «Видение»

Рабочие операции:

- назначение атрибутов;
- установка и проверка трассируемости;
- управление изменениями требований.

Детализация прецедентов

Цели задачи:

- Описание, как минимум, одного потока событий прецедента, достаточно детализированного для разработки программного обеспечения;
- детализация прецедентов для понимания представления акторов или заказчиков.

Исполнитель: Спецификатор требований

Входные документы: Прецедент, План итерации

Выходные документы: Прецедент, Атрибуты требований

Рабочие операции:

- описание требований прецедента;
- обзор и усовершенствование сценариев;
- детализация потока событий;
- структуризация потока событий;
- представление связей с акторами и другими прецедентами;
- описание специальных требований;
- определение коммуникационных протоколов;

- описание предусловий;
- описание постусловий;
- описание точек расширения;
- проверка результатов.

Детализация требований

Цель этой задачи состоит в том, чтобы собрать, детализировать и организовать набор (пакет) рабочих продуктов, которые полностью описывают программные требования системы.

Детализация требований включает детализацию прецедентов и описание дополнительных спецификаций для наиболее приоритетных требований.

Исполнитель: Спецификатор требований

Входные документы: Документ «Видение», План итерации

Выходные документы: Требования к программному обеспечению, Спецификация требований к программному обеспечению, Атрибуты требований

Рабочие операции:

- детализация требований к программному обеспечению;
- составление вспомогательных отчетов;
- распределение требований в пакеты для рассмотрения.

Управление изменениями требований

Структуризация модели прецедентов

Цели этой задачи:

- определение поведения в прецедентах, которые должны рассматриваться как абстрактные. Примерами такого поведения могут быть общее поведение, возможное поведение, поведение в исключительных ситуациях, поведения, которые можно рассматривать в следующих итерациях;
- определение новых абстрактных акторов, которые описывают роли не-

скольких конкретных акторов.

Исполнитель: Системный аналитик

Входные документы: Модель прецедентов

Выходные документы: Модель прецедентов, Словарь, Дополнительные спецификации, Атрибуты требований

Рабочие операции:

- определение общих требований;
- установка отношений включения между прецедентами;
- установка отношений расширения между прецедентами;
- установка отношений обобщения между прецедентами;
- установка отношений обобщения между акторами;
- группировка содержимого модели прецедентов в пакеты;
- оценка результатов.

Управление зависимостями

Цель этой задачи состоит в том, чтобы использовать атрибуты и трассируемость требований проекта для управления границами проекта и изменениями требований.

Исполнитель: Системный аналитик

Входные документы: План управления требованиями

Выходные документы: План управления требованиями, Атрибуты требований, Документ «Видение»

Рабочие операции:

- назначение атрибутов;
- установка и проверка трассируемости;
- управление изменениями требований.

Проверка требований

Цель задачи заключается в формальной проверке результатов задач дисциплины «Управление требованиями» на соответствие представления заказчика.

Исполнитель: Преподаватель

Входные документы: План итерации, Требования к программному обеспечению

Выходные документы: Результаты проверки

Рабочие операции:

- общие рекомендации;
- назначение даты для обсуждения;
- подготовка и составление отчета об обнаруженных дефектах.

Дисциплина «Анализ и проектирование»

Цели дисциплины «Анализ и проектирование»:

- преобразование требований в проект будущей системы;
- создание устойчивой архитектуры системы;
- адаптация проектного решения к среде реализации и увеличению производительности.

Дисциплина «Анализ и проектирование» связана со следующими дисциплинами:

- дисциплина «Управление требованиями» обеспечивает наличие первоначальной информации для анализа и проектирования;
- дисциплина «Управление проектом» планирует проект и каждую итерацию (описывается Планом итерации).

Технологический процесс

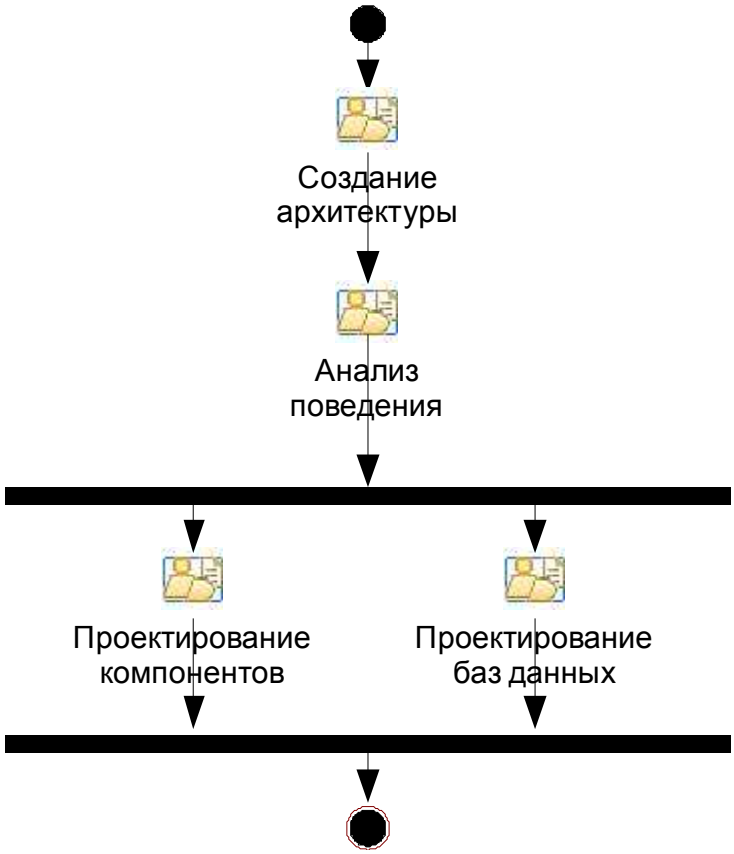


Рисунок 8.4 – Технологический процесс дисциплины «Анализ и проектирование»

Создание архитектуры

Определение контекста системы

Основанное на модели прецедентов создание высокоуровневой модели взаимодействия (высокоуровневое проектирование подсистем), включающей интерфейсы и связи с акторами, внешние входные и выходные объекты, которыми обмениваются система и акторы.

Исполнитель: Системный аналитик

Входные документы: Модель анализа, Модель прецедентов, Дополнительные спецификации

Выходные документы: Модель анализа, Модель прецедентов, Операции

Рабочие операции:

- введение;
- создание начальной контекстной диаграммы;
- уточнение операций и интерфейсов;
- детализация системных операций и других системных характеристик.

Архитектурный анализ

Цели:

- определение кандидата архитектуры системы;
- определение архитектурных шаблонов, ключевых механизмов и проектирование системных правил.

Архитектурный анализ необходим для определения кандидата архитектуры и выбор архитектурных методик, которые будут использоваться в системе. При повторном архитектурном анализе необходимо принимать во внимание предыдущие архитектурные решения. В системах с четко определенной архитектурой архитектурный анализ можно не проводить. Архитектурный анализ, прежде всего, необходим при проектировании новых систем, и систем не имеющих аналогов.

Исполнитель: Архитектор программного обеспечения

Входные документы: Документ «Видение», Словарь, Список рисков

Выходные документы: Модель дизайна, Модель развертывания, Модель анализа, Документ «Архитектура программного обеспечения»

Рабочие операции:

- разработка архитектурного обзора;
- определение высокоуровневой организации подсистем;
- определение ключевых абстракций;
- определение стереотипных взаимодействий;
- разработка вариантов развертывания;
- определение механизмов анализа;
- проверка результатов.

Анализ прецедентов

Цели:

- определение классов, выполняющих поток событий прецедента;
- распределение в эти классы поведения прецедента, используя анализ реализации прецедента;
- определение прав, атрибутов и ассоциаций классов;
- описание использования архитектурных механизмов.

Исполнитель: Проектировщик

Входные документы: Прецедент

Выходные документы: Реализация прецедентов, Модель анализа

Рабочие операции:

- создание анализа реализации прецедента;
- дополнение описания прецедента;
- определение классов анализа из поведения прецедента;
- распределение поведения в классах анализа;
- описание прав;
- описание атрибутов и ассоциаций;
- согласование анализа реализации прецедента;
- определение механизмов анализа;
- установка трассируемости;
- проверка результата.

Анализ операций

Цели:

- разработка текста потока событий «черного ящика» для каждой системной операции в каждом архитектурно-важном прецеденте в последовательности шагов, описанных в терминах действий и взаимодействий подсистем;
- дополнение описаний локальными решениями, процессными решениями и решениями исполнителей;

- описание взаимодействия подсистем с помощью диаграмм последовательности или взаимодействия (модель анализа), основанного на описании последовательности шагов.

В этой задаче Проектировщик начинает преобразование описания поведения на системном уровне, детализируя структуру системы (и связанные взаимодействия) в описания поведения на подсистемном уровне.

Исполнитель: Проектировщик

Входные документы: Модель анализа, Модель прецедентов, Операции

Выходные документы: Модель анализа, Модель прецедентов, Операции, Реализации операций

Рабочие операции:

- преобразование описания «черного ящика» в описания шагов подсистем;
- дополнение шагов локальными, процессными и исполнительскими решениями;
- распределение шагов по подсистемам;
- уточнение взаимодействий для каждой системной операции;
- оценка результатов.

Анализ поведения

Определение элементов проекта

Целью задачи является анализ взаимодействия классов анализа для определения элементов модели дизайна.

Классы анализа, представляющие результат анализа прецедентов, описывают концептуальных исполнителей поведения. В проектировании, классы анализа развиваются в различные виды элементов дизайна:

- классы, представляют набор детально описанных функций;
- подсистемы – набор описанных в широком смысле функций, возможно также составленных из набора подсистем, но в конечном счете, наборе

классов;

- активные классы, представляющие потоки в системе;
- интерфейсы, представляющие абстрактно объявленные функции, предоставляемые классом или подсистемой.

Дополнительно, в дизайне можно также выявить:

- события – привлекающие внимание явления, возникающие во времени и пространстве, обычно (но не всегда) требующие некоторого ответа системы;
- сигналы, представляющие асинхронный механизм, используемый для взаимодействия событий определенного типа в системе.

Выявление таких деталей дает возможность исследовать различные аспекты дизайна:

- события и сигналы дают возможность описать асинхронные триггеры поведения, на которые должна отвечать система;
- классы и подсистемы позволяют группировать функции в относительно независимые модули. Класс – реализация атомарной функции, подсистемы – блоки, составленные из классов или других подсистем. Подсистемы используются для представления рабочих продуктов группы разработчиков как единый функциональный модуль, а также как логический элемент дизайна;
- активные классы используются для представления управляющих потоков в системе, позволяя моделировать параллельное выполнение. Активные классы обычно используются совместно с другими классами, но не обязательно, например, для моделирования общего поведения.
- интерфейсы позволяют исследовать и фиксировать «места соединения» системы, точно определяя, как будут взаимодействовать составные части системы;
- протоколы, используется в системах реального времени для точного определения передаваемых сообщений.

Разделяя задачи и решая каждую задачу в отдельности, упрощается процесс проектирования и разъясняется решение.

В этой задаче также фиксируется трассируемость между моделями системы.

Исполнитель: Архитектор программного обеспечения

Входные документы: Классы анализа

Выходные документы: Модель дизайна, Подсистемы, Сигналы, Пакеты, Классы, События, Интерфейсы

Рабочие операции:

- определение событий и сигналов;
- определение классов, активных классов и подсистем;
- определение интерфейсов подсистем.

Анализ прецедентов

Цели:

- определение классов, выполняющих поток событий прецедента;
- распределение в эти классы поведения прецедента, используя анализ реализации прецедента;
- определение прав, атрибутов и ассоциаций классов;
- описание использования архитектурных механизмов.

Исполнитель: Проектировщик

Входные документы: Прецедент

Выходные документы: Реализация прецедентов, Модель анализа

Рабочие операции:

- создание анализа реализации прецедента;
- дополнение описания прецедента;
- определение классов анализа из поведения прецедента;
- распределение поведения в классах анализа;
- описание прав;
- описание атрибутов и ассоциаций;

- согласование анализа реализации прецедента;
- определение механизмов анализа;
- установка трассируемости;
- проверка результата.

Анализ операций

Цели:

- разработка текста потока событий «черного ящика» для каждой системной операции в каждом архитектурно-важном прецеденте в последовательности шагов, описанных в терминах действий и взаимодействий подсистем;
- дополнение описаний локальными решениями, процессными решениями и решениями исполнителей;
- описание взаимодействия подсистем с помощью диаграмм последовательности или взаимодействия (модель анализа), основанного на описании последовательности шагов.

В этой задаче Проектировщик начинает преобразование описания поведения на системном уровне, детализируя структуру системы (и связанные взаимодействия) в описания поведения на подсистемном уровне.

Исполнитель: Проектировщик

Входные документы: Модель анализа, Модель прецедентов, Операции

Выходные документы: Модель анализа, Модель прецедентов, Операции, Реализации операций

Рабочие операции:

- преобразование описания «черного ящика» в описания шагов подсистем;
- дополнение шагов локальными, процессными и исполнительскими решениями;
- распределение шагов по подсистемам;
- уточнение взаимодействий для каждой системной операции;

- оценка результатов.

Проектирование пользовательского интерфейса

Разработка проекта осмысленного пользовательского интерфейса, расширяемого и удобного в использовании.

При проектировании пользовательского интерфейса необходимо учитывать все пожелания, определенные при сборе требований, единый стиль интерфейса проекта и прототип пользовательского интерфейса, если есть. При возникновении необходимости уточнить пожелания пользователей, системный аналитик должен обновить описания требований (см. задачу Определение требований заказчика).

Исполнитель: Проектировщик пользовательского интерфейса

Входные документы: Требования к программному обеспечению

Выходные документы: Навигационная карта

Рабочие операции:

- описание характеристик пользователей;
- определение основных элементов пользовательского интерфейса;
- создание навигационной карты;
- детализация дизайна элементов интерфейса пользователя.

Создание прототипа пользовательского интерфейса

Прототип интерфейса пользователя системы необходим для проверки функциональности и удобства проекта пользовательского интерфейса.

При создании прототипа интерфейса пользователя необходимо придерживаться проекта пользовательского интерфейса, а также необходимо учитывать все пожелания, определенные при сборе требований, единый стиль интерфейса проекта и прототип пользовательского интерфейса, если есть. При возникновении необходимости уточнить пожелания пользователей, системный аналитик должен обновить описания требований (см. задачу Определение требований заказчика). При необходимости уточнить проект поль-

зовательского интерфейса, нужно выполнить соответствующие дополнения (см. задачу Проектирование пользовательского интерфейса).

Исполнитель: Проектировщик пользовательского интерфейса

Входные документы: Навигационная карта

Выходные документы: Прототип пользовательского интерфейса

Рабочие операции:

- проектирование прототипа пользовательского интерфейса;
- реализация прототипа пользовательского интерфейса;
- апробация прототипа пользовательского интерфейса.

Проверка проекта

Выполнение задачи необходимо для проверки соответствия модели дизайна требованиям проекта и пригодности для реализации.

Исполнитель: Преподаватель

Входные документы: Модель дизайна, Навигационная карта

Выходные документы: Отчет по проверке

Рабочие операции:

- запись общих рекомендаций;
- проверка модели дизайна в целом;
- проверка проекта реализации каждого прецедента;
- проверка каждого элемента проекта;
- подготовка отчета.

Проектирование компонентов

Проектирование прецедентов

Цели задачи:

- определить взаимодействия в реализации прецедентов;
- определить требования к операциям проектируемых классов;
- определить требования к проектируемым подсистемам и/или их интерфейсам.

Эта задача описывает взаимодействия для описания поведения системы с помощью диаграммы последовательности. Диаграммы последовательности особенно полезны, когда поведение системы или подсистем может быть описано синхронным обменом сообщениями. Асинхронный обмен сообщениями, особенно в системах, управляемых событиями, более просто описывается при помощи конечных автоматов и кооперации, позволяя доступно определить возможные взаимодействия между объектами.

Исполнитель: Проектировщик

Входные документы: Прецедент

Выходные документы: Модель дизайна

Рабочие операции:

- создание реализации прецедента;
- описание взаимодействия между объектами дизайна;
- описание статического поведения;
- переопределение описания потока событий;
- унификация классов и подсистем дизайна;
- проверка результатов.

Проектирование подсистем

Цели:

- определить поведение интерфейсов подсистем посредством взаимодействия внутренних модулей и внешних подсистем/интерфейсов;
- описать внутреннюю структуру подсистем;
- определить взаимодействие классов и интерфейсов подсистемы;
- определить зависимости с другими подсистемами.

Пример использования уровней подсистем с прямыми зависимостями...

Пример использования уровней подсистем с интерфейсными зависимостями...

Исполнитель: Проектировщик

Входные документы: Подсистемы дизайна, Интерфейс

Выходные документы: Классы дизайна, Подсистемы дизайна, Модель дизайна, Интерфейс, Компонент

Рабочие операции:

- распределение поведения подсистем в элементы подсистем;
- описание элементов подсистем;
- описание зависимостей подсистем.

Проектирование операций

Цели:

- предварительное представление взаимодействия подсистем в реализацию операций модели дизайна;
- определение операций подсистем.

Исполнитель: Проектировщик

Входные документы: Модель анализа, Операция, Реализация операции, Модель прецедентов

Выходные документы: Модель развертывания, Модель дизайна, Операция, Реализация операции

Рабочие операции:

- создание реализации операций;
- спецификация операций подсистем;
- резюмирование результатов.

Проектирование классов

Цели:

- отражение в классах поведения, требуемого реализацией прецедентов;
- однозначное определение класса для реализации;
- обработка нефункциональных требований, относящихся к классам;
- определение механизмов используемых классом.

Классы – минимальные единицы проекта, выполняющие реальную работу системы. Другие элементы дизайна, такие как подсистемы, пакеты и

кооперации, описывают, как сгруппированы классы и как они взаимодействуют.

Активные классы – классы дизайна, которые координируют и управляют поведением пассивных классов. Активный класс – класс, экземпляры которого – активные объекты, имеющие собственный поток управления.

Исполнитель: Проектировщик

Входные документы: Модель анализа

Выходные документы: Класс дизайна, Модель дизайна, Компонент

Рабочие операции:

- определение паттернов и механизмов;
- создание начального дизайна классов;
- определение статичных классов;
- определение области видимости классов;
- определение операций;
- определение методов;
- определение состояний;
- определение атрибутов;
- определение зависимостей;
- определение ассоциаций;
- определение внутренней структуры;
- определение связей обобщения;
- решение коллизий прецедентов;
- обработка основных нефункциональных требований;
- проверка результатов.

Проверка проекта

Выполнение задачи необходимо для проверки соответствия модели дизайна требованиям проекта и пригодности для реализации.

Исполнитель: Преподаватель

Входные документы: Модель дизайна, Навигационная карта

Выходные документы: Отчет по проверке

Рабочие операции:

- запись общих рекомендаций;
- проверка модели дизайна в целом;
- проверка проекта реализации каждого прецедента;
- проверка каждого элемента проекта;
- подготовка отчета.

Проектирование баз данных

Проектирование классов

Цели:

- отражение в классах поведения, требуемого реализацией прецедентов;
- однозначное определение класса для реализации;
- обработка нефункциональных требований, относящихся к классам;
- определение механизмов используемых классом.

Классы – минимальные единицы проекта, выполняющие реальную работу системы. Другие элементы дизайна, такие как подсистемы, пакеты и кооперации, описывают, как сгруппированы классы и как они взаимодействуют.

Активные классы – классы дизайна, которые координируют и управляют поведением пассивных классов. Активный класс – класс, экземпляры которого – активные объекты, имеющие собственный поток управления.

Исполнитель: Проектировщик

Входные документы: Модель анализа

Выходные документы: Класс дизайна, Модель дизайна, Компонент

Рабочие операции:

- определение паттернов и механизмов;
- создание начального дизайна классов;
- определение статичных классов;

- определение области видимости классов;
- определение операций;
- определение методов;
- определение состояний;
- определение атрибутов;
- определение зависимостей;
- определение ассоциаций;
- определение внутренней структуры;
- определение связей обобщения;
- решение коллизий прецедентов;
- обработка основных нефункциональных требований;
- проверка результатов.

Проектирование базы данных

Цели задачи:

- согласованное и эффективное хранение данных;
- определение поведения, реализуемого в базе данных.

Шаги, представленные в этой задаче предполагают, что модель данных будет выполнена с использованием системы управления реляционными базами данных.

Исполнитель: Проектировщик баз данных

Входные документы: Классы дизайна

Выходные документы: Модель данных

Рабочие операции:

- разработка концептуальной модели данных;
- разработка логической модели данных;
- разработка физической модели данных (необязательно);
- проверка результатов.

Проверка проекта

Выполнение задачи необходимо для проверки соответствия модели дизайна требованиям проекта и пригодности для реализации.

Исполнитель: Преподаватель

Входные документы: Модель дизайна, Навигационная карта

Выходные документы: Отчет по проверке

Рабочие операции:

- запись общих рекомендаций;
- проверка модели дизайна в целом;
- проверка проекта реализации каждого прецедента;
- проверка каждого элемента проекта;
- подготовка отчета.

Дисциплина «Управление проектом»

Цель этой дисциплины состоит в том, чтобы облегчить задачу, обеспечив некоторый контекст для управления проектом. Эта дисциплина представляет подход к управлению проектом, который заметно увеличит вероятность успешной разработки программного обеспечения.

Цели управления проектом:

- предоставление основы для управления преимущественно проектами по разработке программного обеспечения;
- предоставление практических рекомендаций для планирования, управления персоналом, исполнения и контроля проектов;
- предоставление основы для управления рисками.

Однако данная дисциплина не покрывает все аспекты управления проектом, в ней не затрагиваются такие задачи, как:

- управление персоналом: наем, обучение, руководство;
- управление бюджетированием: оценка, распределение, и .т.д.;
- управление договорами с заказчиками и разработчиками.

Дисциплина сосредоточена в основном на важных аспектах процесса итеративной разработки:

- управление рисками;
- планирование в итеративных проектах, как на протяжении всего жизненного цикла, так и на конкретной итерации;
- контроль выполнения итеративных проектов.

Дисциплина «Управление проектом» служит основой для создания и управления проектом. При этом другие дисциплины представляются как часть работы над проектом:

- дисциплина «Проектирование делопроизводства»;
- дисциплина «Управление требованиями»;
- дисциплина «Анализ и проектирование».

Дисциплина не пытается предоставить исчерпывающее руководство по управлению проектами. Здесь описывается только подмножество, непосредственно связанное с разработкой программного обеспечения. Более полно «лучшие практики» по управлению проектами описаны в Своде Знаний по Управлению Проектами (PMBOK®) Института Управления Проектами (PMI®)[42].

42 http://www.pmi.org

Технологический процесс

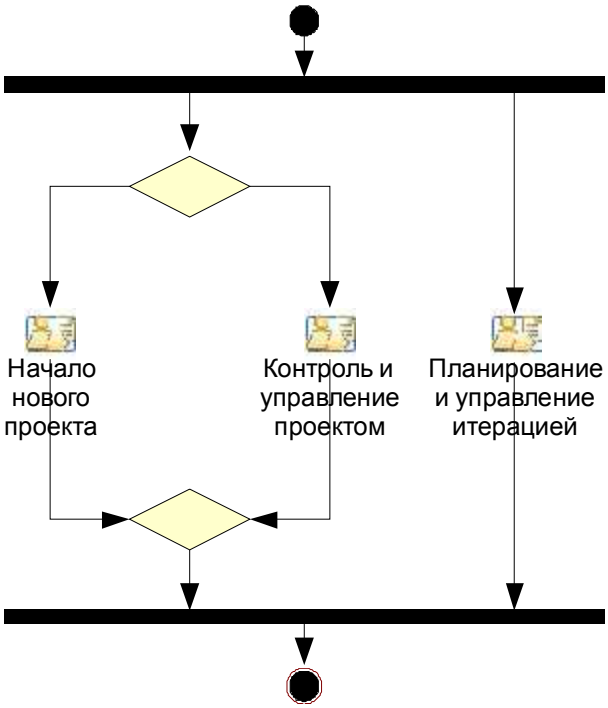


Рисунок 8.5 – Технологический процесс дисциплины «Управление проектом»

Начало нового проекта

Выявление и оценка рисков

Цели задачи:

- выявление, анализ и расстановка приоритетов рисков проекта;
- определение стратегий управления рисками;
- обновление списка рисков для отражения текущего состояния проекта.

Исполнитель: Руководитель проекта

Входные документы: Документ «Видение», План управления рисками

Выходные документы: Список рисков

Рабочие операции:

- выявление потенциальных рисков;
- анализ и расстановка приоритетов рисков;
- определение стратегий предотвращения рисков;
- определение стратегий снижения риска;

- определение альтернативных стратегий;
- пересмотрение рисков во время итерации;
- пересмотрение рисков в конце итерации.

Начало проекта

Целью задачи является назначение ролей участников проекта, а также определение критериев успешного завершения проекта.

Исполнитель: Руководитель проекта

Входные документы: -

Выходные документы: План разработки программного обеспечения

Рабочие операции:

- выбрать эксперта проекта (PRA);
- назначить роли участникам проекта;
- определить критерии успешного завершения проекта.

Разработка плана управления рисками

Задача описывает, как создать план выявления, анализа и расстановки приоритетов рисков, а также как определить стратегию управления рисками для наиболее значимых рисков проекта.

Исполнитель: Руководитель проекта

Входные документы: Список рисков

Выходные документы: План управления рисками

Рабочие операции:

- создать начальный список рисков;
- указать ответственного за риски;
- определить стратегии управления самыми важными рисками проекта;
- запланировать пересмотр списка рисков.

Контроль и управление проектом

Контроль состояния проекта

Задача описывает, как фиксировать текущее состояние проекта и оценивать его в соответствии с планами.

Исполнитель: Руководитель проекта

Входные документы: Список рисков, План разработки программного обеспечения, Список недочетов

Выходные документы: Список рисков, Список недочетов

Рабочие операции:

- определить состояние работ;
- получить показатели хода работ;
- получить показатели качества;
- сопоставить показатели с планами.

Обнаружение проблем и исключительных ситуаций

Задача позволяет обнаружить проблемы и недочеты. Последние могут относиться к проблемам проекта (например, отклонения от планов), проблемам продукта (дефекты, двусмысленности требований, неточности технологии) или возникновению рисковой ситуации. Также задача позволяет обнаружить исключительные ситуации. Под исключительными ситуациями понимаются проблемы, которые не позволяют продолжить работу над проектом (например, недоступность оборудования, сложность в принятии решения). Руководитель проекта должен непрерывно поддерживать актуальность списка проблем, чтобы в любой момент знать какие проблемы и недочеты необходимо решить.

Исполнитель: Руководитель проекта

Входные документы: Список недочетов, План решения проблем

Выходные документы: Список недочетов

Рабочие операции:

- оценка исключительных ситуаций и проблем;
- назначение соответствующих корректирующих действий.

Анализ проделанной работы

Данная задача описывает, как проводить анализ проделанной работы.

Исполнитель: Преподаватель

Входные документы: План итерации

Выходные документы: Заметки

Рабочие операции:

- запланировать встречу участников;
- информировать участников проекта;
- запись результатов анализа;
- планирование следующего анализа.

Планирование и управление итерацией

Определение участников проекта

Целью задачи является создание проектной группы и определение ролей в команде.

Исполнитель: Руководитель проекта

Входные документы: План разработки программного обеспечения

Выходные документы: План разработки программного обеспечения

Рабочие операции:

- формирование команды проекта;
- распределение ролей между участниками проекта;
- обучение при необходимости.

Начало итерации

В начале итерации руководителю проекта необходимо определить задачи итерации для каждого члена команды.

Исполнитель: Руководитель проекта

Входные документы: План разработки программного обеспечения, План итерации

Выходные документы: Порядок работ

Рабочие операции:

- определить перечень работ для каждого сотрудника;
- записать порядок работ.

Выявление и оценка рисков

Цели задачи:

- выявление, анализ и расстановка приоритетов рисков проекта;
- определение стратегий управления рисками;
- обновление списка рисков для отражения текущего состояния проекта.

Исполнитель: Руководитель проекта

Входные документы: Документ «Видение», План управления рисками

Выходные документы: Список рисков

Рабочие операции:

- выявление потенциальных рисков;
- анализ и расстановка приоритетов рисков;
- определение стратегий предотвращения рисков;
- определение стратегий снижения риска;
- определение альтернативных стратегий;
- пересмотрение рисков во время итерации;
- пересмотрение рисков в конце итерации.

Оценка итерации

Цели этой задачи:

- определение успехов и неудач итерации;
- усвоение практического опыта изменения проекта или улучшения процесса.

Одно из наиболее важных преимуществ итеративного подхода в отличие от каскадной модели – то, что итерации предоставляют понятные вехи для оценки выполнения проекта и предотвращения рисков. В течении итерации прогресс проекта и риски также должны оцениваться (хотя и неформально), чтобы гарантировать, что возникающие трудности не остановят проект.

Рисунок 8.6 – Оценка итерации

Исполнитель: Руководитель проекта

Входные документы: План итерации

Выходные документы: Оценка итерации

Рабочие операции:

- сбор показателей;
- оценка результатов итерации;
- определение внешних изменений;
- анализ критериев оценки.

Разработка плана итерации

Разработка плана итерации включает следующее:

- составление структурированного списка задач и назначение ответст-

венных за них;

- определение внутри итеративных вех и результатов;
- определение критериев оценки итерации.

Итерация – ограниченный по времени набор задач, направленный на разработку готового проекта. Внимание концентрируется на непрерывное улучшение проекта и уменьшение рисков.

Выполнение итерации подразумевает изменение некоторых рабочих продуктов. Иначе говоря, для разработки качественного проекта необходимо создавать прототипы и оценивать их, таким образом качество конечного проекта увеличивается, а изменения занимают не так много времени.

Исполнитель: Руководитель проекта

Входные документы: План разработки программного обеспечения, Список рисков, Документ «Видение»

Выходные документы: План итерации

Рабочие операции:

- определение цели итерации;
- определение критериев оценки итерации;
- определение работ итерации.

Фаза «Начало»

Технологический процесс

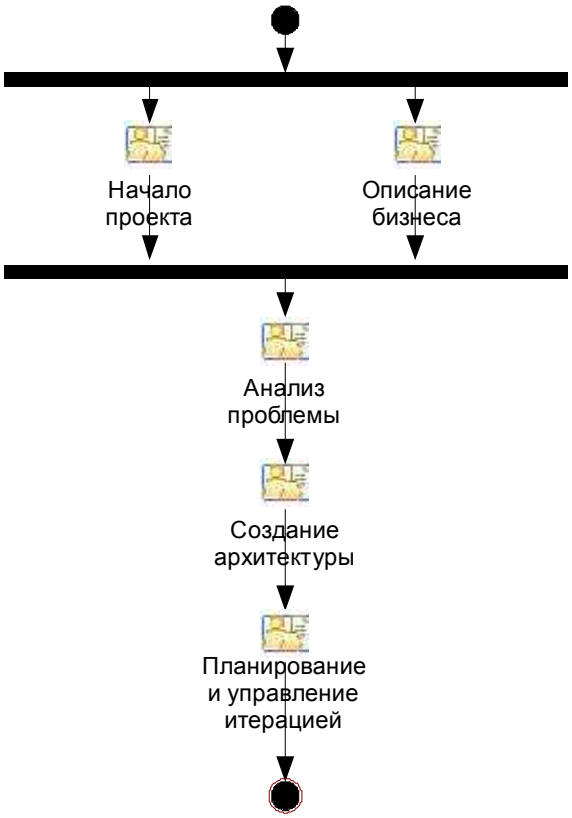


Рисунок 8.7 – Технологический процесс фазы «Начало»

Обзор

Общая цель фазы «Начало» состоит в достижении согласия заинтересованных сторон для целей жизненного цикла проекта. Фаза «Начало», прежде всего, нацелена на определение существенных рисков бизнеса и требований перед продолжением проекта. Для проектов расширения существующих систем фаза «Начало» сокращена, но также направлена на выявление выполнимости проекта.

