Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Ю. П. Ехлаков

ВВЕДЕНИЕ В ПРОГРАММНУЮ ИНЖЕНЕРИЮ

Учебное пособие

Томск
«Эль Контент»
2011

УДК 004.41
ББК 32.973.26-018.2я73
Е 934

Рецензенты:
Тарасенко В. Ф., докт. техн. наук, проф. кафедры системного анализа и управления Томского государственного университета;
Кручинин В. В., докт. техн. наук, проф. кафедры промышленной электроники ТУСУРа.

Ехлаков Ю. П.

Е 934 Введение в программную инженерию : учебное пособие / Ю. П. Ехлаков. — Томск: Эль Контент, 2011. — 148 с.

ISBN 978-5-4332-0018-0

Раскрывается содержание программной инженерии как специфического вида деятельности по разработке и продвижению на рынок программных продуктов. Предназначен для студентов младших курсов по направлению подготовки 231000 «Программная инженерия», 8085000 «Бизнес-информатика», 230100 «Информатика и вычислительная техника».

УДК 004.41
ББК 32.973.26-018.2я73

ISBN 978-5-4332-0018-0

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Традиционный подход к программированию как к «искусству» создания уникальных программ профессионалами-одиночками уходит в прошлое. В настоящее время производство и продажа программных продуктов приобрели черты высокорентабельного вида бизнеса. Это связано как с низкой материалоемкостью процессов производства, так и с высокой долей интеллектуального труда создателей программ.

Сам процесс разработки и продвижения программ приобретает черты промышленного производства с эффективной организацией труда команды программистов, регламентацией и структуризацией технологии создания программных продуктов.

При этом жесткая конкуренция разработчиков в сфере IT-сектора экономики объективно требует от компании обращать должное внимание как на качество создаваемых продуктов, так и на экономику и маркетинг их разработки и продвижения на рынок. Кроме того, индустриальные способы производства должны основываться на современных инструментальных средствах создания программных продуктов, поддерживающих все этапы жизненного цикла, начиная от выявления требований пользователей и заканчивая поставкой им готового продукта, снабженного качественной документацией.

Все эти вопросы входят в круг интересов программной инженерии как методологии промышленного создания и продвижения на рынок качественных и экономичных программных продуктов и систем.

Целью изучения дисциплины «Введение в программную инженерию» является формирование у студента осознания социальной значимости будущей профессии, мотивации к получению профессиональных знаний, пониманию и освоению основных концепций и содержанию программной инженерии как методологии индустриального проектирования прикладных программных продуктов.

Процесс изучения дисциплины направлен на *формирование следующих компетенций*:

1) владение культурой мышления, способность к обобщению, анализу, восприятию информации, постановке цели и выбору путей ее достижения (*ОК-1*);
2) осознание социальной значимости своей будущей профессии, обладание высокой мотивацией к выполнению профессиональной деятельности (*ОК-8*);
3) понимание основных концепций, принципов, теорий и фактов, связанных с информатикой (*ПК-1*);

4) умение готовить презентации, оформлять научно-технические отчеты по результатам выполненной работы, (*ПК-5*).

В результате изучения дисциплины *студент должен*:

знать: причины становления и современное состояние программной инженерии; основные этапы жизненного цикла промышленной разработки и области применения прикладных программных продуктов.

уметь: работать с научно-технической литературой; ясно и конкретно излагать материал, связанный с будущей профессиональной деятельностью.

В первом разделе учебного пособия рассматриваются объективные предпосылки возникновения программной инженерии, специфика инженерной деятельности программиста, требования к выпускнику как к специалисту в области программной инженерии. Технологические процессы создания программных продуктов представлены в виде совокупности взаимосвязанных этапов по преобразованию исходных требований пользователя в готовый продукт. Основу технологического процесса составляют: модели и методы описания бизнес-процессов предметной области, жизненного цикла создания программных продуктов, CASE-технологий индустриального проектирования программ; также описываются отечественные стандарты, регламентирующие процессы жизненного цикла разработки и оценки качества программных систем. Сам процесс проектирования описан в данном разделе в виде сжатого изложения руководства к своду знаний программной инженерии.

Во втором разделе учебника излагаются основы управления программными проектами: определение программного проекта; цели, результаты и ограничения по его созданию; приводится описание основных процессов управления проектом. Приводятся основные понятия и определения риска проекта, перечисляются и описываются основные этапы управления рисками, перечисляются причины появления рисков и пути их снижения. Проблема командной работы над проектом рассматривается как с точки зрения формирования команды, так и создания условий для ее эффективной работы, определяются роли каждого из участников, описываются профессиональные и психологические особенности программиста.

Третий раздел учебного пособия посвящен вопросам описания особенности промышленного рынка программных продуктов, вводятся понятия товара и услуги на этом рынке, описываются роль и проблемы каждого из участников рынка. Производится классификация программных продуктов, описываются методы продвижения ПП в среде Интернет.

Раскрываются основные положения формирования политики ценообразования: существующие типы рынков, возможные цели ценовой политики, формы и методы ценообразования.

В учебном пособии часто встречаются такие выражения, как программное обеспечение (ПО), программная система (ПС), программный продукт (ПП), комплекс программ (КП); в общем случае программное обеспечение необходимо рассматривать как совокупность программных средств, предназначенных для решения тех или иных задач, а программную систему, программный продукт, комплекс программ — как понятия, раскрывающие конкретное содержание ПО, имеющие конкретное название, торговую марку, снабженные документацией.

Соглашения, принятые в книге

Для улучшения восприятия материала в данной книге используются пиктограммы и специальное выделение важной информации.

Эта пиктограмма означает определение или новое понятие.

Пример

Эта пиктограмма означает пример. В данном блоке автор может привести практический пример для пояснения и разбора основных моментов, отраженных в теоретическом материале.

Контрольные вопросы по главе

Глава 1

ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

1.1 Предназначение и основные понятия программной инженерии

Впервые термин программная инженерия (Software Engineering) появился в 1968 году, когда при содействии НАТО была проведена конференция с одноименным названием. Связана она была с тем, что тогда окрестили как «программный кризис» — программирование не успевало за быстрым ростом технических средств (дала сбои система резервирования и управления потоками в авиатранспорте). Основные причины возникновения кризиса заключались в следующем [1]:

- реальные программные проекты выполнялись с отставанием от графика или с превышением сметы расходов;
- программные продукты не обладали требуемыми функциональными возможностями, производительность программного продукта низка, качество не удовлетворяет потребителей;
- возникающая в таких условиях конкурентная борьба существенно затрудняла возможность выпуска качественного программного обеспечения в приемлемые сроки;
- методы и процессы программирования, которые эффективно работали для одного человека или для небольшой команды при разработке программ умеренных размеров, не приводили к успеху при разработке крупных и сложных систем.

Появление неудачных программных проектов объяснялось следующими факторами:

- нечеткая и неполная формулировка требований к ПП,
- недостаточное вовлечение пользователей в работу над проектом,

- отсутствие необходимых ресурсов,
- неудовлетворительное планирование,
- частое изменение требований и спецификации, новизна используемой технологии проектирования ПП,
- отсутствие грамотного управления проектом, недостаточная поддержка со стороны высшего руководства.

Постепенно общество пришло к пониманию, что отношение к разработке программного обеспечения должно быть более серьезным, возникла объективная потребность в соответствующей регламентации и стандартизации этих процессов.

Сегодня программная инженерия является прикладной отраслью фундаментальной математики и информатики, экономики, менеджмента. Как инженерная дисциплина, программная инженерия охватывает все аспекты создания ПО, начиная от формирования требований до создания, сопровождения и снятия с эксплуатации ПО, а также включает инженерные методы оценки трудозатрат, стоимости, производительности и качества ПО.

В настоящее время существует несколько альтернативных определений программной инженерии:

Установление и использование правильных инженерных принципов (методов) для экономичного получения надежного и работающего на реальных машинах программного обеспечения.

Программная инженерия является такой формой инженерии, которая применяет принципы информатики и математики для получения рентабельных решений в области программного обеспечения.

Программная инженерия — методология разработки, внедрения и сопровождения в рамках имеющегося бюджета программного обеспечения с заданными уровнем качества и сроками реализации.

Каждое из этих определений содержит отдельные аспекты, повлиявшие на общее понимание программной инженерии. Однако им всем присуща одна общая черта — программная инженерия это нечто большее, чем просто написание программного кода.

При создании новых продуктов существует набор требований, являющихся не только общими к любой инженерной деятельности, но и существенными для описания основ инженерии как таковой. Основными из этих требований являются: *удовлетворение заданным (возможно, неформальным) функциональным требованиям, накладываемым пользователями; явным и неявным требованиям по эксплу-*

атационным параметрам и ресурсопотреблению; явным и неявным критериям дизайна продукта; требованиям к самому процессу разработки (продолжительности, стоимости, качеству).

Именно эти требования и определяют следующую специфику инженерной деятельности в целом [2]:

1) Инженеры в своей деятельности принимают ряд решений, тщательно оценивая альтернативы и выбирая в каждой точке принятия решения подход, оптимально соответствующий решаемой задаче с учетом существующего контекста. Выбор подхода осуществляется в процессе анализа альтернатив, во время которого тщательно сопоставляются возможные затраты и ожидаемая прибыль.
2) Инженеры, по возможности, работают с использованием измеримых количественных характеристик; они совершенствуют и уточняют существующие методы измерений и при необходимости выдают приближенные решения на основе опыта и эмпирических данных.
3) Инженеры придают особое значение соблюдению технологического процесса при осуществлении проекта и понимают важность вопросов эффективной организации командной работы.
4) Инженеры могут отвечать за выполнение самого широкого спектра задач, начиная с исследований, разработки, проектирования, производства, тестирования, внедрения, эксплуатации и управления и заканчивая продажами, сопровождением, консультированием и обучением.
5) Инженеры в процессе выполнения своих обязанностей широко используют инструментальные средства. Поэтому выбор и использование наиболее эффективных средств является крайне важным вопросом.
6) Инженеры повторно используют результаты проектирования и проектные артефакты. Проектирование является важной составляющей любой инженерной деятельности и играет критически важную роль при разработке программного обеспечения.

Следует отметить, что одновременно со значительным сходством между программной инженерией и традиционной инженерией существуют и некоторые отличия:

- Основанием программной инженерии является информатика, а не естественные науки.
- Основной упор делается на дискретной, а не на непрерывной математике.
- Концентрация на абстрактных/логических объектах вместо конкретных/физических артефактов.
- Отсутствие «производственной» фазы в традиционном промышленном смысле.
- «Сопровождение» программного обеспечения в основном связано с продолжающейся разработкой или эволюцией, а не с традиционным физическим износом.

Для того чтобы оценивать возможные решения с учетом различных факторов, связанных с функционированием, стоимостью, производительностью и техноло-

гичностью, инженер должен обладать опытом и образованием в *соответствующей предметной области*. Поэтому выпускники, специализирующиеся на программной инженерии, должны быть знакомы хотя бы с одной из прикладных предметных областей. То есть они должны понимать круг задач, которые определяют предметную область, а также общие подходы, включая стандартные компоненты (если таковые есть), используемые в производстве программного обеспечения для решения задач данной предметной области. Эффективное использование специфических для предметной области методов, средств и компонентов в большинстве случаев обеспечивает успешность разработок с использованием программной инженерии.

Программная инженерия как профессия имеет *определенные обязательства перед обществом*. Продукты, созданные программистами, могут оказать как позитивное, так и негативное влияние на жизнь и деятельность пользователей. Очевидно, что разработчики программного обеспечения должны соблюдать определенную этику и профессионализм в своей деятельности. Кодекс этики программной инженерии определяет мораль, правила и нормы поведения профессионалов в области программной инженерии, их обязательства и ответственность по отношению к обществу и друг к другу. Кодекс состоит из преамбулы и совокупности принципов, которых должны придерживаться профессионалы [3].

Преамбула к «Кодексу этических норм профессионала в области программной инженерии» формулирует это следующим образом:

1) Вследствие специфики своих ролей в процессе создания программного обеспечения инженеры имеют неограниченные возможности приносить пользу или причинять вред как самостоятельно, так и способствуя другим либо влияя на других.
2) Инженеры должны принять на себя обязательство сделать программную инженерию полезной и уважаемой профессией, чтобы быть уверенными в том, что их работа используется во благо.

В кодексе задекларировано восемь принципов, которые касаются соответственно:

1) согласования профессиональной деятельности с интересами общества;
2) взаимоотношений между пользователем, работодателем и исполнителем разработки;
3) достижения соответствия качества программного продукта лучшим профессиональным стандартам;
4) соблюдения честности и независимости при профессиональных оценках качества процессов и продуктов;
5) соблюдения этических норм в менеджменте и в сопровождении разработок;
6) поддержки становления профессии в соответствия с кодексом этики;
7) соблюдения этических норм во взаимоотношениях между коллегами;
8) постоянного совершенствования программной инженерии как области профессиональной деятельности.

В [2] отмечается, что по окончании университетского обучения выпускники направления подготовки «Программная инженерия» должны:

1) Демонстрировать владение знаниями и навыками в области программной инженерии, а также иметь профессиональные качества, необходимые для начала работы в качестве инженера по программному обеспечению.

   Студентам необходимо развивать уверенность в своих возможностях посредством постоянного укрепления знаний и практики на протяжении всего периода обучения программной инженерии. Также студенты должны обрести понимание и способность самостоятельно решать профессиональные вопросы, связанные с этикой профессионального поведения, экономикой и общественными потребностями.

2) В процессе работы над программными продуктами быть способными эффективно решать поставленные перед ними задачи как индивидуально, так и в команде. В реальной жизни студентам предстоит выполнять значительное количество проектов в одиночку, однако большинство задач требует работы в команде с другими людьми. Соответственно, студенты должны овладеть максимально полной информацией о сущности работы коллектива и ролях в команде. Они должны понимать важность таких вопросов, как необходимость придерживаться установленных сроков и оценка как индивидуальной, так и командной производительности в работе над проектом.

3) Разрешать возникающие противоречия в стоящих перед проектом целях, находя приемлемые компромиссы в рамках существующих ограничений (стоимость, время, качество и т. п.). В процессе обучения студенты должны выполнять задания, умышленно содержащие противоречивые и даже изменяющиеся требования.

4) Проектировать решения в одной или более предметных областях, используя подходы программной инженерии, балансирующие этические, общественные, юридические и экономические интересы различных заинтересованных сторон.

   На протяжении обучения студентам необходимо научиться использовать множество различных подходов к инженерному проектированию в общем и к решению специфических проблем в конкретных предметных областях в частности. Студенты должны понимать достоинства и недостатки различных доступных альтернатив и последствия выбора того или иного подхода в каждой конкретной ситуации. В предлагаемых ими проектных решениях должны адекватно учитываться этические, общественные, юридические, экономические факторы, а также вопросы безопасности.

5) Демонстрировать понимание и способность к применению передовых теорий, моделей и методов, которые обеспечивают современную базу для идентификации и анализа проблем, проектирования, разработки, реализации, аттестации и документирования программного обеспечения.

   В этом отношении существенным является итоговый дипломный проект, который представляет собой крайне важную деятельность, логически завершающую обучение. Дипломный проект дает студентам возможность самостоятельно либо в команде выполнить реальный проект и продемонстрировать умение объединять знания из различных курсов и эффективно их применять. Это позволяет студентам продемонстрировать понимание

широкого спектра знаний программной инженерии и способность применять приобретенные навыки для достижения желаемого эффекта, а также критически оценивать собственные действия и достижения.

6) Демонстрировать понимание важности и способность к ведению переговоров, способность результативно работать, осуществлять руководство и эффективно общаться с заинтересованными лицами в типичных для разработки программного обеспечения ситуациях.

   В программе обучения обязательно должно быть предусмотрено осуществление хотя бы одной достаточно серьезной деятельности, требующей разработки решения для некоторого заказчика. Программные инженеры должны осознавать, что им необходимо создавать программное обеспечение, прежде всего являющееся полезным. По возможности, необходимо объединить в программе обучения период производственного опыта, лекции приглашенных практикующих инженеров и даже участие в конкурсах по созданию программного обеспечения. Все вместе это способствует получению более насыщенного опыта и созданию необходимой среды для подготовки высококвалифицированных специалистов по программной инженерии.

7) Изучать новые модели, методы и технологии по мере их появления, а также осознавать необходимость постоянного профессионального роста. По завершении программы обучения студенты должны продемонстрировать способность и стремление к самообучению на протяжении всей жизни.

1.2 Основные положения индустриального проектирования программных продуктов

1.2.1 Основные компоненты технологии создания программных продуктов

Процесс разработки программного обеспечения представляет собой специфический технологический процесс преобразования исходных требований заказчика о предметной области в готовый программный продукт и состоит из совокупности взаимосвязанных этапов (технологических операций) (рис. 1). В процессе преобразования участвуют специалисты, выполняющие определенную работу с помощью специфических инструментальных средств и предметов производства. На вход этого процесса в виде требований поступают сведения о предметной области. На выходе процесса — разработанное программное обеспечение и техническая документация. Механизмами, обеспечивающими выполнение процесса, являются специалисты и используемые ими инструментальные средства. Весь процесс протекает в соответствии с принятыми на предприятии стандартами и нормативными документами, регламентирующими требования к процессам жизненного цикла, качества, документирования программного продукта и оценке технологической зрелости организаций-разработчиков. К такими документам в частности относятся:

- Международный стандарт ISO/IEC 12207 «Информационная технология. Жизненный цикл процессов разработки программного обеспечения».

- ГОСТ Р ИСО/МЭК 12207-99 «Информационная технология. Процессы жизненного цикла программных средств».
- ГОСТ Р ИСО/МЭК 15288-2005 «Системная инженерия. Процессы жизненного цикла систем».
- ГОСТ Р ИСО/МЭК 15910-2002 «Процесс создания документации пользователя программного средства».
- ГОСТ Р ИСО 9127-94 «Документация пользователя и информация на упаковке для потребительских программных пакетов».
- ГОСТ Р ИСО/МЭК 9126-93 «Оценка программного продукта. Характеристики качества и руководящие указания по их применению».
- CMM — Capability Maturity Model (Модель зрелости процесса конструирования ПО).

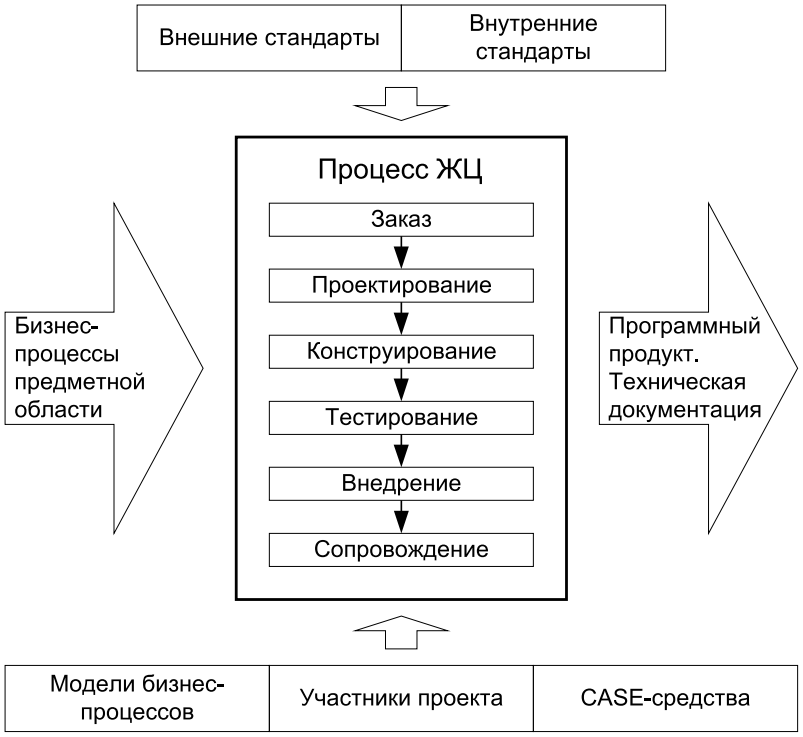


Рис. 1.1 – Модель технологического процесса создания программного продукта

Исходя из вышеизложенного можно констатировать, что технологический процесс (схема) создания программного продукта сводится к преобразованию сведений о какой-либо предметной области в комплекс исполняемых программных модулей, способных обеспечить решение определенных задач пользователей, такие схемы получили название моделей жизненного цикла (ЖЦ).

1.2.2 Модели описания бизнес-процессов предметной области

Предметная область представляет собой набор бизнес-процессов, адекватно описывающих деятельность организации по удовлетворению общества в определенных продуктах и услугах.