Цели

Первичными целями фазы «Начало» являются:

- установление целей и границ проекта, включая видение продукта, а также что должен выполнять продукт, а что – нет;

- определение критичных прецедентов системы, первичных сценариев операций, управляющих основными взаимодействиями;
- создание начальной архитектуры, включающей основные сценарии;
- оценка длительности проекта;
- оценка потенциальных рисков.

Основные действия

В фазе «Начало» основными действиями являются следующие:

- определение области действия проекта, т.е. фиксация контекста, важнейших требований и ограничений для определение критериев приемлемости завершения проекта;
- оценка альтернатив для управления рисками, управление персоналом, плана проекта;
- проектирование возможной архитектуры, оценить допущения в проекте и определение, какие компоненты будут создаваться, какие – покупаться, а какие – использоваться повторно, с тем, чтобы оценить доступные ресурсы и план работ.

Веха

Фаза «Начало» завершается первой из основных вех проекта:цели жизненного цикла проекта. Веха служит для оценки выполнимости проекта. Критерии оценки:

- согласие заинтересованных сторон целей и длительности проекта;
- выявления набора требований и общедоступное понимание этих требований;
- подтверждение плана работ, приоритетов, рисков и процесса разработки;
- идентифицированы все риски и определены стратегии их снижения.

Если проект не пройдет данную веху, он должен быть отменен или существенно пересмотрен.

Артефакты

Артефакт (в порядке значимости)	Состояние в вехе
Документ «Видение»	Зафиксированы основные требования, особенности и ограничения
Список рисков	Определены начальные риски проекта
План проекта	Определены фазы, их цели и продолжительность
План итерации	Завершенный и проверенный план первой итерации
Словарь	Определены основные термины, составлен начальный словарь
Модель прецедентов (акторы, прецеденты)	Выявлены важные акторы и прецеденты, а также описаны потоки событий самых критичных прецедентов

Фаза «Исследование»

Технологический процесс

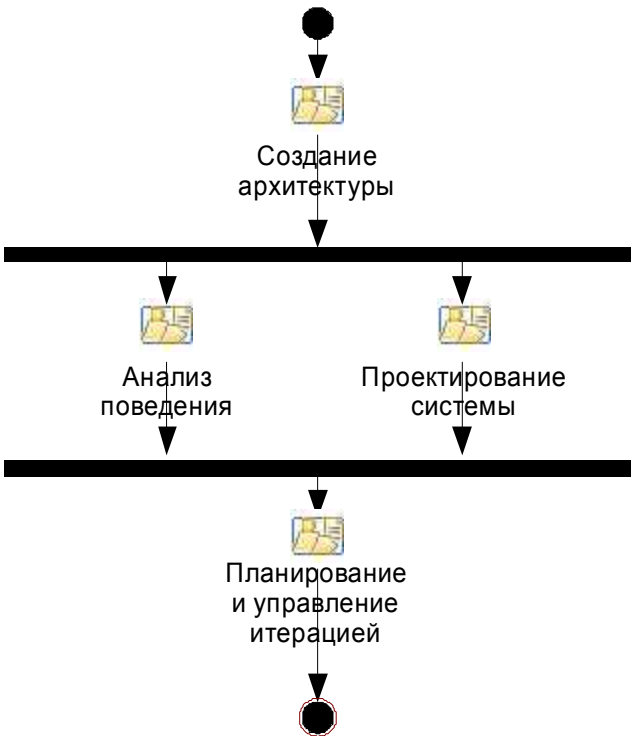


Рисунок 8.8 – Технологический процесс фазы «Исследование»

Обзор

Цель фазы исследование – разработка архитектуры системы, обеспечиваемое проектированием, проводимым в течении фазы. Большое влияние на архитектуру оказывают рассмотрение наиболее значимых требований и оценка рисков. Стабильность архитектуры оценивается посредством архитектурных прототипов.

Цели

Основные цели фазы «Исследование»:

- разработка устойчивой архитектуры, требований и планов, разработка стратегий уменьшения рисками;
- определение всех рисков, влияющих на архитектуру;
- создание архитектурной основы, опирающейся на сценарии снижения рисков проекта;
- создание эволюционного прототипа для снижения таких рисков, как:
  - согласование проекта / требований;
  - многократное использование компонентов;
  - представление проекта заказчикам и конечным пользователям.

Основные действия

Основные действия фазы «Исследование»:

- создание и проверка архитектуры системы на быстродействие и практичность;
- усовершенствование документа «Видение», основываясь на новой информации, полученной во время фазы, а также анализе критических прецедентов, управляющих архитектурными решениями;
- усовершенствование архитектуры и выбор компонентов.

Веха

Фаза «Исследование» завершается второй важной вехой проекта: архитектура жизненного цикла. Для этого исследуется область действия системы,

уточняются цели, а также осуществляется выбор архитектуры и стратегии уменьшения рисков. Критерии оценки:

- описаны требования и документ «Видение»;
- стабильная архитектура;
- основные риски или элементы рисков разрешены;
- согласие заинтересованных сторон с точкой зрения, описанной в документе «Видение».

Если проект не пройдет данную веху, он может быть отменен или значительно пересмотрен.

Артефакты

Артефакт (в порядке значимости)	Состояние в вехе
Прототипы	Разработаны один или несколько архитектурных прототипов для исследования функциональных возможностей и важных архитектурных сценариев
Список рисков	Обновленный и проверенный список рисков. Новые риски касаются нефункциональных требований
Документ «Архитектура системы»	Создание, обоснование и детальное описание важных архитектурных прецедентов, ключевых механизмов и модулей
Модель системы	Определены и описаны все элементы модели, а также сценарии выполнения
Модель данных	Определены и описаны основные элементы модели данных
Документ «Видение»	Усовершенствован, с учетом новой информации, полученной во время фазы и прецедентов, управляющих архитектурой
Модель прецедентов (акторы, прецеденты)	Модель прецедентов разработана полностью, все акторы и прецеденты описаны
Дополнительные спецификации	Дополнительные требования, включая нефункциональные требования, описаны и проверены

Необходимость прототипирования

Унифицированный процесс дает свободу создания прототипов нескольких типов архитектору программного обеспечения и руководителю проекта для разработки стратегий снижения рисков. Некоторые из этих прототипов могут быть только исследовательскими и, в последствии отклоненными. Однако, более вероятно (особенно для больших или новых систем), архитектура будет создана в результате эволюции прототипов, каждый из которых покрывает различные проблемы. Не нужно разрабатывать несколько фрагментов архитектуры, которые не могут быть интегрированы.

Контрольные вопросы

1. Сколько фаз проходит проект до получения конечного результата?
2. В чем отличие фазы от итерации?
3. На какие дисциплины разделена фаза «Начало», фаза «Исследование»?
4. Назовите основные роли проекта.
5. Какие документы необходимы в любом проекте? Почему?
6. Для чего строить бизнес-модель организации?
7. Какие модели создаются на фазе «Начало»?
8. Что является результатом фазы «Начало»?
9. Какие сведения содержит документ «Видение»?
10. Из каких элементов состоит проектная модель?
11. В какой из дисциплин описывается управление рисками?
12. Какая роль ответственная за определение требований?
13. Сколько раз в проекте нужно планировать итерации?
14. Какие действия необходимо выполнить для проверки архитектуры?
15. Для чего нужны вехи в проекте?
16. Нужно ли закончить итерацию при изменении требований? Почему?
17. Каким методом можно получить список рисков?
18. Какие рабочие продукты описывают взаимодействие подсистем?

9. Виртуальная среда исследования

Этап бизнес-моделирования или исследования предметной области является неотъемлемой частью процесса проектирования информационной системы. Любая деятельность или комплекс деятельности, в которой используются ресурсы для преобразования входов в выходы, может рассматриваться как процесс, который описывается различными документами. Любой бизнес-процесс характеризуется набором действий, исполнителями, входными, выходными документами и т.п., поэтому важно выделить общие документы для любого бизнес-процесса вне зависимости от вида деятельности предприятия. Документация дает возможность передать смысл и последовательность действий, представленных в виде модели бизнес-процесса. Таким образом, создание модели сводится к анализу документов и выделению из них характеристик бизнес-процесса.

Поэтому для описания бизнес-процесса предлагается следующая структура, включающая четыре основных типа документов для исследования:

Общий: документ, регламентирующий деятельность организации в целом.

К общим документам можно отнести следующие документы:

- документы, определяющие функционирование организации в целом;
- документы, определяющие направления ее деятельности;
- документы, определяющие правила и принципы осуществления стратегического управления;
- стратегический план (план развития) организации;
- и т.д.

Результатом систематизации информации, полученной из регламентирующих документов является информация, отражающая:

- общие принципы функционирования организации;
- структуру подразделений;
- перечень должностей, определяющий состав организационных подразделений;
- направления деятельности;
- правила взаимодействия компании с внешними организациями;
- основные бизнес-процессы;
- и т.д.

Должностная инструкция: нормативный документ, регламентирующий назначение и место работника в организации, его функциональные обязанности, права, ответственность и нормы поощрения.

Должностная инструкция включает в себя:

- общие положения (требования к квалификации, руководство, документация и т.д.);
- должностные обязанности (владелец бизнес-процессов, выполняемые действия, прочие обязанности);
- права;
- ответственность;
- взаимоотношения с другими сотрудниками;
- условия работы;
- и т.д.

В результате изучения данного документа определены функциональные обязанности, порядок выполнения операций, работа с документами соответствующей должности.

Интервью с ответственным: запрос данных, производящийся у владельца процесса по каждому бизнес-процессу.

Полученная в ходе интервью информация, описывает выполнение бизнес-процессов в следующей форме:

- название бизнес-процесса;
- условия начала выполнения бизнес-процесса;
- документы и данные, необходимые для выполнения бизнес-процесса и их источники;
- документы, создаваемые в результате выполнения бизнес-процесса и их получатели;
- действующие лица, принимающие участие в выполнении бизнес-процесса;
- материальные ценности, необходимые для выполнения бизнес-процесса, если таковые есть;
- материальные ценности – результат выполнения бизнес-процесса, если таковые есть;
- результаты выполнения бизнес-процесса;
- цель данного бизнес-процесса, его место и роль в общих задачах (процессах) компании;
- проблемы, возникающие при выполнении бизнес-процесса;
- нештатное завершение (выполнение) бизнес-процесса;
- последовательность действий выполнения бизнес-процесса;
- и т.д.

Интервью содержит наиболее ценную и реальную информацию о том, как происходит бизнес-процесс, т. к. в нем последовательно изложена беседа эксперта предметной области с интервьюируемым, ответственным за бизнес-процесс.

Интервью с участником может быть выделено в отдельный дополнительный тип бизнес - документа.

Результатом анализа интервью является информация о порядке выполнения действий бизнес-процесса конкретным исполнителем.

Документ: материальный объект, содержащий информацию в зафиксированном виде.

Все перечисленные документы перед началом работ по описанию бизнес-процессов рекомендуется собрать, структурировать и в дальнейшем использовать, как один из источников информации.

Документы могут быть представлены в виде:

- формы;
- схемы;
- записи;
- и т.д.

Изучив соответствующие формы документов и отчетов организации, можно выделить многие элементы бизнес-процесса.

Для последовательного составления модели бизнес-процесса необходимо задать соответствующий порядок работы с документами. Последовательность шагов составления модели позволяет контролировать процесс изучения предметной области и проверять каждый этап исследования бизнес-процесса. Методика контроля знаний может варьироваться.

Для реализации описанной методики исследования бизнес-процессов был разработан специализированный учебный программный комплекс «Виртуальная среда исследования бизнес-процессов».

Формализованное описание предметной области формируется в модуле «Автор задач», представленном в виде Win32 приложения.

Исследование студентом бизнес-процесса осуществляется в модуле «Студент», реализованного в виде тонкого клиента представленного Web-приложением.

Главное окно приложения, реализующее функции подготовки заданий, представлено на рисунке 9.1. Элементами интерфейса являются меню, четыре закладки, дерево элементов, панели отображения данных и строка состояния.

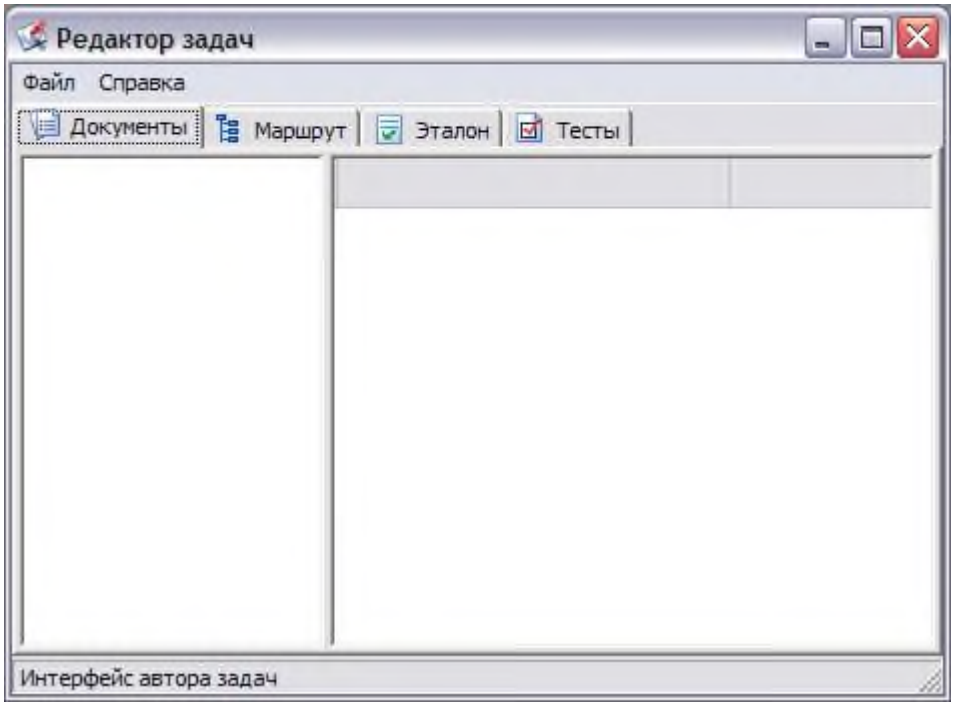


Рисунок 9.1 – Главное окно модуля «Автор задач»

Чтобы приступить к работе непосредственно с проектом, редактор позволяет либо создать новый проект, либо открыть уже существующий, в соответствии с рисунком 9.2.

Уже открытый проект можно сохранить или закрыть (если проект не сохранен, выведется сообщение о сохранении проекта).

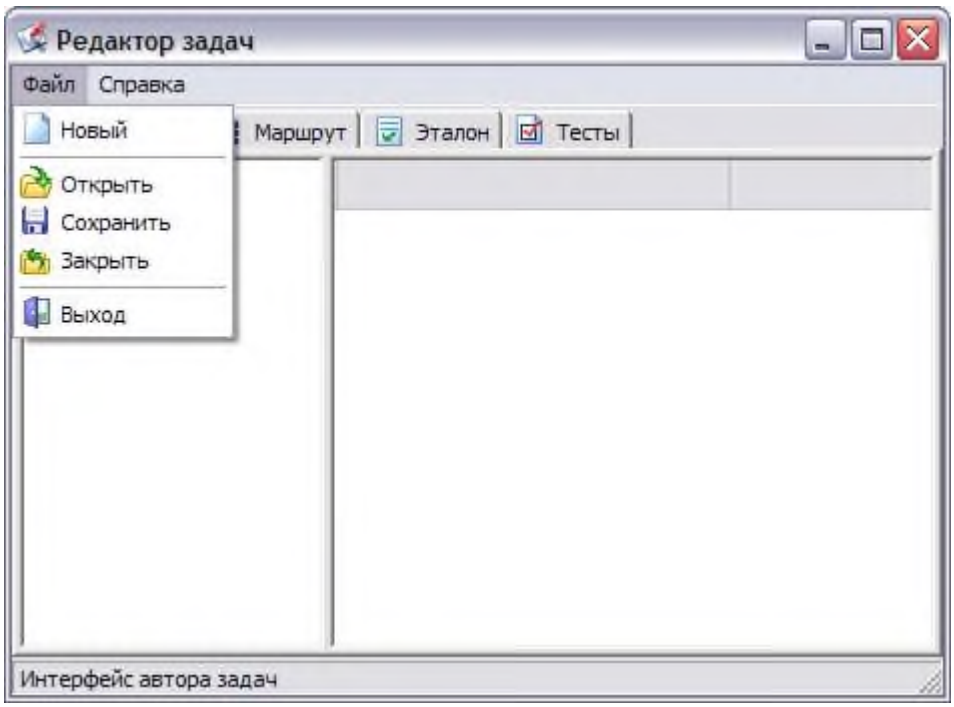


Рисунок 9.2 – Меню «Файл» интерфейса «Автор задач»

После открытия проекта, в окне приложения на вкладке «Документы» отображаются все документы предметной области, и содержание этих документов, в соответствии с рисунком 9.3. Панель отображения состоит в данном случае из структурных единиц соответствующего документа.

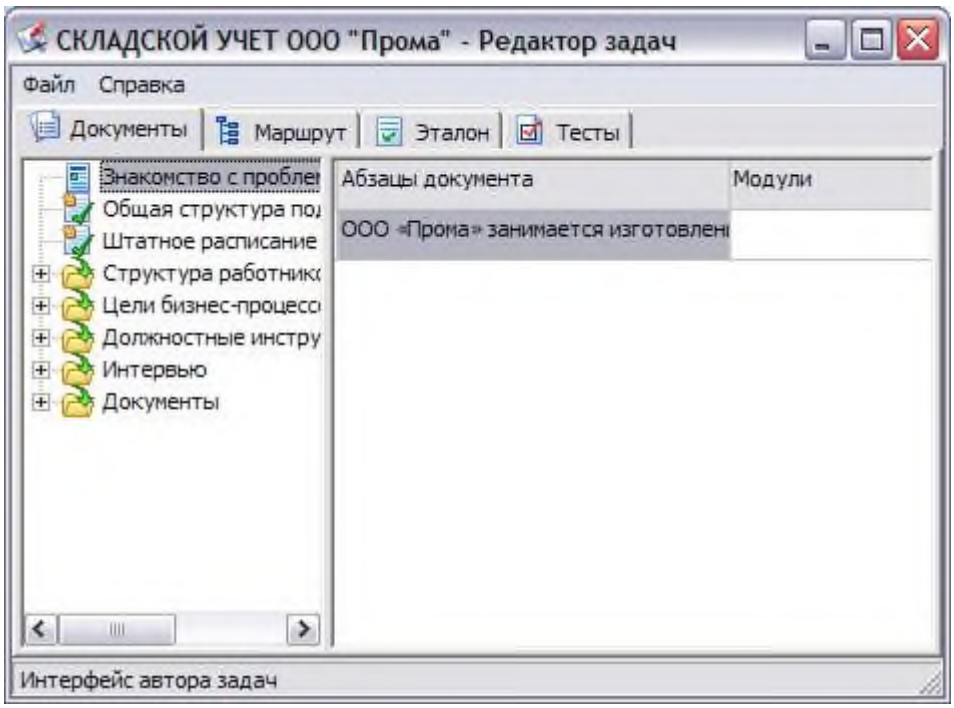


Рисунок 9.3 – Окно программы с загруженным проектом

Структурные единицы документов, расположенные на втором уровне дерева документов (в папках), можно добавить в один или несколько бизнес-процессов (модулей), в соответствии с рисунком 9.4, входящих в эталонное решение, вкладка «Эталон».

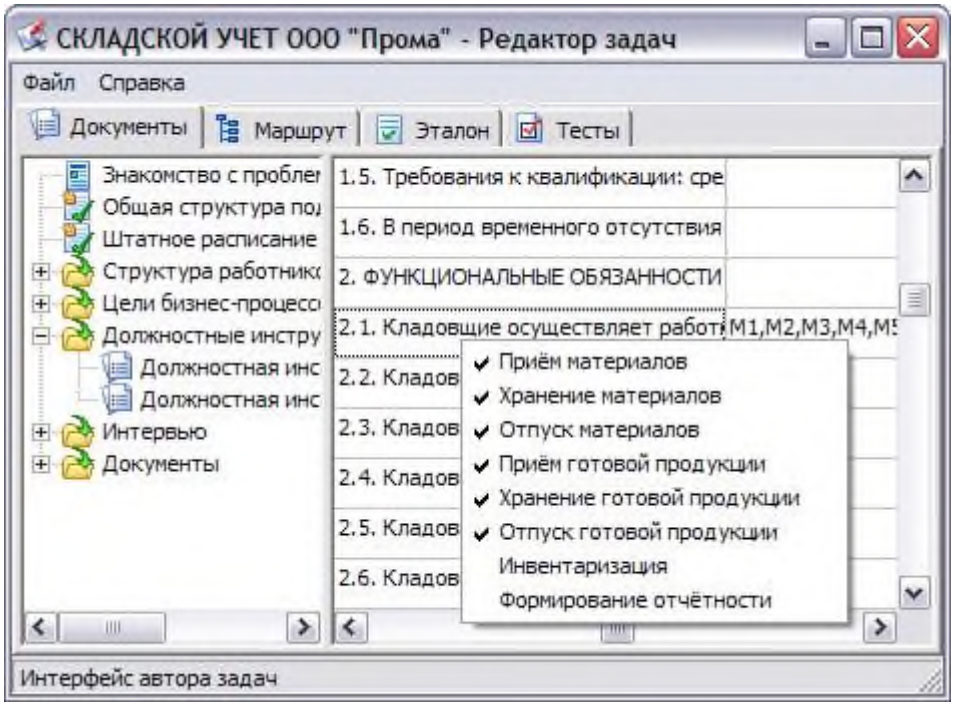


Рисунок 9.4 – Добавление структурной единицы в модуль

Добавление и удаление структурной единицы в модуль осуществляется установкой или снятием галочки напротив соответствующего модуля.

На показанной вкладке «Документы» можно также добавлять, удалять и переименовывать документы, автоматически осуществляется сортировка документов по типу, в соответствии с рисунком 9.5.

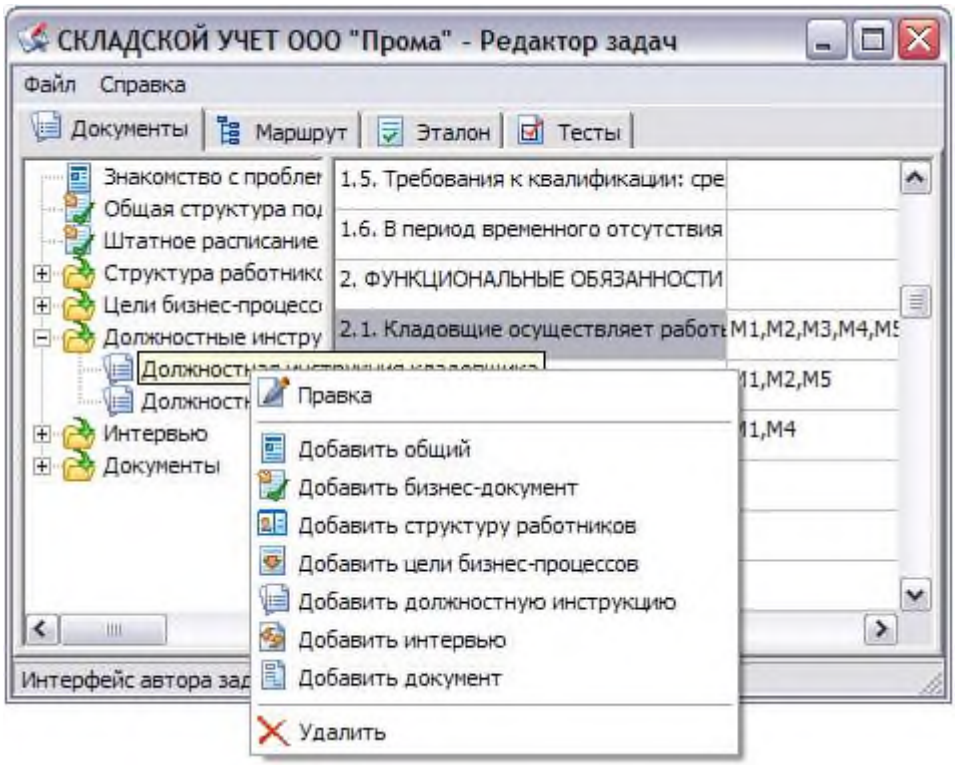


Рисунок 9.5 – Функции работы с документами

Добавлять, удалять модули, а также просматривать и удалять абзацы модулей позволяет вкладка «Эталон», рисунок 9.6. При удалении модуля удаляются все ссылки из абзацев на этот модуль, при этом выводится соответствующее предупреждение.

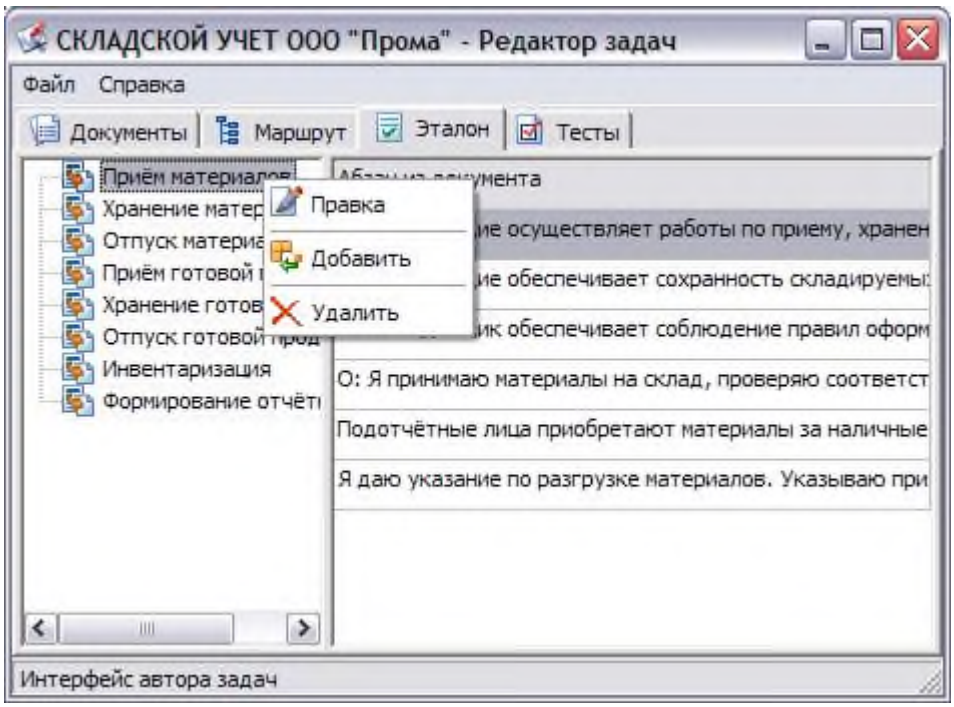


Рисунок 9.6 – Вкладка «Эталон», обеспечивающая работу с модулями

Наравне с документами и модулями в проекте задачи присутствуют тесты, состоящие из вопросов и нескольких ответов на каждый вопрос (один или несколько из которых могут быть правильными). Для указания правильности ответа, необходимо в поле ответа ввести любой символ напротив нужного варианта. Для указания неправильности – очистить поле, в соответствии с рисунком 9.7.

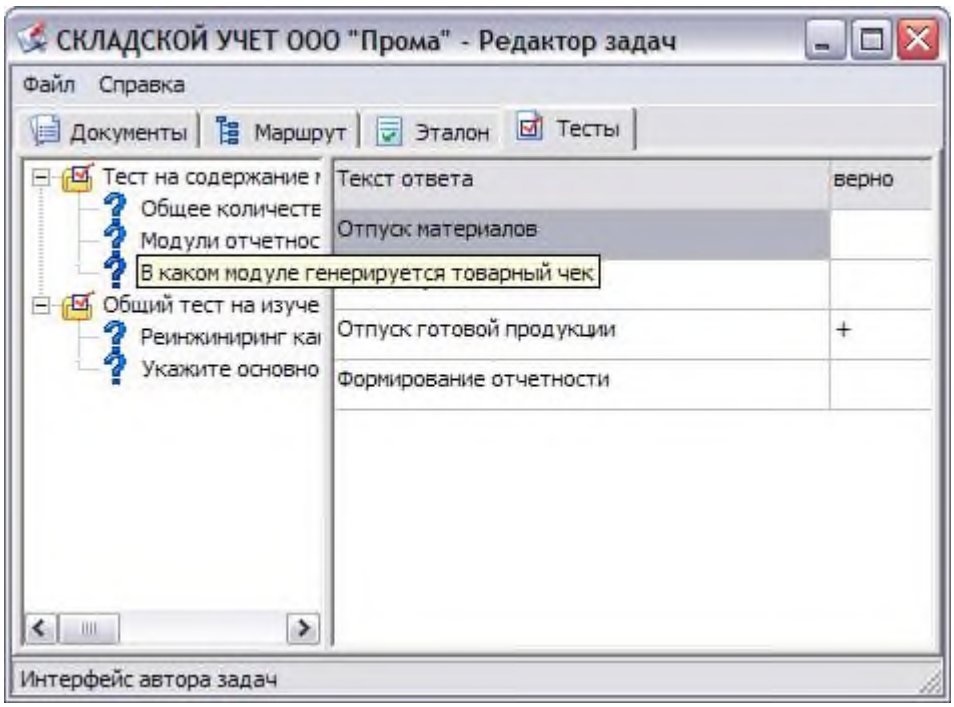


Рисунок 9.7 – Вкладка «Тесты», реализующая подготовку тестов

После завершения подготовки документов, тестов, эталонного решения следует сформировать маршрут изучения предметной области. Маршрут представляется в виде матрицы смежности, т.е. указывается возможные направления перемещения между документами, рисунок 9.8.

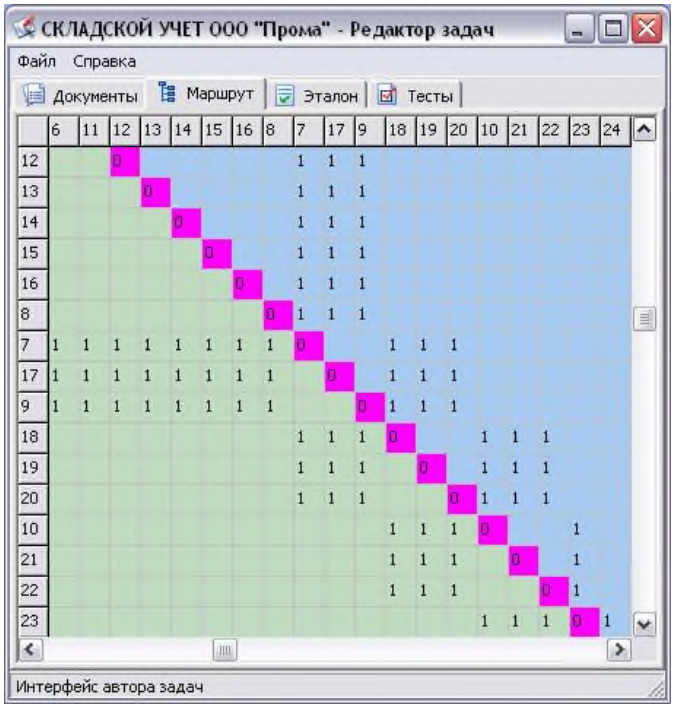


Рисунок 9.8 – Матрица смежности, реализующая маршрут задачи

В представленной матрице строки и столбцы представляют собой документы (в порядке добавления в проект), модули (обозначены ведущей литерой «М»), тесты (обозначены ведущей литерой «Т»).