В свою очередь, под бизнес-процессом понимается:

1) *совокупность взаимосвязанных и взаимодействующих видов деятельности, преобразующая входы в выходы, предоставляющие ценность для клиента;*
2) *логические серии взаимозависимых действий, которые используют ресурсы предприятия для создания или получения в обозримом или измеримо предсказуемом будущем полезного для заказчика выхода, такого как Продукт или Услуга;*
3) *множество внутренних упорядоченных видов деятельности организации по преобразованию исходных ресурсов в готовую продукцию (услугу.)*

Первым шагом по выявлению бизнес-процессов должно стать выделение основных продуктов (услуг) и выстраивание процессов в соответствии с продуктовыми линиями. Это позволяет получить продуктовые «срезы» бизнес-процессов, протекающих в организации. Дальнейшая декомпозиция основных процессов производится по модели жизненного цикла продукта.

Жизненный цикл — строго упорядоченная совокупность процессов, описывающих эволюционное преобразование исходных ресурсов в конечные продукты и услуги (рис 1.2)

Основным инструментом анализа и совершенствования бизнес-процессов является *моделирование*.

Модель бизнес-процесса — это наглядный образ реального процесса, позволяющий акцентировать внимание на его сущности и особенностях.

Модель существующих (текущих) бизнес-процессов дает представление о том, *как* функционирует бизнес, каким образом происходит преобразование входных ресурсов в продукцию (услуги). Такая модель называется *«Как есть»*. Модель про-

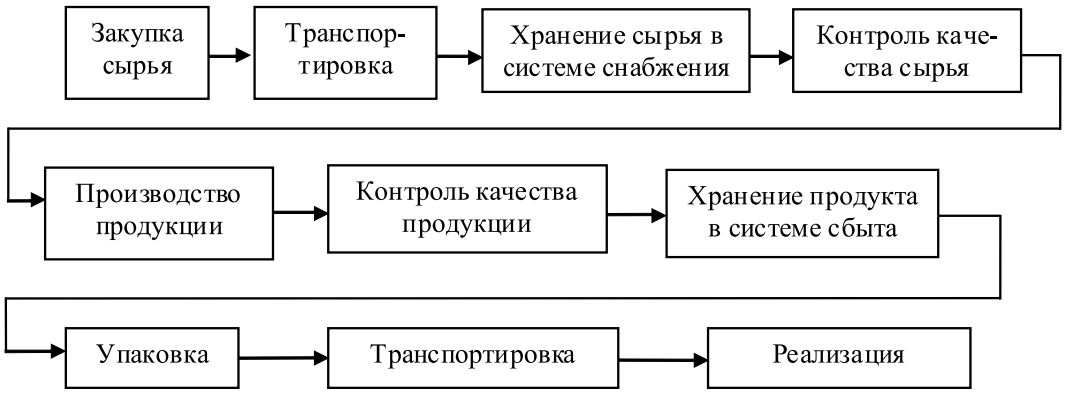


Рис. 1.2 – Модель жизненного цикла создания материального продукта

ектируемых (усовершенствованных) бизнес-процессов дает представление о том, *как должен* функционировать бизнес, чтобы достигались поставленные цели. Это — так называемая модель *«Как должно быть»*.

Для построения бизнес-процессов широкое распространение получили структурные и объектно-ориентированные подходы [4] (рис. 1.3).

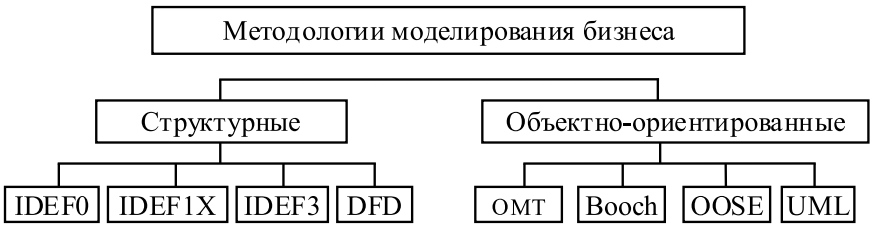


Рис. 1.3 – Классификация методологий моделирования бизнеса

В основе *структурных методов моделирования* бизнеса лежит декомпозиция системы на подсистемы, которые, в свою очередь, делятся на более мелкие подсистемы и т. д. Базовыми принципами структурного подхода являются:

- «разделяй и властвуй» — принцип решения сложных проблем путем их разбиения на множество мелких задач, легких для понимания и решения;
- иерархическое упорядочивание — принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

Наибольшее распространение получили следующие методы структурного моделирования:

- IDEF0 — функциональные модели, основанные на методе структурного анализа и проектирования SADT (Structured Analysis and Design Technique) Дугласа Росса;
- IDEF1X — модели данных, основанные на диаграммах «сущность-связь» (ERD, Entity-Relationship Diagrams);
- IDEF3 — диаграммы потоков работ (Work Flow Diagrams);
- DFD (Data Flow Diagrams) — диаграммы потоков данных.

Главным структурообразующим элементом *в объектно-ориентированном подходе* является объект. При моделировании бизнеса объектами являются, прежде всего, участники бизнес-процесса (активные объекты) — организационные единицы, конкретные исполнители, информационные системы, а также пассивные объекты — материалы, документы, оборудование, над которыми выполняют действия активные объекты. В программировании объектом называется информационная структура, объединяющая данные (атрибуты) и процедуры. Таким образом, в объектно-ориентированных методах модель бизнес-процессов строится вокруг участников процессов и их действий. Язык моделирования UML является общепризнанным стандартом в области объектно-ориентированного моделирования.

Методология моделирования IDEF0

Методология IDEF0 является одной из самых известных и широко используемых методологий моделирования и базируется на методе SADT (Structured Analysis and Design Technique) Росса, предназначенном для решения широкого спектра проблем, включая разработку программного обеспечения, бизнес-анализ, проектирование, планирование и управление производственными системами, управление финансами и материально-техническими ресурсами.

IDEF0-модель использует графический язык для отражения информации о конкретной системе. Модель состоит из диаграмм и фрагментов текста. На диаграммах все функции системы и их взаимодействия представлены как блоки (функции) и дуги (отношения) [5].

Основной конструкцией модели является функциональный блок, представленный в виде прямоугольника и отображающий некоторую функцию (действие, процесс, операцию). Внутри блока записывается его наименование. Оно должно содержать глагол или отглагольное существительное. Например: «разработать проект», «изготовление продукта», «планирование».

Дуги, изображаемые на диаграмме в виде линий со стрелками на конце, играют роль связей блоков с внешней для них средой. Каждая из дуг имеет метку, характеризующую ее. Назначение дуг зависит от стороны блока, в которую стрелка входит или выходит (рис. 1.4):

- «вход» (I — input) — дуги, входящие слева от блока. Они представляют собой предметы или данные, необходимые для выполнения функции блока (сырье, материалы, исходная информация);
- «выход» (O — output) — дуги, выходящие справа из блока. Они показывают предметы или данные, полученные в результате выполнения функции (продукция, услуга, выходные данные);
- «управление» (C — control) — дуги, входящие сверху блока. Они описывают условия или данные, которые управляют выполнением функции (инструкции, требования, стандарты);
- «механизм» (M — mechanism) — дуги, входящие снизу блока. Они обозначают исполнителей или средства, выполняющие функцию (персонал, подразделения фирмы, оборудование, инструменты, информационная система).

Выход и вход показывают, что и из чего делается функциональным блоком, управление показывает, как и почему это делается, а механизм показывает, кем и с помощью чего это делается.

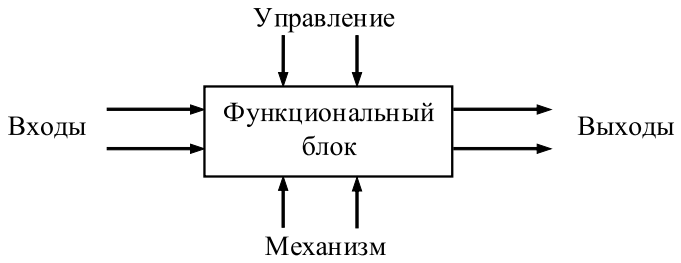


Рис. 1.4 – Функциональный блок IDEF0-диаграммы

Функциональный блок может быть декомпозирован, т. е. представлен в виде совокупности других взаимосвязанных функциональных блоков, которые детально описывают исходный блок. Таким образом, IDEF0-модель состоит из набора иерархически связанных диаграмм (рис. 1.5). На диаграмме корневого уровня представлена вся система в виде одного блока и дуг, изображающих связи с внешним окружением. На диаграмме декомпозиции первого уровня система представлена более детально в виде совокупности блоков-подмодулей, соединенных дугами друг с другом и с окружением. На диаграммах декомпозиции следующего уровня детализируются блоки диаграммы первого уровня и т. д. [4].

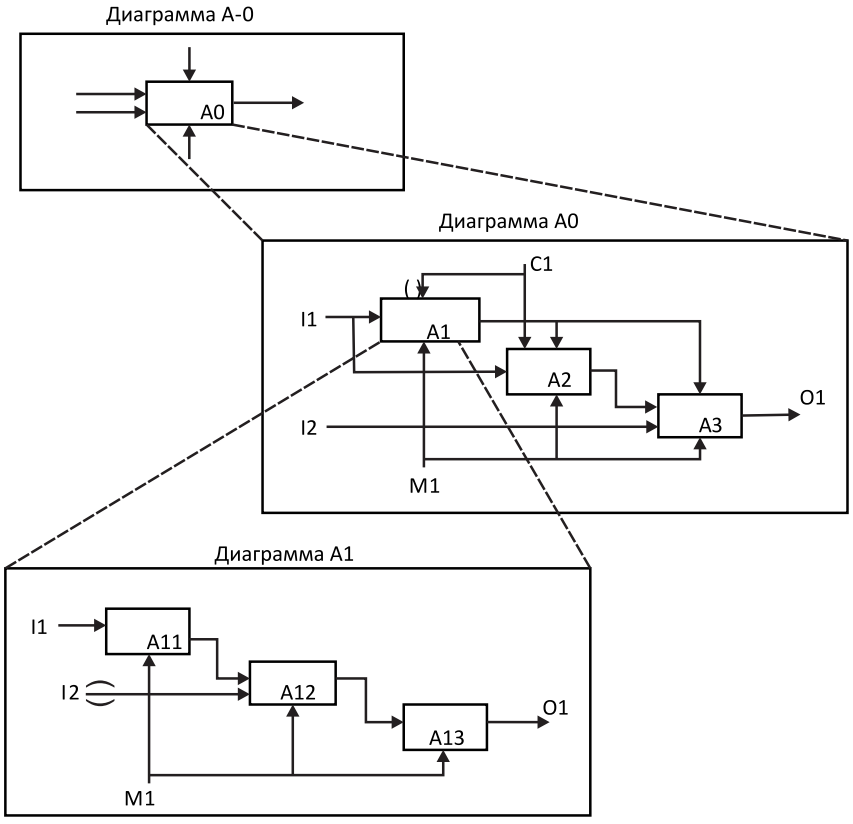


Рис. 1.5 – Иерархия диаграмм IDEF0-модели

Пример декомпозиции бизнес-процесса «Создание продукта» (рис. 1.6 и рис. 1.7).

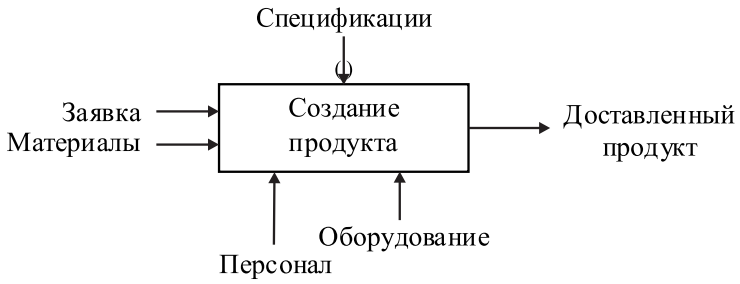


Рис. 1.6 – Пример контекстной диаграммы

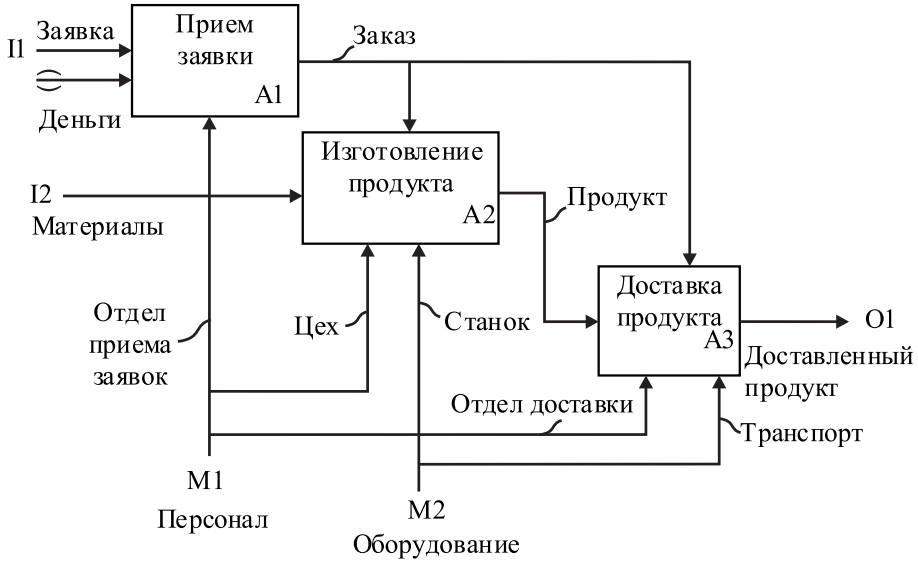


Рис. 1.7 – Пример диаграммы декомпозиции

Объектно-ориентированное моделирование

В основу объектно-ориентированного моделирования с использованием языка UML положен ряд ключевых принципов.

Принцип абстрагирования, который предписывает включать в модель только те аспекты проектируемой системы, которые имеют непосредственное отношение к выполнению системой своих функций или своего целевого назначения.

Принцип многомодельности, который сводится к утверждению о том, что никакая отдельно взятая модель не может с достаточной степенью адекватности описать различные аспекты предметной области и ее основные бизнес-процессы. Применительно к UML это означает, что достаточно полная модель предметной области допускает некоторое число взаимосвязанных представлений, каждое из которых адекватно отражает некоторый аспект функционирования или структуры предметной области. В этом случае интегрированная модель предметной области представляется в виде совокупности диаграмм (рис. 1.8) [6].

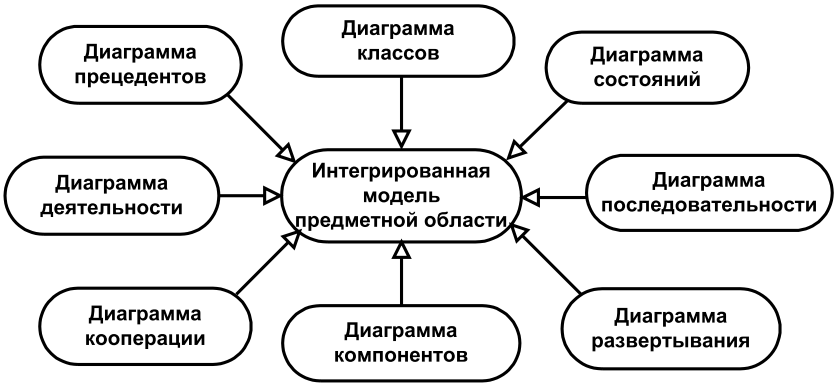


Рис. 1.8 – Интегрированная модель предметной области в нотации UML

Моделирование бизнес-процессов предметной области с помощью UML возможно на основе последовательного построения диаграмм: *прецедентов, деятельности и последовательности.*

Диаграмма прецедентов описывает бизнес в виде графа специального вида, основными элементами которого являются прецеденты, актеры и отношения между ними. Каждый прецедент соответствует отдельному бизнес-процессу, который содержит законченную последовательность действий, приводящую к значимому для потребителя результату. Примеры прецедентов: производство продукта, продажа продукта, сервисное обслуживание, разработка продукта, маркетинг и сбыт. В некоторых русскоязычных изданиях прецеденты называют вариантами использования.

Согласно спецификации UML прецеденты обозначаются эллипсом, внутри которого содержится поясняющий текст, называемый именем прецедента. Имена прецедентам формулируются либо глаголом, либо существительным, обозначающим действие и поясняющим слова.

Любую сущность, взаимодействующую с бизнес-процессом и являющуюся его потребителем либо инициатором, называют актером. В роли актера может выступать человек, техническое устройство, программа или другая система, служащая источником воздействия на бизнес-процессы. Стандартным обозначением актера является пиктограмма (рис. 1.9), под которой располагается имя актера.

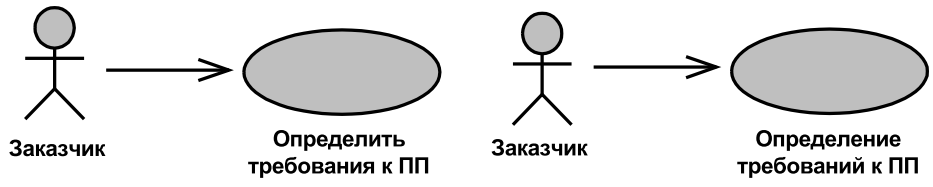


Рис. 1.9 – Эквивалентные прецеденты

Между прецедентами и актерами устанавливаются *отношения коммуникации* которые описывают информационные, материальные и финансовые потоки между ними. Все прецеденты, вводимые в модель, должны быть связаны с актерами: предметная область не должна содержать бизнес-процессы, которые никем не востребованы.

Наиболее важным для описания прецедента является документ, называемый *«потоком событий»*. Он описывает сценарии осуществления прецедента в виде последовательности шагов процесса. Поток событий прецедента может быть представлен в виде *диаграммы деятельности.* На рис. 1.10 приведены условные обозначения основных элементов диаграммы.

Рис. 1.10 – Элементы диаграммы деятельности: *а* — начальное состояние; *б* — конечное состояние; *в* — действие; *г* — переход; *д* — ветвление; *е* — синхронизация

Каждый шаг (событие) прецедента представляет собой некоторое действие, переводящее прецедент в новое состояние. В свою очередь, новое состояние прецедента является стимулом для выполнения следующего шага (события). Таким образом, прецедент рассматривается как *машина состояний-событий.*

Пример

На рис. 1.11 приведен пример диаграммы, иллюстрирующий ход событий прецедента «Продажа продукта».

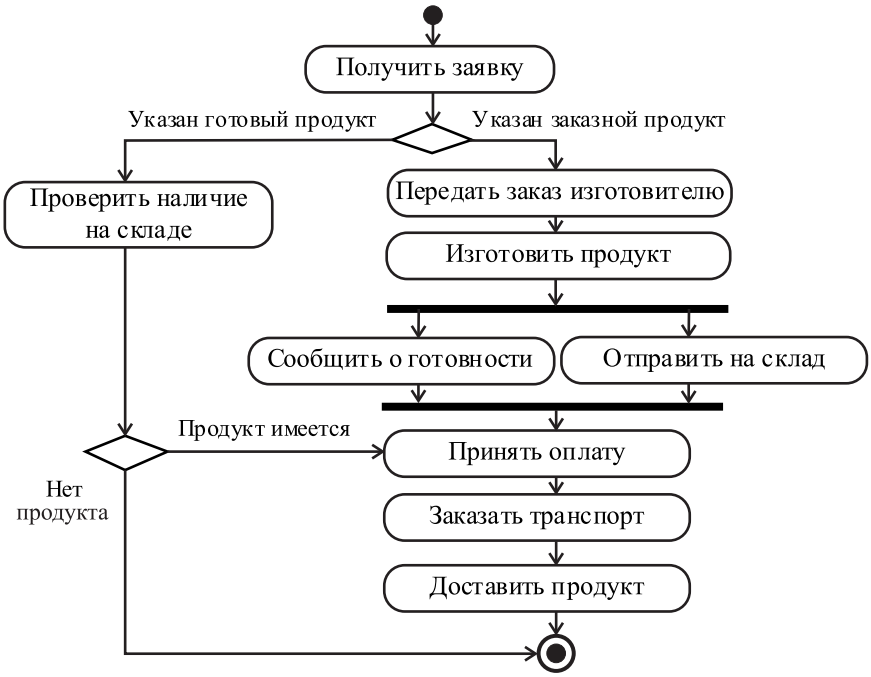


Рис. 1.11 – Диаграмма деятельности прецедента «Продажа продукта»

Для того чтобы отразить участие актеров во время выполнения бизнес-процессов, используется *диаграмма последовательности* (рис. 1.12).

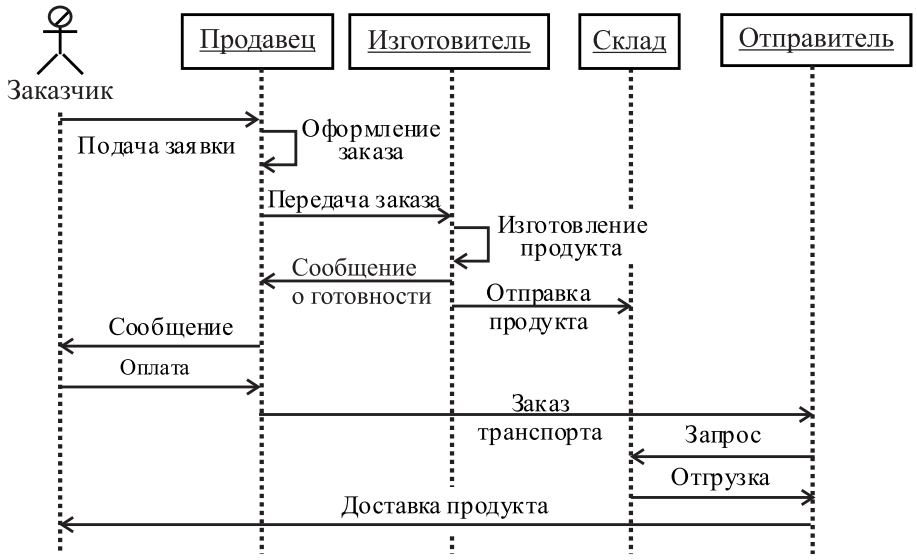


Рис. 1.12 – Диаграмма последовательности прецедента «Продажа продукта»

Каждый актер, участвующий в реализации прецедента, изображается в верхней части диаграммы в виде прямоугольника, от которого вниз проведена линия («линия жизни»). Внутри прямоугольника записывается имя актера.

Между объектами (актерами) устанавливаются отношения сообщений, отражающие аналогично отношениям коммуникации передачу информации (или некоторый материальный поток) между объектами. Сообщение изображается отрезком горизонтальной линии со стрелкой, проведенной от линии жизни объекта (актера), посылающего сообщение, до линии жизни объекта (актера), получающего сообщение. При этом прием сообщения инициирует выполнение определенных действий тем объектом, которому сообщение передано.

Сообщения должны быть упорядочены по времени: первое сообщение изображается вверху диаграммы, следующее — ниже, следующее — еще ниже и т. д.

Использование описанных выше методологий бизнес-процессов «Как должно быть» позволяет упростить взаимодействие заказчиков с разработчиками и разработчиков между собой при проектировании и создании на базе этих моделей программных продуктов.

1.2.3 Модели жизненного цикла программных продуктов

В общем случае модель жизненного цикла описывается в виде последовательности процессов, работ и задач, обеспечивающих разработку, эксплуатацию и сопровождение программного продукта и отражающих эволюцию изменения продукта, начиная от формулировки требований к ней до прекращения ее использования.

Большинство существующих моделей жизненного цикла ПО являются разновидностями трех классических моделей:

- каскадной (ступенчатой или водопадной);
- эволюционной (итеративной или инкрементальной);
- спиральной.

Одной из первых, применяемых на практике моделей была каскадная модель, в которой каждая работа выполняется один раз и в том порядке, как они представлены в выбранной модели ЖЦ.

При этом делается допущение, что каждая работа будет выполнена настолько тщательно, что после ее завершения и перехода к следующему этапу возвращения к предыдущему не потребуется, (возвращение возможно только в процессе сопровождения ПО) [1].

На рис. 1.13 представлены типичные этапы каскадной модели и соответствующие этому этапу элементы программного проекта. Каскадную модель можно рассматривать как модель ЖЦ, пригодную для создания первой версии ПО, для проверки реализованных в ней функций. При сопровождении и эксплуатации могут быть обнаружены разного рода ошибки, которые будут исправляться разработчиком, начиная с первого этапа данной модели.

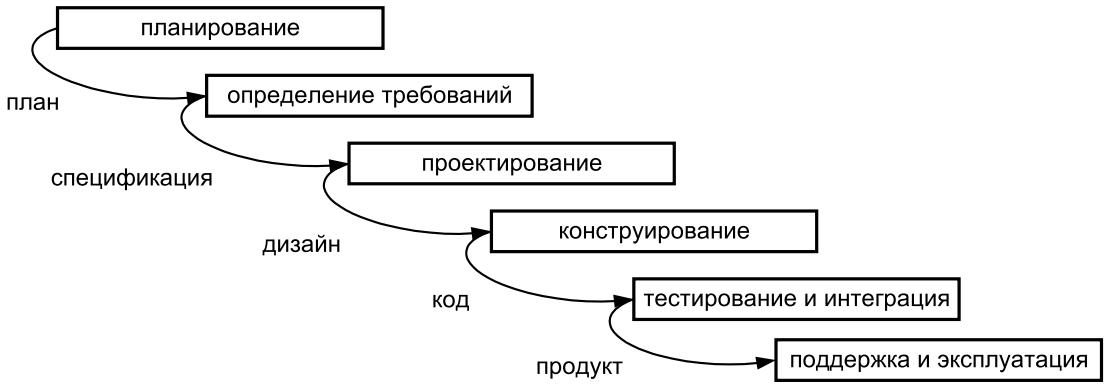


Рис. 1.13 – Каскадная модель жизненного цикла

Недостатком этой модели является то, что в основу ее концепции положена модель фабрики, где продукт проходит стадии от замысла до производства, затем передается заказчику как готовое изделие, изменение которого не предусмотрено, хотя возможна замена на другое подобное изделие в случае рекламации или некоторых ее деталей, вышедших из строя.

Спиральная модель основывается на итерационной процедуре использования каскадной модели.

Отличие этой модели от каскадной состоит в возможности обеспечения многоразового возвращения к процессу формулирования требований и к повторной

разработке с любого процесса выполнения работ. Каждый новый виток спирали предполагает создание фрагмента или версии ПО, уточняются требования к ПО, оценивается качество разработанного фрагмента ПО и планируются работы следующего витка. Таким образом, углубляются и конкретизируются все детали проектируемого ПО и в результате получается окончательный вариант, который удовлетворяет всем требованиям заказчика (рис. 1.14) [1].

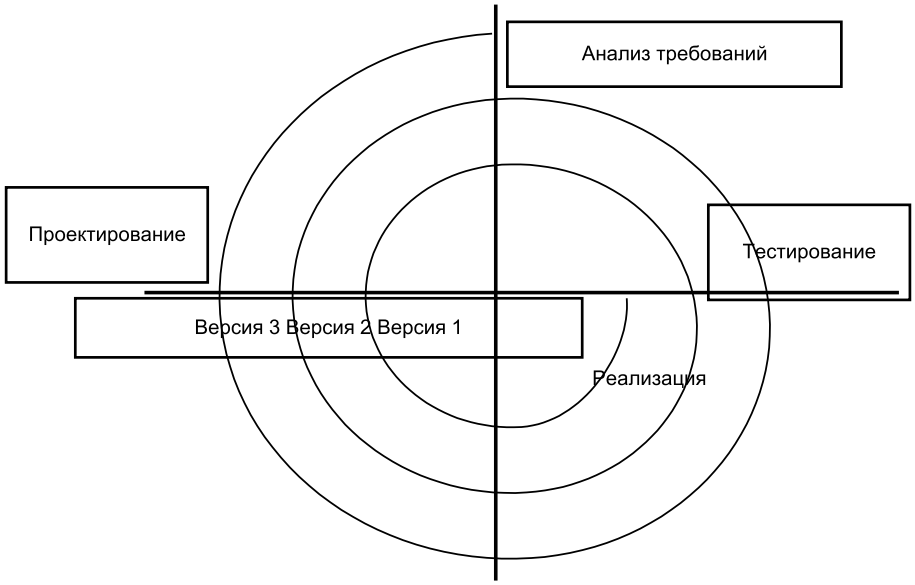


Рис. 1.14 – Спиральная модель жизненного цикла

Использование спиральной модели ориентировано на удовлетворение потребности изменений сразу, как только будет установлено, что созданные артефакты или элементы документации (описание требований, проекта, комментариев различного вида и т. п.) не соответствуют действительному состоянию разработки после внесения некоторых изменений.

Эволюционная модель предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых предполагает создание фрагментов ПО меньшей функциональности, по сравнению с требованиями заказчика.

Цель каждой итерации — получение уже на ранних этапах разработки работающей версии программной продукции, содержащей определенную функциональность.

С точки зрения структуры жизненного цикла такую модель называют *итеративной, с* точки зрения развития продукта — *инкрементальной.* В результате на каждой итерации можно анализировать промежуточные результаты работ и реакцию на них всех заинтересованных лиц и вносить корректирующие изменения на следующих итерациях. Каждая итерация может содержать полный набор принятых этапов ЖЦ: от анализа требований до ввода в эксплуатацию очередной части ПО.

Эволюционная модель подразумевает не только сборку работающей текущей (с точки зрения результатов тестирования) версии системы, но и ее развертывание в реальных операционных условиях с анализом откликов пользователей для определения содержания и планирования следующей итерации.

1.2.4 CASE-технология создания программных продуктов

Современные CASE-технологии представляют собой методологию создания программных продуктов, а так же набор инструментальных средств моделирования, проектирования и разработки, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах создания и сопровождения ПП, разрабатывать приложения в соответствии с потребностями пользователей. В основу большинства CASE-технологий положены методологии структурного или объектно-ориентированного анализа и проектирования.

Функциональные возможности CASE-средств обеспечивают автоматизацию следующих процессов жизненного цикла разработки программного обеспечения [7]:

Моделирование:

построение диаграмм — создание и редактирование диаграмм различных типов, описывающих бизнес-процессы предметной области;

графический анализ — анализ графических объектов, а также хранения и представления проектной информации в графическом представлении. В большинстве случаев графические анализаторы интегрированы со средствами построения диаграмм;

ввод и редактирование спецификаций требований и проектных спецификаций. К спецификациям такого рода относятся описания функций, данных, интерфейсов, структуры, качества, производительности, технических средств, среды, затрат и графиков;

импорт, экспорт и редактирование спецификаций с использованием формального языка спецификации требований;

моделирование данных — ввод и редактирование информации, описывающей элементы данных системы и их отношения;

моделирование процессов — ввод и редактирование информации, описывающей процессы системы и их отношения;

проектирование архитектуры ПО. Проектирование логической структуры ПО — структуры модулей, интерфейсов и др.;

имитационное моделирование — динамическое моделирование различных аспектов функционирования программного обеспечения на основе спецификаций требований и/или проектных спецификаций, включая внешний интерфейс и производительность (например, время отклика, коэффициент использования ресурсов и пропускную способность);

прототипирование — проектирование и генерация первого прототипа или его отдельных компонент на основе спецификаций требований и/или проектных спецификаций. Прототипирование в основном касается внешнего пользовательского интерфейса и осуществляется при непосредственном участии пользователей;

генерация экранных форм — генерация экранных форм на основе спецификаций требований и/или проектных спецификаций;

возможность трассировки — сквозной анализ функционирования программного обеспечения от проверки спецификации требований до корректности работы конечного функционала (установления и отслеживания соответствий и связей между функциональными и другими внешними требованиями, техническими решениями и результатами проектирования);

синтаксический и семантический контроль проектных спецификаций — контроль синтаксиса диаграмм и типов их элементов, контроль декомпозиции функций, проверка спецификаций на полноту и непротиворечивость.

Реализация:

синтаксически управляемое редактирование — ввод и редактирование исходных кодов на одном или нескольких языках с одновременным синтаксическим контролем;

генерация кода — генерация кодов на одном или нескольких языках на основе проектных спецификаций. Типы генерируемого кода могут включать обычный программный код, схему базы данных, запросы, экраны/меню;

компиляция кода — трансляция программы, составленной на исходном языке, и последующая ее компоновка в программу;

конвертирование исходного кода — преобразование кода из одного языка в другой;

анализ надежности — возможность количественно оценивать параметры надежности ПО;

реверсный инжиниринг — анализ исходных кодов и формирование на их основе проектных спецификаций;

реструктуризация исходного кода — модификация формата и/или структуры существующего исходного кода;

анализ исходного кода — определение размера кода, вычисление показателей сложности, генерация перекрестных ссылок и проверка на соответствие стандартам;

отладка — трассировка программ, выделение узких мест и наиболее часто используемых фрагментов кода и т. д.

Тестирование:

описание тестов — генерация тестовых данных, алгоритмов тестирования, требуемых результатов и т. д.;

фиксация и повторение действий оператора — фиксирование данных, вводимых тестером, их редактирование и воспроизведение в тестовых примерах;

регрессионное тестирование — повторение и модификация ранее выполненных тестов для определения различий в системе и/или среде;

автоматизированный анализ результатов тестирования — сравнение ожидаемых и реальных результатов, сравнение файлов, статистический анализ результатов и др.;

анализ тестового покрытия — проверка оснащенности средствами контроля исходного кода и анализ тестового покрытия. Проверяются, в частности, обращения к операторам, процедурам и переменным;

анализ производительности — вычисление и анализ производительности программ (использование центрального процессора, памяти, обращения к определенным элементам данных и/или сегментам кода, временные характеристики и т. д.).

Документирование:

редактирование текстов и графики — ввод и редактирование данных в текстовом и графическом форматах;

редактирование с помощью форм — ввод и редактирование данных в соответствии с формами, определенными пользователями;

возможности издательских систем — поддержка функций и форматов гипертекста, соответствие стандартам документирования, автоматическое извлечение данных из репозитория и генерация документации по спецификациям пользователя.

Управление конфигурацией:

контроль доступа и изменений — контроль доступа на физическом уровне к элементам данных и их изменениям (определения прав доступа к компонентам, извлечения элементов данных для модификации, блокировки доступа к ним на время модификации и помещения обратно в репозиторий);

отслеживание модификаций — фиксация и ведение журнала всех модификаций, внесенных в систему в процессе разработки или сопровождения;

управление версиями — ведение и контроль данных о версиях системы и всех ее коллективно используемых компонентах;

учет состояния объектов конфигурационного управления — получение отчетов обо всех последовательных версиях, содержимом и состоянии различных объектов конфигурационного управления;

генерация версий и модификаций — описание последовательности действий, требуемых для формирования версий и модификаций, и автоматическое выполнение этих действий;

архивирование — автоматическое архивирование элементов данных для последующего использования.

Современный рынок CASE-средств насчитывает сотни систем, различающихся по функциональным возможностям, применяемым методологиям, доступным платформам и цене. Существуют различные классификации CASE-средств [4]. *Классификация по уровню проектирования* в жизненном цикле создания ИС включает три категории:

1) средства верхнего уровня (Upper CASE), предназначенные для анализа предметной области, определения места информационной системы в контуре бизнес-системы;
2) средства среднего уровня (Middle CASE), использующиеся для разработки архитектуры информационной системы, создания проектных спецификаций;
3) средства нижнего уровня (Lower CASE), поддерживающие разработку программного обеспечения.

Классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы жизненного цикла и в основном совпадает с классификацией по уровням. Выделяют следующие типы:

- средства анализа предметной области (соответствуют Upper CASE);
- средства анализа и проектирования (соответствуют Middle CASE);
- средства разработки приложений (соответствуют Lower CASE).

Кроме того, выделяют вспомогательные типы CASE-средств, к которым относятся средства управления проектом, средства тестирования, документирования и т. д.

Рассмотрим основные из перечисленных типов CASE-средств, а также их роли в проектах по оптимизации бизнес-процессов.

Средства анализа предметной области. Средства этого типа позволяют формировать статическую модель предметной области (прежде всего функциональную модель), например в виде диаграмм функциональной декомпозиции или диаграмм потоков данных. Наиболее распространенные методологии, используемые данными средствами, — IDEF0 (SADT), ABC, DFD, IDEF3, UML. К средствам анализа относятся Design/IDEF (Meta Soft-ware), BPwin (Logic Works), CASE Аналитик (МакроПроджект), Rational Rose (Rational Software Corp.).

Средства анализа и проектирования. Результатом использования этих средств являются спецификации компонентов и интерфейсов информационной системы, архитектуры ИС, алгоритмов, структур данных (схем баз данных). Имеется множество методологий, используемых средствами этого типа. Например, для моделирования данных чаще всего используют диаграммы ERD, DSD, IDEF1X. Для моделирования архитектуры ИС используют различные методы структурного проектирования (например, диаграммы архитектуры системы SAD) и объектно-ориентированного проектирования (в частности, язык UML).

К средствам анализа и проектирования относятся Silverrun (CSA), Erwin (Logic Works), Designer/2000 (ORACLE), CASE Аналитик (МакроПроджект), Rational Rose (Rational Software Corp.).

Средства разработки приложений — RAD-средства и средства инжиниринга (реинжиниринга) программного обеспечения. Их основная функция — генерация программного кода на различных языках верхнего уровня, таких как C++, Object Pascal, Java, Visual Basic. При этом зачастую они используют спецификации, созданные средствами анализа и проектирования. К ним относятся Power Builder (Sybase), Delphi (Borland), 4GL (Uniface Compuware), а также генераторы кодов, входящие в состав Rational Rose (Rational Software Corp.), Silverrun (CSA), и др.

Средства управления проектом. Это средства, предназначенные для планирования хода выполнения проекта, а также для сопровождения проекта (контроля и корректировки планов выполнения работ). Поскольку в качестве проекта может выступать не только разработка информационной системы, но и любой бизнес-проект (например, разработка нового изделия, проведение рекламной компании), средства данного типа относят как к CASE-средствам, так и к средствам моделирования бизнеса.

Наиболее распространенные средства данного типа: Microsoft Project (Microsoft), Time Line (Symantec), CA-SuperProject (Computer Associates International):

- средства проектирования баз данных, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся пакеты программ: ERwin (Logic Works), S-Designer (SDP) и DataBase Designer (ORACLE). Средства проектирования баз данных имеются также в составе CASE-средств Vantage Team Builder, Designer/2000, Silverrun и PRO-IV;

- средства реинжиниринга, обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций. Средства анализа схем БД и формирования ERD входят в состав Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, ERwin и S-Designor. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программ на языке C++ (Rational Rose (Rational Software)), Object Team (Cayenne));
- средства конфигурационного управления, обеспечивающие управляемость и контролируемость процессов разработки и сопровождения ПО. К ним относятся (PVCS (Intersolv));
- средства тестирования, обеспечивающие проверку соответствия приложения предъявляемым бизнес-требованиям, или для проверки и оценки производительности приложений. К ним относятся (Quality Works (Segue Software));
- средства документирования предназначены для автоматизации разработки проектной документации на всех фазах ЖЦ ПО. К ним относятся (SoDA (Rational Software)).

Использование компьютерных инструментальных средств позволяет существенно повысить эффективность процесса моделирования и анализа бизнес-процессов, так как позволяет упростить процедуру построения моделей, уменьшить количество ошибок, сократить трудозатраты и время, повысить качество моделей и результатов анализа.

1.3 Руководство к Своду знаний по программной инженерии (Guide to the Software Engineering Body of Knowledge — SWEBOK)

Руководство к Своду Знаний по Программной Инженерии — «SWEBOK» [SWEBOK, 2004] было издано в 2004 году и включает базовые определения и описания областей знаний, которые составляют суть профессии *инженера-программиста*. [10]

Описание областей знаний в SWEBOK построено по иерархическому принципу, как результат структурной декомпозиции, и включает 10 областей знаний, которые условно можно разбить на две группы:

1) Знания, конкретизирующие жизненный цикл разработки программных продуктов:
   - *определение требований;*
   - *проектирование, конструирование;*
   - *тестирование, эксплуатация (поддержка).*
2) Знания, раскрывающие содержание методов и инструментария разработки программных продуктов:

- *конфигурационное управление;*
- *управление инженерной деятельностью;*
- *процессы инженерной деятельности;*
- *инструменты и методы программной инженерии;*
- *качество программного обеспечения.*

Следует отметить, что данное руководство не касается вопросов конкретизации применения тех или иных языков программирования, архитектурных решений или, тем более, рекомендаций, касающихся использования каких-либо технологий проектирования и разработки ПП.

Остановимся в дальнейшем более подробно на описании содержания областей знаний, конкретизирующих основные этапы жизненного цикла разработки программных продуктов (рис. 1.15) [1].