Таким образом, подготовленные в модуле «Автор задач» представления бизнес-процесса исследуются студентом в модуле «Студент».

Для начала изучения предметной области необходимо выбрать задачу анализа.

Бизнес-процесс в задаче представлен пятью информационными блоками:

1. Меню пользователя, отображающее название предметной области, имя пользователя и ссылки на разделы системы.
2. Бизнес - документы, описывающие предметную область, накопление которых происходит по мере прохождения маршрута изучения предметной области.
3. Модули, описывающие действия бизнес-процесса.

4. Тесты, предназначенные для контроля качества анализа изучения предметной области. Тесты представлены в виде набора вопросов и ответов.
5. Содержание, отображающее информацию по бизнес - документам, модулям или тестам.

Исследование ПО состоит в изучении бизнес-документов ПО, поэтому пользователь может выбрать один из доступных бизнес-документов для получения необходимой информации о бизнес-процессе, в соответствии с рисунком 9.9.

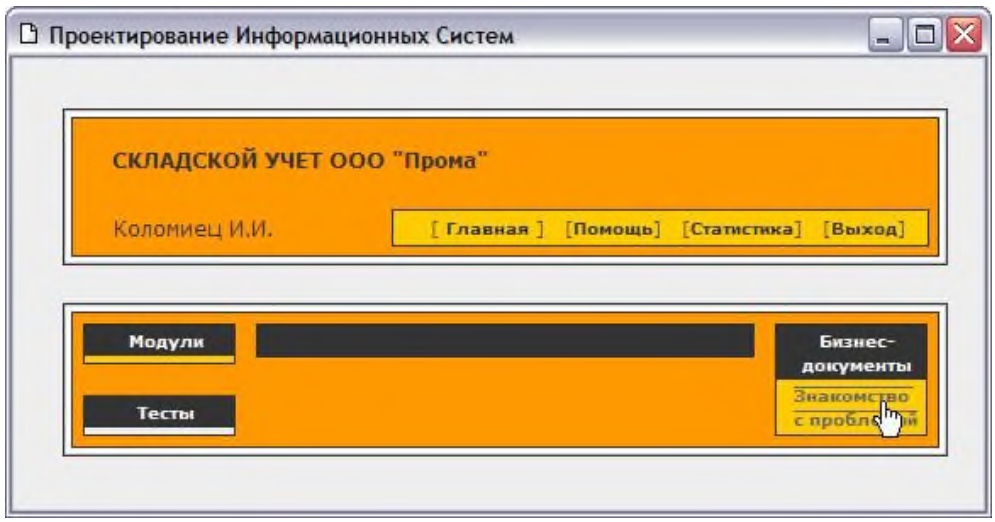


Рисунок 9.9 – Страница работы с предметной областью модуля «Студент»

При ознакомлении с одним из бизнес-документов, список бизнес-документов дополняется новыми, для более детального изучения предметной области, рисунок 9.10.

Рисунок 9.10 – Страница с дополненным списком бизнес-документов

Система позволяет работать с бизнес - документами разных типов, причем структурная единица бизнес-документа, детально описывающая тот или иной порядок действий, может использоваться в одном или нескольких модулях, рисунок 9.11.

Рисунок 9.11 – Страница, отражающая бизнес-документ общего типа.

В процессе изучения ПО, пользователю предлагается пройти тесты, являющиеся контрольной точкой в процессе исследования, в соответствии с рисунком 9.12.

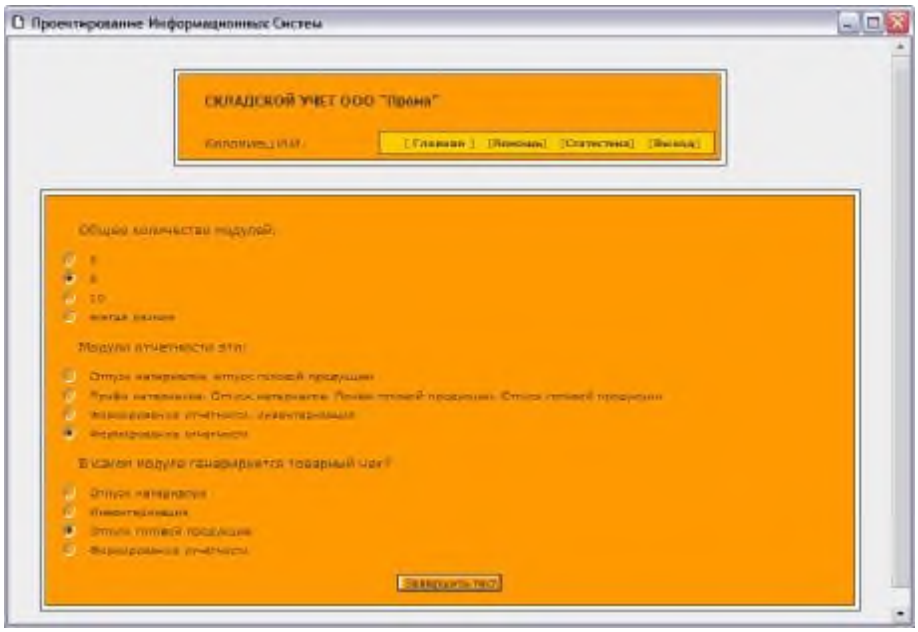


Рисунок 9.12 – Страница с тестовым испытанием

После успешного прохождения контрольной точки, пользователь возвращается на страницу работы с бизнес-документами, рисунок 9.13.

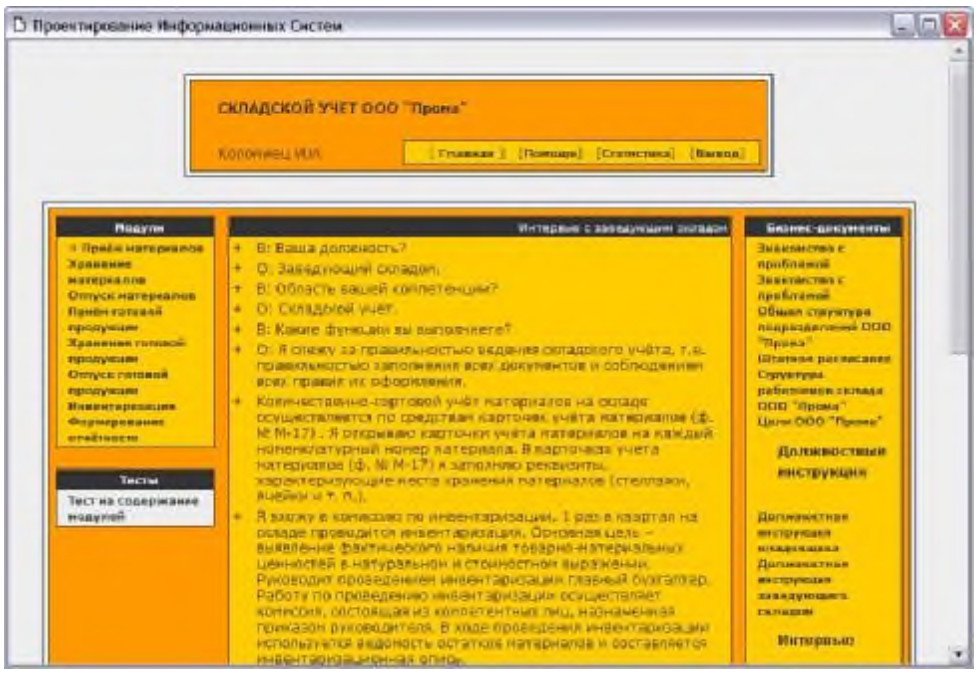


Рисунок 9.13 – Страница работы с бизнес-документами

При изучении бизнес-документов и выборе соответствующих структурных единиц пользователь должен описать бизнес-процесс, помеченный символом “»” в списке модулей, который является текущим.

Пользователь может сам выбирать формируемый модуль, переходя на его название, рисунок 9.14.

Рисунок 9.14 – Страница просмотра текущего модуля

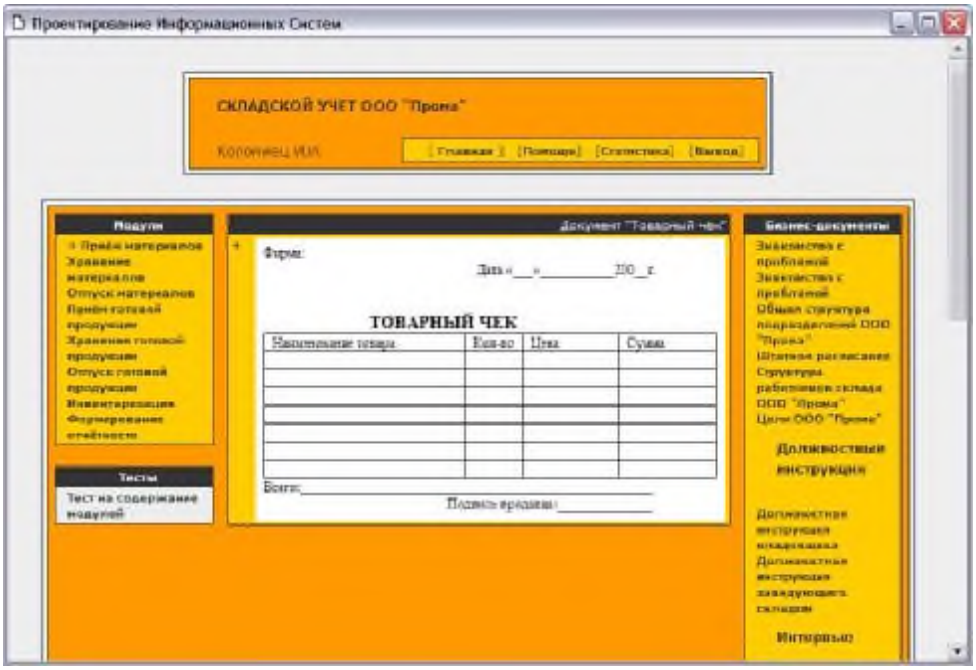


Рисунок 9.15 – Страница с формой документа

Описание предметной области также включает формы документов, используемые в бизнес-процессе, в соответствии с рисунком 9.15.

В результате последовательного анализа документов предметной области, формируется детальное представление исследуемого бизнес-процесса.

Контрольные вопросы:

1. Дайте определение бизнес-процесса?
2. Какими типами бизнес-документов представлен бизнес-процесс?
3. Каким набором элементов характеризуется бизнес-процесс?
4. Какие документы предметной области можно отнести к типу «Общий»?
5. Какими пунктами представлен документ «Должностная инструкция»?
6. Дайте определение владельца бизнес-процесса?
7. Дайте определение исполнителя бизнес-процесса?
8. Может ли структурная единица быть включена сразу в несколько бизнес-процессов?
9. Дайте определение бизнес-документа?
10. Что определяет маршрут задачи?

10. Объектно-ориентированная разработка

Под объектно-ориентированной разработкой понимают способ мышления, основанный на абстракциях, существующих как в реальном мире, так и в программах. Целью объектно-ориентированной разработки (ООР) является идентификация и упорядочение концепций приложения, а не окончательная реализация на языке программирования /68/.

Концепция ООП пришла из программирования. ООП дало программистам более гибкий инструмент, не характерный для традиционных языков программирования. ООР – это концептуальный процесс, независимый от языка программирования до последней фазы реализации.

ООР имеет стандартный язык моделирования UML. UML - это графический язык, предоставляющий инструмент визуализации модели с хорошо формализованным, и детально проработанным, аппаратом графических символов. UML является языком специфицирования, позволяющим описывать все существенные решения, касающиеся анализа, проектирования и реализации, которые должны приниматься в процессе разработки и развертывания системы программного обеспечения.UML - язык конструирования, так как модели, созданные с его помощью, могут быть использования для генерации кода на различных языках программирования. Как язык документирования, UML позволяет решить проблему документирования системной архитектуры и всех ее деталей, предлагает язык для формулирования требований к системе и определения тестов. Предоставляет средства для моделирования работ на этапе планирования проекта и управления версиями. UML используют для моделирования любых систем /69,70/.

В тоже время необходимо понимать, что UML - это всего лишь язык, он не привязан к методологии разработки. UML реализует объектно-ориентированный подход, но не воплощает методологию ООР. В этом сила данного языка, как инструмента, используемого в разработке программного

обеспечения. Именно поэтому он стал стандартом в области ООР, которая представляет довольно большой конгломерата различных методов и методологий.

У разработчика есть инструмент, который признан стандартом в ИТ мире. С его помощью он может разрабатывать любые системы. Как он будет разрабатывать? Какие методологии он будет использовать? Это зависит только от него (у Вас есть часы – можете измерять время, а можете забивать гвозди).

Формат данного пособия не даёт нам возможность познакомиться со всеми аспектами языка UML. Его описание в документе «UML 2.0 Superstructure Specification» занимает более 800 страниц, и это не единственный документ, специфицирующий данный язык. Наша задача - получить общее представление об этом инструменте.

Основное достоинство UML авторы видят в возможности моделировать систему с различных точек зрения. В отличие от структурного моделирования, ООП привносит динамическую составляющую, которая коренным образом изменяет представление о модели. Вспомним модели структурного анализа. Их состав: диаграммы одного вида и текст, поясняющий эти диаграммы. UML 2.0 определяет 13 видов диаграмм; UML 1.5 определяет 9 видов диаграмм. Диаграмма в UML не является семантическим элементом. Она дает представление о семантических элементах модели, однако, их значение не зависит от того, каким образом они представляются. Этим достигается организация единого семантического поля всех диаграмм. С другой стороны, диаграмма организует элементы UML модели для представления ее с какой-либо точки зрения. В книге /69/ выделяется 5 представлений модели, а в книге /68/ только три из этих моделей рассматриваются при построении ПО: классов, состояния, взаимодействия (не надо путать модели с диаграммами). UML вообще не определят такое понятия, как точка зрения. Точка зрения используется только для структурирования сложного языка UML. При модели-

ровании системы используется статическое представление и динамическое[43] представление. Каждый вид представления связан с определенным набором диаграмм. Моделирование системы это представление системы как статической структуры, так и динамическое поведение ее.

Поведение (behavior) - одно из фундаментальных понятий UML, определяющее спецификацию изменения состояния классификатора[44] с течением времени. Конкретными видами поведения является деятельность, взаимодействие, конечные автоматы и прецеденты. Поведение – это общий термин, обозначающий спецификацию того, как состояние экземпляра классификатора изменяется с течением времени в ответ на внешние события и внутренние вычисления.

В книге /68/ рассматривается методология разработки программного продукта, использующая модель классов для описания структуры системы и модели поведения для исследования динамики изменения и взаимодействия элементов системы (под системой можно понимать: бизнес - систему, информационную систему и компьютерную систему).

Можно считать такой подход упрощенным, но он даст нам возможность проследить весь процесс от бизнес - анализа до реализации. Рассмотрим элементы этих моделей по отдельности. Возьмем во внимание, что используется единый язык, в котором элемент языка не закреплен за отдельной диаграммой, следовательно, элементы, разобранные в одной модели, могут использоваться в других.

---

[43]Один их аспектов модели, представляет описание и реализацию поведения системы в течение времени, в отличие от статической структуры, которая изображается в структурном представлении.

[44] Классификатор – элемент модели, описывающий черты структуры и поведения системы. К классификаторам относятся: актеры, ассоциации, поведение, классы, кооперации, компоненты, типы данных, интерфейсы, узлы, сигналы, подсистемы и прецеденты. Наиболее общим видом классификатора является класс.

Поведение системы рассматривается в четырех аспектах: конечные автоматы, функциональная модель системы (модель прецедентов), модель взаимодействия, деятельность.

Статическое представление

Модель классов описывает статистическую структуру системы: элементы системы и их отношения между ними, атрибуты и операции для каждого класса. Модель классов считают одной из самых важных моделей /68/. Дж. Рамбо и М. Блах считают, что в основе всей системы находятся объекты, а не требуемая функциональность. Такое утверждение весьма спорное. Так, например, методология RUP проповедует подход, основанный на функциональности системы (прецедентно ориентированный подход).

Модель классов одна их самых важных моделей системы потому, что понятие классов, объектов и их отношения являются основными понятиями ООП, а именно, парадигма ООП лежит в основе UML.

Синтаксис и семантика элементов, используемых в модели классов

Объект(object) – это концепция, абстракция и сущность, обладающая индивидуальностью и имеющая смысл в рамках приложения. Объекты могут быть именами собственными или конкретными ссылками; могут существовать в реальном мире в виде материальных объектов или концепций (уравнение Тейлора); могут существовать только в компьютерной системе (массив А) и т.д. Все объекты обладают индивидуальностью и отличаются друг от друга. Объект является экземпляром класса.

Класс (class) описывает группу объектов (дескриптор[45] множества объектов) с одинаковыми свойствами, операциями, методами, поведением, отношениями и семантикой (человек, дом, город и т.д.).

Классы в UML описываются атрибутами, операциями и отношениями с другими элементами. Объекты одного класса имеют одинаковые атрибуты (но могут иметь разное значение этих атрибутов) и одинаковые операции, отношения с другими объектами могут быть разными.

Класс объединяет объекты с целью рассмотрения их на более высоком уровне абстракции. Объекты и классы в нотации UML показаны на рисунке 10.1 и 10.2 (графические примеры этой главы выполнены в инструменте Rational Rose и Visio).

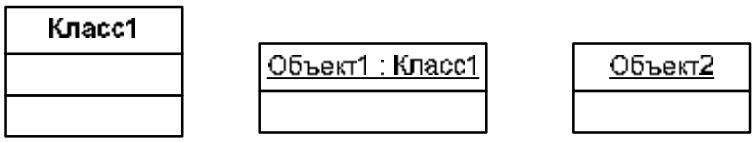


Рисунок 10.1 – Пример изображение класса и объекта

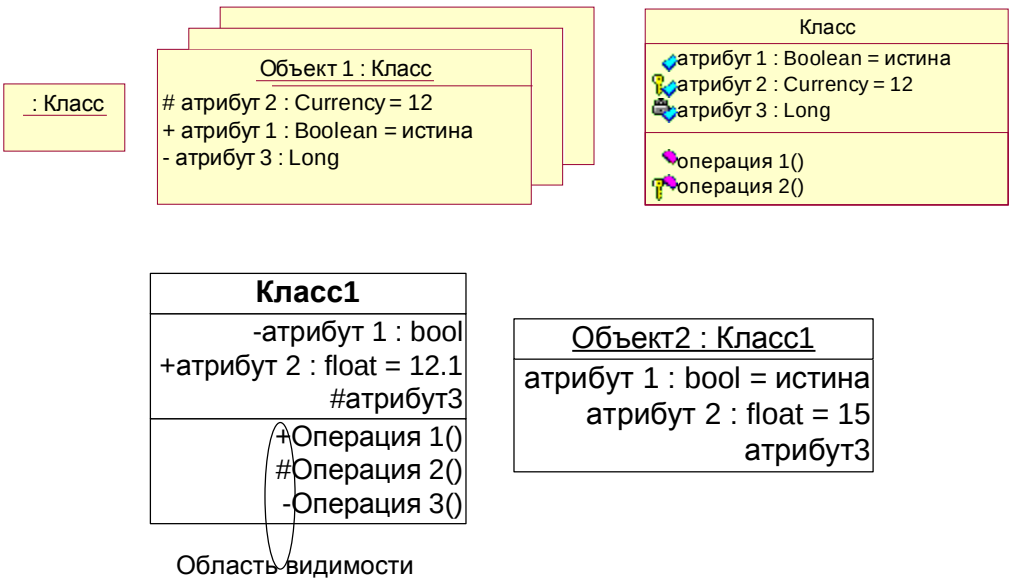


Рисунок 10.2 – Пример изображение классов. У атрибутов и операций показана область видимости. Вверху инструмент Rational Rose, внизу – Visio.

[45] Дескриптор - элемент модели, описывающий общие свойства некоего множества экземпляров, включая их структуру, отношения, поведения, ограничения, назначения и т.д.

Операция - эта функция или процедура, которая может быть применена к объектам класса. Каждая операция в качестве неявного аргумента принимает свой класс (объект всегда знает свой класс, поэтому он знает функции).

Метод (method) – реализация операции. Если операция не абстрактна, то у нее должен быть метод или событие вызова, определенных в классе или унаследованных от предка класса. Абстрактная операция – это операция, у которой отсутствует реализация в виде программного кода (есть спецификация, но нет метода). Абстрактная операция может быть только у абстрактного класса[46] (рисунок 10.3)

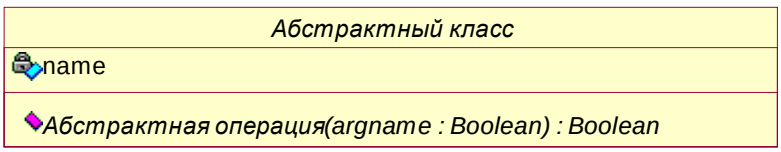


Рисунок 10.3 – Пример абстрактного класса (имена отмечены курсивом).

Уместно будет вспомнить основные положения объектно-ориентированной концепции: наследование, инкапсуляция и полиморфизм. Все они затрагивают операции и атрибуты классов.

На примере классов в UML вводится понятие классификаторов. Классификатор – элемент модели, описывающий черты структуры и поведения системы. К классификаторам относятся: актеры, ассоциации, поведение, классы, кооперации, компоненты, типы данных, интерфейсы, узлы, сигналы, подсистемы и прецеденты. Наиболее общим видом классификатора является класс. Большая часть свойств, присущих классу, есть у прочих классификаторов. У классификаторов есть имя, операции и атрибуты. Он может участвовать в различных отношениях. Внутри пространства имен классификатор обладает свойством множественности и видимостью.

---

[46] Абстрактный класс- это класс, экземпляр которого нельзя создать, т.к. его описание не полное (отсутствует метод для одного или несколько операций).

Связи и ассоциации позволяют устанавливать отношения между объектами и классами. Связь - экземпляр ассоциации и является единичным соединением двух и более объектов. Пример связи: Вася Иванов сдал предмет «Проектирование информационных систем».

Ассоциация – семантическое отношение между двумя или более классификаторами, определяющее связи между их экземплярами. Связи могут быть между объектами, а ассоциации - между классами (рисунок 10.4).

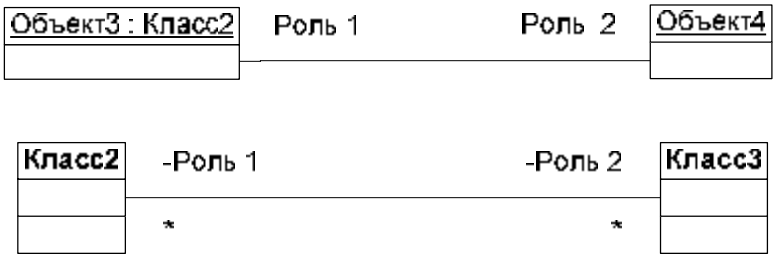


Рисунок 10.4 - Связи между объектами и ассоциации между классами/

Множественность (multiplicity) (иногда называют кардинальность) - это количество экземпляров одного класса, которые могут быть связаны с одним экземпляром другого класса через одну ассоциацию. Множественность может быть указана на полюсах ассоциации, атрибутах, частях композитных классов, повторах сообщений и т.д. Множественность необходимо отличать от кардинального числа (cardinality), которое является конкретным числом, а не диапазоном чисел, как в множественности. Множественность может иметь бесконечное число вариантов (1..*, 0..*,*..*, 3..6 и т.д.)[47] (рисунок 10.5).

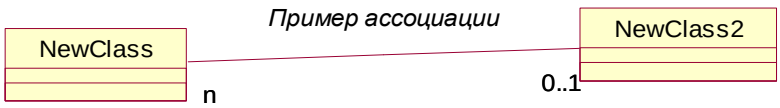


Рисунок 10.5 – Применение множественности.

Для связи не существует понятия множественность, конкретный экземпляр соединяется с одним конкретным экземпляром в бинарной связи.

[47] *- обозначает много.

Может быть несколько ассоциаций между двумя классами, в этом случае объекты этих классов имеют несколько связей, но в каждой связи участвуют только два экземпляра.

Ассоциация может иметь имя. Каждая присоединенная ассоциация к классу называется полюсом ассоциации. Вся основная информация об ассоциации прикрепляется к полюсу (имя, видимость, множественность, навигация) – рисунок 10.6.

Полюса ассоциации различимы, даже если они находятся в пределах одного класса (рисунок 10.7). Навигация прослеживает связь в графе с целью соотнесения объекта и значения (может использоваться на стадии проектирования для указания эффективного доступа к объекту). Не надо путать навигацию с направлением чтения имени ассоциации (рисунок 10.7).

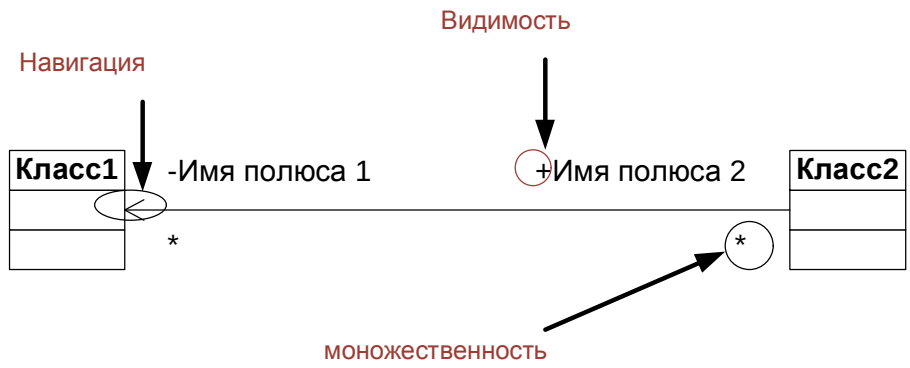


Рисунок 10.6 – Свойства полюса.

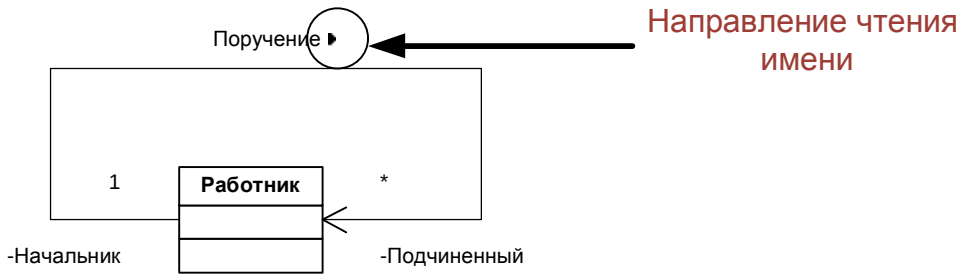


Рисунок 10.7 – Рефлексивная ассоциация.

Если значение атрибута ассоциации является уникальным в группе связанных объектов, то такое значение называется квалификатором, а ассоциа-

ция называется квалифицированная. Квалификатор позволяет выбрать отдельный объект из множества, тем самым уменьшает множественность с «много» до «1» и образует четкий маршрут для поиска целевого объекта по исходному (рисунок 10.8, 10.9).

Рисунок 10.8 - Пример ассоциации без квалификатора.

Рисунок 10.9 – Пример ассоциации с квалификатором.

Ассоциация может быть не только между двумя классами. Могут встречаться n-арные ассоциации (рисунок 10.10).

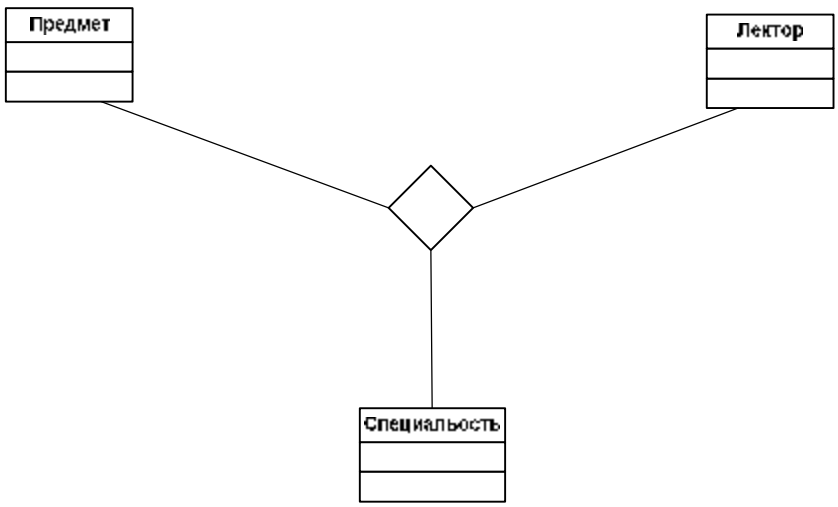


Рисунок 10.10 - Пример N- арной ассоциации

Агрегация - это частный случай ассоциации, описывающая отношение «часть - целое» между агрегатом (целое) и его составной частью (рисунок 10.11).

Связь, являющаяся экземпляром агрегации, подчиняется определенным правилам. Отношение агрегации транзитивно[48] и ассиметрично[49] во всех связях агрегации.

Существует более сильная форма агрегации, которая называется композицией (composition). Композит - это агрегат, который имеет дополнительные ограничения: объект может быть частью только одного композита, а композитный объект несет всю ответственность за управление своими частями (за их создание и уничтожение). Агрегация, не являющаяся композицией, называется разделяемой агрегацией. В разделяемой агрегации часть может принадлежать одному или нескольким агрегатам, а может существовать независимо от них. Агрегат не может обойтись без частей, так как он представляет собой их совокупность. Например, университет состоит из студентов, без студентов он не может существовать. Студент может обучаться сразу в нескольких университетах, а может вообще нигде не обучаться (отчисленный студент). Пример композиции: факультет состоит из кафедр, он управляет их возникновением и реорганизацией, кафедра без факультета не может существовать (будем так считать). Иллюстрация композиции и агрегации приведена на рисунке 10.11.

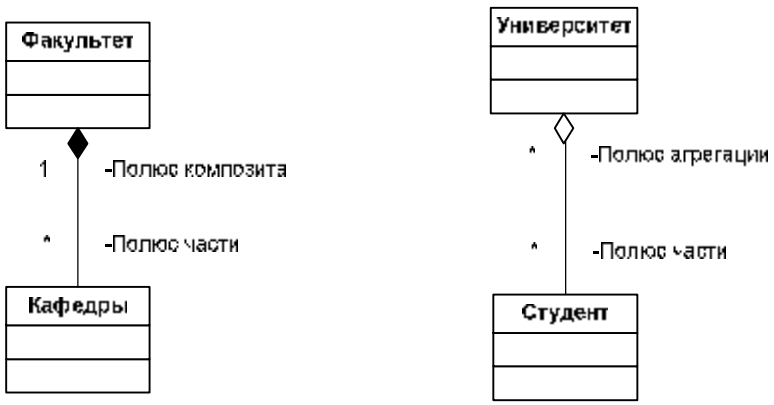


Рисунок 10.11 – Пример композиции и агрегации.

[48] В математике бинарное отношение R на множестве X называется транзитивным, если для любых трёх элементов множества a,b,c выполнение отношений aRb и bRc влечёт выполнение отношения aRc.