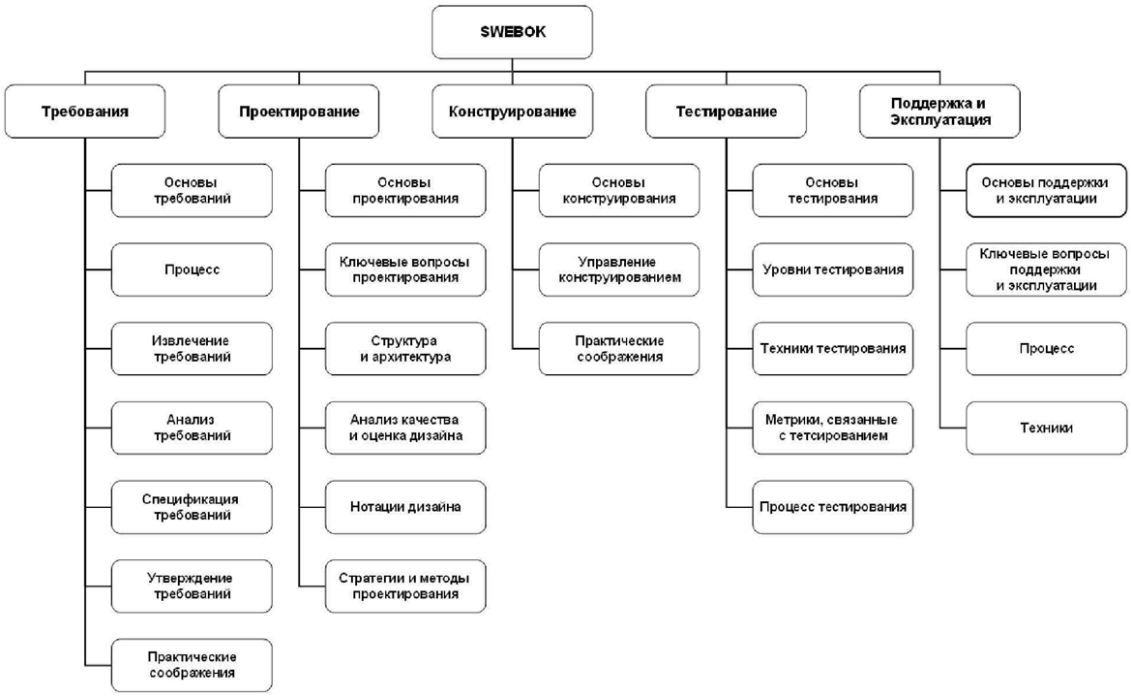


Рис. 1.15 – Содержание этапов жизненного цикла программных продуктов

1.3.1 Определение требований

В данном случае подразумевается не процесс определения (извлечения, сбора, формирования, формулирования) требований, а дается само понятие «требования», как такового, и описываются его основные характеристики. Под требованиями в данном документе понимаются потребности людей (заказчиков, пользователей, разработчиков), заинтересованных в создании ПП, которые должны быть надлежащим образом представлены и оформлены в виде технического задания или иного документа.

Все множество требований условно разбито на следующие группы: требования к продукту и процессу, функциональные и нефункциональные требования, системные и программные требования, требования с количественной оценкой ПО (рис. 1.16).

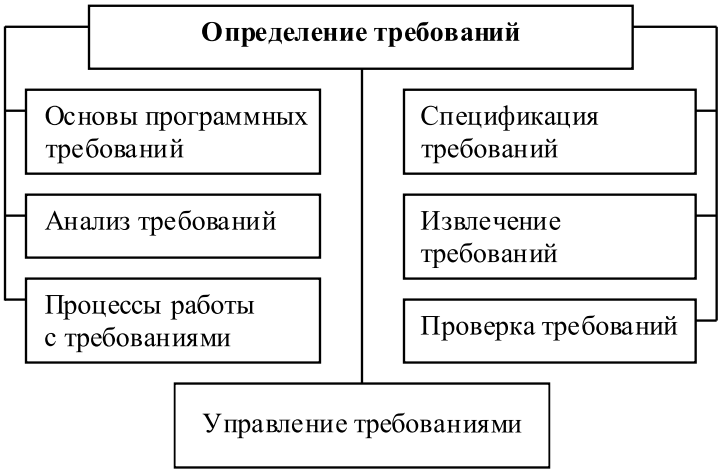


Рис. 1.16 – Состав раздела «Определение требований»

Основы программных требований

Требования к продукту и процессу — должны описывать свойства продукта, который необходимо получить, и процесса, с помощью которого продукт будет создаваться.

Функциональные требования характеризуют функциональные возможности программного обеспечения, методы передачи и преобразования входных данных в результаты.

Нефункциональные требования определяют условия и среду выполнения функций (например, защита и доступ к БД, взаимодействие компонентов и др.).

Системные требования описывают требования к программному продукту, состоящему из взаимосвязанных программных и аппаратных подсистем и разных приложений. Требования могут оцениваться количественно (например, количество запросов в единицу времени), значительная часть требований относится к атрибутам качества: безотказность, надежность и др.

Следует отметить, что в российских стандартах излагается несколько другой подход к составу требований. Так, например, в серии стандартов *ГОСТ 34.602-89 «Информационная технология. Техническое задание на создание автоматизированных систем»* приводится следующий состав требований:

- Требования к функциям (задачам), выполняемым системой.
- Требования к структуре и режимам функционирования системы.
- Требования к видам обеспечения: математическое обеспечение; информационное обеспечение; программное обеспечение; техническое обеспечение; организационное обеспечение.
- Общие требования к приемке работ по стадиям, порядок согласования и утверждения приемочной документации.

- Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие.
- Требования к документированию системы.

Процессы и работы с требованиями

Процесс работы с требованиями заключается в преобразовании предложенных заказчиком требований в описание требований к программному продукту, его спецификация и верификация инициируются в начале проекта и продолжаются на протяжении всего жизненного цикла, вплоть до его завершения.

Участниками процесса работы с требованиями являются:

- *Пользователи:* группа людей, кто будет непосредственно использовать программное обеспечение. Пользователи могут описать задачи, которые они планируют решать с использованием программного обеспечения, а также ожидания по отношению к атрибутам качества, отображаемые в пользовательских требованиях.
- *Заказчики* — лицо или организация, которые приобретают или получают программный продукт или программную услугу от поставщика и получают прямую или косвенную выгоду от их использования.
- *Инженеры по программному обеспечению, инженеры-программисты:* лица, проявляющие обоснованный интерес к разработке требований, например их может интересовать процесс повторного использования тех или иных компонент, библиотек, средств и инструментов. Именно инженеры ответственны за техническую оценку способов решения поставленных задач и последующую реализацию требований заказчиков.

Извлечение (выявление) требований

Данный раздел освещает вопросы сбора требований как с точки зрения организации процесса, так и определения источников, откуда поступают требования. Определение заинтересованных лиц в создании ПО, их взаимодействия, выполняемых ими бизнес-процессов — все это является ключевыми вопросами, без четкого и однозначного ответа на которые даже не стоит думать об успешности проекта.

Один из ключевых принципов программной инженерии заключается в обеспечении взаимодействия между пользователями и инженерами. Прежде всего, специалисты «по требованиям» — аналитики определяют способы коммуникаций и взаимопонимания между заказчиками и исполнителями, которые необходимы для решения задач проекта.

Необходимо идентифицировать все возможные источники требований, значимые для решения задач проекта (описание функциональных обязанностей, должностных инструкций, договоров, материалов аналитиков по задачам и функциям системы и т. д.). Идентифицировав источники требований, необходимо перейти к сбору требований с использованием методов: дерева целей, анкетирования, разработки сценариев, интервьюирования и т. д.

Анализ требований

Данный раздел посвящен описанию процессов анализа требований, то есть трансформации информации, полученной от пользователей (и других заинтересованных лиц), в четко и однозначно определенные требования, передаваемые инженерам для реализации в программном коде. Анализ требований включает:

- процессы изучения потребностей и целей пользователей;
- классификацию требований и их преобразование к требованиям программного обеспечения; к общесистемному аппаратно-программному обеспечению;
- установление и разрешение конфликтов между требованиями;
- определение их приоритетов и принципов взаимодействия со средой функционирования.

Результаты этапа отражаются в специальном документе (техническом задании), по которому проводится *согласование требований* для достижения взаимопонимания между заказчиком и разработчиком.

Спецификация требований

В разделе «спецификация требований» отражается структура ПО, требования к функциям, качеству и документации, задаются в общих чертах: архитектура ПО, алгоритмы, логика управления и структура данных, требования к взаимодействию с другими компонентами и платформами.

Проверка (аттестация) требований

Аттестация требований — это процесс проверки правильности спецификаций требований на непротиворечивость, полноту и выполнимость, а также на соответствие стандартам. На этом этапе заказчик и разработчик ПО проводят экспертизу сформированного варианта требований с тем, чтобы разработчик мог далее проводить разработку ПО. В результате проверки требований подписывается согласованный выходной документ, устанавливающий полноту и корректность требований к ПО, а также возможность продолжить проектирование ПО. Одним из методов аттестации является прототипирование, т. е. быстрая реализация отдельных требований в виде прототипа будущего ПО, анализ масштабов изменения требований, измерение объема функциональности, трудозатрат и стоимости.

Управление требованиями

Требования часто меняются в силу изменения внешних условий и внутренних бизнес-процессов. Необходимо понимать неизбежность изменений и планировать шаги по уменьшению проблем, связанных с изменениями. Данный процесс, точнее комплекс процессов, охватывает весь жизненный цикл программного обеспечения. Управление изменениями, сопровождение и поддержка актуальности требований и их реализации — ключ к успешным процессам программной инженерии. Управление изменениями не может быть хаотическим, поэтому отношение к управлению

требованиями как к постоянно действующему бизнес-процессу. Восприятие изменений и возможность их своевременной обработки — вопрос способности проектной команды работать в постоянно меняющихся условиях. Так или иначе, понимание меняющейся природы требований — один из факторов адекватного реагирования на сами изменения, а следовательно, и возможности успешного завершения проекта.

1.3.2 Проектирование ПО

Процесс «Проектирование ПО» предназначен для описания, на основе анализа требований, архитектуры (программного дизайна) ПО в виде иерархической совокупности компонентов и интерфейсов между ними (рис.1.17).

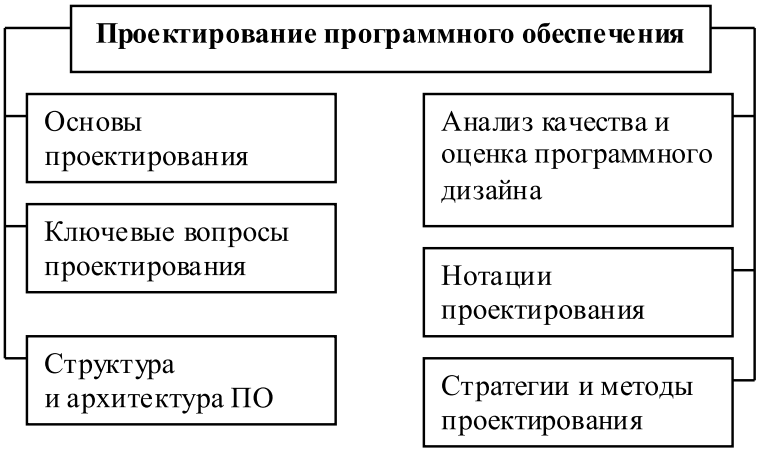


Рис. 1.17 – Состав области знаний «Проектирование ПО»

Основы проектирования

В данном разделе документа приводятся понятия и терминология в качестве основы для понимания роли и содержания проектирования (как деятельности) и дизайна (архитектуры, как результата) программного обеспечения. Для понимания содержания проектирования программного обеспечения важно выбрать одну из моделей жизненного цикла, на основе которой будет осуществляться проектирование. Сам процесс проектирования состоит из двух частей:

- архитектурный дизайн — описание многоуровневой структуры компонентов системы;
- детализированная архитектура — описание поведения характеристик отдельных компонент.

Ключевые вопросы проектирования

К ключевым вопросам проектирования ПО относятся декомпозиция архитектуры на функциональные компоненты для независимого и параллельного их выполнения, принципы и механизмы их взаимодействия между собой, обеспечения качества и живучести программного обеспечения в целом.

Один из вариантов декомпозиции описан в ГОСТ 34.003-90 «Информационная технология. Автоматизированные системы. Термины и определения», где предлагается выделять и описывать следующие элементы программного продукта:

- *интегрированный программный продукт* — совокупность двух и более программных продуктов, в которых функционирование одного из них зависит от результатов функционирования другого;
- *программный продукт*- совокупность программных компонент, реализующих конкретный бизнес-процесс;
- *сложный программный компонент* — совокупность программных кодов, реализующих сложную функцию бизнес-процесса;
- *программный компонент* — совокупность программных кодов, реализующих элементарную функцию бизнес-процесса.

Такая декомпозиция на модули сложного программного обеспечения производится с целью получения более мелких и относительно независимых программных компонентов, каждый из которых несет различную функциональность.

Структуры и архитектуры ПО

В строгом значении архитектура программного обеспечения содержит описание логики отдельных компонентов системы, достаточное для проведения работ по кодированию, и связей между ними.

По способам организации внутренней структуры, описания и конструирования ее элементов и связей между ними архитектура программного обеспечения может рассматриваться с разных точек зрения — например, поведенческой (динамической), структурной (статической), логической (удовлетворение функциональным требованиям), физической (распределенной).

В результате будут получены различные архитектурные представления и дизайн системы. Архитектурное представление может быть определено как частные аспекты программной архитектуры, рассматривающие специфические свойства программного обеспечения. В свою очередь, дизайн системы — комплекс архитектурных представлений, достаточный для реализации системы и удовлетворения требований, предъявляемых к ней.

Анализ качества и оценка программного дизайна

Данный раздел включает мероприятия по анализу сформулированных в требованиях атрибутов оценки качества различных аспектов ПО. Все множество атрибутов предлагается условно разбить на три группы:

- применимые ко времени реализации программного обеспечения; например, среднее время отклика системы, позволяющей оценить качество дизайна с точки зрения производительности;
- позволяющие оценивать качество получаемого дизайна еще на этапе проектирования, например средняя нагруженность классов бизнес-методами, характеризующая легкость поддержки, модификации и развития системы с такой внутренней структурой.

Проведение качественного анализа результатов проектирования возможно путем статического анализа, моделирования и прототипирования.

Нотации проектирования

В общем случае нотация определяется как соглашение о представлении (описании) структуры и поведения системы и взаимосвязей между ее элементами. Существует два типа нотаций: структурные и поведенческие, и соответственно множество различных методов их представления.

Структурные нотации являются графическими, они используются для представления структурных аспектов проектирования, компонентов и их взаимосвязей, элементов архитектуры и их интерфейсов. Нотации включают языки описания архитектуры и интерфейса, диаграммы классов и объектов, диаграммы «сущность-связь», компонентов, развертывания, а также структурные диаграммы и схемы.

Такие нотации создаются с использованием формальных языков проектирования: UML (Unified Modeling Language), ERD (Entity-Relation Diagrams) и т. д.

Поведенческие нотации отражают динамический аспект поведения систем и их компонентов. Таким нотациям соответствуют диаграммы: Data Flow — один из основных инструментов структурного анализа и проектирования информационных систем; Decision Tables — способ компактного представления модели со сложной логикой. Аналогично условным операторам в языках программирования, они устанавливают связь между условиями и действиями; Activity — диаграмма поведения, на которой показан автомат и подчеркнуты переходы потока управления от одной деятельности к другой; Colloboration — диаграмма поведения, на которой показано взаимодействие и подчеркнута структурная организация объектов, посылающих и принимающих сообщения; Sequence — диаграмма поведения, на которой показано взаимодействие и подчеркнута временная последовательность событий, и др.

Стратегия и методы проектирования ПО

Данный раздел знаний представляет различные стратегии и методы, которые используются при проектировании.

Функционально-ориентированные (структурные) методы ориентированы на идентификацию функций и их уточнение сверху-вниз, после чего проводится разработка диаграмм потоков данных и описание процессов.

Объектно-ориентированное проектирование базируется на концепции объектов, ключевую роль играет наследование, полиморфизм и инкапсуляция, а также абстрактные структуры данных и отображение объектов.

Подходы, ориентированные на структуры данных, базируются на методе Джексона и используются для задания входных и выходных данных структурными диаграммами. В данном подходе фокус сконцентрирован в большей степени на структурах данных, которыми управляет система, чем на функциях системы. Инженеры по программному обеспечению часто вначале описывают структуры данных входов и выходов, а затем разрабатывают структуру управления этими данными.

Компонентное проектирование ориентировано на использование и интеграцию компонентов (особенно компонентов повторного использования) и их интерфейсов, обеспечивающих взаимодействие компонентов.

1.3.3 Конструирование ПО

Конструирование программного обеспечения в описании области знаний SWEBOK заключается в создании рабочего программного обеспечения посредством комбинации кодирования, верификации (проверки), модульного тестирования, интеграционного тестирования и отладки (рис. 18).

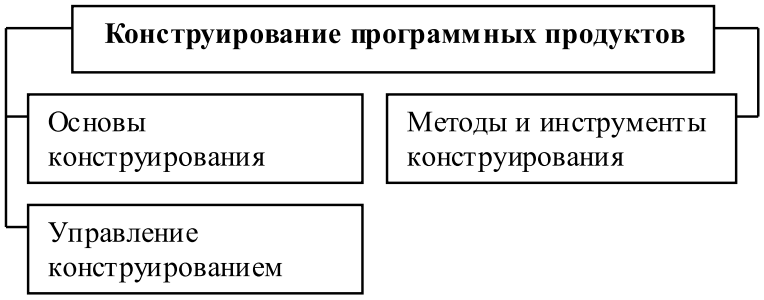


Рис. 1.18 – Содержание раздела «Конструирование ПО»

Данная область знаний тесно связана с другими областями проектированием и тестированием, так как конструирование отталкивается от результатов проектирования, а тестирование (в любой своей форме) предполагает работу с результатами конструирования. Достаточно сложно определить границы между проектированием, конструированием и тестированием, так как все они связаны в единый комплекс процессов жизненного цикла и, в зависимости от выбранной модели жизненного цикла и применяемых методов (методологии), такое разделение может выглядеть по-разному.

Основы конструирования

Основы конструирования программного обеспечения определяются следующими положениями:

- Минимизация сложности при создании программного кода.
- Готовность (ожидание) периодических изменений требований.
- Конструирование с возможностью проверки программного кода.
- Обязательное использование стандартов в конструировании.

Уменьшение сложности в конструировании программного обеспечения достигается при уделении особого внимания созданию простого и легко читаемого кода, пусть и в ущерб стремлению сделать его идеальным (например, с точки зрения гибкости или следования тем или иным представлениям о красоте, утонченности кода, ловкости тех или иных приемов, позволяющих его сократить в ущерб размерам, и т. п.).

Большинство программного обеспечения изменяется с течением времени, так как программное обеспечение не является изолированным от внешнего окружения. Более того, программное обеспечение является частью изменяющейся среды и должно меняться вместе с ней, а иногда, и быть источником изменений самой среды.

«Конструирование для проверки» предполагает, что построение программного обеспечения должно вестись таким образом, чтобы само программное обеспечение помогало вести поиск причин сбоев, будучи прозрачной для применения различным методов проверки как на стадии независимого тестирования (например, инженерами-тестировщиками), так и в процессе эксплуатации, когда особенно важна возможность быстрого обнаружения и исправления возникающих ошибок.

Конструирование зависит от внешних стандартов, связанных с языками программирования, используемым инструментальным обеспечением, техническими интерфейсами. Внешние стандарты создаются разными источниками, например международными организациями по стандартизации, производителями платформ, операционных сред, производителями инструментов, систем управления базами данных и т. п. Определенные стандарты, соглашения и процедуры могут быть также созданы внутри организации или даже проектной команды. Эти стандарты поддерживают координацию между определенными видами деятельности, группами операций, минимизируют сложность, могут быть связаны с вопросами ожидания и обработки изменений, рисков и вопросами конструирования для проверки и дальнейшего тестирования.

Управление конструированием

Управление конструированием базируется на моделях конструирования, планировании и внесении изменений в процессы конструирования ПО.

Модели конструирования включают набор операций, последовательность действий и результаты. Виды моделей определяются стандартом ЖЦ, методологиями и практиками. Основные стандарты ориентированы на экстремальное программирование (XP — eXtreme Programming — упрощенная методология организации разработки программ для небольших и средних по размеру команд разработчиков, занимающихся созданием программного продукта в условиях неясных или быстро меняющихся требований) и использование методологии Rational Unified Process (RUP) — четко определенный процесс (технологическая процедура), описывающий структуру жизненного цикла проекта, роли и ответственности отдельных исполнителей, выполняемые ими задачи и используемые в процессе разработки модели, отчеты и т. д.