[49] Ассиметрично – в данном случае предполагает отсутствие циклов при обходе агрегации.

Ассоциации, также как и классы, могут иметь атрибуты. В этом случае UML предоставляет инструмент класса ассоциаций. Класс ассоциации – это ассоциация, которая одновременно является и классом. Не следует путать класс ассоциации с ассоциацией, которые были выделены в отдельный класс.

Так, например, классы Работник и Компания связаны ассоциацией, которую лучше представить, как класс ассоциации, так как данная ассоциация характеризуется зарплатой, датой найма и т.д. Класс ассоциации порождает один – единственный класс для каждой пары экземпляров Работник и Компания. Напротив, в модели, где Работник и Компания ассоциированы с классом Работник_Компании, может быть сколько угодно экземпляров Работники_Компании между конкретными экземплярами классов Работник и Компания (рисунок 10.12).

Обобщение (generalization) - это отношение между классом (суперклассом) и подклассом. Это отношение между общим (суперкласс) и специализированным (подкласс) элементом. Экземпляр специализированного эле мента является косвенным экземпляром общего элемента и наследует его характеристики (рисунок 10.13).

Рисунок 10.12 – Пример использование класс - ассоциации и ассоциации, выделенной в отдельный класс.

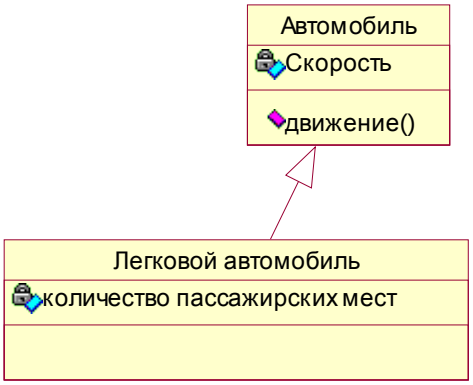
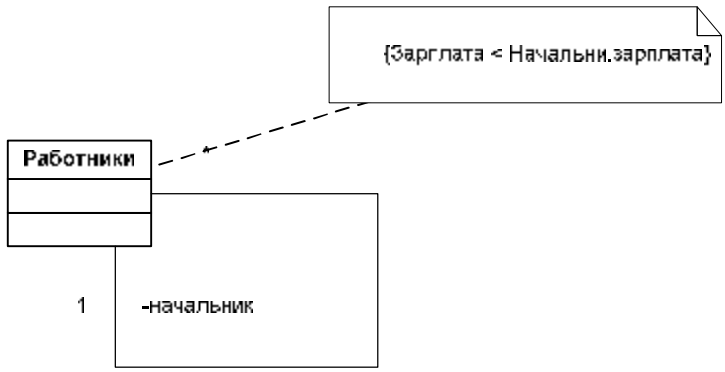


Рисунок 10.13 – Пример обобщение.

Обобщение используется для структурированного описания объектов (представление общих свойств в суперклассе), поддержки полиморфизма, повторного использование кода. С обобщением связаны термины наследования и перекрытия, которые известны из ООП. Подкласс может использоваться везде, где используется суперкласс, но не наоборот.

Довольно часто используется суперкласс для определения сигнатуры операций. Сами операции будут реализованы в подклассах. Например, в суперклассе Фигура объявляется метод Прорисовка, реализация которого будет индивидуальна для каждого подкласса (Окружность, Квадрат и т.д.). В этом случае Суперкласс объявляется абстрактным (он не может иметь конкретных экземпляров) с нереализованным, абстрактным методом (рисунок 10.3). В работе /68/ рекомендуется избегать конкретных суперклассов.

Ограничение (constraint) - это логическое условие, накладываемое на элементы модели, такие как объекты, классы, атрибуты, связи, ассоциации и набор обобщений. Ограничения можно записывать на естественном и на формальном языке (например, OCL –Object Constraint Language). Рекомендуется объявлять ограничения в модели. UML имеет две альтернативных системы обозначения ограничений: поместить в фигурные скобки, поместить в прямоугольник – комментарии (note) (рисунок 10.14).



{Зарплата < Начальни.зарплата}

Рисунок 10.14 – Способы показа ограничений.

Модель классов можно уместить на одной диаграмме, если решаются небольшие задачи. Большую модель трудно охватить целиком, рекомендуется разбивать её на части. Пакет – это группирующая сущность, которая позволяет разбить модель на логические части, объединенные общей темой.

Следует придерживаться следующих рекомендаций при определении пакета /68/:

- имена классификаторов в каждом пакете должны быть уникальны и по возможности согласованы в разных пакетах;
- класс, как и другие классификаторы, определяется только в одном пакет, в других пакетах, ссылающихся на этот класс, лучше использовать упрощенное изображение (для класса это прямоугольник с одним только именем);
- ассоциации и обобщения должны умещаться внутри одного пакета, как и другие отношения, используемые в разных моделях, классы могут использоваться в разных пакетах, тем самым связывать их.

Пакеты могут быть изображены на диаграмме пакетов. Нет строгого разграничения между различными структурными диаграммами, поэтому диаграмма пакетов используется только для удобства. Очень часто на диаграмме пакетов приводится отношение зависимости (Рисунок 10.15).

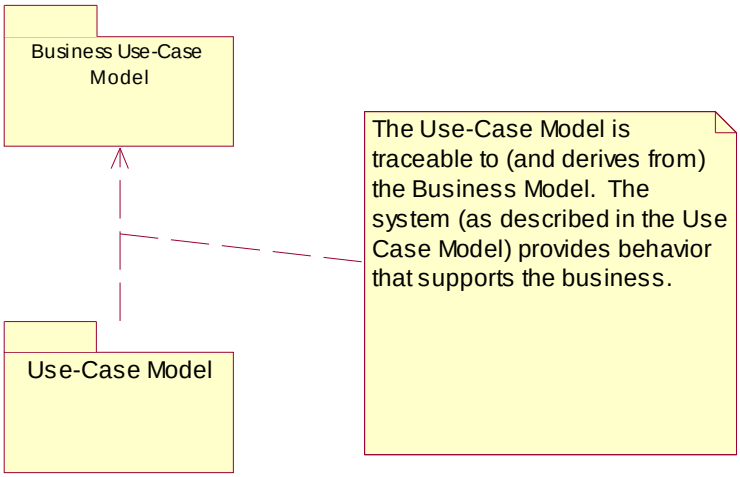


Рисунок 10.15 - Пример диаграммы пакетов, созданной шаблоном Rational Rose (между пакетами отношение зависимости).

Зависимость – отношение между двумя элементами, при котором изменение одного элемента (поставщика) может затронуть другой элемент (клиент) или предоставить необходимую ему информацию. В нашем примере клиент – пакет Use-Case Model, а поставщик Business Use-Case Model.

Синтаксические элементы UML могут быть расширены за счет использования стереотипов[50], теговых значений[51] и профилей[52].

В Ration Rose некоторые стереотипы имеют свое графическое представление (рисунок 10.16, 10.17).

[50] Стереотип - это новый элемент модели, который определяется в самой модели на основе существующего элемента.

[51] Теговое значение - это определение атрибута, принадлежащего элементу модели как таковому.

[52] Профиль - это пакет, задающий некоторое подмножество базовой метамодели.

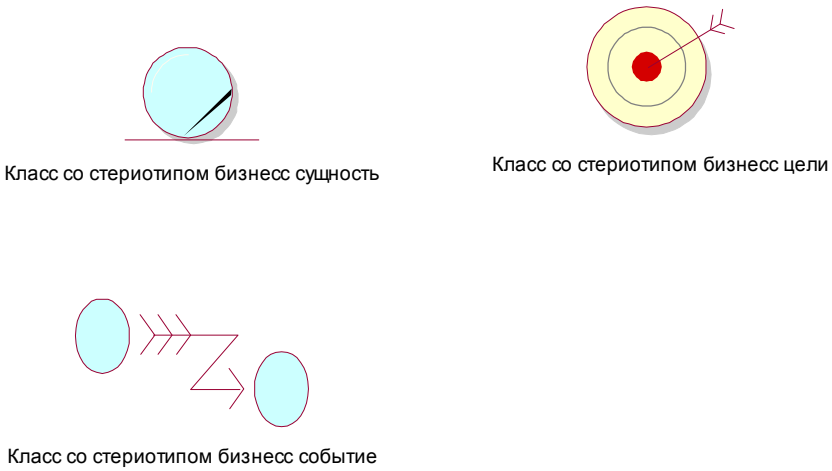


Рисунок 10.16– Пример стереотипов классов, используемых в Rose.

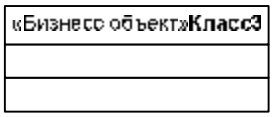


Рисунок 10.17 – Пример стереотипа

На рисунке 10.18 показан пример модели бизнес - объектов.

Модель поведения

Синтаксис и семантика элементов модели состояния

Представленная модель классов требуется для описания статической структуры, то есть структуры объектов и отношений в фиксированный момент времени. Модель состояний описывает последовательность операций, происходящих в системе в ответ на внешние воздействия. Модель состояний включает несколько диаграмм состояний (State machine diagram -диаграмма автоматов) по одной на каждый класс. Диаграмма состояний описывает возможный цикл объектов и состояний. Основными элементами диаграммы являются состояния, соединенные переходами.

Состояние - это такой период жизненного цикла объекта, когда он удовлетворяет определенным условиям. Некоторое событие может привести к переходу, в результате которого объект окажется в новом состоянии. При

переходе может выполняться действие или осуществляться деятельность, предписанная данному переходу.

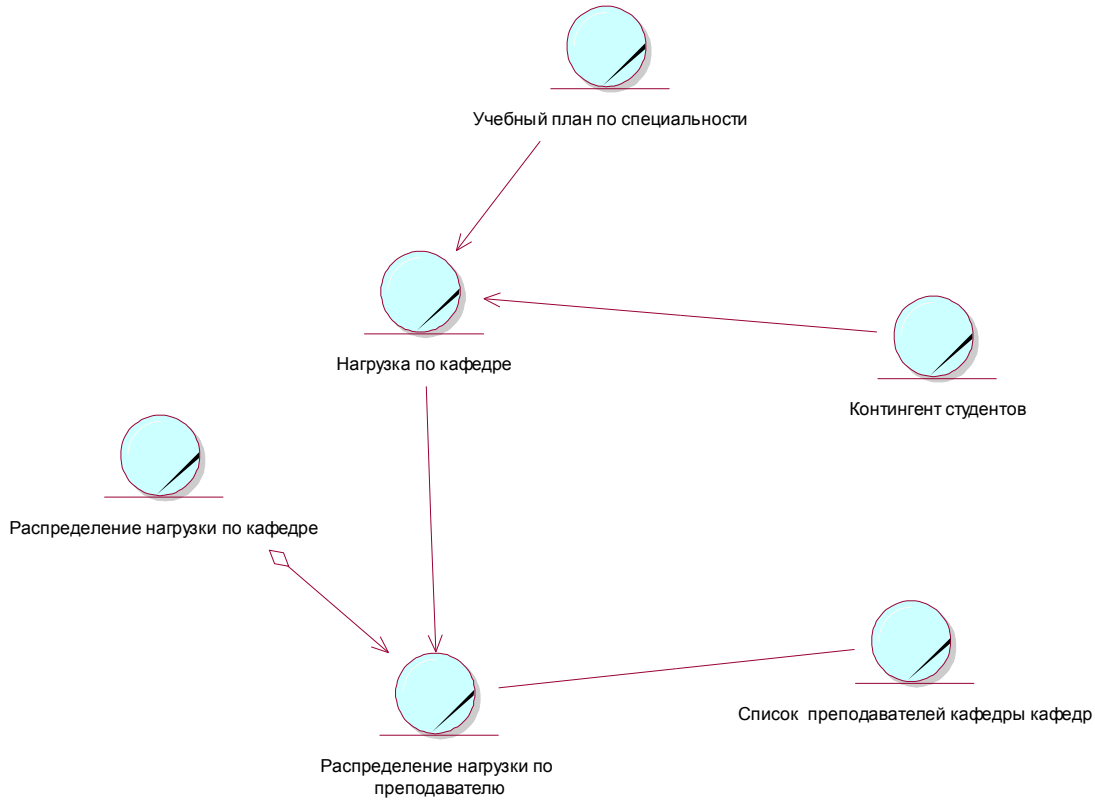


Рисунок 10.18 – Пример модели бизнес – объектов, используемых в процессе распределения нагрузки на кафедре.

Событие (event) – это происшествие, случившееся в определенный момент времени, например, подписание приказа о назначение на должность. События могут быть логически связаны друг с другом (предшествовать одно другому), могут быть не связаны, в этом случае они называются параллельными.

События можно разделить на события сигналов, события изменения и события времени.

Событие сигналов – это событие получения или отправки сигнала. Сигнал (signal) – это явная односторонняя передача информации от одного объекта другому. Сигнал отличается от вызова подпрограммы, т.к. вызов

может возвратить значение. Каждая передача сигнала это уникальное происшествие, которое можно сгруппировать в класс сигналов (рисунок 10.19).

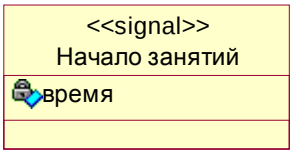


Рисунок 10.20 – Пример класса сигнала.

Событие изменения (change event) – это событие, вызванное выполнением логического выражения. В UML событие изменения обозначается ключевым словом *when* , за которым следует логическое выражение (например, *when* $T_{комнаты} > 35^0C$).

Событие времени (time event) – это событие, вызванное достижением момента времени либо в абсолютном измерении (ключевое слово *when*), либо в относительном временном интервале (ключевое слово *after*).

Состояние (state) – этап жизненного цикла объекта, во время которого оно удовлетворяет некоторому условию, выполняет определенную собственную деятельность или ожидает какого-либо события.

Объект класса обладает конечным числом возможных состояний. В каждый момент времени отдельный объект может находиться ровно в одном состоянии. Состояние объекта описывает отклик объекта на получаемые события. Событие во времени не имеет протяженности, а состояние занимает некоторый интервал времени между двумя точками событий. Состояние можно характеризовать: именем; описанием; событием, приводящим в данное состояние; как условие, характеризующее данное состояние; как событие и отклики объекта, возможные в данном состоянии.

Переход (transition) – это мгновенная смена одного состояния другим. Переход запускается, когда происходит связанное с ним событие. К переходу может быть прикреплена деятельность.

Сторожевое условие (guard condition) – это логическое выражение, которое должно быть истинным, чтобы разрешить выполнения перехода. Сто-

рожевое условие проверяется только один раз в тот момент, когда осуществляется событие, и если условие истинно, то переход выполняется (событие проверяется непрерывно).

Удовлетворение сторожевого условия запускает переход и приводит к выполнению деятельности. После этого исходное состояние перестает быть активным, а целевое, наоборот, становится активным. Продолжительная собственная деятельность, подлежащая выполнению, может быть связана с состоянием. Собственная деятельность может быть текущей, при входе и при выходе. Состояния могут быть простыми, композитными или состояния вложенного автомата. Событие в UML – стрелка, соединяющая исходное состояние с целевым (рисунок 10.21).

Часто назначением состояния является последовательное выполнение некоторой деятельности, когда деятельность завершается, запускается переход в следующее состояние. Такой переход называются - переход по завершению. Если состояние имеет переход по завершению, то сторожевое условие должно охватывать весь спектр возможных событий.

Синтаксис и семантика модели прецедентов

Прецедент – спецификация последовательностей действий (вариантов последовательностей и ошибочные последовательности), которые может осуществлять система, подсистема или класс, взаимодействуя с внешним актером.

Последовательность действий можно определить более строго. Прецедент - это описание множества последовательностей действий (включая их варианты), которые выполняются системой для того, чтобы актер получил результат, имеющий для него определенное значение. Последовательность действий должна быть важна для актера. Это описание действий системы с точки зрения актера, т.е. с точки зрения объекта окружающей среды, который взаимодействует с системой.

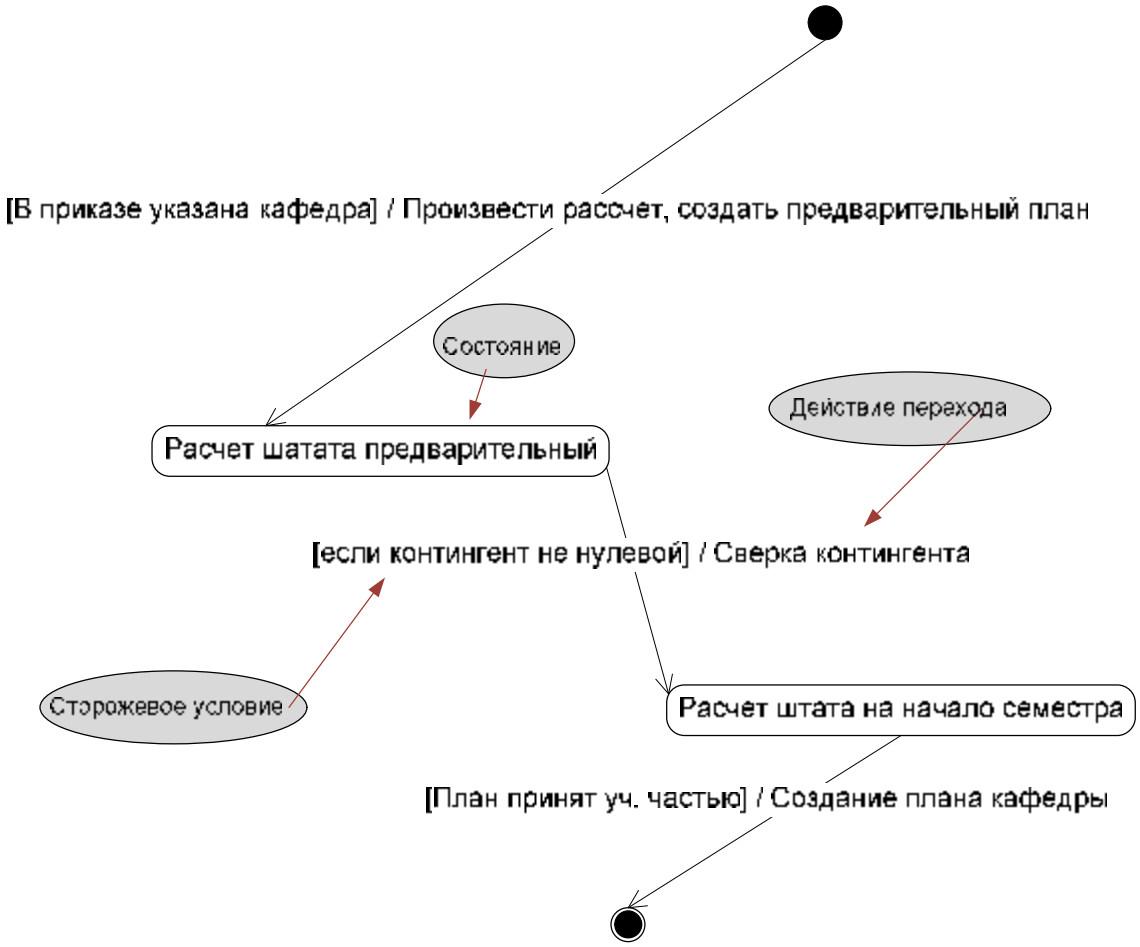


Рисунок 10.21 – Пример диаграммы состояния.

Актер (actor) – классификатор для сущности, находящийся вне некоторой системы, который взаимодействует с этим субъектом. Актер взаимодействует с элементом Use Case (прецедент) или согласованным набором прецедентов с целью воплощений особых намерений. Актер характеризует роль внешнего пользователя по отношению к системе. Это не значит, что актер должен быть человеком (рисунок 10.22).

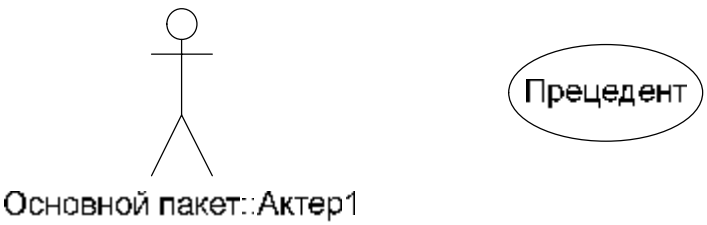


Рисунок 10.22 – Актер и прецедент

Спецификация прецедента должна иметь примерно следующую схему:

- название прецедента;
- краткое описание;
- потоки событий;
- основной поток;
- альтернативный поток1;
- альтернативный поток 2;
- альтернативный поток n;
- специальные требования;
- предусловия;
- постусловия;
- точки расширения.

На практике для именования прецедентов используют короткие глагольные фразы в активной форме, обозначающие некоторое поведение и взятые из словаря моделируемой системы.

Как правило, в начале работы потоки событий прецедента описывают в текстовой форме. По мере уточнения требований к системе будет удобнее перейти к графическому изображению потоков на диаграммах взаимодействия. Для описания основного потока прецедента обычно используют диаграмму последовательностей, а для альтернативных потоков - ее варианты.

Сценарий - это некоторая последовательность действий, иллюстрирующая поведение системы. Сценарии находятся в таком же отношении к прецедентам, как экземпляры к классам, то есть сценарий - это экземпляр прецедента.

Прецедент описывает желательное поведение системы, но в последствии, прецеденты необходимо реализовать. Для этого будет создан набор классов и других элементов, в результате совместной работы которых, будет достигнуто желаемое поведение. Такой набор классов, включая его динамическую и статическую структуру, называется в UML кооперацией (рисунок 10.23).

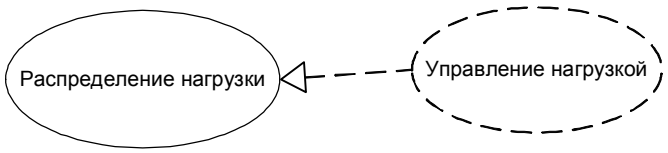


Рисунок 10.23 - Пример реализации прецедента с помощью кооперации.

На диаграмме показана кооперация[53] (collaboration), соединенная с прецедентом отношением реализации[54]. Основная задача системной архитектуры - нахождение минимального набора хорошо структурированных кооперации, реализующих определенный во всех прецедентах системы поток событий.

Диаграмма прецедентов включает не очень много элементов (рисунок 10.24). Между актером и прецедентом допускается только ассоциация. Ассоциация превращается в связь между экземпляром актера и экземпляром прецедентом, поэтому, иногда навигацию на ассоциации определяют, как направление инициализации взаимодействия.

Между актерами допускается отношение генерализации (обобщение). Благодаря этому отношению выделяют роль с общими чертами взаимодействия с прецедентами. Потомки будут реализовывать специфические элементы взаимодействия.

Между прецедентами допускается всего три типа отношений: генерализация, включение и расширение.

Генерализация позволяет описывать вариации базового прецедента, как и обобщение для классов. Прецедент, являющийся родителем, описывает общую последовательность поведения, а дочерние прецеденты к поведению

---

[53] Кооперация задает отношения контекста между экземплярами, которые взаимодействуют между собой для реализации определенной функциональности.

[54] Реализация – отношение между спецификацией и ее программной реализацией, указывает на то, что повеление наследуется без структуры.

предка добавляют новые элементы в его последовательность.. Прецедент предок может быть абстрактным или конкретным.

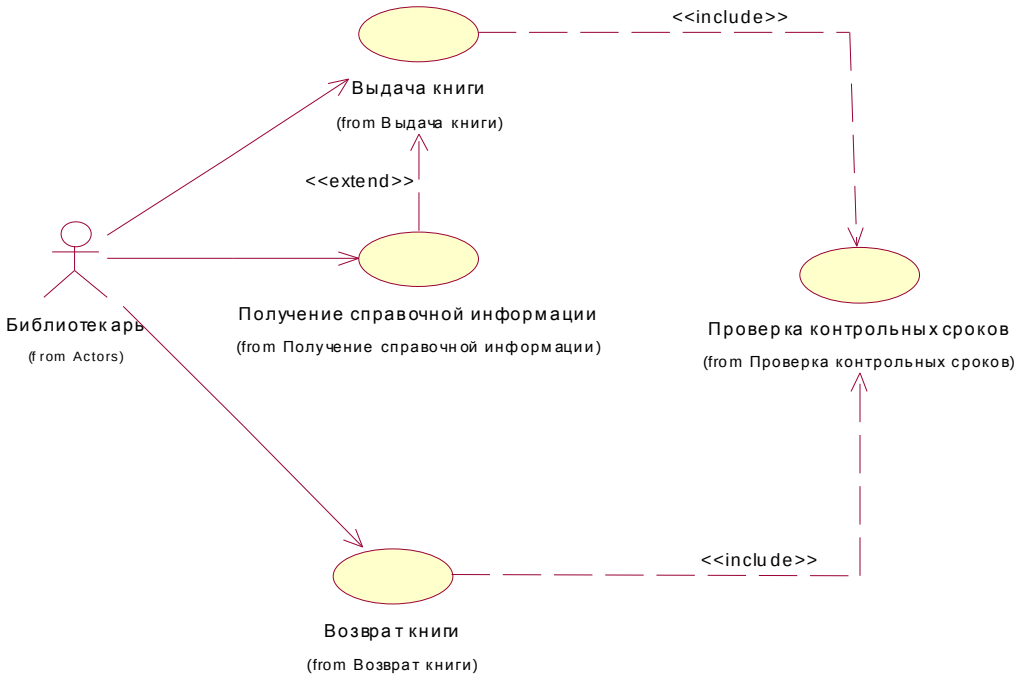


Рисунок 10.24 - Пример диаграммы прецедентов.

В работе /68/ не рекомендуется использовать в модели конкретные прецеденты в качестве родительских (рисунок 10.25).

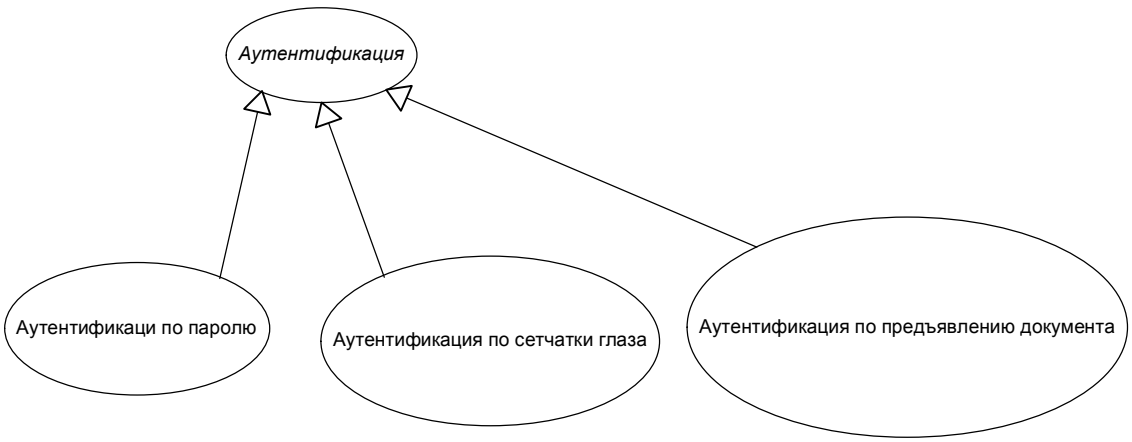


Рисунок 10.25 Пример использование обобщения.

Отношение включения (include) означает, что в некоторой точке базового прецедента инкорпорировано поведение другого прецедента. Включаемый прецедент никогда не существует автономно, а инстанцируется только как часть объемлющего прецедента. Благодаря наличию отношений включе-

ния удается избежать многократного описания одного и того же потока событий. Реализуется делегирование, при котором ряд обязанностей системы описывается в одном месте (во включаемом прецеденте), а остальные прецеденты, когда необходимо, включают эти обязанности в свой набор (рисунок 10.26).

Рисунок 10.26- Пример использования отношения включения.

Для обозначения включения используется пунктирная стрелка со стереотипом «include», направленная от исходного (включающего) прецедента к включаемому. В спецификации прецедента должны быть указаны точки включения.

Отношение расширения добавляет к варианту использования дополнительное поведение. Отношение расширения описывает ситуацию: определен базовый прецедент, затем к нему добавляют поведение. Отношение расширение применяется в том случае, если описывается последовательность действий, которые будут происходить крайне редко (рисунок 10.27). Направление стрелки «extend» противоположно стрелке «include», от прецедента - который расширяет поведение к прецеденту - у которого будет расширено поведение.

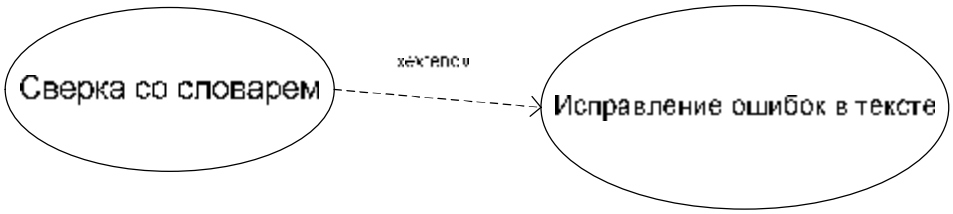


Рисунок 10.27 – Пример отношения расширения.

Для представления модели бизнес - прецедентов в Rational Rose используются стереотипы: business – actor, business – worker, business Use Case (рисунок 10.28).

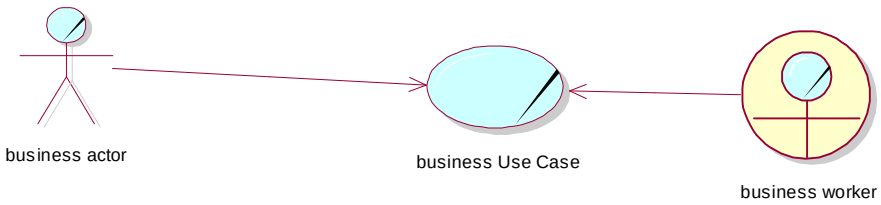


Рисунок 10.28 – Бизнес моделирование.

Понятно, что добавление «business» показывает, что речь идет о бизнес системе. «Business – actor» и «business – worker» имеют существенное концептуальное различие: актер - это элемент окружающей среды по отношению к моделируемой бизнес - системе (например, информационной системе), а worker является исполнителем системы, он будет актером по отношению к компьютерной системе. Вот, кстати, пример того, как терминологическая путаница, рассмотренная в главе «Терминология», может привести к различию в моделях прецедентов. Если считать, что в информационную систему входят люди (а мы именно так должны считать), то эти люди при моделировании информационной системы будут «business – worker». Если мы будем считать неправильно- люди не входят в информационную систему, то люди, работающие с компьютерной системой будут «Business – actor».

Синтаксис и семантика модели взаимодействия

Модель состояния показывает поведение одного объекта, а модель взаимодействия – поведение множества объектов.

Взаимодействие (interfction) –спецификация порядка передачи во времени сообщений между ролями (в контексте выполнения некоторых задач).

Взаимодействие задает определенный паттерн[55] поведения. Контекст обеспечивается классификатором или кооперацией. Внутри экземпляра взаимодействия объекты привязаны к своим ролям, а поток сообщений между ними соответствует спецификации.

В UML 2 взаимодействие значительно изменилось по сравнению с UML 1.5. Мы рассматриваем основные элементы взаимодействия, являющиеся подмножеством и той и другой версии.

Взаимодействие отображается с помощью диаграмм последовательности и диаграмм коммуникации. Поскольку эти две диаграммы изоморфны (одна диаграмма взаимно однозначно соответствует другой), мы в дальнейшем рассмотри только диаграмму последовательности.

В работе /68/ к модели взаимодействия отнесены диаграммы прецедентов и диаграммы деятельности. Действительно, на диаграмме прецедентов показывается взаимодействие актера с системой. На диаграмме деятельности показаны потоки управления и потоки данных. С их помощью документируется последовательность, необходимая для реализации операций.

В силу того, что взаимодействие это - устоявшийся термин, используемый в UML, мы придерживаемся того, что взаимодействие моделируется диаграммами последовательности и коммуникации.

Описание поведения каждого варианта использования требует, по крайне мере, одну диаграмму последовательности. Правильнее сказать, что каждый поток действий (альтернативные и главный), описанный в прецеденте, требует моделирования его на отдельной диаграмме последовательности.

Диаграмма последовательности изображается в виде двумерной схемы. По вертикали проходит временная ось (задает последовательность сообще-

---

[55] Паттерн - это шаблон кооперации. Паттерн – специализированная кооперация, представляющая собой множество параметризированных классификаторов, отношений и поведений, которые можно применить в различных ситуациях путем привязывания ролям паттерна элементов модели (обычно этими элементами модели являются классы).

ний), а по горизонтали располагаются отдельные роли[56] кооперации. Каждой роли предоставлена линия жизни (пунктирная вертикальная линия), показывающая время существования роли, а также время выполнения процедур объекта (вертикальная двойная линия). Сверху колонки помещается прямоугольник. Если объект существовал с самого начала взаимодействия, то прямоугольник изображается в верхней чисти диаграммы. В противном случае прямоугольник размещается у стрелки (сообщение для действия создания). Имя роли задается как *имя[селектор[57]]:Тип*. Каждое сообщение выглядит как стрелка (асинхронное сообщение – обычная стрелка, синхронная – с закрашенным наконечником). Возврат управления обозначается пунктирной стрелкой. Хвост стрелки может заканчивать активацию роли (двойная черта на линии жизни прерывается) или обозначает уничтожение объекта (линия жизни заканчивается символом «Х»). При вызове того же самого объекта на двойную линию, обозначающую активность объекта, накладывается другая двойная линия (активность второго объекта). В UML2.0 синтаксис и семантика диаграммы последовательности существенно расширены, но при этом не все инструменты поддерживают новые элементы модели. Пример диаграммы последовательности приведен на рисунке 10.29.

Синтаксис и семантика деятельности

Деятельность- это спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчиненных элементов – вложенных видов деятельности и, в конечном счете, отдельных действий, соединенных между собой потоками, которые идут от выходов одного узла к входу другого. Деятельность может вызываться дейст-

---

[56] Роль – составной элемент структурного классификатора, описывающий появление экземпляра (или множества экземпляров) в контексте, определяемом структурой классификатора. Экземпляр кооперации не является владельцем объектов, появляющихся в его контексте, он просто ссылается на них, а они могут присутствовать в разных кооперациях.

[57] Селектор - это выражение, которое позволяет выделить элемент из множества.

виями и являться составной частью другого вида поведения. Она моделируется в виде графа узлов деятельности, соединенных управляющими и объектными потоками. Деятельность имеет входные и выходные параметры, представляющие данные, которые предаются перед выполнением и после выполнения.

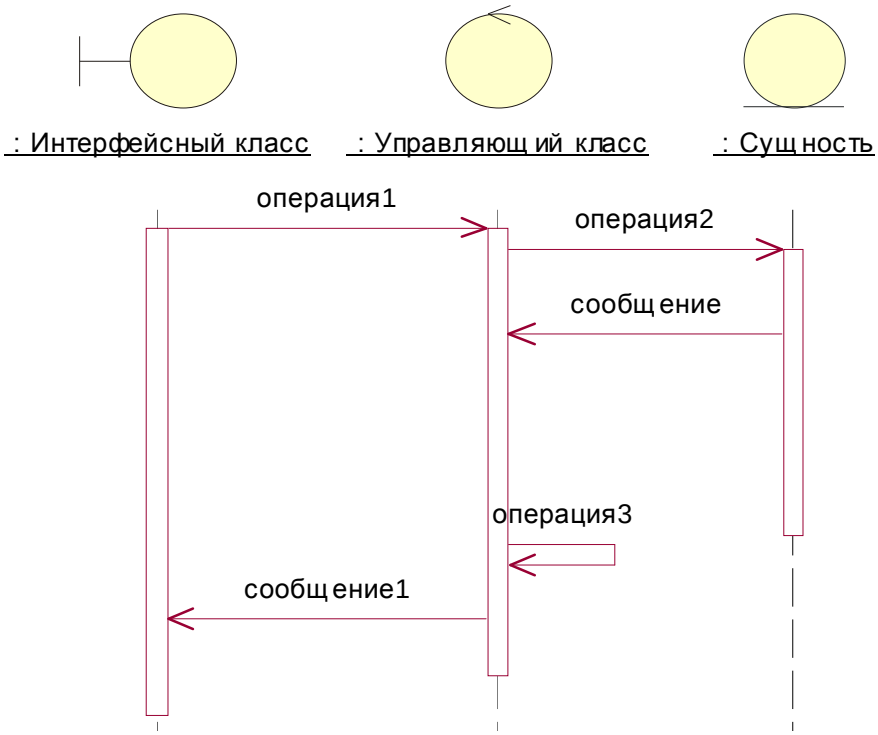


Рисунок 10.29 – Пример диаграммы последовательности.

Структура видов деятельности и их семантика основаны на свободной интерпретации сетей Петри. Процесс выполнения можно представить себе, как обход экземпляра графа этой деятельности, содержащего маркеры. Маркеры указывают на наличие управления или данных на некотором этапе процесса вычисления, представленном узлом или потоком. Маркеры управления не имеют внутренней структуры. Маркеры данных содержат значения, представляющие промежуточные значения на некотором этапе вычисления. Граф может содержать параллельно выполняемые вычисления. В каждый момент времени может существовать несколько маркеров.

Действие (в нотации используется узел действий или состояние действий, рисунок 10.30) может быть выполнено, если во всех входных управляющих потоках присутствует маркер, а на всех контактах действия присутствуют данные.

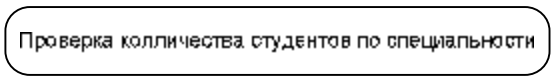


Рисунок 10.30 – Действие.

Когда действие начинается, то оно поглощает маркер. Когда действие заканчивается, то на все его выходные контакты выдаются выходные значения, на все управляющие ребра выдаются маркеры управления. Действие считается «быстрым», т.е. система тратит на него ничтожно малое время. Если действие прерывается досрочно, то это не оказывает влияния на состояние системы. Некоторые виды действий допускают аварийное завершение. Элементами деятельности являются различные управляющие конструкции. Каждый вид узлов имеет свои собственные правила выполнения, которые описаны в Словаре терминов. Используются следующие узлы управления: разветвление, объединение, разделение, слияние, начало, окончание деятельности и окончание потока.

Разветвление (DecisionNode). Узел разветвления имеет одно входное ребро и множество выходных. Обычно для выходного ребра задано сторожевое условие. Когда на входном ребре появляется маркер, он передается в то выходное ребро, сторожевое условие в котором истинно. Даже если истинным является сторожевое условие в нескольких ребрах, маркер передается на одно ребро. Если ни одно ребро не удовлетворяет сторожевому условию, модель построена неправильно. Разветвление сочетается с последующим объединением для формирования условного элемента (рисунок 10.31) или с предыдущим объединением для организации цикла (рисунок 10.32).

Рисунок 10.31 – Пример использования разветвления и объединения для организации условного элемента.

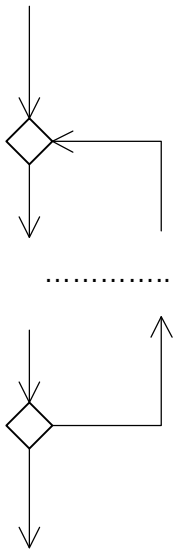


Рисунок 10.32 – Пример использования разветвления и объединения для организации цикла.

Объединение (MergeNode) обратно разветвлению. Когда маркер появляется на одном из входных ребер, то инициируется и выходное ребро.

Разделение (ForkNode) – узел, который имеет одно входное и множество выходных ребер. Обычно разделение сочетается с последующим слиянием

(JoinNode). Тем самым организуется параллельное выполнение. При появлении маркера на входном ребре инициируются т маркеры на всех выходных ребрах. На узле слияния, когда маркеры присутствуют на всех входных ребрах, то инициируется маркер на выходном ребре. В простых ситуациях использование слияния и разделения эквивалентно применению нескольких выходных и входных ребер на действиях (рисунок 10.33).

У начального (InitialNode) узла нет входного ребра, но есть одно выходное, в то время, как у конечно узла (ActivityFinal) могут быть несколько входных и ни одного выходного. Появление маркера на одном ребре деятельности прекращает всю деятельность.

Узел окончания потока имеет одно или несколько входных ребер и не имеет выходных. Узел поглощает все маркеры на входных ребрах.

На диаграммах могут быть показаны объекты, которые в свою очередь могут участвовать в объектных потоках. Объектный поток отличается от потока управления только присутствием данных в объектном потоке.

Для того, что бы диаграмму разбить на группы по различным критериям используются разделы. Каждая группа будет представлять какую-то ответственность за действия и объекты этой группы (рисунок 10.34). Группы иногда называются плавательными дорожками (swimlanes).

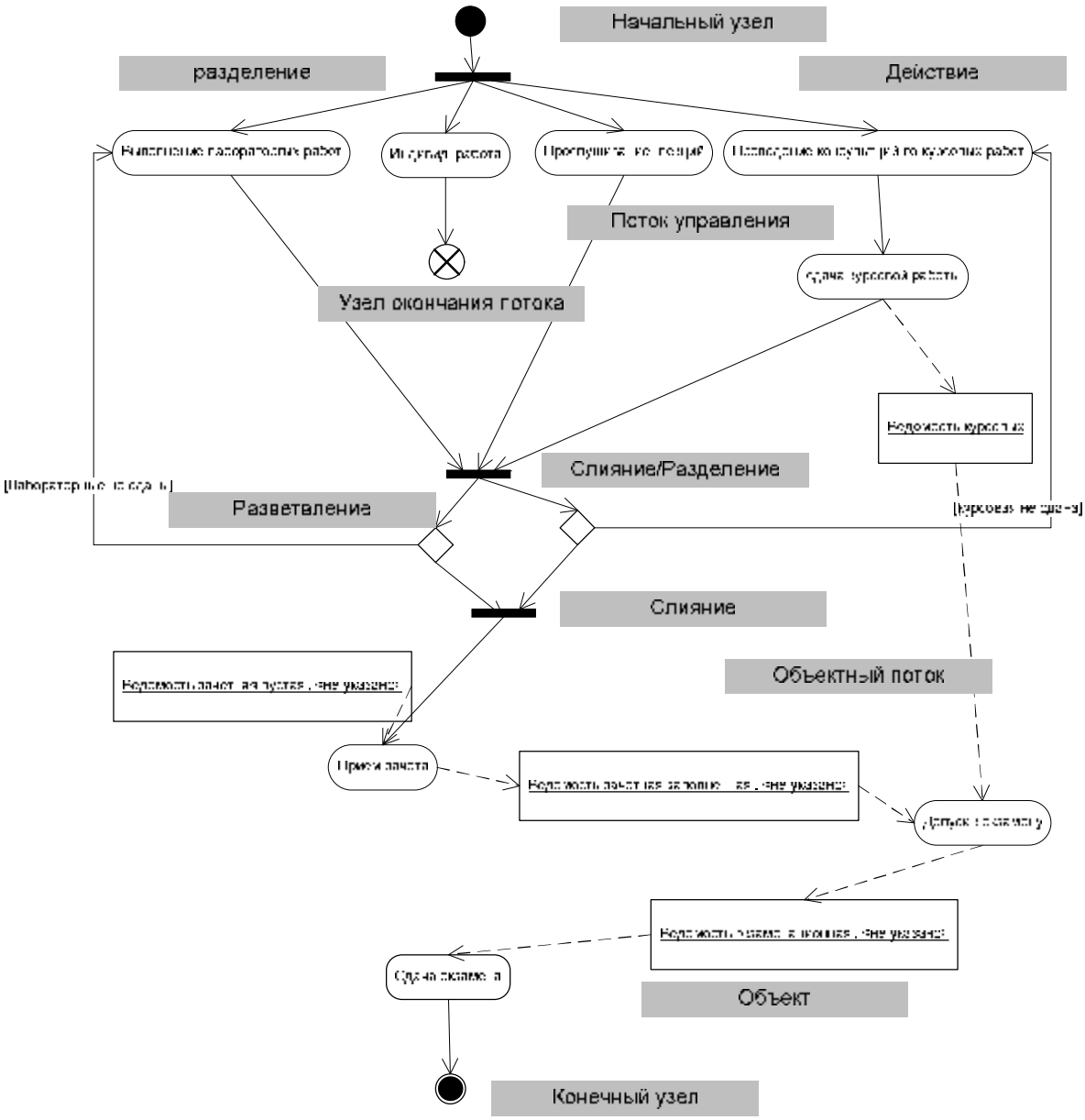


Рисунок 10.33 – Пример диаграммы деятельности (серым цветом отмечены названия элементов диаграммы).

Деятельность подобна конечному автомату в том отношении, что оба понятия описывают последовательности состояний, возникающих с течением времени, и условий, которые вызывают изменение в наборе состояний. Различие между ними состоит в том, что конечный автомат описывает состояния объекта, выполняющего вычисления, или состояния предмета вычисления. Деятельность задает состояния вычисления, как такового, и явным образом моделирует управляющие и информационные потоки между узлами деятельности.

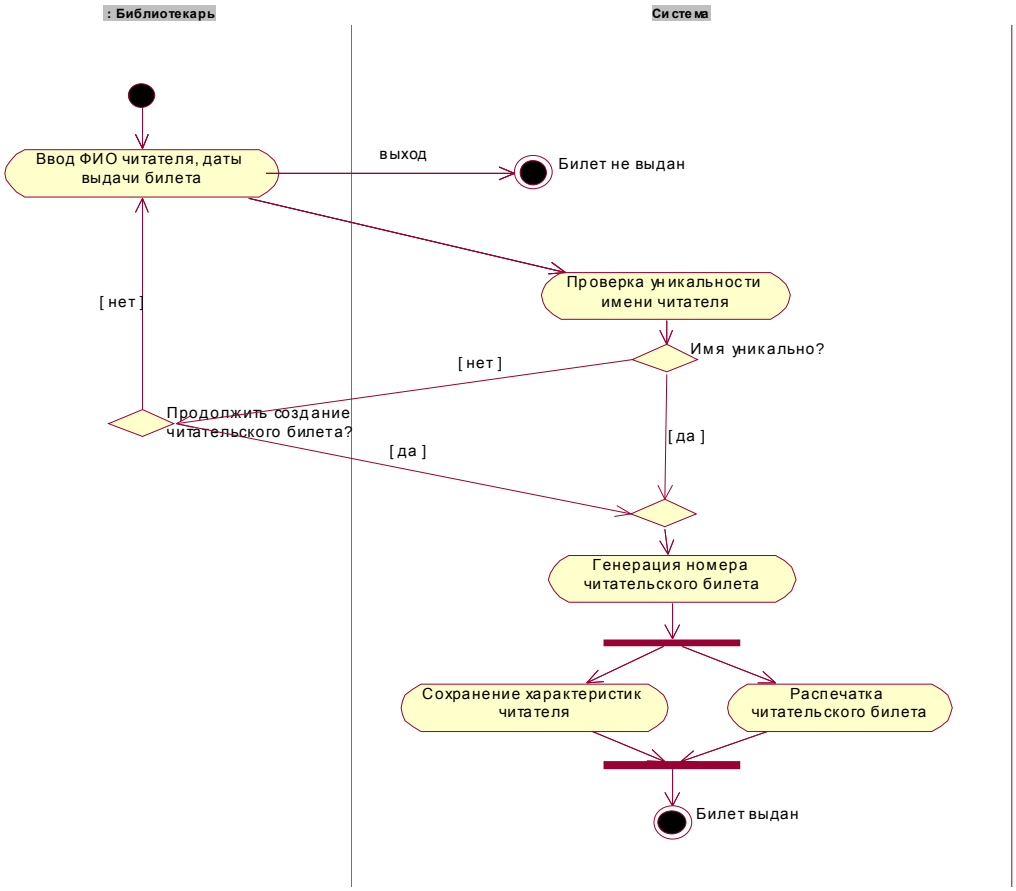


Рисунок 10.34 – Пример диаграммы действий с разделами (с плавательными дорожками).

Другие виды диаграмм

Спецификация UML определяется рядом документов, опубликованных object Management Group (http://www.omg.org). Язык UML определяется двумя спецификациями UML2.0 Infrastructure и UML2.0 Superstructure. В документе инфраструктуре изложены основные концепции, которые используются другими спецификациями (MOF Metamodel-Object Specification – спецификация метаобъектов, CWM Common Warehous Metadata –хранилище метаданных).

Сверхструктура UML определяет язык с точки зрения его использования. Документ состоит из вводной части, трех основных разделов и семи приложений, объединенных в четвертый раздел. Первый раздел посвящен

статическим конструкциям языка (рисунок 10.35), а второй раздел - динамическим, поведенческим конструкциям (рисунок 10.36).

Модель UML состоит из элементов типа пакетов, классов и ассоциаций. UML диаграммы – часть модели в форме графического представления. Они содержат графические элементы (узлы, ребра), которые представляют элементы модели UML. Каждая диаграмма имеет область содержания. Она может помещаться в рамку, изображенную на рисунке 10.37.

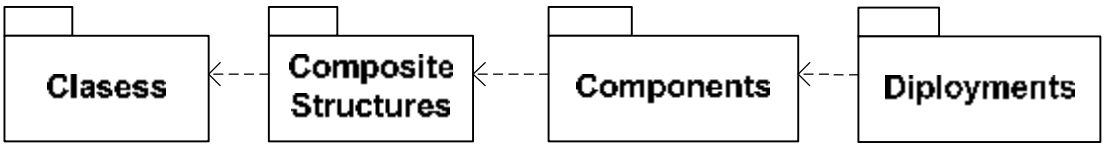


Рисунок 10.35 - UML пакеты, которые поддерживают структурное моделирование.

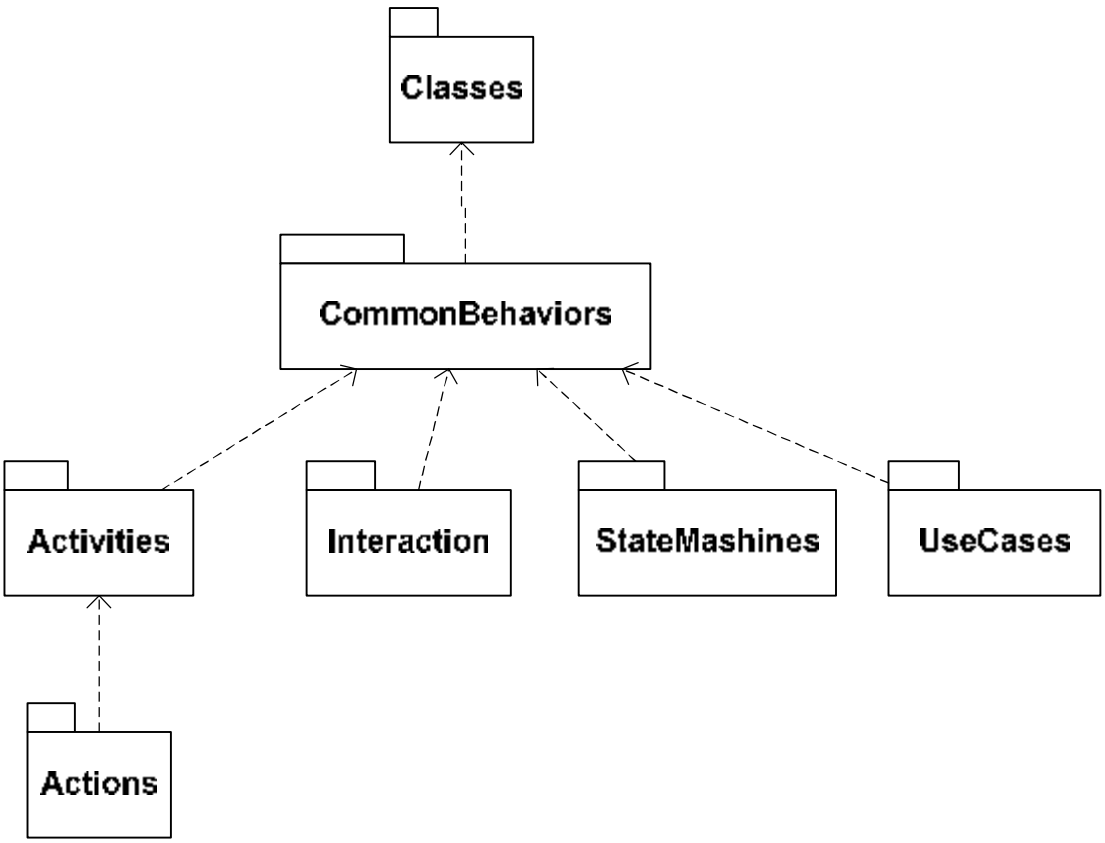


Рисунок 10.36 - UML пакеты, которые поддерживают поведенческое моделирование.

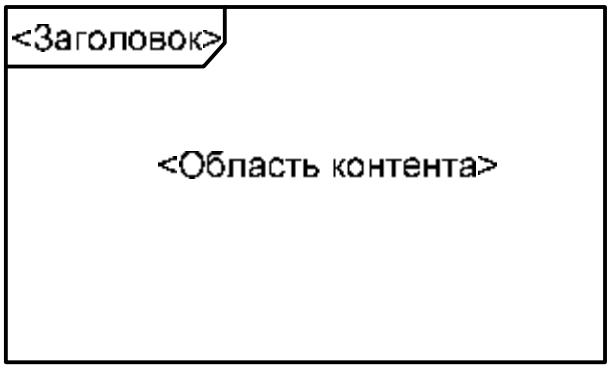


Рисунок 10.37 – Рамка для представления диаграмм UML.

На рисунке10.38 изображена классификация UML диаграмм, приведенных в Суперструктуре.

Структурные диаграммы показывают статическую структуру объектов в системе. Они изображают те элементы спецификации, которые являются независимыми от времени. Элементы в структурных диаграммах представляют значащие концепции применения, и могут включать классы объекты и пакеты.

Поведенческие диаграммы показывают динамическое поведение объектов системы, включая их методы, кооперации, действия и историю. Данная классификация дает возможность структурировать основные диаграммы, однако никто не запрещает использовать комбинацию различных диаграмм, в том числе и структурные, и поведенческие диаграммы. Следовательно, границы между различными типами диаграмм строго не регламентируется.

В данном параграфе мы коротко опишем следующие диаграммы: диаграммы кооперации, диаграммы коммуникации, диаграммы развертывания и диаграмма обзора взаимодействий.

Диаграммы кооперации одно из представлений композитной структуры. Кооперация - это контекстное отношение множества объектов, выполняющих совместную работу для достижения некоторой цели. Кооперация определяет набор взаимодействующих участников, которые необходимы для выполнения ее задач. Роли кооперации будут сыграны отдельными объекта-

ми при взаимодействии друг с другом. Кооперация представлена примером диаграммы, приведенной на рисунке 10.39.

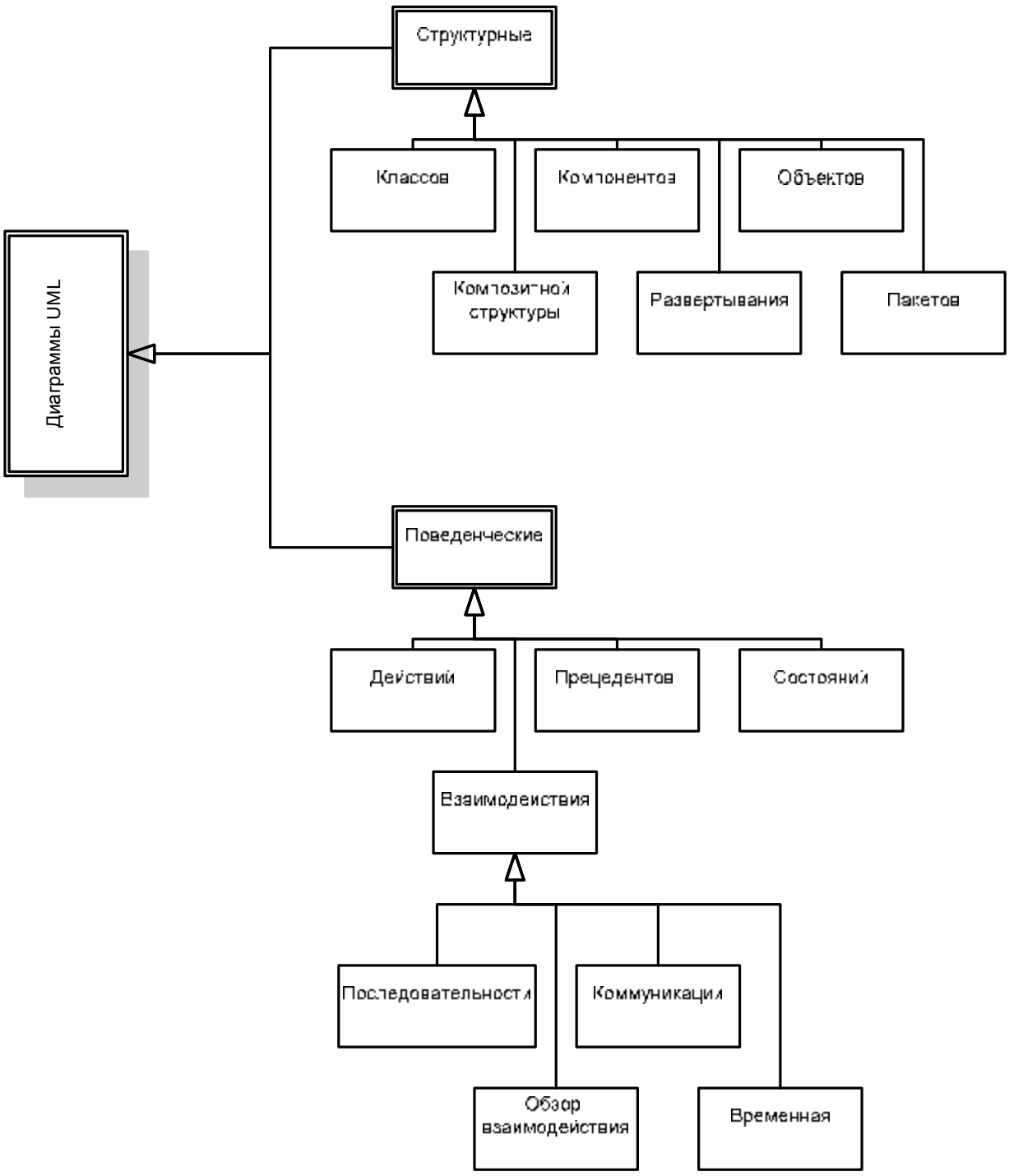


Рисунок 10.38 – Классификация диаграмм UML

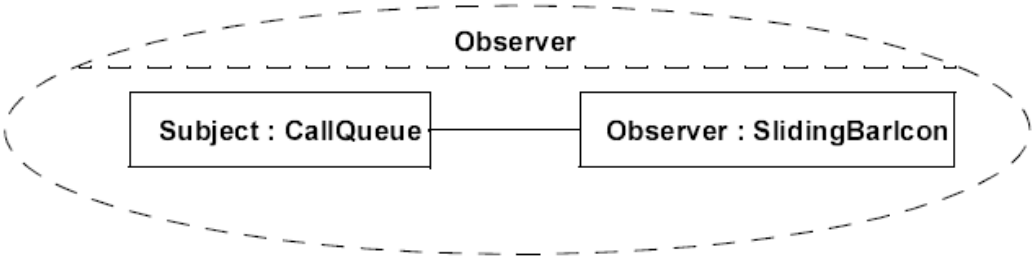


Рисунок 10.39 – диаграмма кооперации.

Диаграмма коммуникации показывает роли, участвующие во взаимодействии с описанием жизненной истории объекта. Одно из предназначений диаграмм коммуникации - показать реализацию какой-либо операции. При реализации поведения последовательность сообщения на диаграмме коммуникации соответствуют структуре вложенных вызовов и прохождению сигналов в программе. В диаграмме коммуникации главная задача - геометрически отобразить отношение между ролями и связывание сообщений с соединителями.

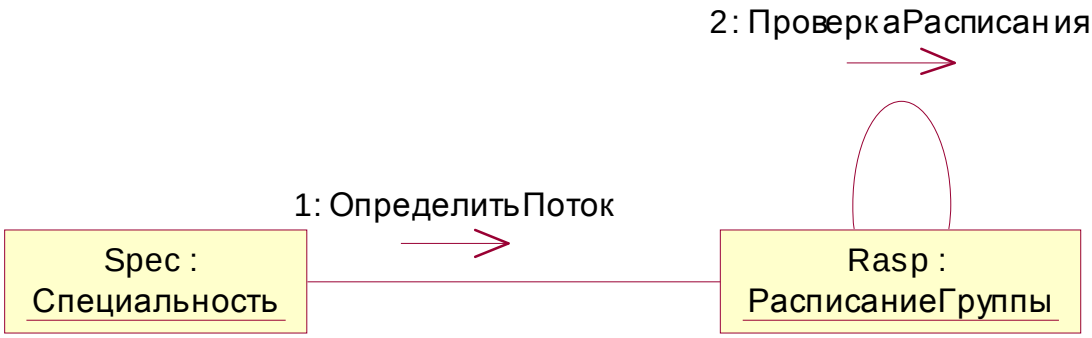


Рисунок 10.38 – Пример диаграммы классификации.

Диаграмма развертывания представляет артефакты[58] времени выполнения на узлах. Узел – это ресурс, используемый во время выполнения программы (компьютер, память и т.д.). Данная модель (рисунок 10.39) дает возможность оценить последствия принятого варианта поставки и размещения ресурсов.

[58] Артефакты - физический элемент информации, используемого или порождаемого в процессе разработки программного обеспечения. В данном случае это может быть компонент или файл, или база данных, или драйвер и т.д.

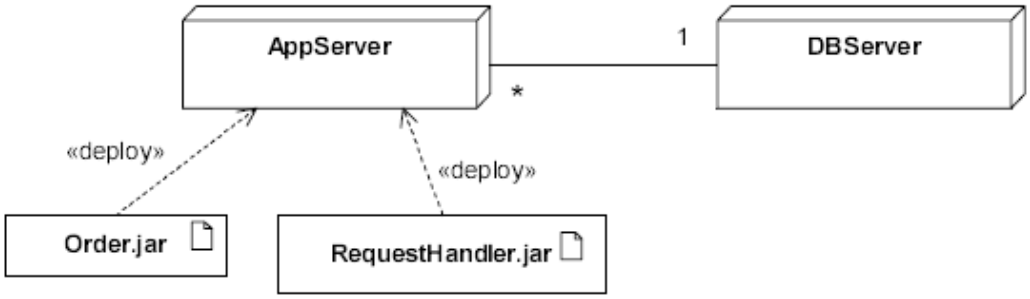


Рисунок 10.39 - Пример диаграммы развертывания.