Планирование состоит в определении порядка создания компонентов и методов обеспечения качества. Измерение в конструировании ориентировано на количественную оценку объема кода, степени повторного использования кода (ПИК), вероятности появления дефектов и количественных показателей качества ПО.

Внесение изменений проводится с целью сохранения функциональной целостности системы на основе проведенного метрического анализа необходимости проведения изменений в конструируемое ПО.

Методы и инструменты конструирования ПО

К методам и инструментам конструирования ПО отнесены языки программирования и конструирования, а также программные методы и инструментальные системы (компиляторы, СУБД, генераторы отчетов, системы управления версиями,

конфигурацией, тестированием и др.). К формальным средствам описания процесса конструирования ПО, взаимосвязей между человеком и компьютером и с учетом среды окружения отнесены языки конструирования. Основу данного раздела составляет задача понижения сложности конструирования программного продукта, которая обеспечивается применением наиболее подходящих *стилей конструирования*.

Лингвистический стиль основан на использовании словесных инструкций и выражений для представлений отдельных элементов (конструкций) программ. Он используется при конструировании несложных конструкций и приводится к виду традиционных функций и процедур, логическому и функциональному их программированию и др.

Формальный стиль используется для точного, однозначного и формального определения компонентов системы. В результате его применения обеспечивается конструирование сложных систем с минимальным количеством ошибок, которые могут возникнуть в связи с неоднозначностью определений или обобщений при конструировании ПО неформальными методами.

Визуальный стиль является наиболее универсальным стилем конструирования ПО. Он позволяет разработчикам проекта представлять в наглядном виде сложные программные конструкции. Например, графический интерфейс пользователя освобождает разработчика от подбора необходимых координат и свойств объектов интерфейса. Визуальный язык проектирования UML представляет разработчику набор удобных диаграмм для задания статической и динамической структуры ПО.

При применении визуального стиля конструирования создается текстовое и диаграммное описание структуры ПО, которое выводится на экран дисплея не только для рассмотрения, но и корректировки.

В процессе конструирования должны использоваться внешние стандарты языков программирования (Ада 95, C++ и др.), языков описания данных (XML, SQL и др.), средств коммуникации (COM, CORBA и др.), интерфейсов компонентов (POSIX, IDL, APL), сценариев UML и др.

1.3.4 Тестирование ПО

Тестирование ПО — это процесс проверки работоспособности, основанный на использовании конечного набора тестовых, сформированных на базе требований к ПО и сравнения полученных результатов с целевыми показателями качества, заложенными в проекте (рис. 1.19).

Основы тестирования

В разделе *основы тестирования* приводятся терминология, методы и инструменты подготовки и проведения процесса тестирования, а также показатели оценки данных статистического анализа процесса тестирования. В соответствии с принятой терминологией при тестировании выявляются следующие недостатки: отказы и дефекты, сбои, ошибки. Важно четко разделять *причину* нарушения работы прикладных программ, обычно описываемую терминами «*недостаток*» или «*дефект*», и наблюдаемый нежелательный эффект, вызываемый этими причинами —

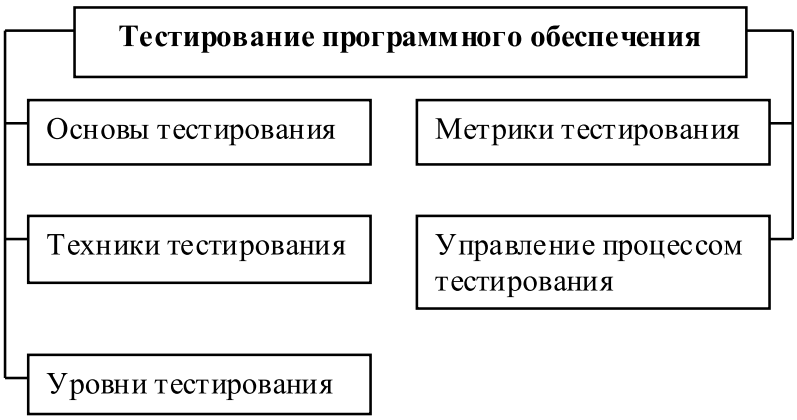


Рис. 1.19 – Область знаний «Тестирование ПО»

сбой. Термин «*ошибка*», в зависимости от контекста, может описывать и причину сбоя, и сам сбой. Тестирование позволяет обнаружить дефекты, приводящие к сбоям.

Базовым понятием тестирования является также тест, который выполняется в заданных условиях и на проверочных наборах данных. Степень тестируемости зависит от задания критериев покрытия системы тестами и вероятности появления сбоев.

Уровень тестирования

Уровень тестирования определяет, «над чем» производятся тесты: над отдельным модулем, группой модулей или системой в целом. При этом ни один из уровней тестирования не может считаться приоритетным. В документе выделены следующие уровни тестирования:

- *модульное тестирование,* предполагающее проверку отдельных, изолированных и независимых *элементов* (компонентов) ПО;
- *интеграционное тестирование,* которое ориентировано на проверку связей и способов взаимодействия (интерфейсов) компонентов друг с другом;
- *тестирование программного обеспечения,* предназначенное для проверки правильности функционирования в целом, с обнаружением отказов и дефектов, и их устранения. При этом контролируется и выполнение нефункциональных требований (безопасность, надежность и т. д.), а так же правильность задания и выполнения внешних интерфейсов со средой окружения.

Тестирование проводится в соответствии с определенными целями проверки качества и работоспособности ПО. В документе выделены следующие, наиболее распространенные цели (и, соответственно виды) тестирования:

- *функциональное тестирование*, которое заключается в проверке соответствия выполнения функциональных возможностей ПО;
- *регрессионное тестирование* — тестирование программного обеспечения или его компонентов после внесения в них изменений;

- *тестирование эффективности функционирования ПО* — проверка в соответствии со спецификациями требований производительности, пропускной способности, максимального объема данных, системных ограничений и т. д.;
- *нагрузочное (стресс) тестирование* — проверка поведения ПО при максимально допустимой нагрузке;
- *внутреннее и внешнее тестирование* — альфа(без плана тестирования) и бета (с планом тестирования) тестирование соответственно;
- *тестирование конфигурации* — проверка структуры и идентификации системы на различных наборах данных, а также проверка работы системы в различных конфигурациях;
- *приемочное тестирование* — проверка в соответствии с заранее подготовленной программой и методикой испытаний поведения системы на этапе приемки-сдачи ее заказчику.

Техники тестирования

Техники тестирования условно разбиты в документе на следующие группы:

- *Техники, базирующиеся на интуиции и опыте инженера*, рассматривающего проблему с точки зрения имевшихся ранее аналогий.
- *Техники, базирующиеся на блок-схеме ПО,* строятся исходя из покрытия всех условий и решений блок-схемы. Максимальная отдача от тестов на основе блок-схемы получается, когда тесты покрывают различные пути блок-схемы. Адекватность таких тестов оценивается как процент покрытия всех возможных путей блок-схемы.

Тестирование, ориентированное на дефекты, направлено на обнаружение наиболее вероятных ошибок, предсказываемых заранее, например в результате анализа возможных рисков программного проекта.

Техники, базирующиеся на условиях использования, позволяют оценить поведение системы в реальных условиях. Входные параметры тестов задаются на основе вероятностного распределения соответствующих параметров или их формирования в процессе эксплуатации.

Техники, базирующиеся на природе приложения, находят применение в зависимости от технологической или архитектурной природы приложений, среди таких техник можно выделить:

- Объектно-ориентированное тестирование.
- Компонентно-ориентированное тестирование.
- Web-ориентированное тестирование.
- Тестирование на соответствие протоколам.
- Тестирование систем реального времени.

Метрики тестирования

Метрики тестирования предназначены для измерения процессов и результатов планирования, проектирования и тестирования ПО.

Измерение результатов тестирования касается оценки качества получаемого продукта. *Оценка программ в процессе тестирования* должна базироваться на размере программ (например, в терминах количества строк кода или функциональных точек) или их структуре (например, с точки зрения оценки ее сложности в тех или иных архитектурных терминах). Структурные измерения могут также включать частоту обращений одних модулей программы к другим.

Эффективность тестирования может быть измерена путем определения, какие типы дефектов могут быть найдены в процессе тестирования ПО и как изменяется их частота во времени. Эта информация позволяет прогнозировать качество ПО и совершенствовать в дальнейшем процесс разработки в целом.

Тестируемая программа может оцениваться также *на основе подсчета и классификации найденных дефектов*. Для каждого класса дефектов можно определить отношение между количеством соответствующих дефектов и размером программы.

В ряде случаев процесс тестирования, требует систематического выполнения тестов для определенного набора элементов программы, задаваемых ее архитектурой или спецификацией. Соответствующие метрики позволяют оценить степень охвата характеристик системы и глубину их детализации. Такие метрики помогают прогнозировать вероятностное достижение заданных параметров качества системы.

Документация на тестирование включает описание тестовых документов, их связи между собой и с процессом тестирования. Без документации на процессы тестирования невозможно провести сертификацию и оценку зрелости программного продукта. После завершения тестирования рассматриваются вопросы стоимости и рисков, связанных с появлением сбоев и недостаточно надежной работой системы. Стоимость тестирования является одним из ограничений, на основе которого принимается решение о прекращении или продолжении тестирования.

Управление тестированием

Концепции, стратегии, техники и измерения тестирования должны быть объединены в единый процесс тестирования (от планирования тестов до оценки их результатов) как деятельности по обеспечению качества ПО, поддерживающей «правила игры» для членов команды тестирования. Только в том случае, если тестирование рассматривать как один из важных процессов всей деятельности по созданию и поддержке программного обеспечения, можно добиться оценки стоимости соответствующих работ и, в конце концов, выполнить те ограничения, которые определены для проекта.

Работы по управлению процессом тестирования, ведущиеся на разных уровнях, должны быть организованы в единый процесс, на основе учета четырех элементов и связанных с ними факторов: *участников процесса* (в том числе, в контексте организационной структуры и культуры), *инструментов, регламентов и количественных оценок измерения.*

В состав этого процесса должны входить:

а) планирование работ по тестированию (составление планов, тестов, наборов данных) и измерению показателей качества ПО;

б) генерация необходимых тестовых сценариев, соответствующих среде выполнения ПО:

в) проведение тестирования с учетом следующих положений:

- все работы и результаты процесса тестирования должны обязательно фиксироваться;
- форма журналирования работ и их результатов должна быть такой, чтобы соответствующее содержание было понятно, однозначно интерпретируемо и повторяемо другими лицами;
- тестирование должно проводиться в соответствии с заданными и документированными процедурами;
- тестирование должно производиться над однозначно идентифицируемой версией и конфигурацией ПО;
- должен осуществляться сбор данных об отказах, ошибках и др. непредвиденных ситуациях при выполнении программного продукта;
- подготовка отчетов по результатам тестирования и оценки характеристик ПО должны составляться отчеты.

1.3.5 Сопровождение ПО

Фаза сопровождения в жизненном цикле, обычно, начинается сразу после приемки/передачи продукта и действует в течение периода гарантии или технической поддержки. Объективная потребность в сопровождении связана:

- с обнаружением при эксплуатации ПП скрытых дефектов и необходимости устранение сбоев;
- улучшением дизайна;
- реализацией новых функциональных возможностей;
- созданием интерфейсов взаимодействия с внешними системами;
- адаптацией для обеспечения возможности работы ПП на другой программно-аппаратной платформе;
- изменением бизнес-процессов в предметной области;
- выводом программного обеспечения из эксплуатации.

Процесс сопровождения выполняется как перед вводом системы в эксплуатацию, так и после этого и состоит из планирования деятельности по сопровождению системы, организации перехода к ее полнофункциональному использованию, обучения пользователей и их ежедневной поддержки при работе с текущей версией продукта. Если новая система должна заменить старую систему, важно обеспечить плавный переход со старой системы на новую максимально естественно для пользователей.

Область знаний «Сопровождение ПО» состоит из следующих описаний разделов (рис. 20).

Рис. 1.20 – Область знаний «Сопровождение ПО»

Основы сопровождения

Этот раздел содержит описание содержания концепции и терминологию, формирующие основы понимания роли и содержания работ по сопровождению ПО.

Сопровождение программного обеспечения определяется стандартами как:

- *совокупность деятельности, необходимой для обеспечения эффективной (с точки зрения затрат) поддержки ПО;*
- *модификация программного продукта после передачи в эксплуатацию для устранения сбоев, улучшения показателей производительности и/или других характеристик (атрибутов) продукта, или адаптации продукта для использования в модифицированном окружении;*
- *процесс модификации программного продукта в части его кода и документации для решения возникающих проблем при эксплуатации или реализации новых потребностей пользователей, улучшения тех или иных характеристик продукта;*
- *модификация программного продукта в процессе эксплуатации при условии сохранения целостности продукта.*

Деятельность персонала сопровождения включает четыре ключевых аспекта: поддержку контроля программного обеспечения в течение всего цикла эксплуатации; поддержку модификаций ПО; совершенствование существующих функций; предотвращение снижения производительности ПО до критического уровня.

В зависимости от исходных условий состояния ПО в стандартах выделяется четыре категории сопровождения:

- Корректирующее сопровождение: «реактивная» модификация программного продукта, выполняемая уже после передачи в эксплуатацию для устранения сбоев.
- Адаптирующее сопровождение: модификация программного продукта на этапе эксплуатации для обеспечения продолжения его использования с заданной эффективностью (с точки зрения удовлетворения потребностей пользователей) в изменившемся или находящемся в процессе изменения окружении, в первую очередь подразумевается изменение бизнес-окружения, порождающее новые требования к ПО.

- Совершенствующее сопровождение: модификация программного продукта на этапе эксплуатации для повышения характеристик производительности и удобства сопровождения.
- Профилактическое сопровождение: модификация программного продукта на этапе эксплуатации для идентификации и предотвращения скрытых дефектов до того, когда они приведут к реальным сбоям.

Ключевые вопросы сопровождения программного обеспечения

Для обеспечения эффективного сопровождения ПО необходимо решать целый комплекс вопросов и проблем, связанных с соответствующими работами. Эти вопросы и проблемы сгруппированы в стандарте по следующим темам:

- Технические вопросы.
- Управленческие вопросы.
- Вопросы оценка стоимости сопровождения.
- Вопросы измерения.

К группе технических вопросов по сопровождению ПО относятся:

- анализ и понимание персоналом программного кода ПО, которое он не разрабатывал, а также необходимость внести исправления или изменения в код;
- организация работ по тестированию модификаций эксплуатируемого ПО, вплоть до предварительного планирования и разработки регламентов, в соответствии с которыми персоналом сопровождения будут проводиться стандартные процедуры;
- анализ возможных последствий и влияний изменений, вносимых в существующее ПО, оценка рисков, связанных с внесением изменений.

Сущность управленческих вопросов состоит в контроле качества ПО в процессе модификации функционала и недопущении снижения производительности.

Оценка стоимости затрат на сопровождение зависит от типа ПО, квалификации персонала, платформы и др. Знание этих факторов позволяет не только их сохранить, но и уменьшить.

Вопросы измерения относятся к оценке характеристик ПО после его модификации. В качестве типичных метрик оценки работ по сопровождению, соответствующих распространенной классификации эксплуатационных характеристик программного обеспечения, рекомендуется использовать:

- анализируемость: оценка ресурсов, необходимых для диагностики недостатков или причин сбоев, а также для идентификации тех фрагментов ПО, которые должны быть модифицированы;
- изменяемость: оценка ресурсов, необходимых для проведения заданных модификаций;
- стабильность: оценка режимов непредусмотренного поведения ПО, включая ситуации, обнаруженные в процессе тестирования;

- тестируемость: оценка трудозатрат персонала сопровождения и пользователей по тестированию модифицированного программного обеспечения.

Процесс сопровождения

Необходимо рассматривать процесс сопровождения как эволюционную модификацию ПО, поскольку сданное в эксплуатацию ПО не всегда является полностью завершенным, его надо изменять в течение срока эксплуатации. В результате ПО становится более сложным и плохо управляемым.

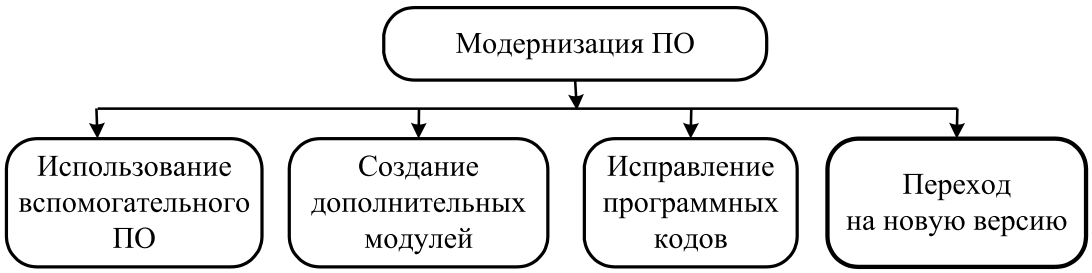


Рис. 1.21 – Варианты модернизации ПО

В этом случае возможны четыре варианта развития событий при модификации ПО (рис. 1.21):

1) пользователям приходится смириться с отсутствием некоторых функциональных возможностей программного обеспечения или их несоответствием новым реалиям и выполнять их либо вручную, либо при помощи подручных инструментов, например Word и Excel. Соответственно возрастает трудоемкость обработки, вероятность ошибок (человеческий фактор), требуется обучение и коррекция мотивации персонала, а главное — теряется драгоценное время;
2) в обход ограничений ПО (врожденных или приобретенных в ходе эволюции бизнеса) создаются дополнительные компоненты («заплатки»). Здесь к риску допуска новых ошибок добавляются издержки на исследование «обходных путей», их программирование и отладку;
3) если первые два варианта решения неприемлемы, то тогда разработчик начинает реализацию изменений на уровне исходных текстов, что требует ресурсов, а главное — времени на завершение полного цикла разработки, тестирование и разворачивание новой версии ПО;
4) наиболее пессимистичное развитие ситуации, когда критичные для заказчика требования не только не покрываются ПО и всеми его штатными средствами настройки, но и разработчик в силу различных причин не берется реализовать их путем доработок или переработок программного кода и предлагает заново осуществлять выбор новой платформы и разрабатывать новое решение. Это может привести просто к катастрофе бизнеса, сокращению доли рынка, реальных доходов от продаж, снижению конкурентных преимуществ.

Техники сопровождения

Для реализации изменений программисты тратят значительную часть времени на чтение и формирование понимания в большинстве случаев чужого программного кода. Средства работы с кодом являются ключевым инструментом для решения этой задачи. Четкая, однозначная и лаконичная документация обеспечивает адекватное понимание ПО.

Общепринятыми техниками, используемыми в процессе сопровождения ПО, являются: реинжиниринг, обратный инжиниринг и рефакторинг.

Реинжиниринг — это улучшение возможностей, функций в устаревшем ПО путем его реорганизации и реструктуризации, перепрограммирования или настройки на другую платформу или среду с обеспечением удобства его сопровождения.

Обратный инжиниринг состоит в анализе программного обеспечения с целью идентификации программных компонент и связей между ними, восстановлении спецификации по полученному коду ПО (особенно, когда в нее внесено много изменений) для наблюдения за ней на более высоком уровне. Конечными целями обратного инжиниринга могут быть: создание новой документации на существующее ПО; восстановление дизайна и т. д.

Рефакторинг — трансформация программного обеспечения, в процессе которого ПО реорганизуется (не переписываясь) с целью улучшения структуры, без изменения ее функционала. Рефакторинг ориентирован на улучшение структурных характеристик и качественных показателей объектно-ориентированных программ без изменения их поведения. Этот процесс реализуется путем изменения отдельных операций над текстами, интерфейсами, средой программирования и выполнения ПО, а также настройки или внесения изменений в инструментальные средства поддержки ПО.

1.4 Государственный стандарт РФ ГОСТ Р ИСО/МЭК 12207-99. «Информационная технология. Процессы жизненного цикла программных средств»

В основу стандарта положены следующие базовые понятия:

система (system): комплекс, состоящий из процессов, технических и программных средств, устройств и персонала, обладающий возможностью удовлетворять установленным потребностям или целям;

модель жизненного цикла (life cycle model): 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;

процесс (process): набор взаимосвязанных работ, которые преобразуют исходные данные в выходные результаты [5].

Все работы, которые могут выполняться в жизненном цикле программных средств, распределены по пяти основным, восьми вспомогательным и четырем организационным процессам (рис.1.22). Каждый процесс жизненного цикла разделен на набор работ; каждая работа разделена на набор задач. Процесс, работа или задача по мере необходимости инициируются и выполняются другим процессом, причем нет заранее определенных последовательностей.