Диаграмма обзора взаимодействия

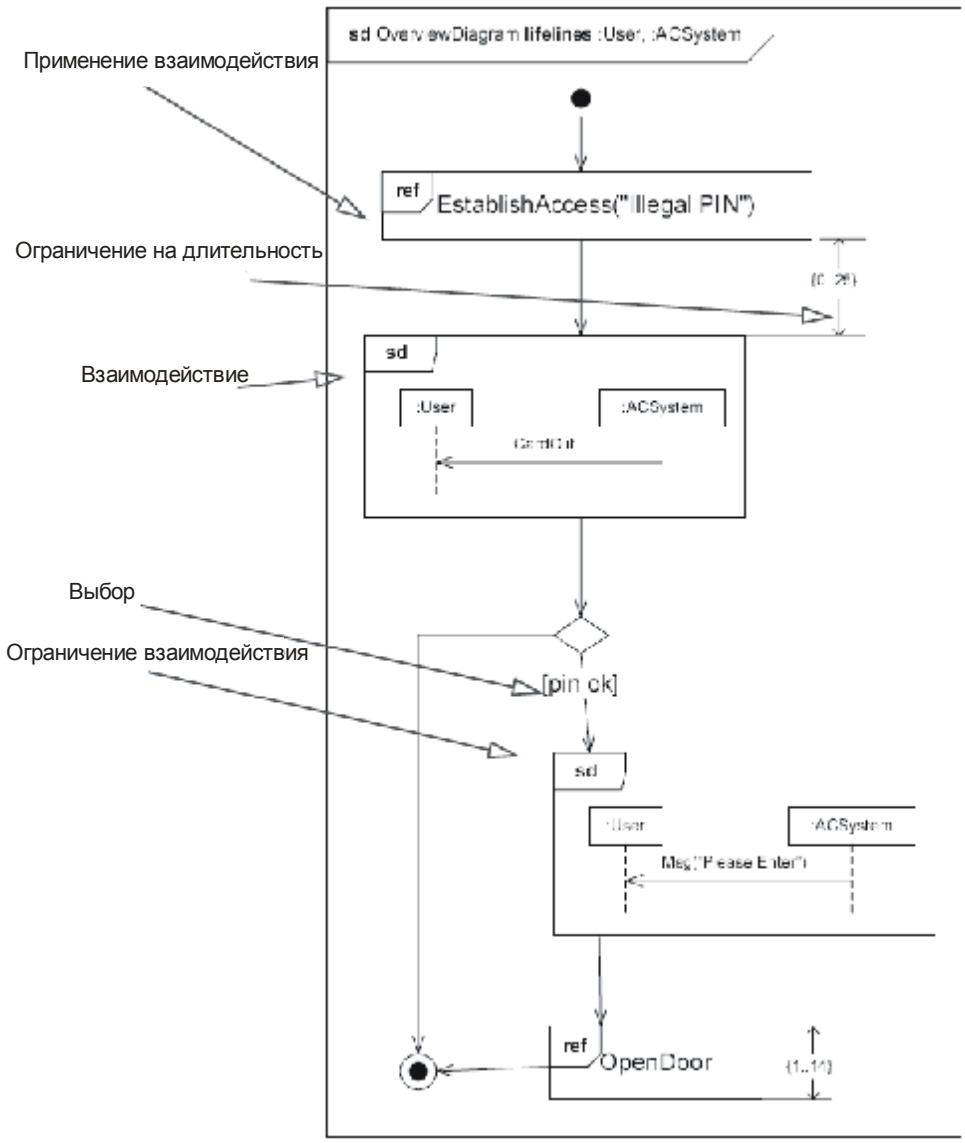


Рисунок 10.40 - Диаграмма обзора взаимодействия.

Диаграмма обзора взаимодействия (рисунок 10.40) представляет взаимодействие через вариант диаграммы деятельности, на которой представлен краткий обзор потоков управления. На диаграмме акцентируется внимание на описании потока управления, соединяющих узлах взаимодействия и реализации взаимодействия (InteractionUses). Жизненный цикл и сообщения не показываются на этом уровне.

Контрольные вопросы

1. Что отличает модель статическую от модели динамической в UML?
2. Что понимают под динамическим моделированием в UML и в чем отличие от моделирования поведения?
3. Что отличает модель от диаграммы в UML? Приведите пример, когда модель не ассоциирована с конкретной диаграммой.
4. Есть ли отличие в семантике одних и тех же элементов, изображенных на различных диаграммах?
5. Что общего в элементах модели UML актере и классе?
6. Что отличает понятия класс и классификатор?
7. Есть ли классификаторы, не имеющие графического элемента в нотации UML? Приведите примеры.
8. Можно ли UML считать методологией проектирования информационных систем?
9. Что отличает статические модели UML от функциональных моделей?
10. Приведите пример модели UML, которая может отвечать на теже вопросы, что и модель IDEF0?
11. Какая UML модель, по Вашему мнению, может использоваться при моделировании данных?

Список использованных источников

1. Большая советская энциклопедия. http://www.rubricon.com/
2. Википедии — свободная энциклопедия. http:// ru.wikipedia.org/
3. Руководство к Своду знаний по управлению проектами. Третье издание. Американский национальный стандарт ANSI/PMI 99-001-2004
4. ISO/TR 10006: 1997 (E). Quality Management - Guidelines to quality in project management - p. 1.
5. Стадии проектирования. ЕСКД ГОСТ 2.103-68
6. Воройский, Ф.С. Информатика. Энциклопедический систематизированный словарь-справочник. (Введение в современные информационные и телекоммуникационные технологии в терминах и фактах). - М.: 2007.
7. ГОСТ 19.001-77 ЕСПД. Общие положения.
8. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
9. ГОСТ 19.102-77 ЕСПД. Стадии разработки.
10. ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содер-жанию и оформлению.
11. Зиндер, Е. З. Соотнесение и использование стандартов организации жизненных циклов систем //Системы управления базами данных – 1997- №03
12. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения.
13. ГОСТ 34.601-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. (Взамен ГОСТ 24.601-86, ГОСТ 24.602-86)
14. ГОСТ 34.602-89 Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
15. ГОСТ Р ИСО/МЭК 12207-99 "Информационная технология. Процессы жизненного цикла программных средств"
16. ГОСТ Р ИСО/МЭК ТО 15271-2002 "Информационная технология. Руководство по применению ГОСТ Р ИСО/МЭК 12207 (Процессы жизненного цикла программных средств)"
17. IEEE Std 1074-1997 IEEE Standard for Developing Software Life Cycle Processes
18. ГОСТ Р ИСО/МЭК 15288—2005. Информационная технология. Системная инженерия. Процессы жизненного цикла систем.
19. Модель зрелости процессов разработки программного обеспечения. /Марк Паулк и др./ М., Богородский печатник, 2002.- 256 стр.
20. ISO/IEC TR 15504-1 TR «Information technology - Software process assessment - Part 1: Concepts and introductory guide
21. ISO/IEC TR 15504-2 TR «Information technology - Software process assessment - Part 2: A reference model for processes and process capability
22. ISO/IEC TR 15504-3 TR «Information technology - Software process assessment - Part 3: Performing an assessment
23. ISO/IEC TR 15504-4 TR «Information technology - Software process assessment - Part 4: Guide to performing assessments
24. ISO/IEC TR 15504-5 TR «Information technology - Software process assessment - Part 5: An assessment model and indicator guidance
25. ISO/IEC TR 15504-6 TR «Information technology - Software process assessment - Part 6: Guide to competency of assessors ISO/IEC TR 15504-7 TR «Information technology - Software process assessment - Part 7: Guide for use in process improvement

26. ISO/IEC TR 15504-8 TR «Information technology - Software process assessment - Part 8: Guide for use in determining supplier process capability
27. ISO/IEC TR 15504-9 TR «Information technology - Software process assessment - Part 9: Vocabulary
28. ISO/IEC 15504-1:2004 Information technology -- Process assessment -- Part 1: Concepts and vocabulary
29. ISO/IEC 15504-2:2003 Information technology -- Process assessment -- Part 2: Performing an assessment
30. ISO/IEC 15504-3:2004 Information technology -- Process assessment -- Part 3: Guidance on performing an assessment
31. ISO/IEC 15504-4:2004 Information technology -- Process assessment -- Part 4: Guidance on use for process improvement and process capability determination
32. ISO/IEC 15504-5:2006 Information technology -- Process Assessment -- Part 5: An exemplar Process Assessment Model
33. Оценка и аттестация зрелости процессов создания и сопровождения программных средств и информационных систем (ISO/IEC TR 15504 CMM) / Пер. с англ. А.С. Агапова, С.В. Зенина, Н.Э. Михайловского, А.А. Мкртумяна. М.: Книга и бизнес, 2001.-348 с.
34. Capability Maturity Model® Integration (CMMISM), Version 1.1. CMMISM for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1). Continuous Representation. CMU/SEI-2002-TR-011 ESC-TR-2002-011
35. Capability Maturity Model® Integration (CMMISM), Version 1.1. CMMISM for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1). Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1). Staged Representation. CMU/SEI-2002-TR-012 ESC-TR-2002-012
36. Capability Maturity Model® Integration (CMMISM), Version 1.1. CMMISM for Software Engineering (CMMI-SW, V1.1). Continuous Representation. CMU/SEI-2002-TR-028 ESC-TR-2002-028
37. Capability Maturity Model® Integration (CMMISM), Version 1.1. CMMISM for Software Engineering (CMMI-SW, V1.1). Staged Representation CMU/SEI-2002-TR-029 ESC-TR-2002-029
38. Capability Maturity Model® Integration (CMMISM), Version 1.1. CMMISM for Systems Engineering, Software Engineering, and Integrated Product and Process Development (CMMI-SE/SW/IPPD, V1.1). Continuous Representation. CMU/SEI-2002-TR-003 ESC-TR-2002-003
39. Capability Maturity Model® Integration (CMMISM), Version 1.1. CMMISM for Systems Engineering, Software Engineering, and Integrated Product and Process Development (CMMI-SE/SW/IPPD, V1.1). Staged Representation CMU/SEI-2002-TR-004 ESC-TR-2002-004
40. Capability Maturity Model® Integration (CMMISM), Version 1.1. CMMISM for Systems Engineering and Software Engineering (CMMI-SE/SW, V1.1). Continuous Representation. CMU/SEI-2002-TR-001 ESC-TR-2002-001
41. Capability Maturity Model® Integration (CMMISM), Version 1.1. CMMISM for Systems Engineering and Software Engineering (CMMI-SE/SW, V1.1). Staged Representation CMU/SEI-2002-TR-002 ESC-TR-2002-002
42. CMMI® for Development, Version 1.2 (CMMI-DEV, V1.2). CMU/SEI-2006-TR-008 ESC-TR-2006-008
43. Липаев, В.В. Модели зрелости программной инженерии – CMMI. Содержание и применение. Информационный бюллетень № 6 (157). - 2006

44. The Standish Group International The Standish Group International, Inc., Extreme Chaos, 2000 - http://www1.standishgroup.com//sample_research/PDFpages/extreme_chaos.pdf
45. Фатрел, Р., Шафер, Д., Шафер, Л. Управление программными проектами: достижение оптимального качества при минимуме затрат. – М.: Издательский дом «Вильямс», 2004. – 1136 с.
46. ANSI/PMI 99-001-2004 Руководство к Своду знаний по управлению проектами Третье издание. (Руководство PMI BOK)
47. Буч, Г., Якобсон, А., Рамбо, Дж. UML. Классика CS. -2-е изд., [пер. с англ.]; под общ. ред. проф. С. Оролова – СПб: Питер, 2006.- 736с.
48. http://en.wikipedia.org/wiki/V-Model_%28software_development%29
49. http://en.wikipedia.org/wiki/Spiral_model
50. Boehm, B. W.A spiral model of software development and enhancement Computer Volume 21, Issue 5, May 1988 Page(s):61 – 72
51. http://en.wikipedia.org/wiki/Integrated_Computer-Aided_Manufacturing
52. Хансен, Г., Хансен, Д.. Базы данных: разработка и управление.- М.: ЗАО «издательство БИНОМ», 1999 – 704 с.
53. P. Chen The Entity-Relationship Model-Toward a Unified View of Data. ACM Transactions on Database Systems, Vol. 1, No. 1. March 1976, Pages 9-36.
54. Конноли, Т., Брег, К., Страчан, А., Базы данных: проектирования реализация и сопровождения. Теория и практика, 2-е изд.- М.: Издательский «Вильямс», 2001.- 1120с.
55. Вендров, А.М.CASE-технологии. Современные методы и средства проектирования информационных систем. http://www.citforum.ru/database/case/index.shtml
56. Ожегов, С.И., Шведов, Н.Ю. Толковый словарь русского языка - 2-е издание – М.: ФЗЪ, 1995.-938с.
57. Малый энциклопедический словарь Брокгауза и Ефрона. http://slovari.yandex.ru/dict/brokminor.
58. David A. Marca, Clement L. McGowan Sadt: Structured Analysis and Design Techniques I- Mcgraw-Hill (Tx) - 1987 ISBN-13: 978-0070402355. 398 с. (русский перевод http://www.interface.ru/home.asp?artId=1852)
59. ГОСТ Р 50.1.028—2001 «Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования»
60. http://www.idef.com/IDEF0.html
61. Ed Yourdon. Just Enough Structured Analysis http://www.yourdon.com/jesa/jesa.php
62. Калашян, А.Н., Калянов, К.Н. Структурные модели бизнеса: DFD-технология Под ред.Г.Н.Калянова.-М.:Финансы и статистика, -2003.-256с.
63. Калянов, Г.Н. Консалтинг при автоматизации предприятий (подходы, методы, средства) // М.: СИНТЕГ, -1997, -316с.
64. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем: Учебник М.: Финансы и статистика ISBN 527902144X 2002 -352 с. (http://mirknig.com/2007/09/26/vendrov_a_m_proektirovanie_programmnogo_obespechenija_jekonomicheskikh_informacionnykh_system.html)
65. IEEE Std 610.12-1990 IEEE, Standard Glossary of Software Engineering Terminology
66. Лингвистический энциклопедический словарь/ ред. Ярцева В.Н., Издание 2-е, доп., ISBN: 5-85270-239-0 - М.: Большая Российская энциклопедия. 2002 – 709с.
67. Integration Definition For Information Modeling (IDEF1X). http://www.idef.com/IDEF1X.html
68. Рамбо, Дж., Блаха, М. UML2.0. Объектно-ориентированное моделирование и разработка. 2-е изд. – Спб.:Питер, -2007. – 544 с.

69. Буч, Г., Якобсон, А., Рамбо, Дж. UML. Классика CS. 2-е изд. – Спб.:Питер, -2006. – 736с.
70. Буч, Г., Якобсон, А., Рамбо, Дж. Язык UML. Руководство пользователя. 2-е изд. – М:ДМК Пресс СПб.: Питер, 2004.-432с.
71. Фаулер, М., Кендалл. С. UML. Основы – Спб.:Символ плюс, 2002- 192 с.
72. UML 2.0 Superstructure Specificationversion 2.0. ptc/04-10-02 (convenience document) October 8, 2004. http//www.omg.com
73. Нейбург, Эрик, Максимчук, Дж., Роберт, А. Проектирование баз данных с помощью UML.: Пер. с англ. — М.: Издательский дом "Вильяме", 2002. — 288 с.: ил. ISBN 5-8459-0355-6

Приложение А

Требования к оформлению курсовых и дипломных работ*

1 Структурные элементы отчета

Структурными элементами отчета о НИР являются:

- **титульный лист (рисунок А1)**;
- **реферат**;
- содержание;
- определения;
- обозначения и сокращения;
- **введение**;
- **основная часть**;
- **заключение**;
- список использованных источников;
- приложения.

Обязательные структурные элементы выделены полужирным шрифтом. Остальные структурные элементы включают в отчет по усмотрению исполнителя НИР с учетом требований разделов 5 и 6.

2 Требования к содержанию структурных элементов отчета

2.1 Реферат

Реферат должен содержать: сведения об объеме отчета, количестве иллюстраций, таблиц, приложений, количестве частей отчета, количестве использованных источников; перечень ключевых слов; текст реферата. Текст реферата должен отражать: объект исследования или разработки; цель работы; метод или методологию проведения работы; результаты работы; технико-эксплуатационные характеристики; итоги внедрения; область применения; экономическую эффективность или значимость работы; прогнозные предположения о развитии объекта исследования. Если отчет не содержит сведений по какой-либо из перечис-

* требования выполнены в соответствии с ГОСТ 7.32-2001 Отчет о научно-исследовательской работе. Структура и правила оформления

ленных структурных частей реферата, то в тексте реферата они опускаются.

2.2 Содержание

Содержание включает введение, наименование всех разделов, подразделов, пунктов (если они имеют наименование), заключение, список использованных источников и наименование приложений с указанием номеров страниц, с которых начинаются эти элементы отчета.

2.3 Определения

Перечень определений начинают со слов: “В настоящем отчете применяют следующие термины с соответствующими определениями”.

2.4 Обозначения и сокращения

Структурный элемент “Обозначения и сокращения” содержит перечень обозначений и сокращений, применяемых в данном документе. Запись обозначений и сокращений проводят в порядке приведения их в тексте отчета с необходимой расшифровкой и пояснениями.

2.5 Введение

Введение должно содержать оценку современного состояния решаемой научно-технической проблемы, основание и исходные данные для разработки темы, обоснование необходимости проведения проекта. Во введении должны быть показаны актуальность и новизна темы, связь данной работы с другими работами.

2.6 Основная часть должна содержать:

а) выбор направления исследований, включающий обоснование направления исследования, методы решения задач и их сравнительную оценку, описание выбранной общей методики;

б) процесс теоретических и (или) экспериментальных исследований, включая определение характера и содержания теоретических исследований, методы исследований, методы расчета, обоснование необходимости проведения экспериментальных работ, принципы действия разработанных объектов, их характеристики;

в) обобщение и оценку результатов исследований, включающих оценку полноты решения поставленной задачи и предложения по дальнейшим направлениям работ, оценку достоверности полученных результатов и их сравнение с аналогичными результатами отечественных и зарубежных работ, обоснование необходимости проведения дополнительных исследований, отрицательные результаты, приводящие к не-

обходимости прекращения дальнейших исследований.

2.7 Заключение

Заключение должно содержать:

- краткие выводы по результатам выполнений работы или отдельных ее этапов;

- оценку полноты решений поставленных задач;

- разработку рекомендаций и исходных данных по конкретному использованию результатов работы;

- оценку технико-экономической эффективности внедрения;

- оценку научно-технического уровня выполненной работы в сравнении с лучшими достижениями в данной области.

2.8 Список использованных источников

Список должен содержать сведения об источниках, использованных при составлении отчета. Сведения об источниках приводятся в соответствии с требованиями ГОСТ 7.1.

2.9 Приложения

5.12.1 В приложения рекомендуется включать материалы, связанные с выполненной работой, которые по каким-либо причинам не могут быть включены в основную часть. В приложения могут быть включены:

- промежуточные математические доказательства, формулы и расчеты;

- таблицы вспомогательных цифровых данных;

- протоколы испытаний;

- инструкции, методики, разработанные в процессе выполнения работы;

- иллюстрации вспомогательного характера;

- копии технического задания, программы работ или другие исходные документы для выполнения;

- акты внедрения результатов работы и др.

3 Правила оформления отчета

3.1 Общие требования

Отчет о работе должен быть выполнен на одной стороне листа белой бумаги формата А4 через полтора интервала. Цвет шрифта должен быть черным, высота букв, цифр и других знаков — не менее 1,8 мм (кегль не менее 12).

Текст отчета следует печатать, соблюдая следующие размеры полей: правое — 10 мм, верхнее — 20 мм, левое и нижнее — 20 мм.

Сокращение русских слов и словосочетаний в отчете — по ГОСТ 7.12.

3.2 Построение отчета

Основную часть отчета следует делить на разделы, подразделы и пункты. Пункты, при необходимости, могут делиться на подпункты. При делении текста отчета на пункты и подпункты необходимо, чтобы каждый пункт содержал законченную информацию.

Разделы, подразделы, пункты и подпункты следует нумеровать арабскими цифрами и записывать с абзацного отступа. Разделы должны иметь порядковую нумерацию в пределах всего текста, за исключением приложений.

Номер подраздела или пункта включает номер раздела и порядковый номер подраздела или пункта, разделенного точкой. После номера раздела, подраздела, пункта и подпункта в тексте точку не ставят. Если текст отчета подразделяют только на пункты, их следует нумеровать, за исключением приложений, порядковыми номерами в пределах всего отчета. Если раздел или подраздел имеет только один пункт, или пункт имеет один подпункт, то нумеровать его не следует.

Пример

1 Типы и основные размеры

1.1
1.2 } ***Нумерация пунктов первого раздела документа***
1.3

2 Технические требования

2.1
2.2 } ***Нумерация пунктов второго раздела документа***
2.3

Внутри пунктов или подпунктов могут быть приведены перечисления.

Пример

а) _____

б) _____

1) _____

2) _____

в) _____

Разделы, подразделы должны иметь заголовки. Пункты, как правило, заголовков не имеют. Заголовки должны четко и кратко отражать содержание разделов, подразделов. Заголовки разделов, подразделов и пунктов следует печатать с абзацного отступа, с прописной буквы без точки в конце, не подчеркивая. Если заголовок состоит из двух предложений, их разделяют точкой.

3.3 Нумерация страниц отчета

Страницы отчета следует нумеровать арабскими цифрами, соблюдая сквозную нумерацию по всему тексту отчета. Номер страницы проставляют в центре нижней части листа без точки. Титульный лист включают в общую нумерацию страниц отчета. Номер страницы на титульном листе не проставляют. Нумерация страниц отчета и приложений, входящих в состав отчета, должна быть сквозная.

3.4 Иллюстрации

Иллюстрации (чертежи, графики, схемы, компьютерные распечатки, диаграммы, фотоснимки) следует располагать в отчете непосредственно после текста, в котором они упоминаются впервые, или на следующей странице.

Иллюстрации, за исключением иллюстрации приложений, следует нумеровать арабскими цифрами сквозной нумерацией. Если рисунок один, то он обозначается “Рисунок 1”. Слово “рисунок” и его наименование располагают посередине строки. Допускается нумеровать иллюстрации в пределах раздела. В этом случае номер иллюстрации состоит из номера раздела и порядкового номера иллюстрации, разделенного точкой. Например, Рисунок 1.1. Иллюстрации, при необходимости, могут иметь наименование и пояснительные данные (подрисуночный текст). Слово “Рисунок” и наименование помещают после пояснительных данных и располагают следующим образом: Рисунок 1 — Детали прибора.

Иллюстрации каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения. Например, Рисунок А.3.

При ссылках на иллюстрации следует писать “... в соответствии с рисунком 2”.

3.5 Таблицы

На все таблицы должны быть ссылки в отчете. При ссылке следует писать слово “таблица” с указанием ее номера.

Пример оформления таблицы приведен на рисунке А1.

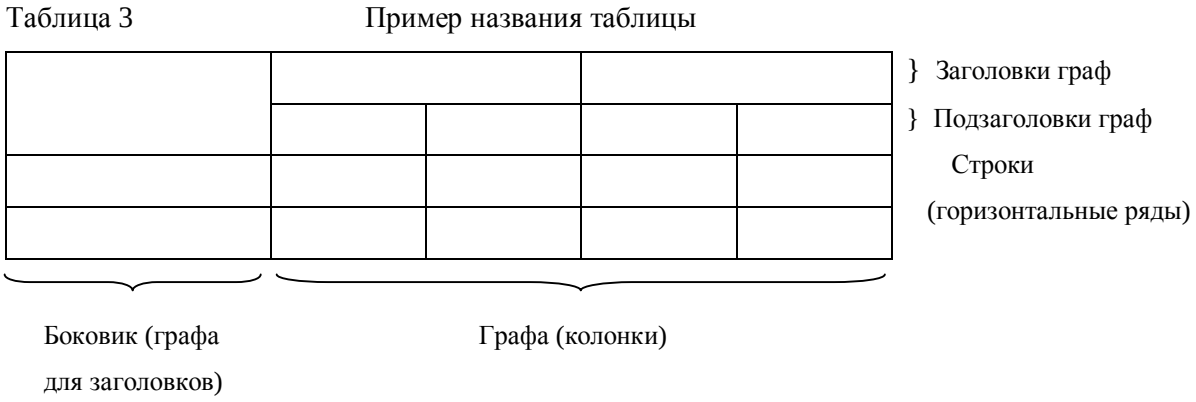


Рисунок А1

3.6 Примечания

Примечания приводят в документах, если необходимы пояснения или справочные данные к содержанию текста, таблиц или графического материала. Примечания не должны содержать требований.

Если примечание одно, то после слова “Примечание” ставится тире и примечание печатается с прописной буквы. Одно примечание не нумеруют. Несколько примечаний нумеруют по порядку арабскими цифрами без проставления точки. Примечание к таблице помещают в конце таблицы над линией, обозначающей окончание таблицы.

Пример

Примечание -_____

_____

Несколько примечаний нумеруются по порядку арабскими цифрами.

Пример

Примечания

1_____

2_____

3_____

3.7 Формулы и уравнения

Уравнения и формулы следует выделять из текста в отдельную строку. Выше и ниже каждой формулы или уравнения должно быть оставлено не менее одной свободной строки. Если уравнение не умещается в одну строку, то оно должно быть перенесено после знака равенства (=) или после знаков плюс (+), минус (-), умножения (х), деления (:), или других математических знаков, причем знак в начале следующей строки повторяют. При переносе формулы на знаке, символизирующем операцию умножения, применяют знак “Х”.

Пояснение значений символов и числовых коэффициентов следует приводить непосредственно под формулой в той же последовательности, в которой они даны в формуле. Формулы в отчете следует нумеровать порядковой нумерацией в пределах всего отчета арабскими цифрами в круглых скобках в крайнем правом положении на строке.

Пример

A=a:Ь, (1)

B=c:e. (2)

Ссылки в тексте на порядковые номера формул дают в скобках. Пример –... в формуле (1).

Допускается нумерация формул в пределах раздела. В этом случае номер формулы состоит из номера раздела и порядкового номера формулы, разделенного точкой, например (3.1).

3.8 Список использованных источников

Сведения об источниках следует располагать в порядке появления ссылок на источники в тексте отчета и нумеровать арабскими цифрами без точки и печатать с абзацного отступа. Оформление библиографическое описание осуществляется согласно ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. Ниже приводится краткое описание правил библиографического описания.

3.9 Приложения

6.14.1 Приложение оформляют как продолжение данного документа на последующих его листах или выпускают в виде самостоятельного документа.

Приложения обозначают заглавными буквами русского алфавита, начиная с А,

за исключением букв Ё, З, Й, О, Ч, Ь, Ы, Ъ. После слова “Приложение” следует буква, обозначающая его последовательность.

4. Примеры библиографического описания

Однотомные издания

Перроун, П. Д. Создание корпоративных систем на базе Java 2 Enterprise Edition [Текст] : рук. разработчика : [пер. с англ.] / Поль Дж. Перроун, Венката С. Р. «Кришна», Р. Чаганти. - М. [и др.] : Вильямс, 2001. - 1179 с.

Агафонова, Н. Н. Гражданское право [Текст] : учеб. пособие для вузов / Н. Н. Агафонова, Т. В. Богачева, Л. И. Глушкова ; под. общ. ред. А. Г. Калпина ; авт. вступ. ст. Н. Н. Поливаев ; М-во общ. и проф. образования РФ, Моск. гос. юрид. акад. - Изд. 2-е, перераб. и доп. - М. : Юристъ, 2002. - 542 с.

«Воспитательный процесс в высшей школе России», межвузовская науч.-практическая конф. (2001 ; Новосибирск). Межвузовская научно-практическая конференция «Воспитательный процесс в высшей школе России», 26-27 апр. 2001 г. [Текст] : [посвящ. 50-летию НГАВТ : материалы] / редкол.: А. Б. Борисов [и др.]. - Новосибирск : НГАВТ, 2001. - 157 с.

История России [Текст] : учеб. пособие для студентов всех специальностей / В. Н. Быков [и др.] ; отв. ред В. Н. Сухов ; М-во образования Рос. Федерации, С.-Петерб. гос. лесотехн. акад. - 2-е изд., перераб. и доп. / при участии Т. А. Суховой. - СПб. : СПбЛТА, 2001. - 231 с. - ISBN 5-230-10656-5.

Отдельный том

Казьмин, В. Д. Справочник домашнего врача [Текст]. В 3 ч. Ч. 2. Детские болезни / Владимир Казьмин. - М. : АСТ : Астрель, 2002. - 503 с.

Законодательные материалы

Российская Федерация. Конституция (1993). Конституция Российской Федерации [Текст] : офиц. текст. - М. : Маркетинг, 2001. - 39 с.; - ISBN 5-94462-025-0.

Правила

Правила безопасности при обслуживании гидротехнических сооружений и гидромеханического оборудования энергоснабжающих организаций [Текст] : РД 153-34.0-03.205-2001: утв. М-вом энергетики Рос. Федерации 13.04.01 : ввод. в действие с 01.11.01. - М. : ЭНАС, 2001. -158 с. ; 22 см. - В надзаг.: …РАО «ЕЭС России». - 5000 экз. - ISBN 5-93196-091-0.

Стандарты

ГОСТ Р 517721-2001. Аппаратура радиоэлектронная бытовая. Входные и выходные параметры и типы соединений. Технические требования [Текст]. - Введ. 2002-01-01. - М. : Изд-во стандартов, 2001. -IV, 27 с.

Сборник стандартов

Система стандартов безопасности труда : [сборник]. - М. : Изд-во стандартов, 2002. - 102 с.

Патентные документы

Пат. 2187888 Российская Федерация, МПК7 H 04 B 1/38, H 04 J 13/00. Приемопередающее устройство [Текст] /Чугаева В. И. ; заявитель и патентообладатель Воронеж. науч.-ислед. ин-т связи. - № 2000131736/09 ; заявл. 18.12.00 ; опубл. 20.08.02, Бюл. № 23 (II ч.). - 3 с.

А. с. 1007970 СССР, МКИЗ B 25 J 15/00. Устройство для захвата неориентированных деталей типа валов [Текст] / В. С. Ваулин, В. Г. Ке-майкин (СССР). - № 3360585/25-08 ; заявл. 23.11.81 ; опубл. 30.03.83, Бюл. № 12. -2с.

Промышленные каталоги

Оборудование классных комнат общеобразовательных школ [Текст] : каталог / М-во образования РФ, Моск. гос. пед. ун-т. - М. : МГПУ, 2002. - 235 с.

Отдельный том

Казьмин, В. Д. Справочник домашнего врача [Текст]. В 3 ч. Ч. 2. Детские болезни / Владимир Казьмин. - М. : АСТ : Астрель, 2002. - 503, [1] с.

Многотомные издания

Казьмин, В. Д. Справочник домашнего врача [Текст]. В 3 ч. Ч. 2. Детские болезни / Владимир Казьмин. - М. : АСТ : Астрель, 2002. - 503, [1] с.

Депонированные научные работы

Разумовский, В. А. Управление маркетинговыми исследованиями в регионе [Текст] / В. А. Разумовский, Д. А. Андреев ; Ин-т экономики города. - М., 2002. -210 с.: схемы. - Библиогр.: с. 208-209. - Деп. в ИНИ-ОН Рос. акад. наук 15.02.02, № 139876.

Отчеты о научно-исследовательской работе

Формирование генетической структуры стада [Текст] : отчет о НИР (промежуточ.) : 42-44 / Всерос. науч.-исслед. ин-т животноводства ; рук. Попов В. А. ; исполн.:

Алешин Г. П. [и др.]. - М., 2001. - 75 с. - Библи-огр.: с. 72-74. - № ГР 01840051145. - Инв. № 04534333943.