Основные процессы жизненного цикла включают в себя: процесс заказа, процесс поставки, процесс разработки, процесс эксплуатации, процесс сопровождения.

Процесс заказа состоит из следующих работ, выполняемых заказчиком:

- подготовка заявки на подряд;
- подготовка и корректировка договора;
- надзор за поставщиком, приемка и закрытие договора.

Процесс начинается с определения потребностей заказчика в программном продукте или программной услуге. Далее следуют подготовка и выпуск заявки на подряд, выбор поставщика и управление процессом заказа вплоть до завершения приемки программного продукта или программной услуги.

Процесс подготовки начинается с описания потребности приобретения готового ПП. Далее заказчик формулирует функциональные, коммерческие, организационные и потребительские свойства к ПП, а также требования безопасности,

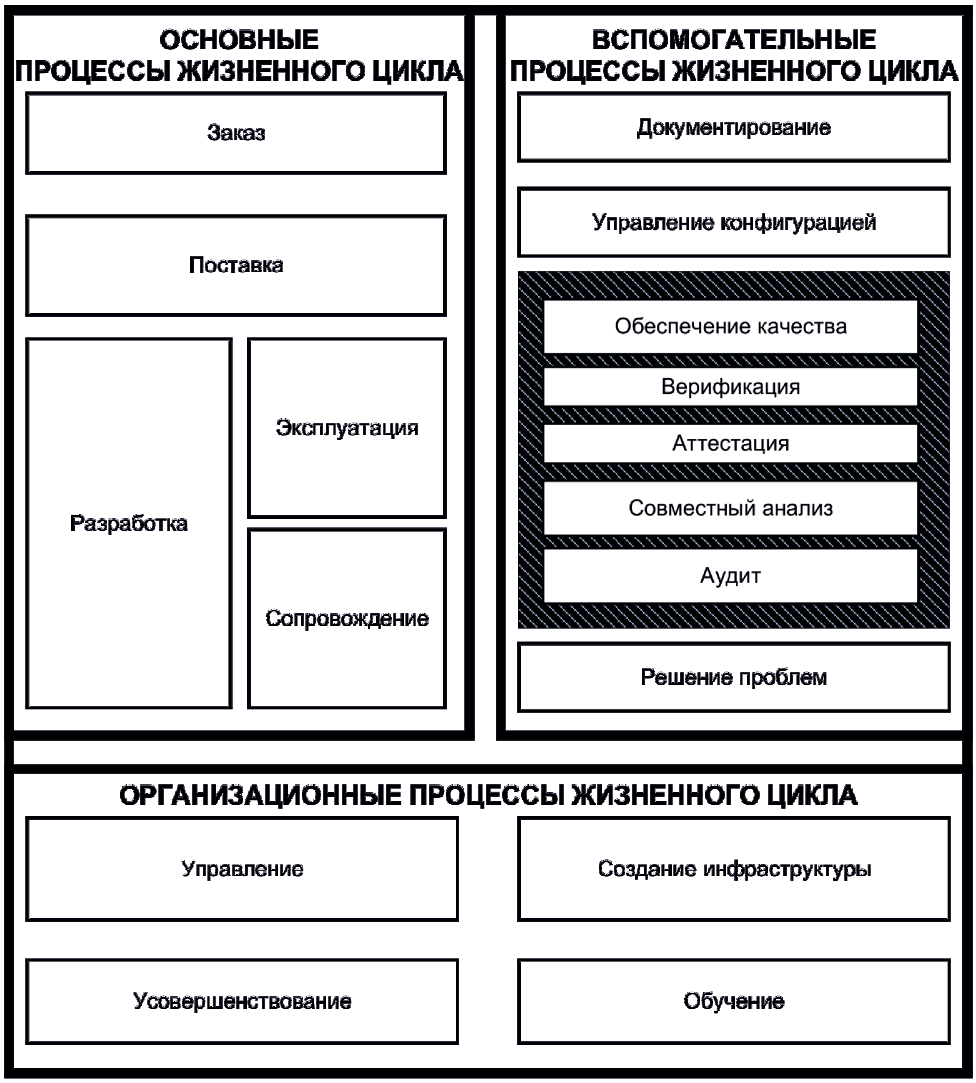


Рис. 1.22 – Структура стандарта

защиты и другие критические требования наряду с требованиями к проектированию, тестированию и соответствующим стандартам и процедурам.

Процесс поставки включает работы, выполняемые поставщиком:

- подготовку ответа, подготовку договора;
- планирование, выполнение и контроль;
- проверку и оценку;
- поставку и закрытие договора.

Процесс может быть начат с решения о подготовке предложения в ответ на заявку на подряд, присланную заказчиком, или с подписания договора и вступления с заказчиком в договорные отношения по поставке программного продукта или программной услуги. Процесс продолжается определением процедур и ресурсов, необходимых для управления и обеспечения проекта, включая разработку и выполнение проектных планов по поставке ПП или программной услуги заказчику.

Процесс разработки состоит из следующих работ, выполняемых разработчиком:

- анализ требований к системе;
- проектирование системной архитектуры;
- анализ требований к программным средствам;
- проектирование программной архитектуры;
- техническое проектирование программных средств;
- программирование и тестирование программных средств;
- сборка программных средств;
- квалификационные испытания программных средств;
- сборка системы; квалификационные испытания системы;
- ввод в действие программных средств;
- обеспечение приемки программных средств.

Процесс включает работы по анализу требований, проектированию, программированию, сборке, тестированию, вводу в действие и приемке программных продуктов. В данный процесс могут быть включены работы, связанные с разработкой ПП, если это оговорено в договоре. Разработчик выполняет или обеспечивает выполнение работ по данному процессу в соответствии с условиями договора.

Процесс эксплуатации состоит из следующих работ, выполняемых группой внедрения заказчика:

- эксплуатационные испытания;
- эксплуатация системы;
- поддержка пользователя.

Процесс охватывает эксплуатацию программного продукта и поддержку пользователей в процессе эксплуатации.

Процесс сопровождения состоит из следующих работ, выполняемых группой сопровождения разработчика (поставщика):

- анализ проблем и изменений; внесение изменений;
- проверка и приемка при сопровождении;
- перенос; снятие с эксплуатации.

Данный процесс реализуется при изменениях (модификациях) программного продукта и соответствующей документации, вызванных возникшими проблемами или потребностями в его модернизации или настройке. Целью процесса является изменение существующего программного продукта при сохранении его целостности. Данный процесс охватывает вопросы переносимости и снятия программного продукта с эксплуатации. Процесс заканчивается снятием программного продукта с эксплуатации.

Вспомогательные процессы жизненного цикла ПП сформулированы в ГОСТе следующим образом: процесс документирования; процесс управления конфигурацией; процесс обеспечения качества; процесс верификации; процесс аттестации; процесс совместного анализа; процесс аудита; процесс решения проблем.

Процесс документирования является формализованным описанием документации, создаваемой в процессе проектирования ПП. Данный процесс состоит из набора работ, при помощи которых планируют, проектируют, разрабатывают, выпускают, редактируют, распространяют и сопровождают те документы, в которых нуждаются все заинтересованные лица, такие как администраторы, системные инженеры и пользователи программного продукта. Данный процесс состоит из следующих работ:

- проектирование и разработка;
- выпуск;
- сопровождение.

Процесс управления конфигурацией является процессом применения административных и технических процедур на всем протяжении жизненного цикла программных средств (продуктов, комплексов, компонентов):

- для обозначения, определения и установления состояния программных объектов;
- управления изменениями и выпуском объектов;
- описания и информирования о состояниях объектов и заявок на внесение изменений в них;
- обеспечения полноты, совместимости и правильности объектов;
- управления хранением, обращением и поставкой объектов.

Данный процесс состоит из следующих работ:

- определение конфигурации;
- контроль конфигурации;
- учет состояний конфигурации;
- оценка конфигурации;
- управление выпуском и поставка.

Процесс обеспечения качества является процессом обеспечения соответствующих гарантий того, что программные продукты и процессы в жизненном цикле проекта соответствуют установленным требованиям и утвержденным планам. С точки зрения беспристрастности обеспечение качества должно быть организационно и полномочно независимым от субъектов, непосредственно связанных с разработкой программного продукта или выполнением процесса в проекте. Обеспечение качества может субъективно (внутренне или внешне) зависеть от того, демонстрируются ли доказательства качества продукта или процесса под управлением поставщика или заказчика. При обеспечении качества могут использоваться результаты других вспомогательных процессов, таких как верификация, аттестация, совместные анализы, аудит и решение проблем. Данный процесс состоит из следующих работ:

- обеспечение качества продукта в соответствии с условиями договора;
- обеспечение качества процесса (соответствие условиям договора);
- обеспечение систем качества (соответствие ГОСТ Р ИСО 9001).

Процесс верификации является процессом определения того, что программные продукты функционируют в полном соответствии с требованиями или условиями, сформулированным в соответствующих документах. Для оценки эффективности затрат и выполняемых работ верификация должна как можно раньше реализовываться в соответствующих процессах (таких как поставка, разработка, эксплуатация или сопровождение). Данный процесс может включать анализ, проверку и испытание (тестирование).

Данный процесс может выполняться с различными степенями независимости исполнителей. Степень независимости исполнителей может распределяться как между различными субъектами в самой организации, так и субъектами в другой организации с различными степенями распределения обязанностей. Данный процесс называется процессом независимой верификации, если организация-исполнитель не зависит от поставщика, разработчика, оператора или персонала сопровождения.

Процесс аттестации является процессом определения полноты соответствия установленных требований созданного программного продукта к его функциональному назначению. Аттестация может проводиться на начальных этапах работы. Данный процесс может проводиться как часть работы по обеспечению приемки программных продуктов. Условия реализации данного процесса идентичны условиям реализации процесса верификации.

Процесс совместного анализа является процессом оценки состояний и, при необходимости, результатов работ по проекту. Совместные анализы применяются как на уровне управления проектом, так и на уровне технической реализации проекта и проводятся в течение всего жизненного цикла договора. Данный процесс может выполняться двумя любыми сторонами, участвующими в договоре, когда одна сторона (анализирующая) проверяет другую сторону (анализируемую). Данный процесс состоит из следующих работ: анализы управления проектом; технические анализы. Процессы анализов хода работ проводятся в сроки, установленные проектным планом, а также в сроки, определяемые заинтересованной стороной.

Процесс аудита является процессом определения соответствия требованиям, планам и условиям договора. Данный процесс также может выполняться двумя любыми сторонами, участвующими в договоре, когда одна сторона (ревизующая) проверяет другую сторону (ревизуемую). Аудиторские проверки должны проводиться в сроки, установленные проектным планом(ами).

Процесс решения проблем является процессом анализа и решения проблем (включая обнаруженные несоответствия), независимо от их происхождения или источника, которые обнаружены в ходе выполнения разработки, эксплуатации, сопровождения или других процессов. Целью данного процесса является обеспечение способов своевременного, ответственного и документируемого анализа и решения всех обнаруженных проблем и определения причин их возникновения. При выявлении проблем (включая обнаруженные несоответствия) в программном продукте или работе должен быть подготовлен отчет по проблеме, описывающий каждую выявленную проблему. Отчет по проблеме должен являться составной частью вышеописанного процесса, охватывая вопросы:

- выявления проблем;
- их исследования, анализа и решения, а также причин их возникновения;
- определения тенденций, способствующих возникновению проблем.

Организационные процессы жизненного цикла включают: процесс управления; процесс создания инфраструктуры; процесс усовершенствования; процесс обучения.

Процесс управления состоит из общих работ, которые могут быть выполнены как разработчиком так и заказчиком, управляющими соответствующим процессом(ами). Администратор отвечает за управление проектом, работами и задачами соответствующего процесса(ов), такими как заказ, поставка, разработка, эксплуатация, сопровождение или вспомогательные процессы. Данный процесс состоит из следующих работ:

- подготовка и определение области управления;
- планирование;
- выполнение и контроль;
- проверка и оценка; завершение.

Процесс создания инфраструктуры является процессом установления и обеспечения (сопровождения) инфраструктуры, необходимой для любого другого процесса. Инфраструктура может содержать технические и программные средства, инструментальные средства, методики, стандарты и условия для разработки, эксплуатации или сопровождения. Данный процесс состоит из следующих работ:

- создание инфраструктуры;
- сопровождение инфраструктуры.

Процесс усовершенствования является процессом идентификации, оценки, измерения, контроля и улучшения любого процесса жизненного цикла программного продукта. Данный процесс состоит из следующих работ:

- оценка процесса;
- усовершенствование процесса.

Процесс обучения является процессом обеспечения первоначального и продолженного обучения персонала. Заказ, поставка, разработка, эксплуатация и сопровождение программных продуктов в значительной степени зависят от квалификации персонала. Например, персонал разработчика должен быть соответствующим образом обучен управлению проектами и технологии программирования. Поэтому обязательно должно быть запланировано и заранее выполнено обучение персонала с целью готовности его к работам по заказу, поставке, разработке, эксплуатации или сопровождению программного проекта. Данный процесс состоит из следующих работ:

- разработка учебных материалов;
- реализация плана обучения.

Настоящий стандарт содержит полный набор описания процессов жизненного цикла ПП для некоторого типового проекта с максимально возможным составом процессов, работ и задач, которые используются: при приобретении системы, содержащей программные средства, или отдельно поставляемого программного продукта; при оказании программной услуги, а также при поставке, разработке, эксплуатации и сопровождении ПП. Поэтому при решении практических задач стандарт необходимо адаптировать к конкретному проекту.

Ниже описывается вариант применения стандарта для реализации проекта: «Организация работ по поставке готового ПП».

Пример

Заказчик и поставщик ведут переговоры и вступают в договорные отношения, используя при этом соответственно описания процессов заказа и поставки (рис.1.23).

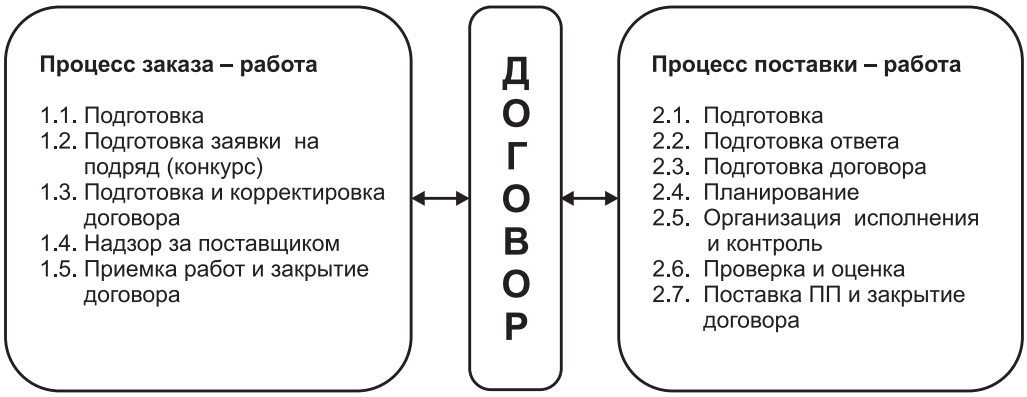


Рис. 1.23 – Содержание процессов

Последовательность выполнения работ при поставке готового ПП поставщиком заказчику представлена на рис. 1.24.

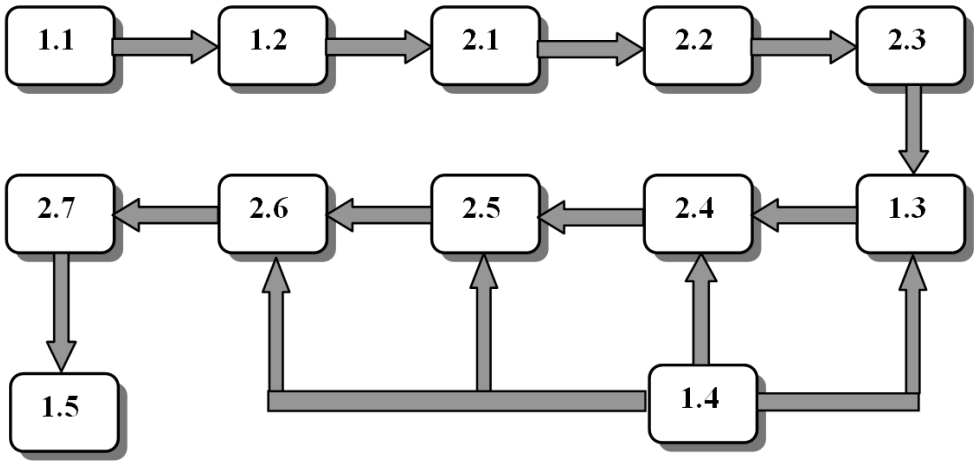


Рис. 1.24 – Последовательность выполнения работ

1.1. Подготовка. Заказчик начинает процесс заказа, описывая потребность в приобретении готового программного продукта, удовлетворяющего определенным требованиям. Требования к ПП должны охватывать функциональные, коммерческие, организационные и потребительские аспекты системы, а также без-

опасность, соответствие стандартам, проектирование, тестирование и т. д. Заказчик может выполнить формирование и анализ требований к ПП сам или поручить решение этой задачи потенциальному поставщику. В этом случае в задачи заказчика входит согласование этих требований.

Кроме того, заказчик должен потребовать гарантии, что при приобретении готового программного продукта удовлетворены следующие условия:

- программный продукт соответствует установленным требованиям;
- имеется в наличии соответствующая документация;
- соблюдены права собственности, использования, лицензирования и гарантии;
- предусмотрена последующая поддержка программного продукта.

Работа заканчивается документальным оформлением принятых правил и условий (критериев) реализации будущего договора на поставку ПП.

1.2. Заявка на подряд должна содержать: требования к системе; описание области применения системы; указания для участников торгов; список программных продуктов; сроки и условия реализации заказа; критерии оценки реализации заказа; правила контроля над выполнением заказа; технические ограничения (например, по условиям эксплуатации). На основании заявки объявляется конкурс на выполнение работ.

1.3. Подготовка и корректировка договора. На первоначальном этапе заказчик должен на конкурсной основе выбрать поставщика, исходя из оценки предложений, поступивших от потенциальных поставщиков, их возможностей и других рассматриваемых факторов.

Далее уточняются условия договора с поставщиком, включая стоимость и календарный план на поставку программного продукта. В договоре должны быть оговорены также права собственности, использования, лицензирования и гарантии, связанные с использованием готового программного продукта.

В ходе реализации договора заказчик должен контролировать изменения, вносимые в договор, обсуждая их с поставщиком. Изменения, вносимые в договор, должны быть изучены с точки зрения их влияния на договорные планы и цены, эффективность и качество.

1.4. Надзор за работами поставщика заказчик осуществляет в соответствии с процессами совместного анализа и аудита, при необходимости проводится текущий надзор согласно процессам верификации и аттестации.

1.5. Приемка и закрытие договора. Заказчик должен на основе программы и методики испытаний подготовить контрольные примеры, данные для процедуры тестирования и проведения испытаний, определить степень участия поставщика при проведении приемки. При выполнении всех условий приемки заказчик обязан принять поставленный ПП и закрыть договор на поставку.

2.1. Подготовка поставщика. Поставщик проводит анализ требований, установленных в заявке на проведение конкурса, принимая во внимание организационные вопросы и другие установленные правила, и в случае принятия решения об участии в конкурсе на поставку ПП готовит конкурсную документацию.

2.2. Подготовка договора. После подведения итогов конкурса поставщик-победитель вступает в договорные отношения с организацией заказчика по поставке

программного продукта. При этом он по согласованию с заказчиком может предложить внести изменения в текст договора по согласованию с заказчиком.

2.3. Планирование. На первом этапе поставщик должен провести анализ требований в целях создания системы управления по реализации проекта и обеспечения требуемого качества поставляемого ПП, разработать и документально оформить план управления проектом, отразив в нем следующие вопросы:

- организационную структуры проекта, полномочий и обязанностей каждого участника, включая сторонние организации;
- процедуру распределения заданий по процессам и работам жизненного цикла, включая состав исполнителей, требуемые материальные ресурсы, графики выполнения работ;
- процедуру верификации и аттестации;
- процедуры взаимоотношений с заказчиком: совместные анализы ситуаций, аудиторские проверки, совещания, рабочие и другие контакты;
- процедуры взаимоотношений с непосредственными пользователями, которые реализуются такими средствами, как выполнение требуемых настроек, демонстрация прототипов и оценки;
- алгоритм управления критическими ситуациями, то есть управления областями проекта, которые связаны с потенциальными техническими, финансовыми и плановыми затруднениями;
- подтверждение статуса поставляемой продукции, обеспечиваемое такими средствами, как инструкции, обязательная сертификация, права собственности, использования и распространения, гарантии и лицензионные права;
- план обучения персонала.

2.4. Выполнение и контроль. Поставщик должен:

- полностью реализовать планы управления проектом и поставить ПП в соответствии с процессом разработки;
- провести опытную эксплуатацию программного продукта (в соответствии с процессом эксплуатации);
- сопровождать программный продукт (в соответствии с процессом сопровождения);
- осуществлять контроль и анализ всех этапов жизненного цикла исполнения договора;
- осуществлять надзор за технической реализацией, расходами, выполнением планов и отчетностью о ходе проекта;
- постоянно проводить аудит по выявлению возникающих проблем, их документальное оформление, анализ и выработку решений по их устранению.

Поставщик должен выполнять все установленные договором требования, гарантирующие, что поставляемый заказчику программный продукт соответствует исходным договорным требованиями.

2.5. Проверка и оценка. Поставщик должен:

- координировать работы по проверке выполнения договора, взаимодействуя с организацией заказчика;
- проводить или участвовать в совещаниях по подготовке к приемке и приемочным испытаниям, совместных с заказчиком аудиторских проверках в соответствии с договором и проектными планами;
- выполнять верификацию и аттестацию для того, чтобы продемонстрировать заказчику полное соответствие программных продуктов установленным требованиям;
- предоставлять заказчику отчеты о проведенных испытаниях, аудиторских проверках, испытаниях и реализованных решениях по возникшим проблемам.

2.6. Поставка и закрытие договора. Поставщик должен поставить программный продукт заказчику и обеспечить поддержку поставленного программного продукта в соответствии с условиями договора.

В заключение следует отметить, что все вышеперечисленные стандарты носят рекомендательный характер и в соответствии с Законом РФ «О стандартизации» становятся обязательными *на контрактной основе*, т. е. при ссылке на них в договоре на разработку (поставку) информационных технологий и программных продуктов.

1.5 Практические рекомендации по взаимодействию разработчика и заказчика при создании программного обеспечения

Деятельность разработчиков и заказчиков, направленная на снижение рисков, связанных с ошибками в оценке длительности и стоимости разработки заказного ПО, может осуществляться по двум сценариям:

1) взаимоотношения заказчика и разработчика строятся на взаимном доверии, просчеты в оценке проекта берет на себя в основном разработчик — *мягкое внедрение*;
2) взаимоотношения заказчика и разработчика строго регламентированы и обязательны для исполнения обеими сторонами, спорные моменты часто могут приводить к конфликтам — *жесткое внедрение.*

1.5.1 Первый сценарий (мягкое внедрение)

Первый сценарий (мягкое внедрение) ориентирован на проекты с небольшой продолжительностью (до 3-х месяцев) либо проекты, которые можно разбить на отдельные этапы: постановочный, уточняющий, стабилизирующий, внедрение. Планирование трудоемкости и оценка стоимости проводятся по каждому этапу отдельно.

Постановочный этап

Данный этап проводится по договору о консалтинге, ввиду невозможности спланировать заранее стоимость проекта. Оценка затрат производится по суммарной трудоемкости в человеко/днях. Результаты выполнения этапа оформляются в виде документа «Техническое задание» (ТЗ). Данный документ должен определять цель проекта и включать в себя описание проекта и список ключевых требований без подробной расшифровки. Несмотря на отсутствие подробного описания, состав работ должен поддаваться статистической оценке трудоемкости со стандартным отклонением (риском) в разумных рамках. Кроме того, в данном документе необходимо представить предварительную оценку экономической эффективности от внедрения проекта.

При расчете трудоемкости целесообразно использовать статистику трудоемкости (эффективности) аналогичных проектов. При отсутствии данной статистики неизбежны ошибки в оценках, в этом случае следует попробовать получить статистику, опираясь на результаты разработки прототипов. Для оценки рисков рекомендуется разработать как минимум 2 простейших прототипа:

1) «*интерфейсный прототип*», имитирующий 1- 2 важнейших диалога программы и ориентированный на изучение рисков, связанных с модификацией требований будущих пользователей;
2) «*архитектурный прототип*», отображающий наиболее критические места будущей архитектуры и предназначенный для оценки технологических рисков.

В дальнейшем данные прототипы не должны использоваться при разработке системы. Это связано с тем, что прототипы служат для нахождения оптимальных решений, но не являются сами оптимальными решениями.

Оценку рисков требуется выразить в виде возможного превышения трудоемкости (пессимистическая оценка). Именно из данной оценки следует исходить при определении общей стоимости проекта.

Условием завершения этапа является подписание сторонами технического задания.

Уточняющий этап

На данном этапе производится создание серии рабочих прототипов будущей системы, проводится согласование ее функциональности с ключевыми пользователями. Стоимость этапа составляет примерно 30% от общей стоимости разработки.

Одновременно в виде пошаговых сценариев создается «Руководство пользователя», разрабатываются отдельные пункты документа «Описание применения». Сначала формируется общее описание системы, затем готовятся должностные инструкции для пользователей. Пользователи оценивают прототип и документацию одновременно.

На данном этапе «Руководство пользователя» фактически заменяет классическое ТЗ. Такой подход имеет ряд преимуществ:

1) включение пользователя в анализ своей рабочей документации непосредственно на первых этапах разработки программы;

2) отсутствие необходимости в одновременной правке технического задания и «Руководства пользователя»;
3) достижение соответствия создаваемой документации текущему состоянию будущего проекта.

Условием завершения этапа является подписание письменного соглашения заказчика и разработчика о принятии системы при наличии ее соответствия последней согласованной версии «Руководства пользователя», стабильности архитектуры и требований к системе (допустимые изменения в ходе следующего этапа составляют не более 20%).

В случае если не удается достигнуть согласия ключевых пользователей с прототипом или описанием в документации, заказчик должен принять волевое решение на уровне топ-менеджера и определиться с требованиями. Если этого не удается достичь или требования выходят за рамки ТЗ с учетом надбавок на риск, рекомендуется пересмотреть трудоемкость/цену проекта или прекратить его разработку. Указанная возможность прекращения проекта должна быть отражена в договоре.

Стабилизирующий этап

На данном этапе устраняются недостатки в прототипах и документации и выпускается «Релиз системы». Стоимость этапа составляет примерно 50% от общей стоимости разработки.

В начале этапа составляется и согласовывается документ «Программа и методика испытаний». Данный документ содержит описание тестов, успешное выполнение которых является необходимым и достаточным условием приемки системы. Иными словами, документ описывает минимально гарантированное качество реализации. После тестирования отдельных компонентов системы по инициативе тестера и согласию заказчика «Руководство пользователя» может быть изменено.

После исправления дефектов, выявленных тестерами, выпускается первая версия системы, проходит ее окончательное тестирование и в случае успешного тестирования система объявляется готовой к сдаче в опытную эксплуатацию.

Внедрение

На данном этапе заказчик выявляет дефекты программы, обнаруженные в процессе опытной эксплуатации, а разработчик их устраняет. Ошибка это или доработка — решается на основании «Руководства пользователя». Стоимость этапа составляет примерно 10% от разработки.

После доработки разработчик изменяет «Руководство пользователя», устанавливает систему на рабочей станции заказчика, проводит обучение пользователей. Пользователи должны изучить свои должностные инструкции и подтвердить возможность эксплуатации системы в соответствии с данными инструкциями. Эту процедуру можно проводить в виде аттестации рабочего места пользователя. Если все ошибки, выявленные в процессе опытной эксплуатации, исправлены и окончательная версия программы соответствует «Руководству пользователя», то по согласованию с заказчиком разрабатываются остальные документы и принимается решение о приемке системы в промышленную эксплуатацию.

1.5.2 Второй сценарий (жесткое внедрение)

Второй сценарий (*жесткое внедрение*) предполагает соблюдение разработчиком и заказчиком одного из отечественных либо зарубежных стандартов по жизненному циклу создания программных продуктов (предпроектное обследование, проектирование, разработка, документирование, тестирование, опытная эксплуатация, сдача в промышленную эксплуатацию) и документирование каждого из его этапов. На начальном этапе заказчик представляет будущий ПП, удовлетворяющий некоторому набору требований. Эти требования формулируются, как правило, нечетко (расплывчато). Поэтому первая версия системы это обычно не готовый программный продукт, а некоторый прототип. Отличие же готового продукта от прототипа заключается в следующем: функционирование системы не требует участия разработчика, функции системы соответствуют технической документации, отсутствуют ошибки в программах.

Причина некачественной первой версии заключается, в первую очередь, в принципиальной нечеткости требований, а не в ошибках проектирования и программирования. Заказчик считает, что его требования правильны, причем это может происходить как по причине его консервативности, так и по причине нежелания признавать свои ошибки и просчеты. В этом случае разработчик может взять ответственность за нечеткость требований на себя, но потребовать увеличения сроков и стоимости проекта.

В противном случае налицо конфликтная ситуация. Во избежание этого сформулируем ряд рекомендаций по формализации отношений между заказчиком и разработчиком:

1) документы, формализующие взаимоотношения, могут быть двух типов: планы работ, определяющие, что надо сделать; функциональные и информационные модели, описывающие бизнес-процессы заказчика (как надо делать);
2) все требования и планы работ должны оформляться в документальном виде с указанием сроков и исполнителей и утверждаться первым руководителем организации;
3) требования к системе и планы работ должны детализироваться до простейших задач, имеющих однозначную трактовку;
4) необходимо заранее согласовать с заказчиком контрольные тесты (примеры) и договориться о том, что именно они являются критерием корректности работы системы.

1.6 Базовые стандарты оценки качества программных продуктов и баз данных

Формализации показателей качества программных средств посвящена также целая серия стандартов. В базовом международном стандарте ISO/МЭК 9126:1991 «Оценка программного продукта. Характеристики качества и руководство по их применению», при отборе минимума стандартизируемых показателей, описываемых в стандарте разработчиками, выдвигались и учитывались следующие принципы:

- ясность и измеряемость значений;
- независимость между используемыми показателями;
- соответствие установившимся понятиям и терминологии;
- возможность последующего уточнения и детализации.

Выделенные в стандарте характеристики позволяют оценивать ПО *с позиции пользователя, разработчика и управляющего проектом* [11].

Пользователи в основном проявляют заинтересованность к возможностям применения программного обеспечения, его производительности и результатам использования и оценивают ПО без изучения его внутренних аспектов или условий создания программного обеспечения.

Пользователя могут интересовать следующие вопросы:

- наличие требуемых функций в программном обеспечении;
- надежность программного обеспечения;
- эффективность программного обеспечения;
- удобство при использовании ПО;
- простота переноса ПО в другую среду.

Процесс создания требует от пользователя и разработчика использования одних и тех же характеристик качества программного обеспечения, так как они применяются для установления требований и приемки. Когда разрабатывается программное обеспечение для продажи, в требованиях должны быть отражены предполагаемые потребности.

Так как разработчики отвечают за создание программного обеспечения, которое должно удовлетворять требованиям качества, они заинтересованы в качестве промежуточной продукции так же, как и в качестве конечной продукции.

Представление пользователя также должно включать представление о характеристиках качества, требуемое тем, кто сопровождает программное обеспечение.

Руководитель проекта может быть больше заинтересован в общем качестве, чем в конкретной характеристике качества, и по этой причине будет нуждаться в определении важности значений, отражающих коммерческие требования для индивидуальных характеристик.

Руководителю также может потребоваться сопоставление повышения качества с критериями управляемости, такими как плановая задержка или перерасход стоимости, потому что он желает оптимизировать качество в пределах ограниченной стоимости, трудовых ресурсов и установленного времени.

В стандарте рекомендуется использовать шесть основных характеристик качества ПО, каждая из которых детализируется еще несколькими субпоказателями. В настоящее время подготовлен проект модернизации стандарта, в котором предполагается выделить четыре части:

1) модель показателей качества и их развития в жизненном цикле ПО;
2) внешние метрики качества, которые отражают наиболее общие характеристики программного обеспечения;
3) внутренние метрики качества, характеризующие структуру и конструктивные особенности ПО;

4) метрики качества ПО с позиции пользователя:
   - эффективность применения ПО пользователем;
   - производительность при решении основных функциональных задач;
   - степень защищенности от внешних угроз;
   - удовлетворенность пользователей качеством решения задач.

Близкими к описанному стандарту по идеологии, структуре и содержанию являются следующие отечественные стандарты:

- ГОСТ 28806-90 «Качество программных средств. Термины и определения». Стандарт устанавливает термины и определения в области качества программных средств. Термины, установленные настоящим стандартом, обязательны для применения во всех видах документации и литературы по вычислительной технике и программным средствам;
- ГОСТ 28195-89 «Оценка качества программных средств. Общие положения». Стандарт устанавливает общие положения по оценке качества программных средств вычислительной техники (ПО), поставляемых через фонды алгоритмов и программ (ФАП), номенклатуру и применяемость показателей качества ПО;
- ГОСТ Р ИСО/МЭК 9126-93 «Оценка программного продукта. Характеристики качества и руководящие указания по их применению». Стандарт определяет шесть характеристик, которые с минимальным дублированием описывают качество программного обеспечения, данные характеристики образуют основу для дальнейшего уточнения и описания качества программного обеспечения. Термины, установленные настоящим стандартом, обязательны для применения во всех видах документации и литературе по аналого-цифровой вычислительной технике, входящих в сферу работ по стандартизации и (или) использующих результаты этих работ. Определения характеристик и соответствующая модель процесса оценки качества, приведенные в стандарте, применимы тогда, когда определены требования для программной продукции и оценивается ее качество на всех этапах жизненного цикла: приобретение, разработка, эксплуатация, поддержка, сопровождение. Эти характеристики могут применяться к любому виду программного обеспечения, включая программы ЭВМ и данные, входящие в программно-технические средства (встроенные программы);
- ГОСТ Р 51188-98 «Испытания программных средств на наличие компьютерных вирусов. Типовое руководство». Стандарт распространяется на испытания ПО и его компонентов, цели которых — обнаружить в этом ПО и устранить из него компьютерные вирусы силами специальных предприятий (подразделений), и устанавливает общие требования к организации и проведению таких испытаний. Стандарт предназначен для применения в испытательных лабораториях, проводящих сертификационные испытания ПО на выполнение требований защиты информации;
- ГОСТ Р ИСО/МЭК 12119-2000 «Пакеты программ. Требования к качеству и тестирование». Настоящий стандарт применяется для пакетов прикладных программ: текстовых процессоров, электронных таблиц, программ баз

данных, графических пакетов, программ, реализующих технические и научные функции, сервисных программ (утилит), которые являются объектами продажи и поставки;

- ГОСТ Р ИСО/МЭК 15026-2002 «Уровни целостности систем и программных средств». Стандарт содержит основные положения по определению уровней целостности, определяет процессы для установления уровней целостности, требований к сохранению целостности программных средств. Стандарт предназначен для применения разработчиками, пользователями, поставщиками и экспертами программных продуктов или систем, содержащих программные средства, а также для административной и технической поддержки данных продуктов и систем.

В вышеуказанных документах для оценки качества ПО выделяются шесть характеристик: функциональные возможности, надежность, практичность, эффективность, сопровождаемость, мобильность.

Функциональные возможности — набор атрибутов, относящихся к сути набора реализуемых в программном продукте функций и их конкретным свойствам (установленные или предполагаемые потребности).

Надежность — набор атрибутов, относящихся к способности программного обеспечения сохранять свой уровень качества функционирования при установленных условиях за установленный период времени.

Практичность — набор атрибутов, относящихся к объему работ, требуемых для использования и индивидуальной оценки такого использования определенным и предполагаемым кругом пользователей.

Эффективность — набор атрибутов, относящихся к соотношению между уровнем качества функционирования программного обеспечения и объемом используемых ресурсов при установленных условиях.

Сопровождаемость — набор атрибутов, относящихся к объему работ, требуемых для проведения конкретных изменений (модификаций).

Мобильность — набор атрибутов, относящихся к способности ПО быть перенесенным из одного окружения в другое.

В случае необходимости относительная важность каждой характеристики может изменяться в зависимости от специфики программного продукта и области применения. Например, надежность наиболее важна для программного обеспечения боевых критичных систем, эффективность наиболее важна для ПО критичных по времени систем реального времени, а практичность — для ПО диалога конечного пользователя.

В дальнейшем каждая характеристика должна быть детализирована на подхарактеристики. В стандарте ISO/МЭК 9126:1991 предлагается набор подхарактеристик, представленный на рис. 1.25

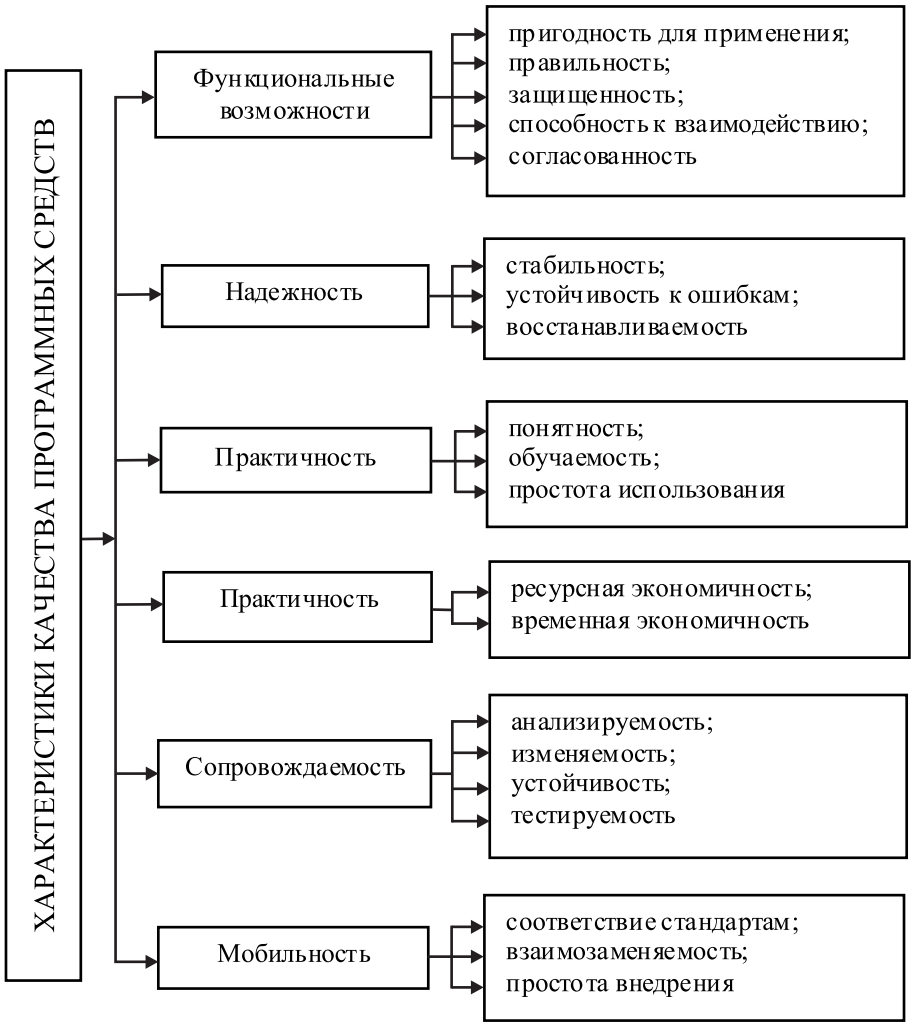


Рис. 1.25 – Характеристики качества ПО

Детализация каждой характеристики на подхарактеристики является необходимым этапом построения качественной модели оценивания программного продукта.

В настоящее время в литературе имеется описание ряда подобных моделей, однако степень их завершенности пока не позволяет создавать какой-либо единый стандарт.