Диссертации

Вишняков, И. В. Модели и методы оценки коммерческих банков в условиях неопределенности [Текст] : дис …. канд. экон. наук : 08.00.13 : защищена 12.02.02 : утв. 24.06.02 / Вишняков Илья Владимирович. - М., 2002. - 234 с. - Библиогр.: с. 220-230. - 04200204433.

Журнал

Актуальные проблемы современной науки [Текст] : информ.-аналит. журн. / учредитель ООО «Компания «Спутник +». - 2001, июнь - . -М. : Спутник +, 2001- . - Двухмес. - ISSN 1680-2721

Электронные ресурсы

Художественная энциклопедия зарубежного классического искусства [Электронный ресурс]. - Электрон. текстовые, граф., зв. дан. и прикладная прогр. (546 Мб). - М. : Большая Рос. энцикл. [и др.], 1996.

Статья

Двинянинова, Г. С. Комплимент : Коммуникативный статус или стратегия в дискурсе [Текст] / Г. С. Двинянинова // Социальная власть языка : сб. науч. тр. / Воронеж. межрегион. ин-т обществ. наук, Воронеж, гос. ун-т, Фак. романо-герман. истории. - Воронеж, 2001. - С. 101-106.

Боголюбов, А. Н. О вещественных резонансах в волноводе с неоднородным заполнением [Текст] / А. Н. Боголюбов, А. Л. Делицын, М. Д. Малых // Вестн. Моск. ун-та. Сер. 3, Физика. Астрономия. - 2001. - № 5. - С. 23-25.

Приложение Б

Требования к выполнению лабораторной работы №1(7семестр) по курсу «Проектирование информационных систем» на тему «Методология функционального моделирования IDEF0».

1 Назначение лабораторной работы

а) Сформировать у студентов представление о сущности функционального подхода к моделированию на примере задачи анализа бизнес-процессов в заданной предметной области.

б) Способствовать в освоении основных положений федерального стандарта США «FIPS 183. Integration Definition for Function Modeling (IDEF0)» и рекомендаций по стандартизации Госстандарта России «Р 50.1.028-2001. Методология функционального моделирования».

2 Цель лабораторной работы

В результате выполнения работы студенты должны подтвердить знание следующих элементов методологии IDEF0:

- синтаксис графического языка;
- семантика языка;
- правила построения графических диаграмм модели.

3 Требования к оформлению отчёта

Оформление отчёта по лабораторной работе должно удовлетворять общим требованиям к текстовым документам, представленным в ГОСТ 7.32-2001.

Отчёт должен иметь следующую структуру:

- титульный лист;
- содержание;
- основная часть.

4 Требования к содержанию отчёта

Основная часть отчёта должна содержать следующие разделы и подразделы:

а) Описание предметной области на естественном языке.

б) Модель IDEF0 предметной области.

1) Цель создания модели.

2) Границы моделирования.

3) Точка зрения разработчика.

4) Целевая аудитория.

5) Глоссарий.

6) Графические диаграммы (дерево узлов модели, контекстная диаграмма, диаграммы декомпозиции (не менее двух уровней), иллюстрации (FEO-diagrams)).

4.1 Описание предметной области на естественном языке

Описание предметной области на естественном языке должно представлять собой связный текст и может включать в себя таблицы и рисунки. В описание должны быть включены все бизнес-процессы, которые предполагается подвергнуть функциональному анализу. В описании важно раскрыть характеристики этих процессов: определения основных участников, последовательности реализующих процедур, организационные и случайные факторы, влияющие на их протекание и т.д.

Описание на естественном языке служит основой для процесса формализации предметной области, реализуемого в ходе построения IDEF0-модели. Модель не может содержать каких-либо аспектов предметной области, не зафиксированных в данном описании.

Раздел описания не может быть озаглавлен как «Описание предметной области». Заголовок должен отражать контент описания. Примеры правильных заголовков: «Описание процессов учёта товаров на складе компании «Тюменская

продуктовая компания», «Описание порядка начисления зарплаты в АО «ТюменьПромРесурс».

4.2 Модель IDEF0

Данный раздел должен содержать основные компоненты модели IDEF0, построенные в ходе функционального анализа предметной области: цели и границы моделирования, определения точки зрения разработчика и целевой аудитории, глоссарий предметной области и набор графических диаграмм.

Каждый компонент должен быть оформлен в виде отдельного подраздела отчёта. В зависимости от сложности предметной области и целей проводимого анализа, отчёт может не содержать определений точки зрения разработчика и целевой аудитории или может быть дополнен подразделами с текстовыми комментариями к диаграммам.

4.2.1 Цель создания модели

Формулировка цели выражает причину создания модели, то есть содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет её структуру.

4.2.2 Границы моделирования

Границы моделирования предназначены для обозначения ширины охвата предметной области и глубины детализации и являются логическим продолжением уже определенного назначения модели. Как читающий модель, так и непосредственно ее автор, должны понимать степень детальности ответов на поставленные в назначении модели вопросы.

4.2.3 Точка зрения разработчика

Под точкой зрения понимается перспектива, с которой наблюдалась система при построении модели. Точка зрения выбирается таким образом, чтобы учесть уже обозначенные границы моделирования и назначение модели. Однажды выбранная точка зрения остается неизменной для всех элементов модели. При необходимости могут быть созданы другие модели, отображающие систему с

других точек зрения. Приведем несколько примеров точек зрения при построении моделей: клиент, поставщик, владелец, редактор.

4.2.4 Целевая аудитория

Целевая аудитория определяет множество людей, для нужд которых создается модель. Зачастую от этого зависит уровень детализации, с которым должна создаваться модель. Перед построением модели необходимо иметь представление о том, какие сведения о предмете моделирования уже известны, какие дополнительные материалы и/или техническая документация для понимания модели могут быть необходимы для целевой аудитории, какие язык и стиль изложения являются наиболее подходящими.

4.2.5 Глоссарий

Глоссарий предназначен для определения аббревиатур, ключевых слов и фраз, используемых в качестве имён и меток на диаграммах. Глоссарий определяет понятия и термины, которые должны быть одинаково понимаемы всеми участниками разработки и пользователями модели.

4.2.6 Диаграммы

Диаграммы должны быть оформлены в соответствии с Р 50.1.028-2001, за следующим рядом исключений:

- разрешается не использовать описанный в Р 50.1.028-2001 стандартный бланк диаграмм;

- определения цели разработки модели и точки зрения разработчика могут быть приведены вне контекстной диаграммы.

Диаграммы должны отражать, как минимум, два уровня функциональной декомпозиции предметной области.

В случаях, когда это необходимо, в качестве дополнений, поясняющих специфику содержания основных диаграмм, могут быть использованы диаграммы-иллюстрации (For Exposition Only diagrams – FEO diagrams). Диаграмма FEO может не соответствовать синтаксическим правилам IDEF0.

Разработанная модель IDEF0 со всеми уровнями структурной декомпозиции должна быть представлена на диаграмме специального вида - дерево узлов модели. Формат для изображения этого дерева может быть произвольным.

5 Инструментарий для построения диаграмм

Для построения диаграмм рекомендуется использовать следующий программный инструментарий:

- Microsoft Visio Professional 2000/2002/2003.

В случае, если у вас нет возможности использовать рекомендуемые программные средства, диаграммы могут быть построены вручную с помощью средств пакета Microsoft Office.

6 Рекомендуемые источники

6.1 Литература

1. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000. – 352 с.

2. Калянов, Г.Н. CASE-технологии. Консалтинг в автоматизации бизнес-процессов. –3-е изд. – М.: Горячая линия-Телеком, 2002. – 320 с.: ил.

3. Маклаков, С.В. BPwin и ERwin. CASE-средства разработки информационных систем. – 2-е изд., испр. и дополн. – М.: Диалог-МИФИ, 2001. – 304 с.

4. Черемных, С.В., Семёнов, И.О., Ручкин, В.С. Структурный анализ систем: IDEF-технологии. – М.: Финансы и статистика, 2003. – 208 с.: ил.

6.2 Стандарты

1. ГОСТ 7.32-2001 Отчет о научно-исследовательской работе. Структура и правила оформления.

2. Р 50.1.028-2001. Методология функционального моделирования. Введ. 02.07.2001.

3. FIPS 183. Integration Definition for Function Modeling (IDEF0). 1993 December 21.

6.3 Web-ресурсы

1. http://www.idef.com;
2. http://www.idef0.ru;
3. http://www.interface.ru.

Требования к выполнению лабораторной работы №2(7семестр) по курсу «Проектирование информационных систем» на тему «Структурный анализ потоков данных (DFD)»

1 Назначение лабораторной работы

в) Сформировать у студентов представление о сущности функционального подхода к моделированию на примере задачи анализа бизнес-процессов в заданной предметной области.

г) Способствовать в освоении метода структурного анализа потоков данных (DFD) в нотации Гейна-Сэрсона или Йордана.

2 Цель лабораторной работы

В результате выполнения работы студенты должны подтвердить знание следующих элементов диаграммной техники DFD:

- синтаксис графического языка;
- семантика языка;
- правила построения графических диаграмм, словаря данных и спецификации процессов.

3 Требования к оформлению отчёта

Оформление отчёта по лабораторной работе должно удовлетворять общим требованиям к текстовым документам, представленным в ГОСТ 7.32-2001.

Отчёт должен иметь следующую структуру:

- титульный лист;
- содержание;
- основная часть.

4 Требования к содержанию отчёта

Основная часть отчёта должна содержать следующие разделы и подразделы:

а) Описание предметной области на естественном языке.

б) Модель DFD предметной области.

1) Цель создания модели.

2) Графические диаграммы (контекстная диаграмма, диаграммы декомпозиции (не менее двух уровней)).

3) Словарь данных.

4) Спецификации процессов.

4.1 Описание предметной области на естественном языке

Описание предметной области на естественном языке должно представлять собой связный текст и может включать в себя таблицы и рисунки. В описание должны быть включены все бизнес-процессы, которые предполагается подвергнуть функциональному анализу. В описании важно раскрыть характеристики этих процессов: определение основных участников, последовательности реализующих процедур, организационные и случайные факторы, влияющие на их протекание и т.д.

Описание на естественном языке служит основой для процесса формализации предметной области, реализуемого в ходе построения DFD-модели. Модель не может содержать каких-либо аспектов предметной области, не зафиксированных в данном описании.

Раздел описания не может быть озаглавлен как «Описание предметной области». Заголовок должен отражать контент описания. Примеры правильных заголовков: «Описание процессов учёта товаров на складе компании «Тюменская продуктовая компания», «Описание порядка начисления зарплаты в АО «ТюменьПромРесурс».

4.2 Модель DFD

Данный раздел должен содержать основные компоненты модели DFD, построенные в ходе функционального анализа предметной области: цели моделирования, набор графических диаграмм, словарь данных и спецификации процессов.

Каждый компонент должен быть оформлен в виде отдельного подраздела отчёта.

4.2.1 Цель создания модели

Формулировка цели выражает причину создания модели, то есть содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет её структуру.

4.2.2 Диаграммы

Диаграммы должны быть выполнены в соответствии с нотацией Гейна-Сэрсона (Chris Gane, Trish Sarson) и отражать, как минимум, два уровня функциональной декомпозиции предметной области.

4.2.3 Словарь данных

Словарь данных представляет собой организованный список всех элементов данных системы с их точным определением.

Для каждого потока данных в словаре необходимо определить имя, тип и атрибуты потока. Среди атрибутов группового потока необходимо обязательно указать его структурированное.

4.2.4 Спецификации процессов

Каждый процесс, представленный на DFD-диаграмме, должен иметь спецификацию. Спецификация описывает тело процесса – реализуемый алгоритм преобразования потоков данных. Спецификация должна содержать:

- номер и имя процесса;
- списки входных и выходных потоков данных;
- описание алгоритма на структурированном естественном языке.

Синтаксис спецификации процесса можно найти в книге Г.Н. Калянова «CASE-технологии. Консалтинг в автоматизации бизнес-процессов». В приложении Б приведены необходимые выдержки из этой книги.

Структурированный естественный язык – это компромисс между детерминированностью формального языка и выразительностью языка

естественного. Степень его структурированности зависит от целей моделирования и сложности описываемого алгоритма. Синтаксис используемого вами структурированного естественного языка может отличаться от синтаксиса языка, предлагаемого Г.Н. Каляновым.

5 Инструментарий для построения диаграмм

Для построения DFD-диаграмм рекомендуется использовать следующий программный инструментарий:

- Computer Associates BPwin 4.0.
- Microsoft Visio Professional 2000/2002/2003.

В случае, если у вас нет возможности использовать рекомендуемые программные средства, диаграммы могут быть построены вручную с помощью средств пакета Microsoft Office.

6 Рекомендуемые источники

6.1 Литература

1. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000. – 352 с.

2. Гейн, К., Сарсон, Т. Структурный системный анализ: средства и методы. – М: Эйтекс, 1992 – 274 с.

3. Калашян, А.Н., Калянов, Г.Н. Структурные модели бизнеса: DFD-технологии. – М.: Финансы и статистика, 2003. – 256 с.: ил.

4. Калянов, Г.Н. CASE-технологии. Консалтинг в автоматизации бизнес-процессов. –3-е изд. – М.: Горячая линия-Телеком, 2002. – 320 с.: ил.

5. Маклаков, С.В. BPwin и ERwin. CASE-средства разработки информационных систем. – 2-е изд., испр. и дополн. – М.: Диалог-МИФИ, 2001. – 304 с.

6.2 Стандарты

1. ГОСТ 2.105-95. Общие требования к текстовым документам. - Взамен ГОСТ 2.105-79, ГОСТ 2.906-71; Введ. 01.07.96.

2. ГОСТ 7.32-2001. Отчёт о научно-исследовательской работе. Структура и правила оформления. - Взамен ГОСТ 7.32-91; Введ. 01.07.2002 г.

6.3 Web-ресурсы

1. http://www.interface.ru/fset.asp?Url=/case/defs0.htm

Приложение А
(обязательное)

Порядок заполнения словаря данных

Словарь данных представляет собой организованный список всех элементов данных системы с их точными определениями, что дает возможность различным категориям пользователей (от системного аналитика до программиста) иметь общее понимание всех входных и выходных потоков и компонент хранилищ. Определения элементов данных в словаре осуществляются следующими видами описаний:

- описанием значений потоков и хранилищ, изображенных на DFD;

- описанием композиции агрегатов данных, движущихся вдоль потоков, т.е. комплексных данных, которые могут расчленяться на элементарные символы (например, АДРЕС ПОКУПАТЕЛЯ содержит ПОЧТОВЫЙ ИНДЕКС, ГОРОД, УЛИЦУ и т.д.);

- описанием композиции групповых данных в хранилище;

- специфицированием значений и областей действия элементарных фрагментов информации в потоках данных и хранилищах;

- описанием деталей отношений между хранилищами.

Для каждого потока данных в словаре необходимо хранить имя потока, его тип и атрибуты. Информация по каждому потоку состоит из ряда словарных статей, каждая из которых начинается с названия потока, которое является ключевым словом на диаграмме (заголовку предшествует символ “@”).

По типу потока в словаре содержится информация, идентифицирующая:

- простые (элементарные) или групповые (комплексные) потоки;

- внутренние (существующие только внутри системы) или внешние (связывающие систему с другими системами) потоки;

- потоки данных или потоки управления;

- непрерывные (принимающие любые значения в пределах определенного диапазона) или дискретные (принимающие определенные значения) потоки.

Атрибуты потока данных включают:

- имена-синонимы потока данных в соответствии с узлами изменения имени (ключевое слово **ИМЯ**);

- тип потока, в случае группового потока данных (например, дискретный, управляющий) (ключевое слово **ТИП**);

- единицы измерения потока (ключевое слово **ЕДИНИЦА ИЗМЕРЕНИЯ**);

- диапазон значений для непрерывного потока, типичное его значение и информацию по обработке экстремальных значений (ключевое слово **ДИАПАЗОН)**;

- список значений и их смысл для дискретного потока (ключевое слово **ОПИСАНИЕ**);

- список номеров диаграмм различных типов, в которых встречается поток (ключевое слово **ССЫЛКИ**);

- список потоков, в которые данный поток входит (ключевое слово **ВХОЖДЕНИЕ**);

- комментарий, включающий дополнительную информацию (например, о цели введения данного потока) (выделяются звездочками).

Элементы конструкции языка, используемы в словарной статье.

:=	Составлен из
+	и
( )	дополнительный элемент(может присутствовать или отсутствовать)
{ }	повторение элементов
[ ]	альтернативные варианты
**	комментарии
@	ключевое поле в хранилище
|	отдельные альтернативные элементы выбора в конструкции [ ]

Так, например, определение адреса в потоке данных:

адрес = название страны + (название федерального округа) + (название типа субъекта федерации) + (название субъекта федерации) + (название района субъекта федерации)+ название типа населенного пункта + название населенного пункта + (название района населенного пункта) + название типа объекта

населенного пункта + (название объекта населенного пункта) +номер дома + (номер корпуса)+ (номер квартиры)

название типа населенного пункта = [город | районный центр | поселок | село | рабочий поселок | и т.д]

имя субъекта федерации = {слова}

Важно понимать, что точные определения потоков содержатся в словаре данных, а не на диаграммах. Например, на диаграмме может иметься групповой узел с входным потоком X и выходными подпотоками Y и Z. Однако это вовсе не означает, что соответствующее определение в словаре данных обязательно должно быть X=Y+Z. Это определение может быть следующим:

X=A+B+C; Y=A+B; Z=B+C

Ее синтаксис имеет вид:

@ИМЯ = <структурированное-выражение>,

Ниже приведен пример описания потока данных с помощью БНФ:

@ИМЯ = ВОСЬМЕРИЧНАЯ ЦИФРА

@ТИП = дискретный поток

@ОПИСАНИЕ = [«0» ! «1» ! «2» ! «3» ! «4» ! «5» ! «6» ! «7»]

@ИМЯ = ДАННЫЕ КРЕДИТНОЙ КАРТЫ

@ТИП = дискретный поток

@ОПИСАНИЕ = ПАРОЛЬ + ДЕТАЛИ КЛИЕНТА + ЛИМИТ ДЕНЕГ

@ИМЯ = ДАННЫЕ ПО БАЛАНСУ

@ТИП = дискретный поток

@ОПИСАНИЕ = *текущий баланс счета клиента*

@ЕДИНИЦА ИЗМЕРЕНИЯ = доллар

@ДИАПАЗОН = +/- 100000

@ТОЧНОСТЬ = .01

Приложение Б
(обязательное)

Порядок оформления спецификации процессов

Спецификация процесса (СП) используется для описания функционирования процесса в случае отсутствия необходимости детализировать его с помощью DFD (если он невелик и его описание занимает до одной страницы текста). Фактически, СП представляет собой алгоритмы описания задач, выполняемых процессами: множество всех СП является полной спецификацией системы. СП содержит номер и/или имя процесса, списки входных и выходных данных и тело (описание) процесса, являющееся спецификацией алгоритма или операции, трансформирующей входные потоки данных в выходные. Известно большое число разнообразных методов, позволяющих задать тело процесса. Соответствующий язык может варьироваться от структурированного естественного языка или псевдокода до визуальных языков проектирования (типа FLOW-форм и диаграмм Насси-Шнейдермана) и формальных компьютерных языков.

Независимо от используемой нотации, спецификация процесса должна начинаться с ключевого слова «спецификация процесса» (мы используем ключевое слово **СП**). Требуемые входные и выходные данные должны быть специфицированы следующим образом:

@ИМЯ = <имя процесса и его номер>

@ВХОД = <имя символа данных>

@ВЫХОД = <имя символа данных>

@ВХОДВЫХОД = <имя символа данных>

@ПРЕДУСЛОВИЕ=<выражение>

@ПОСТУСЛОВИЕ=<выражение>

@СП <описание>

где <имя символа данных> - соответствующее имя из словаря данных,

<выражение> - описание условия на структурированном языке;

<описание> - спецификация процесса с использованием структурированного языка.

Например:

@ВХОД = СЛОВА ПАМЯТИ

@ВЫХОД = ХРАНИМЫЕ ЗНАЧЕНИЯ

@СП

Для всех СЛОВ ПАМЯТИ выполнить:

Распечатать ХРАНИМЫЕ ЗНАЧЕНИЯ

@

Спецификации должны удовлетворять следующим требованиям:

- для каждого процесса нижнего уровня должна существовать одна и только одна спецификация;
- спецификация должна определять способ преобразования входных потоков в выходные;
- нет необходимости (на данном этапе) определять метод реализации этого преобразования;
- спецификация должна стремиться к ограничению избыточности - не следует переопределять то, что уже было определено на диаграмме или в словаре данных;
- набор конструкций для построения спецификации должен быть простым и стандартным.

К спецификации не предъявляется требование – обязательное использование структурированного языка.

Структурированный естественный язык применяется для читабельного, строгого описания спецификаций процессов. Он является разумной комбинацией строгости языка программирования и читабельности естественного языка. Язык состоит из подмножества слов, организованных в определенные логические структуры, арифметических выражений и диаграмм.

В состав языка входят следующие основные символы:

- глаголы, ориентированные на действие и применяемые к объектам;

- термины, определенные на любой стадии проекта ПО (например, задачи, процедуры, символы данных и т.п.);

- предлоги и союзы, используемые в логических отношениях;

- общеупотребительные математические, физические и технические термины;

- арифметические уравнения;

- таблицы, диаграммы, графы и т.п.;

- комментарии.

Управляющие структуры языка имеют один вход и один выход. К ним относятся:

а) последовательная конструкция:

ВЫПОЛНИТЬ функция1

ВЫПОЛНИТЬ функция2

ВЫПОЛНИТЬ функция3

б) конструкция выбора:

ЕСЛИ <условие> ТО

ВЫПОЛНИТЬ функция1

ИНАЧЕ

ВЫПОЛНИТЬ функция2

КОНЕЦЕСЛИ

в) итерация:

ДЛЯ <условие>

ВЫПОЛНИТЬ функция

КОНЕЦДЛЯ

или

ПОКА <условие>

ВЫПОЛНИТЬ функция

КОНЕЦПОКА

При использовании структурированного естественного языка приняты следующие соглашения:

а) Логика процесса выражается в виде комбинации последовательных конструкций, конструкций выбора и итераций.

б) Ключевые слова ЕСЛИ, ВЫПОЛНИТЬ, ИНАЧЕ и т.д. должны быть написаны заглавными буквами.

в) Слова или фразы, определенные в словаре данных, должны быть написаны заглавными буквами.

г) Глаголы должны быть активными, недвусмысленными и ориентированными на целевое действие (заполнить, вычислить, извлечь, а не модернизировать, обработать).

д) Логика процесса должна быть выражена четко и недвусмысленно.

Ниже приведен пример спецификации процесса:

@ИМЯ = 1.1 ВВЕДЕННИЕ ПАРОЛЯ
@ВХОД = ПАРОЛЬ
@ВЫХОД = СООБЩЕНИЕ
@ВЫХОД = КОРРЕКТНЫЙ ПАРОЛЬ
@СПЕЦПРОЦ 1.1 ПОЛУЧИТЬ ПАРОЛЬ
ВЫПОЛНИТЬ
    выдать СООБЩЕНИЕ клиенту, запрашивающее ввод пароля
    принять ВВЕДЕННЫЙ ПАРОЛЬ
ДОТЕХПОРПОКА ВВЕДЕННЫЙ ПАРОЛЬ = ПАРОЛЬ
    или были сделаны три попытки ввода
КОНЕЦВЫПОЛНИТЬ
ВЫПОЛНИТЬ
    установить флаг КОРРЕКТНЫЙ ПАРОЛЬ в случае равенства
@**КОНЕЦ СП** 1.1

Требования к выполнению лабораторной работы №3(7семестр) по курсу «Проектирование информационных систем» на тему «Концептуальное моделирование данных (EER model)»

1 Назначение лабораторной работы

а) Сформировать у студентов представление об использовании высокоуровневых концептуальных моделей данных для проектирования баз данных.

б) Способствовать в освоении основных концепций модели «сущность-связь» (Entity-Relationship model, ER model) и её расширений (Enhanced Entity-Relationship model, EER model).

2 Цель лабораторной работы

В результате выполнения работы студенты должны подтвердить знание:

- основных концепций модели «сущность-связь» (ER model) и её расширений (EER model);
- этапов разработки концептуальной модели данных;
- диаграммной техники ER-моделирования (ERD – Entity-Relationship Diagrams, ERD) Питера Чена (Dr. Peter P. Chen, MIT).

3 Требования к оформлению отчёта

Оформление отчёта по лабораторной работе должно удовлетворять общим требованиям к текстовым документам, представленным в ГОСТ 7.32-2001.

Отчёт должен иметь следующую структуру:

- титульный лист;
- содержание;
- основная часть.

4 Требования к содержанию отчёта

Основная часть отчёта должна содержать следующие разделы и подразделы:

а) Описание предметной области на естественном языке.

б) Модель «Сущность-связь».

1) Цель создания модели.

2) Словарь данных.

3) EER-диаграмма.

4.1 Описание предметной области на естественном языке

Описание предметной области на естественном языке должно представлять собой связный текст и может включать в себя таблицы и рисунки. В описание должны быть включены все понятия (объекты, сущности и т.д.), которые имеют бизнес-смысл для рассматриваемой предметной области. В описании важно раскрыть характеристики этих понятий: определение, характерные свойства, взаимосвязи и т.д.

Описание на естественном языке служит основой для процесса формализации предметной области, реализуемого в ходе построения EER-модели. Модель не может содержать каких-либо аспектов предметной области, не зафиксированных в данном описании.

Раздел описания не может быть озаглавлен как «Описание предметной области». Заголовок должен отражать контент описания. Примеры правильных заголовков: «Описание процессов учёта товаров на складе компании «Тюменская продуктовая компания», «Описание порядка начисления зарплаты в АО «ТюменьПромРесурс».

4.2 Модель «Сущность-связь»

Данный раздел должен содержать основные компоненты EER-модели: документацию модели в виде словаря данных и графическое представление модели в виде EER-диаграммы.

Каждый компонент должен быть оформлен в виде отдельного подраздела отчёта.

4.2.1 Цель создания модели

Формулировка цели выражает причину создания модели, то есть содержит перечень вопросов, на которые должна отвечать модель, что в значительной мере определяет её структуру.

4.2.2 Словарь данных

Словарь данных должен являться основной спецификацией модели «сущность-связь». Он включает в себя информацию о выявленных:

- типах сущностей;
- типах связей;
- доменах атрибутов;
- связях специализации (генерализации);
- связях категоризации.

Информация о выявленных атрибутах должна быть приведена вместе с описанием типов сущностей и типов связей.

Словарь данных рекомендуется оформить в виде подразделов отчёта, соответствующих указанному выше списку компонентов модели. Информацию словаря данных желательно представлять в табличной форме.

4.2.2.1 Типы сущностей

Для каждого выявленного типа сущности в словаре данных следует указать:

а) имя типа сущности;

б) краткое описание;

в) разновидность (сильный тип/слабый тип);

г) ожидаемое количество экземпляров (опционально);

д) список атрибутов;

е) список потенциальных ключей;

ж) первичный ключ.

Список атрибутов рекомендуется представлять в виде таблицы. Структура таблицы приведена на рисунке 1.

Имя атрибута	Краткое описание	Домен	Тип по составу (простой / составной)	Тип по значению (однозначный / многозначный)	Определение метода вычисления для производного атрибута
			В случае составного атрибута, указать из каких простых атрибутов он состоит.		

Рисунок 1 – Структура таблицы для представления списка атрибутов.

4.2.2.2 Типы связей

Для каждого выявленного типа связи в словаре данных следует указать:

а) имя типа связи;

б) краткое описание;

в) степень связи;

г) список типов сущностей, участвующих в связи;

д) список атрибутов.

Список атрибутов рекомендуется представлять в виде таблицы. Структура таблицы приведена на рисунке 1.

Список типов сущностей, участвующих в связи, также рекомендуется представлять в виде таблицы. Структура таблицы приведена на рисунке 2.

Имя типа сущности	Кардинальность	Степень участия

Рисунок 2 – Структура таблицы для представления списка типов сущностей, участвующих в связи

4.2.2.3 Домены атрибутов

Для каждого выявленного домена в словаре данных следует указать:

а) имя домена;

б) краткое описание;

в) список подчинённых доменов (в случае, если домен представляет собой комбинацию других доменов).

4.2.2.4 Специализация (генерализация)

Для каждой выявленной связи специализации (генерализации) в словаре данных следует указать:

а) краткое описание;

б) имя суперкласса;

в) список подклассов;

г) ограничение пересечения (пересекающаяся/непересекающаяся);

д) ограничение участия (участие полное/частичное).

4.2.2.5 Категоризация

Для каждой выявленной связи категоризации в словаре данных следует указать:

а) краткое описание;

б) список суперклассов;

в) имя подкласса;

г) ограничение участия (участие полное/частичное).

4.2.3 EER-диаграмма

Диаграммы должны быть выполнены в соответствии с нотацией П. Чена (Peter P. Chen) (нотация представлена в книге Т. Коннолли «Базы данных: проектирование, реализация и сопровождение. Теория и практика»).

Если EER-диаграмма получается слишком громоздкой и неудобной для демонстрации разрешается не отображать на ней всю информацию модели. Например, можно скрыть атрибуты типов связей и те атрибуты типов сущностей, которые не входят в первичный ключ. Однако при этом, скрываемая информация должна сохраниться в словаре данных.

5 Инструментарий для построения диаграмм

Для построения ER-диаграмм рекомендуется использовать Microsoft Visio Professional 2000/2002/2003.

6 Рекомендуемые источники

6.1 Литература

1. Chen P. The Entity-Relationship Model-Toward a Unified View of Data – ACM Transactions on Database Systems, Vol. 1, No. 1. March 1976, Pages 9-36.

2. Коннолли, Т., Бегг, К., Страчан, А. Базы данных: проектирование, реализация и сопровождение. Теория и практика. 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 1120 с.

6.2 Стандарты

1. ГОСТ 7.32-2001. Отчёт о научно-исследовательской работе. Структура и правила оформления. - Взамен ГОСТ 7.32-91; Введ. 01.07.2002 г.

6.3 Web-ресурсы

1. http://bit.csc.lsu.edu/~chen/chen.html (Dr. Peter P. Chen's Homepage).

Требования к выполнению лабораторной работы №4(7семестр) по курсу «Проектирование информационных систем» на тему «Логическое моделирование данных (IDEF1X)»

1 Назначение лабораторной работы

а) Сформировать у студентов представление об использовании логической модели данных при проектировании баз данных.

б) Способствовать в освоении основных положений федерального стандарта США «FIPS 184. Integration Definition Method for Information Modeling (IDEF1X)».

2 Цель лабораторной работы

В результате выполнения работы студенты должны подтвердить знание:

а) синтаксиса и семантики графического языка методологии IDEF1X;

б) правил преобразования концептуальной модели данных в логическую модель данных.

3 Требования к оформлению отчёта

Оформление отчёта по лабораторной работе должно удовлетворять общим требованиям к текстовым документам, представленным ГОСТ 7.32-2001.

Отчёт должен иметь следующую структуру:

- титульный лист;
- содержание;
- основная часть.

4 Требования к содержанию отчёта

Основная часть отчёта должна содержать следующие разделы:

1) Цель создания модели.

2) Границы моделирования.

3) Глоссарий.

4) Диаграмма IDEF1X.

4.1 Цель создания модели

Формулировка цели выражает причину создания модели.

Цель должна отражать роль создаваемой логической модели данных в процессе проектирования базы данных.

4.2 Границы моделирования

Границы моделирования задают основную область, которая охватывается данной моделью.

Здесь может быть приведена информация, руководствуясь которой Вы ограничивали область моделирования при переходе от концептуальной модели данных к модели логической.

Данный раздел отчёта является опциональным.

4.3 Глоссарий

Глоссарий содержит описания всех сущностей и доменов атрибутов, присутствующих в модели IDEF1X.

Описание должно представлять собой общепринятое и однозначное определение сущности (домена атрибута). Определения доменов должны содержать ссылку на родительский (или базовый) домен.

4.4 Диаграмма IDEF1X

Диаграммы должны быть выполнены в соответствии с правилами синтаксиса и семантики, определёнными стандартом IDEF1X. Имена сущностей, связей и доменов должны соответствовать лексическим соглашениям IDEF1X.

При построении диаграмм связи «многие ко многим» (non-specific relationships), следует обязательно преобразовывать в связи «один ко многим» (specific relationships) с дополнительной сущностью.

На диаграммах рекомендуется в произвольной форме отображать домены атрибутов.

5 Инструментарий для построения диаграмм

Для построения диаграмм рекомендуется использовать следующий программный инструментарий:

- Microsoft Visio Professional 2000/2002/2003.

В случае, если у вас нет возможности использовать рекомендуемые программные средства, диаграммы могут быть построены вручную с помощью средств пакета Microsoft Office.

6 Рекомендуемые источники

6.1 Литература

1. Вендров,. А. М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000. – 352 с.

2. Коннолли,. Т., Бегг, К., Страчан, А. Базы данных: проектирование, реализация и сопровождение. Теория и практика. 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 1120 с.

6.2 Стандарты

1. ГОСТ 2.105-95. Общие требования к текстовым документам. - Взамен ГОСТ 2.105-79, ГОСТ 2.906-71; Введ. 01.07.96.

2. ГОСТ 7.32-2001. Отчёт о научно-исследовательской работе. Структура и правила оформления. - Взамен ГОСТ 7.32-91; Введ. 01.07.2002 г.

3. FIPS 184. Integration Definition Method for Information Modeling (IDEF1X). 1993 December 21.

6.3 Web-ресурсы

1. http://www.idef.com;

2. http://www.idef.ru;

3. http://www.interface.ru.

Приложение В

Требования к выполнению лабораторной работы №1 (8семестр) по курсу «Проектирование информационных систем» на тему «Business use case model»

1. **Назначение лабораторной работы**

Сформировать у студентов представление о сущности объектно-ориентированного подхода к моделированию. Изучить сведения о предметной области в виде моделирования описаний деятельности.

2. **Цель лабораторной работы**

В результате выполнения лабораторной работы студенты должны научиться строить модели бизнес-анализа, используемые в процессе бизнес-моделирования, определять элементы модели и связи между ними:

- Business Actor
- Business Worker
- Business Use Case

3. **Требования к оформлению отчета**

Оформление отчета по лабораторной работе должно соответствовать шаблону, для форматирования должны использоваться стандартные стили.

Отчет должен иметь следующую структуру:

- титульный лист;
- содержание;
- основная часть.

4. **Требования к содержанию отчета**

Основная часть отчета должна содержать следующие разделы:

1) Графические диаграммы

2) Описание прецедентов:
   2.1) Название прецедента
   2.2) Цель прецедента
   2.3) Дата создания
   2.4) Дата изменения
   2.5) Предусловия
   2.6) Постусловия
   2.7) Ограничения
   2.8) Предположения
   2.9) Основной поток
   2.10) Альтернативный поток

5. Инструментарий для построения диаграмм

1) Rational Rose Enterprise Edition
2) StarUML
3) Rational XDE
4) Microsoft Visio (Software -> UML)

6. Рекомендуемые источники

1) Нейбург, Эрик, Дж., Максимчук, Роберт, А. Проектирование баз данных с помощью UML.: Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 288 с.: ил.
2) Буч Г. / Язык UML. Руководство пользователя. / Грэйди Буч, Джеймс Рамбо, Айвар Джекобсон: Пер. с англ. Слинкин А. А. – 2-ое изд., стер. – М.: ДМК «Пресс»; СПБ.: Питер, 2004 – 432 с.: ил.
3) Крачтен, Филлип / Введение в Rational Unified Process. 2-ое изд..: Пер. с англ – М. : Издательский дом «Вильямс», 2002. –240 с.: ил. – Парал. тит. англ.
4) Электронное руководство Rational Unified Process: Team → Artifacts → Business Modeling Artifact Set → Business Analysis Model.

Требования к выполнению лабораторной работы №2 (8семестр) по курсу «Проектирование информационных систем» на тему «Business ANALYSIS (Object) Model»

1. Назначение лабораторной работы

Сформировать у студентов представление о сущности объектно-ориентированного подхода к моделированию. Изучить сведения о предметной области в виде моделирования объектов.

2. Цель лабораторной работы

В результате выполнения лабораторной работы студенты должны научиться строить диаграмму классов, которая отображает взаимосвязи между соответствующими сущностями бизнес-прецедентов, определять элементы модели и связи между ними:

- Business Actor
- Business Worker
- Business Entity

3. Требования к оформлению отчета

Оформление отчета по лабораторной работе должно соответствовать шаблону, для форматирования должны использоваться стандартные стили.

Отчет должен иметь следующую структуру:

- титульный лист;
- содержание;
- основная часть.

4. Требования к содержанию отчета

Основная часть отчета должна содержать следующие разделы:

1) Графические диаграммы
2) Описание бизнес-правил сущности

5. Инструментарий для построения диаграмм

1) Rational Rose Enterprise Edition
2) StarUML
3) Rational XDE
4) MS Visio

6. Рекомендуемые источники

1) Нейбург, Эрик, Дж., Максимчук, Роберт, А. Проектирование баз данных с помощью UML.: Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 288 с.: ил.
2) Буч Г. / Язык UML. Руководство пользователя. / Грэйди Буч, Джеймс Рамбо, Айвар Джекобсон: Пер. с англ. Слинкин А. А. – 2-ое изд., стер. – М.: ДМК «Пресс»; СПБ.: Питер, 2004 – 432 с.: ил.
3) Крачтен, Филлип / Введение в Rational Unified Process. 2-ое изд..: Пер. с англ – М. : Издательский дом «Вильямс», 2002. –240 с.: ил. – Парал. тит. англ.
4) Электронное руководство Rational Unified Process: Team → Artifacts → Business Modeling Artifact Set → Business Analysis Model.

Требования к выполнению лабораторной работы №3 (8семестр) по курсу «Проектирование информационных систем» на тему «Vision»

1. Назначение лабораторной работы

Сформировать у студентов представление о проектируемой системе.

2. Цель лабораторной работы

В результате выполнения лабораторной работы студенты должны научиться сбирать, анализировать и определять в общем виде потребности пользователей к проектируемой системе, а также ее возможности.

3. Требования к оформлению отчета

Оформление отчета по лабораторной работе должно соответствовать шаблону, для форматирования должны использоваться стандартные стили.

Отчет должен иметь следующую структуру:

- титульный лист;
- документ Видение.

4. Рекомендуемые источники

1) Электронное руководство Rational Unified Process: Team → Artifacts → Requirements Artifact Set → Vision.

Требования к выполнению лабораторной работы №4 (8семестр) по курсу «Проектирование информационных систем» на тему «Use Case Model»

1. **Назначение лабораторной работы**

Сформировать у студентов представление о пользователях и функциях проектируемой системы.

2. **Цель лабораторной работы**

В результате выполнения лабораторной работы студенты должны научиться строить модель вариантов использования (прецедентов), выявлять пользователей и функции системы, а также отражать требования пользователей к системе, определять элементы модели и связи между ними:

- Actor
- Use Case
- Use-Case Package

3. **Требования к оформлению отчета**

Оформление отчета по лабораторной работе должно соответствовать шаблону, для форматирования должны использоваться стандартные стили.

Отчет должен иметь следующую структуру:

- титульный лист;
- содержание;
- основная часть.

4. **Требования к содержанию отчета**

Основная часть отчета должна содержать следующие разделы:

1) Графические диаграммы
2) Спецификация варианта использования (прецедента):
    а) Название прецедента
        – Краткое описание
    б) Потоки событий

– Основной поток
– Альтернативные потоки
1. <Альтернативный поток 1>
2. <Альтернативный поток 2>

в) Специальные требования
– <Специальное требование 1>

г) Предусловия
– <Предусловие 1>

д) Постусловия
– <Постусловие 1>

е) Точки расширения
– <Название точки расширения 1>

5. Инструментарий для построения диаграмм

1) Rational Rose Enterprise Edition
2) StarUML
3) Rational XDE
4) MS Visio

6. Рекомендуемые источники

1) Буч Г. / Язык UML. Руководство пользователя. / Грэйди Буч, Джеймс Рамбо, Айвар Джекобсон: Пер. с англ. Слинкин А. А. – 2-ое изд., стер. – М.: ДМК «Пресс»; СПБ.: Питер, 2004 – 432 с.: ил.
2) Буч Г., Якобсон А., Рамбо Дж. / UML. Классика CS. 2-е изд./ Пер. с англ.; Под общей редакцией проф. С. Орлова – СПб.:Питер, 2006.- 736 с.: ил.
3) Фаулер М. / UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 192 с., ил.
4) Электронное руководство Rational Unified Process: Team → Artifacts → Business Modeling Artifact Set → Business Analysis Model.

Требования к выполнению лабораторной работы №5 (8семестр) по курсу «Проектирование информационных систем» на тему «Activity Model»

1. **Назначение лабораторной работы**

Сформировать у студентов представление о выполнении вариантов использования (прецедентов).

2. **Цель лабораторной работы**

В результате выполнения лабораторной работы студенты должны научиться выявлять узкие места в работе вариантов использования (прецедентов), определять процессы в системе (внутренние или внешние), которые можно усовершенствовать, получать информацию по требованиям конкретного варианта использования (прецедента), определять элементы модели и связи между ними:

- Activity states
- Transitions
- Decisions
- Alternative threads
- Synchronization bars
- Guard conditions
- Concurrent threads

3. **Требования к оформлению отчета**

Оформление отчета по лабораторной работе должно соответствовать шаблону, для форматирования должны использоваться стандартные стили.

Отчет должен иметь следующую структуру:

- титульный лист;
- содержание;
- основная часть.

4. Требования к содержанию отчета

Основная часть отчета должна содержать следующие разделы:

1) Графические диаграммы

5. Инструментарий для построения диаграмм

1) Rational Rose Enterprise Edition
2) StarUML
3) Rational XDE
4) MS Visio

6. Рекомендуемые источники

1) Буч Г. / Язык UML. Руководство пользователя. / Грэйди Буч, Джеймс Рамбо, Айвар Джекобсон: Пер. с англ. Слинкин А. А. – 2-ое изд., стер. – М.: ДМК «Пресс»; СПБ.: Питер, 2004 – 432 с.: ил.
2) Электронное руководство Rational Unified Process: Team → Artifacts → Business Modeling Artifact Set → Business Analysis Model.

Требования к выполнению лабораторной работы №6 (8семестр) по курсу «Проектирование информационных систем» на тему «Class Diagram»

1. Назначение лабораторной работы

Сформировать у студентов представление о концептуальных сущностях системы.

2. Цель лабораторной работы

В результате выполнения лабораторной работы студенты должны научиться выявлять группы объектов с общими свойствами (атрибутами), поведением (операциями), отношениями с другими объектами и семантикой.

3. Требования к оформлению отчета

Оформление отчета по лабораторной работе должно соответствовать шаблону, для форматирования должны использоваться стандартные стили.

Отчет должен иметь следующую структуру:

- титульный лист;
- содержание;
- основная часть.

4. Требования к содержанию отчета

Основная часть отчета должна содержать следующие разделы:

1) Графические диаграммы

5. Инструментарий для построения диаграмм

1) Rational Rose Enterprise Edition
2) StarUML
3) Rational XDE
4) MS Visio

6. Рекомендуемые источники

1) Буч Г. / Язык UML. Руководство пользователя. / Грэйди Буч, Джеймс Рамбо, Айвар Джекобсон: Пер. с англ. Слинкин А. А. – 2-ое изд., стер. – М.: ДМК «Пресс»; СПБ.: Питер, 2004 – 432 с.: ил.
2) Электронное руководство Rational Unified Process: Team → Artifacts → Business Modeling Artifact Set → Business Analysis Model.

Требования к выполнению лабораторной работы №7 (8семестр) по курсу «Проектирование информационных систем» на тему «Sequence Diagram»

1. Назначение лабораторной работы

Сформировать у студентов представление о реализациях вариантов использования (прецедентов) в логическом представлении системы.

2. Цель лабораторной работы

В результате выполнения лабораторной работы студенты должны научиться выявлять объекты и классы, используемые в сценарии, и последовательность сообщений, которыми обмениваются объекты для выполнения сценария.

3. Требования к оформлению отчета

Оформление отчета по лабораторной работе должно соответствовать шаблону, для форматирования должны использоваться стандартные стили.

Отчет должен иметь следующую структуру:

- титульный лист;
- содержание;
- основная часть.

4. Требования к содержанию отчета

Основная часть отчета должна содержать следующие разделы:

1) Графические диаграммы

5. Инструментарий для построения диаграмм

1) Rational Rose Enterprise Edition
2) StarUML
3) Rational XDE
4) MS Visio

6. Рекомендуемые источники

1) Буч Г. / Язык UML. Руководство пользователя. / Грэйди Буч, Джеймс Рамбо, Айвар Джекобсон: Пер. с англ. Слинкин А. А. – 2-ое изд., стер. – М.: ДМК «Пресс»; СПБ.: Питер, 2004 – 432 с.: ил.
2) Электронное руководство Rational Unified Process: Team → Artifacts → Business Modeling Artifact Set → Business Analysis Model.

Приложение Г

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

.

Проектирование информационных систем

Рабочая программа для специальности

080801.65 - Прикладная информатика в экономике

Тюмень 2006

1. Пояснительная записка

1.1. Цели и задачи дисциплины

Цель изучения дисциплины - получение студентами знаний по основам структурного системного анализа и проектирования информационных систем. Данная дисциплина должна подготовить будущих специалистов к решению следующих задач: создание информационно-логических моделей объектов, разработка нового программного и информационного обеспечения в предметной области; оптимизация информационных процессов обработки информации; решение задач унификации профессионально-ориентированного программного и информационного обеспечения предметной области

Задачи изучения дисциплины.

1.Расширение представлений о методах и средствах проектирования современных информационных систем.

2. Приобретение навыков в использовании CASE-систем проектирования информационных систем.

3. Развитие самостоятельности при разработке информационных систем на базе корпоративных СУБД.

1.2. Требования к уровню освоения содержания дисциплины

В результате изучения дисциплины студенты должны

знать:

- перспективные информационные технологии проектирования;
- методы научных исследований по теории, технологии разработки;

уметь:

- формулировать и решать задачи проектирования профессионально-ориентированных информационных систем с использованием различных методов и решений;
- ставить задачу системного проектирования и комплексирования локальных и глобальных сетей обслуживания пользователей информационных систем;
- создавать и внедрять профессионально-ориентированные информационные системы в предметной области;

должен владеть:

- методиками анализа предметной области и проектирования профессионально-ориентированных информационных систем;
- методами системного анализа в предметной области;

иметь опыт:

- работы с основными объектами, явлениями и процессами, связанными с информационными системами, и использования методов их научного исследования;
- разработки проектных решений и их реализации в заданной инструментальной среде.

Объем дисциплины и виды учебной работы в седьмом семестре

Вид занятий	Всего часов	Семестры
		7
Общая трудоемкость	147	147
Аудиторные занятия	54	54
Лекции	36	36
Лабор. Занятия	18	18
Индивидуальная работа	10	10
Самостоятельная работа	83	83
Контрольные работы		+
Курсовая работа		
Вид итогового контроля		экзамен

Тематический план изучения дисциплины

№ п/п	Наименование темы	Лекции	Лаборат. работы	Самост. работа	Формы контроля
1	Жизненный цикл программного обеспечения	2		3	
2	Жизненный цикл разработки программного обеспечения.	8		10	
3	Проектирование информационной системы .	2	6		
4	Государственные стандарты, регламентирующие работы по разработки программного обеспечения.	2			
5	Основные компоненты технологии проектирования ИС.	4			
6	Диаграммы сущность связь.	6			
7	Автоматизированное проектирование ИС с использованием CASE-	1	3		

	технологии.				
8	Проектирование фактографических БД	9	8		
9	Типовое проектирование ИС.	2			К.р
	Всего:	**30**	**11**	**176**	

Содержание разделов дисциплины

1. Жизненный цикл программного обеспечения. Стандарт ИСО/МЭК 12207: основные понятия, структура, область применения, основные участники процесса.
2. Жизненный цикл разработки программного обеспечения. Обобщенная схема процесса. Причина стандартизации процесса. Основные стандарты, работающие в этой области знаний. Модели жизненного цикла разработки программного обеспечения. Каноническое проектирование ИС. Стадии и этапы процесса проектирования ИС. Состав работ на предпроектной стадии, стадии технического и рабочего проектирования, стадии ввода в действие Содержание RAD-технологии прототипного создания приложений
3. Проектирование информационной системы (ИС). Понятия и структура проекта ИС. Требования к эффективности и надежности проектных решений.
4. Государственные стандарты, регламентирующие работы по разработки программного обеспечения. Достоинства и основные недостатки. Каноническое проектирование ИС. Стадии и этапы процесса проектирования ИС. Состав работ на предпроектной стадии, стадии технического и рабочего проектирования, стадии ввода в действие ИС, эксплуатации и сопровождения. Состав проектной документации.
5. Основные компоненты технологии проектирования ИС. Методы и средства проектирования ИС. Краткая характеристика применяемых технологий проектирования. Требования, предъявляемые к технологии проектирования ИС. Выбор технологии проектирования ИС. Функционально-ориентированный и объектно-ориентированный подходы. Функциональное моделирование DFD.
6. Диаграммы сущность связь. Основные понятия. Технология построения модели. Связь информационной модели с функциональной.
7. Автоматизированное проектирование ИС с использованием CASE-технологии

8. Проектирование фактографических БД: методы проектирования; концептуальное, логическое и физическое проектирование. Принципы и особенности проектирования интегрированных ИС. Система управления информационными потоками как средство интеграции приложений ИС. Методы и средства организации метаинформации проекта ИС.
9. Типовое проектирование ИС. Понятие типового элемента. Технологии параметрически-ориентированного и модельно-ориентированного проектирования.

Лабораторные занятия

1. Функциональное моделирование IDEF0
2. Функциональное моделирование DFD.
3. Концептуальное моделирование схемы данных. ERD в нотации Чена.
4. Логическое моделирование данных. Диаграммы IDEF1X.

Вопросы к экзаменам

1. Жизненный цикл - основные определения
2. Международный стандарт ISO/IEC 12207, назначение, область применения, ограничения
3. Международный стандарт ISO/IEC 12207, структура
4. Международный стандарт ISO/IEC 12207, основные участники процесса (пример)
5. Международный стандарт ISO/IEC 12207. Основные процессы
6. Международный стандарт ISO/IEC 12207. Вспомогательные процессы
7. Международный стандарт ISO/IEC 12207. Организационные процессы
8. Международный стандарт ISO/IEC 12207. Этапы и стадии ЖЦ.
9. ЖЦ разработки ПО. Основные термины.
10. Модель жизненного цикла разработки ПО. SLCM
11. SLCM. Обобщенная структура процесса. Целевая структура инжиниринга ПО.
12. Причина стандартизация процесса разработки ПО.
13. Модель SEI CMM
14. SLCM в Международном стандарте ISO/IEC 12207.
15. Каскадная модель (преимущества, недостатки, область применения)
16. V-образная модель (преимущества, недостатки, область применения)

17. Модель эволюционно - ускоренного прототипирования (преимущества, недостатки, область применения)
18. Быстрая разработка приложений (RAD) (преимущества, недостатки, область применения)
19. Инкрементная модель (преимущества, недостатки, область применения)
20. Спиральная модель (преимущества, недостатки, область применения)
21. ГОСТ Р IDEF0. (понятия системного анализа, преимущества недостатки и область применения)
22. ГОСТ Р IDEF0. Синтаксис графического языка IDEF0
23. ГОСТ Р IDEF0. Семантика языка IDEF0
24. ГОСТ Р IDEF0. Иерархическая структура диаграмм. Ссылочный код.
25. ГОСТ Р IDEF0. Отношения блоков на диаграммах. ICOM - кодирование граничных стрелок. Туннель
26. ГОСТ Р IDEF0. Правила построения диаграмм
27. Методика разработки функциональных моделей среде IDEF 0. Понятия: система, функциональный блок, потоки, информация
28. Методика разработки функциональных моделей среде IDEF 0. Классификация функций, моделируемых блоками IDEF0
29. Организационно-технические структуры и механизмы IDEF0-моделей
30. Методика разработки функциональных моделей среде IDEF 0. Управление
31. Интегрированная структурная модель (расширенная DFD)
32. Базовая нотация DFD
33. Миниспицификации. Критерии для завершения детализации DFD - модели
34. Рекомендации оформления DFD
35. Преимущества DFD
36. Этапы построения моделей в DFD-технологии.
37. Разработка структурной функциональной модели бизнес-системы (DFD).
38. ERD - модель (преимущества, недостатки, область применения)
39. ERD - модель. Сущности.
40. ERD - модель. Связи
41. ERD - модель. Структурные ограничения
42. ERD - модель. Ловушки соединения.
43. EER - модель. Специализация / генерализация
44. EER - модель. Категоризация
45. Методология проектирования

46. Концептуальное проектирование базы данных
47. Логическое проектирование базы данных
48. Физическое проектирование базы данных
49. Факторы успешного завершения проектирования БД
50. Первый этап проектирования БД (задачи и подэтапы).
51. Второй этап проектировании БД (задачи и подэтапы)
52. Третий этап проектирования БД (задачи и подэтапы)
53. Первый этап проектирования БД (характеристика подэтапов).
54. Второй этап проектировании БД (характеристика подэтапов)
55. Третий этап проектирования БД (характеристика подэтапов)
56. Действия на этапе преобразования локальной концептуальной модели данных в локальную логическую модель
57. DBDL (Database Definition Language)
58. Действия на этапе определения набора отношений исходя из структуры локальной логической модели данных
59. Проверка модели в отношении транзакций пользователей
60. IDEF1X
61. IDEF3
62. ГОСТ (СТ СЭВ) 19.201-78, ГОСТ (СТ СЭВ) 19.101-77, ГОСТ 19.102-77.
63. Стандарты комплекса ГОСТ 34.
64. ГОСТ 34.602-89
65. ЕСПД для ПС (преимущества, недостатки, область применения)
66. Краткое представление стандартов ЕСПД. Обозначение ЕСПД

Литература

Основная:

1. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. – М.: Финансы и статистика, 2000. – 352с.
2. Калянов Г. Н. CASE структурный системный анализ (автоматизация и применение). М.: Лори, 1996. - 242с.
3. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. М: Финансы и статистика, 1998. - 176с.
4. Боэм Б.У. Инженерное проектирование ПО. М.: Радио и связь, 1985. - 512с.

5. Липаев В.В. Направления развития методов и стандартов открытых систем // Информатика и вычислителелельная техника. Научно-технический сборник. Вып. 1-2. М.: 1995.
6. Липаев В.В., Филинов Е.Н. Формирование и применение профилей открытых информационных систем // Информационные технологии, 1997. №4. С.3-11.
7. Чери С., Готлоб Г., Танка Л. Логическое программирование и базы данных. М.: Мир, 1992. - 234с.

Дополнительная:

8. Бадд Т. Объектно-ориентированное программирование в действии. СПб.; Питер, 1997. - 464с.
9. Беленков Е.Л. Системы документооборота в электронных офисах //Информатизация и связь. 1998. №2. С. 3 4- 3 8.
10. Объектно-ориентированная технология CORBA. Обзор фирмы Jet Infosystem.
11. Система автоматизации делопроизводства и документооборота ДЕЛО-96: Введение в систему. М.: ЗАО Электронные офисные системы, 1997.67с.
12. Стоунбрейкер Михаэл. Объектно-реляционные системы баз данных // Открытые системы, 1994. Вып. 4(8). С.34-39.
13. Чечкин А.В. Математическая информатика. М.: Наука, 1991. - 416с.

Объем дисциплины и виды учебной работы в восьмом семестре

Вид занятий	Всего часов	Семестры
		8
Общая трудоемкость	143	143
Аудиторные занятия	51	51
Лекции	34	34
Лабор. Занятия	17	17
Индивидуальная работа	9	9
Самостоятельная работа	83	83
Контрольные работы		+
Курсовая работа		+
Вид итогового контроля		экзамен

Тематический план изучения дисциплины

№ п/п	Наименование темы	Лекции	Лаборат. работы	Самост. работа	Формы контроля
	Состав, содержание и принципы организации информационного обеспечения ИС.	2		5	
	Объектно-ориентированный язык UML	4		5	К.р.
	Основы структурного моделирования	4		10	К.р.
	Основы моделирования поведения	4		10	К.р.
	Архитектурное моделирование	4		10	К.р.
	Введение в методологию моделирования RUP	8	2	10	
	Бизнес-моделирование.	2	4	10	
	Процесс разработки	4	11	10	
	Межсистемные интерфейсы и	2		13	

	драйверы				
	Всего:	**34**	**17**	**83**	

Содержание разделов дисциплины

1. Проектирование документальных БД: анализ предметной области, разработка состава и структуры БД, проектирование логико-семантического комплекса.
2. Объектно-ориентированный язык UML. История языка UML. Основные принципы моделирования. Объектно-ориентированное моделирование. Назначение UML. Классификация и характеристика строительных блоков UML.
3. Основы структурного моделирования. Классы и объекты. Этапы моделирования распределения обязанностей в системе и словаря системы. Отношения. Моделирование отношений наследования и структурных отношения. Общие механизмы. Интерфейсы и роли.
4. Основы моделирования поведения. Взаимодействия. Прецеденты Диаграммы: взаимодействия, прецедентов, действия, диаграммы состояния.
5. Архитектурное моделирование. Компоненты Развертывания кооперации. Системы и модели.
6. Введение в методологию моделирования RUP. Обзор методологии agile Основные понятия RUP. Прогрессивные методы разработки программного обеспечения. Основные дисциплины (технологические процессы) RUP. Организация унифицированного процесса во времени. Навигация в RUP. Инструментарий, поддерживающий agile –методологию.
7. Бизнес-моделирование. Технологический процесс, артефакты, роли и их задачи.
8. Процесс разработки. Основные роли их задачи и артефакты.
9. Межсистемные интерфейсы и драйверы; интерфейсы в распределенных системах. Стандартные методы совместного доступа к базам и программам в сложных информационных системах (драйверы ODBC, программная система CORBA и др.).

.

Лабораторные занятия

1. Бизнес-моделирование (бизнес use case, диаграмма действий, диаграмма объектов)
2. Разработка функциональных требований к системе (Документ видение)
3. Моделирование системы с точки зрения конечного пользователя (Диаграмма прецедентов)
4. Моделирование взаимодействия объектов системы (Диаграмма последовательности)
5. Структурное моделирование (диаграмма классов)

Вопросы к экзаменам

1. История UML.
2. 4 основные принципа моделирования от Буча
3. Объектно-ориентированное моделирование
4. Назначение UML и его особенности
5. Структурные сущности
6. Поведенческие сущности
7. Группирующие сущности
8. Аннотационные сущности
9. Отношения (основные и дополнительные)
10. Виды диаграмм в UML
11. Диаграмма классов
12. Диаграмма объектов
13. Диаграмма прецедентов
14. Диаграмма последовательностей
15. Диаграмма кооперации
16. Диаграмма состояний
17. Диаграмма действий
18. Диаграмма компонентов
19. Диаграмма развертывания
20. Механизмы расширения
21. Архитектура 4+1
22. Жизненный цикл разработки ПО и рекомендуемые процессы
23. Классы и основы структурного моделирования
24. Этапы моделирования словаря системы

25. Отношения
26. Зависимости
27. Обобщения
28. Ассоциации
29. Агрегирование
30. Моделирование отношений наследования
31. Моделирование структурных отношений
32. Общие механизмы
33. Прецеденты и их задачи
34. Описание потоков событий
35. Отношения между прецедентами
36. Моделирование поведения элемента
37. Диаграммы прецедентов
38. Диаграммы последовательностей
39. Основные проблемы, возникающие при разработке ПО
40. Лучшие методы
41. Итеративная разработка
42. Управление требованиями
43. Компонентоиспользующая архитектура
44. Визуальное моделирование
45. Постоянно проверяемое качество
46. Управление изменениями
47. Организация временных процессов в RUP
48. Дисциплины RUP
49. Главные вехи
50. Основные фазы RUP
51. Фаза начала. Цели, критерии оценки.
52. Фаза проектирования. Цели, критерии оценки.
53. Фаза построения. Цели, критерии оценки.
54. Фаза внедрения. Цели, критерии оценки.
55. Понятие итерации в RUP
56. Планирование итераций
57. Риски и стратегии управления рисками

Примерные темы курсовых работ по дисциплине

1. Автоматизация обмена данных из программы "Супер Окна" в информационную систему 1С:Предприятие
2. Разработка АРМ сотрудника рекламного отдела
3. Разработка автоматизированного рабочего места специалиста службы по наградам при администрации г. Нижневартовска
4. Разработка системы организации электронного ведения бумажного архива проектной документации предприятия ОАО
5. Разработка информационной системы обязательного автомобильного страхования
6. Создание графического редактора на языке Python
7. Автоматизация учета объектов локальных сетей предприятий
8. Расчет нагрузки на преподавателей факультета по филиалам и составление графика заездов
9. Создание автоматизированной системы паспортизации и мониторинга конфигурации АРМ в ЗАО "Тюмбит АСУ"
10. Создание системы управлением сайта на платформе ASP.NET
11. Организация работы с удаленными базами данных с использованием Web-технологий
12. Информационная система учета контингента студентов на примере ИМиКН ТюмГУ
13. Средства защиты от несанкционированного доступа в ASP.NET на примере реализации Интернет-магазина
14. Применение нейросетей в экономических задачах
15. Разработка инструмента подготовки контента
16. Учет и распределение информационных ресурсов Института дистанционного образования ТюмГУ
17. Использование альтернативных баз данных

Литература

Основная:

1. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++, 2-е изд,.Пер. с англ. -М.: "Издательство Бином", СПб:"Невский диалект", 1998. -560с.

2. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. - М.:ДМК, 2000. -432с.
3. Липаев В.В., Филинов Е.Н. Мобильность программ и данных в открытых информационных системах. - М.: Научная книга, -1997. -368с.
4. Боггс У.,Боггс М. UML и Rational Rose,Пер. с англ. -М.: Издательство "ЛОРИ", 2000. -580с.
5. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. -М. Финансы и статистика, 2000. -352с

Дополнительная:

6. Бадд Т. Объектно-ориентированное программирование в действии. СПб.; Питер, 1997. - 464с.
7. Беленков Е.Л. Системы документооборота в электронных офисах //Информатизация и связь. 1998. №2. С. 3 4- 3 8.
8. Объектно-ориентированная технология CORBA. Обзор фирмы Jet Infosystem.
9. Система автоматизации делопроизводства и документооборота ДЕЛО-96: Введение в систему. М.: ЗАО Электронные офисные системы, 1997.67с.
10. Стоунбрейкер Михаэл. Объектно-реляционные системы баз данных // Открытые системы, 1994. Вып. 4(8). С.34-39.
11. Чечкин А.В. Математическая информатика. М.: Наука, 1991. - 416с.