Набор вышеуказанного стандарта подхарактеристик включает:

функциональные возможности:

пригодность — атрибут ПО, относящийся к наличию и соответствию набора функций конкретных задач;

правильность — атрибуты ПО, относящиеся к обеспечению правильности или соответствия результатов или эффектов;

способность к взаимодействию — атрибуты программного обеспечения, относящиеся к способности его взаимодействовать с конкретными системами;

согласованность — атрибуты ПО, заставляющие программу придерживаться соответствующих стандартов или соглашений, или положений законов, или подобных рекомендаций;

защищенность — атрибуты ПО, относящиеся к его способности предотвращать несанкционированный доступ, случайный или преднамеренный, к программам и данным;

надежность:

стабильность — атрибуты ПО, относящиеся к частоте отказов при ошибках в программном обеспечении;

устойчивость к ошибке — атрибуты программного обеспечения, относящиеся к его способности поддерживать определенный уровень качества функционирования в случаях программных ошибок или нарушения определенного интерфейса;

восстанавливаемость — атрибуты программного обеспечения, относящиеся к его возможности восстанавливать уровень качества функционирования и данные, непосредственно поврежденные в случае отказа, а также к времени и усилиям, необходимым для этого;

практичность:

понятность — атрибуты программного обеспечения, относящиеся к усилиям пользователя по пониманию общей логической концепции и ее применимости;

обучаемость — атрибуты программного обеспечения, относящиеся к усилиям пользователя по обучению его применению (например, оперативному управлению, вводу, выводу);

простота использования — атрибуты ПО, относящиеся к усилиям пользователя по эксплуатации и оперативному управлению;

эффективность:

временная экономичность — атрибуты программного обеспечения, относящиеся к временам отклика и обработки и к скоростям выполнения его функций;

ресурсная экономичность — атрибуты ПО, относящиеся к объему используемых ресурсов и продолжительности такого использования при выполнении функции;

сопровождаемость:

анализируемость — атрибуты программного обеспечения, относящиеся к усилиям, необходимым для диагностики недостатков или случаев отказов или определения составных частей для модернизации;

изменяемость — атрибуты программного обеспечения, относящиеся к усилиям, необходимым для модификации, устранению отказа или для изменения условий эксплуатации;

устойчивость — атрибуты ПО, относящиеся к риску от непредвиденных эффектов модификации;

тестируемость — атрибуты ПО, относящиеся к усилиям, необходимым для проверки модифицированного программного обеспечения;

мобильность:

адаптируемость — атрибуты ПО, относящиеся к удобству его адаптации к различным конкретным условиям эксплуатации без применения других действий или способов, кроме тех, что предназначены для этого в рассматриваемом ПО;

простота внедрения — атрибуты программного обеспечения, относящиеся к усилиям, необходимым для внедрения программного обеспечения в конкретное окружение;

соответствие — атрибуты программного обеспечения, которые заставляют программу подчиняться стандартам или соглашениям, относящимся к мобильности;

взаимозаменяемость — атрибуты ПО, относящиеся к простоте и трудоемкости его применения вместо другого конкретного программного средства в среде этого средства.

Немаловажной проблемой при оценке характеристик и подхарактеристик является выбор соответствующих метрик (показателей) их описания. В литературе существует несколько общепринятых метрик для характеристик (подхарактеристик), описанных выше.

Пример

Примеры описания характеристики практичности представлены в табл. 1.1.

Таблица 1.1 – Метрика характеристики «Практичность»

Характеристики качества	**Мера**	**Шкала**
Понятность:		
четкость концепции;	Порядковая	Отл., хор.
демонстрационные возможности;		Удовл., неудовл.
наглядность и полнота документации		
Простота использования:		
простота управления функциями;	Порядковая	Отл., хор.
комфортность эксплуатации;		Удовл., неудовл.
среднее время ввода заданий;	Секунды	1–1000
среднее время отклика на задание	Секунды	1–1000
Обучаемость:		
трудоемкость изучения применения ПС;	Чел.-ч	1–1000
продолжительность изучения;	часы	1–1000
объем эксплуатационной документации	Страницы	1–1000

Участники процесса оценивания могут разрабатывать и собственные метрики.

Вышеописанные стандарты рекомендуется применять для установления требований к качеству ПО и оценивания (измерения, ранжирования и оценки) программных продуктов после завершения их разработки в следующих случаях:

- определение требований к качеству программной продукции;
- оценивание технических требований к ПО при контроле за выполнением требования к качеству в процессе разработки;
- описание признаков и свойств (атрибутов) внедренного ПО (например, в руководствах пользователя);
- оценивание разработанного ПО перед его поставкой;
- оценивание программного обеспечения перед приемкой.

Для оценки баз данных (БД) пока отсутствуют какие-либо стандарты, регламентирующие их показатели качества. Поэтому ниже представлен набор характеристик, который наиболее часто используется на практике при решении этой проблемы. В системах баз данных доминирующее значение приобретают сами данные, их хранение и обработка. Поэтому при анализе качества БД целесообразно выделить два компонента [8]:

1) программные средства системы управления базой данных, независимые от сферы их применения и смыслового содержания накапливаемых и обрабатываемых данных;
2) информация БД, доступная для обработки и использования в конкретной проблемно-ориентированной сфере применения.

Практически весь набор показателей качества ПО, изложенный выше, в той или иной степени может использоваться при анализе и оценке качества СУБД. Особенности состоят в изменении акцентов при выборе и упорядочении этих показателей качества. Почти во всех случаях важнейшими показателями качества СУБД являются функциональные характеристики процессов формирования и изменения информационного наполнения БД администраторами, а также доступа к данным и представления результатов пользователям БД.

Качество интерфейса специалистов с БД, обеспечиваемого средствами СУБД, оценивается в значительной степени субъективно, однако имеется ряд характеристик, которые можно оценивать достаточно корректно.

Вторым компонентом БД является собственно накапливаемая и обрабатываемая информация в базе данных. Выделяемые показатели качества должны иметь практический интерес для пользователей БД и быть упорядочены в соответствии с приоритетами практического применения. Кроме того, каждый выделяемый для проверки показатель должен быть пригоден для достаточно достоверного измерения и сравнения с требуемым значением при испытаниях и сертификации.

Функциональными показателями качества информации БД являются:

- полнота накопленных описаний объектов — относительное число объектов или документов, имеющихся в БД, к общему числу объектов по данной тематике или к числу объектов в аналогичных БД по той же тематике;

- достоверность — степень соответствия данных об объектах в БД реальным объектам вне ЭВМ в данный момент времени, определяющаяся изменениями самих объектов, некорректностью записей об их состоянии или некорректностью расчетов их характеристик;
- идентичность данных — относительное число описаний объектов, не содержащих ошибки, к общему числу документов об объектах в БД;
- актуальность данных — относительное число морально устаревших данных об объектах в БД к общему числу накопленных и обрабатываемых данных.

К конструктивным показателям качества информации в БД относятся в основном объемно-временные характеристики сохраняемых и обрабатываемых данных:

- объем базы данных — число записей описаний объектов или документов в БД, доступных для хранения и обработки;
- оперативность — степень соответствия изменений данных в процессе сбора и обработки состояниям реальных объектов или величина запаздывания между появлением или изменением характеристик реального объекта и его отражением в БД;
- периодичность — промежуток времени между поставками двух последовательных, достаточно различающихся информацией версий БД;
- глубина ретроспективы — интервал времени от даты выпуска и/или записи в базу данных самого раннего документа до настоящего времени;
- динамичность — относительное число изменяемых описаний объектов к общему числу записей в БД за некоторый интервал времени, определяемый периодичностью издания версий БД.

К конструктивным показателям относятся и все показатели защищенности информации. Защищенность реализуется, в основном, программными средствами СУБД, однако в сочетании с поддерживающими их средствами организации данных. В распределенных базах данных показатели защищенности тесно связаны с характеристиками целостности данных. Эти показатели отражают степень тождественности данных в памяти удаленных компонентов распределенной БД.

При реальном функционировании баз данных важную роль играют временные характеристики взаимодействия конечных пользователей и администраторов БД в процессе эксплуатации базы данных по прямому назначению. Эти характеристики зависят от качества СУБД, а также от объема, структуры и показателей качества используемой информации. Выше они отражены критерием эффективности использования ресурсов ЭВМ программными средствами, в данном случае СУБД.

Для баз данных важнейшим ресурсом является память ЭВМ, занимаемая информацией БД, а также используемость этого ресурса. Эти показатели качества влияют на время реакции БД на разные виды запросов пользователей и на пропускную способность БД при эксплуатации.

Контрольные вопросы по главе 1

1) Назовите и прокомментируйте основные причины появления программной инженерии как методологии разработки программного обеспечения.
2) Перечислите специфику инженерной деятельности как таковой и ее особенности при разработке ПО.
3) Перечислите и прокомментируйте основные принципы «Кодекса этических норм профессионала в области программной инженерии».
4) Раскройте содержание модели технологического процесса создания программного продукта.
5) Раскройте содержание объектно-ориентированного подхода при описании бизнес-процессов предметной области.
6) Раскройте содержание структурного (функционального) подхода при описании бизнес-процессов предметной области.
7) Раскройте содержание и особенности каскадной модели жизненного цикла ПО.
8) Раскройте содержание и особенности спиральной модели жизненного цикла ПО.
9) Раскройте содержание и особенности эволюционной модели жизненного цикла ПО.
10) Дайте понятие CASE-средств и перечислите их функциональные возможности на этапе моделирования предметной области.
11) Дайте понятие CASE-средств и перечислите их функциональные возможности на этапе реализации программного продукта.
12) Дайте понятие CASE-средств и перечислите их функциональные возможности на этапе тестирования и документирования программного продукта.
13) Раскройте содержание руководства к Своду Знаний по Программной Инженерии — «SWEBOK»: раздел «определение требований».
14) Раскройте содержание руководства к Своду Знаний по Программной Инженерии — «SWEBOK»: раздел «проектирование».
15) Раскройте содержание руководства к Своду Знаний по Программной Инженерии — «SWEBOK»: раздел «конструирование».
16) Раскройте содержание руководства к Своду Знаний по Программной Инженерии — «SWEBOK»: раздел «тестирование».
17) Раскройте содержание руководства к Своду Знаний по Программной Инженерии — «SWEBOK»: раздел «эксплуатация».
18) Раскройте содержание Государственного стандарта РФ ГОСТ Р ИСО/МЭК 12207-99 «Информационная технология. Процессы жизненного цикла программных средств».

19) Прокомментируйте особенности «мягкого внедрения» при взаимодействии разработчика и заказчика.
20) Раскройте содержание отечественного стандарта по оценке качества ПС.
21) Раскройте содержание показателей качества и функциональной пригодности.
22) Раскройте содержание показателей надежности и применимости.
23) Раскройте содержание показателей «эффективность» и «сопровождаемость».
24) Раскройте содержание показателя «переносимость».
25) Перечислите и прокомментируйте функциональные показатели информации БД.
26) Перечислите и прокомментируйте конструктивные показатели качества БД.

Глава 2

ОСНОВЫ УПРАВЛЕНИЯ ПРОГРАММНЫМИ ПРОЕКТАМИ

2.1 Основные понятия и определения

Классическое управление проектами выделяет два вида организации человеческой деятельности: операционную и проектную. *Операционная деятельность* (например, функционирование службы поддержки пользователя) применяется, когда внешние условия хорошо известны и стабильны, когда производственные операции хорошо изучены и неоднократно испытаны, а функции исполнителей определены и постоянны. В этом случае основой эффективности служат узкая специализация и повышение компетенции. *Проектная деятельность* (например, создание нового программного продукта) используется в том случае, когда разрабатывается новый продукт, внешние условия и требования к которому постоянно меняются, где применяются в основном оригинальные методы и технологии разработки, где постоянно требуются поиск новых решений, интеллектуальные усилия и творчество. Задача проектной деятельности — достижение конкретной бизнес-цели, задача операционной деятельности — обеспечение нормального течения бизнеса.

Другими словами, операционная и проектная деятельности различаются, главным образом, тем, что операционная деятельность — это продолжающийся во времени и повторяющийся процесс, в то время как проекты являются уникальными по содержанию и всегда ограничены во времени.

Основные особенности управления программными проектами заключаются в следующем:

- программный продукт не материален, его нельзя увидеть в процессе конструирования и, следовательно, оперативно повлиять на его реализацию;
- жизненный цикл ПП в существующих стандартах описан в общем виде и прямо не ориентирован на специфику конкретного продукта, необходимо адаптировать его к виду и типу проекта и разработать методики его выполнения исполнителями;

- программные продукты как результаты творческого труда не поддаются точному оцениванию как по времени создания, так и по требуемому бюджету.

Вследствие этих особенностей только 35% проектов завершились в срок, не превысили запланированный бюджет и реализовали все требуемые функции и возможности.

46% проектов завершились с опозданием, расходы превысили запланированный бюджет, требуемые функции не были реализованы в полном объеме. Среднее превышение сроков составило 120%, среднее превышение затрат 100%, обычно исключалось значительное число функций.

19% проектов полностью провалились и были аннулированы до завершения по следующим причинам: требования заказчика не выполняются; проект не вложился в стоимость; проект не вложился в заданные сроки; этапы работ оказались нескоординированными друг с другом [11].

В литературе приводятся следующие определения проекта как законченного продукта [3, 12, 13]:

1) проект — комплекс взаимосвязанных мероприятий, предназначенных для достижения целевых результатов при решении многокритериальных задач в течение заданного периода времени при установленном бюджете в условиях ограниченных ресурсов;

2) проект — это ограниченное по времени целенаправленное изменение исследуемой системы с установленными требованиями к качеству результатов, возможными объемами расхода средств, ресурсов и специфической организацией управления. Управление проектом — это управление этими изменениями с высокой степенью уверенности в успешном исходе;

3) проект — временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов;

4) проект — это комплекс усилий, предпринимаемых с целью получения конкретных уникальных результатов в рамках отведенного времени и в пределах утвержденного бюджета, который выделяется на оплату ресурсов, используемых или потребляемых в ходе проекта;

5) проект есть комплекс действий, направленных на получение уникального результата, будь то продукт или услуга. Суть результата — его содержание. Для информационной системы — ее функциональность;

6) проект представляет собой комплекс уникальных действий, не опирающийся на организационную структуру, имеющий определенные дату начала и окончания, расписание, стоимость и технические задачи.

Следует различать следующие понятия: цели, результаты, ограничения и допущения.

Цель проекта описывает, какие задачи должны быть решены в результате проекта, желаемый результат, достигнутый в пределах некоторого интервала времени, другими словами, должен отвечать на вопрос, *зачем* данный проект нужен, какие потребности бизнеса он удовлетворяет. Например, целями проекта могут быть:

- завоевание значительной доли растущего рынка за счет вывода на него нового продукта;
- разработка ПО под потребности конкретного заказчика;
- доработка программного продукта в целях приведения его в соответствие с изменениями в законодательстве.

Цели должны быть значимыми, конкретными, измеримыми и достижимыми. Четкое определение бизнес-целей важно, поскольку существенно влияет на все процессы и решения в проекте. Проект должен быть закрыт, если признается, что достижение цели невозможно или стало нецелесообразным. Например, если реальные затраты на проект будут превосходить будущие доходы от его реализации.

Результаты проекта отвечают на вопрос, *что* должно быть получено после его завершения. Результаты проекта должны определять: какие именно бизнес-выгоды получит заказчик в результате проекта; какой продукт или услуга будут получены по окончании проекта; краткое описание и при необходимости ключевые свойства и/или характеристики продукта/услуги.

Следует помнить, что результаты проекта должны быть измеримыми. Это означает, что при оценке результатов проекта должна иметься возможность сделать заключение: достигнуты оговоренные в концепции результаты или нет.

Ограничения являются неотъемлемой частью проекта и сокращают возможности проектной команды в выборе решений. В частности, они могут содержать:

- требования обязательной сертификация продукта, услуги на соответствие определенным стандартам;
- требования на использование конкретной заданной программно-аппаратной платформы;
- специфические требования к защите информации.

Допущения, как правило, тесно связаны с управлением рисками, о которых заказчик должен знать заранее. Например, оценивая проект по схеме с фиксиро-

ванной ценой, можно записать в допущении предположение о том, что стоимость лицензий на ПО, приобретаемое на стороне, не изменится до завершения проекта.

Оптимальная реализация программного проекта должна обеспечить достижение конкретной бизнес-цели при соблюдении ограничений «железного треугольника» (рис. 2.1) [22].

Рис. 2.1 – «Железный треугольник» ограничений проекта

Ни один из углов треугольника не может быть изменен без изменения других. Например, чтобы уменьшить время, потребуется увеличить бюджет и/или сократить содержание.

В приложении к индустрии программного обеспечения к трем основным ограничениям «железного треугольника» добавляют четвертое ограничение — *приемлемое качество* [22]. В этом случае система ограничений должна строиться на основе приоритетов проекта и должна учитывать требования потребителей к создаваемому продукту или услуге. Если необходим жесткий предопределенный набор функциональности — понятно, что «плавающими» характеристиками проекта (вторичными по своей природе, требующими компромисса в контексте требуемого объема функциональности) будут требуемое время и необходимый бюджет (рис. 2.2).

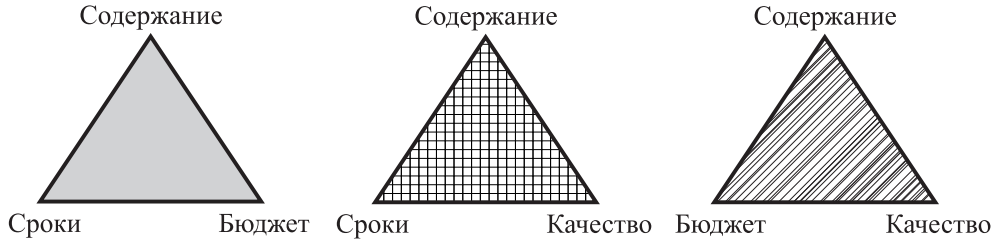


Рис. 2.2 – Возможные варианты «железного треугольника» ограничений проекта

Если бюджет не является жестко определенным и может быть предметом обсуждения, другие ограничения все равно будут играть свою роль. Поэтому считается, что неотъемлемой частью любого программного проекта является *анализ компромиссов*, направленный на выделение функциональных требований, которые можно реализовать в заданных ограничениях проекта, будь то сроки, бюджет, качество или другие характеристики. В общем случае, *задача такого анализа* - нахождение баланса, приемлемого для всех сторон, связанных с проектом; заказчиков, которым нужна определенная функциональность в конкретные сроки, при имеющемся бюджете; исполнителей, которые обладают бюджетом, достаточным для определенной стоимости использования ресурсов, трудоемкости проекта и достижения качества в заданные сроки. Согласно стандарту PMBOK проект считается успешным, если он выполнен в срок в соответствии со спецификациями и в пределах запланированного бюджета.

Общепринято, что не существует единственного правильного процесса разработки ПО. В каждом новом проекте в соответствии с «Законом 4-х П» (рис. 2.3) процесс должен определяться каждый раз заново, в зависимости от проекта, продукта и персонала.

Рис. 2.3 – «Закон 4-х П»

Совершенно разные процессы должны применяться в проектах, в которых участвуют 5 человек, и в проектах, в которых участвуют 500 человек. Если продуктом проекта является критическое ПО, например система управления атомной электростанцией, то процесс разработки должен сильно отличаться от разработки сайта. И, наконец, по-разному следует организовывать процесс разработки в команде начинающих программистов и в команде состоявшихся профессионалов [22].

Управление проектом — это руководство работами команды исполнителей проекта для реализации проекта с использованием общих методов, планирования и контроля работ (видение будущего продукта, стартовые операции, планирование итераций, мониторинг и отчетность), планирование и управление рисками, эффективной организацией работы команды и коммуникационными потоками в команде исполнителей.

В стандарте PMBOK (Project Management Body Of Knowledge — Свод знаний по управлению проектами) приведены 9 процессов (областей знаний) по управлению проектами, каждый из которых состоит из определенного набора работ, и пять этапов (фаз) жизненного цикла проекта: инициация, планирование, исполнение, мониторинг и управление, завершение [1]. При этом процессы взаимосвязаны, а *этапы проекта* могут накладываться во времени друг на друга. Распределение работ по этапам жизненного цикла, рекомендованное стандартом, приведено в табл. 2.1.

Таблица 2.1 – Распределение процессов управления проектом по этапам

Процессы и области знаний	**Группы процессов управления проектами по фазам проекта**				
	Инициация	**Планирование**	**Исполнение**	**Мониторинг и управление**	**Завершение**
1 Интеграция управления проектом	1.1 Разработка Устава проекта 1.2 Разработка предварительного описания содержания проекта	1.3 Разработка плана управления проектом	1.4 Руководство и управление исполнением проекта	1.5 Мониторинг и контроль (управление) работ проекта 1.6 Общее управление изменениями	1.7 Закрытие проекта
2 Управление содержанием проекта		2.1 Планирование содержания проекта 2.2 Определение содержания 2.3 Создание иерархической структуры работ (ИСР)		2.4 Подтверждение содержания 2.5 Управление содержанием	
3 Управление сроками проекта		3.1 Определение состава операций 3.2 Определение взаимосвязей операций 3.3 Оценка ресурсов операций 3.4 Оценка длительности операций 3.5 Разработка расписания		3.6 Управление расписанием	
4 Управление стоимостью проекта		4.1 Стоимостная оценка 4.2 Разработка бюджета расходов		4.3 Управление стоимостью	
5 Управление качеством проекта		5.1 Планирование качества	5.2 Процесс обеспечения качества	5.3 Процесс контроля качества	

продолжение на следующей странице

Таблица 2.1 – Распределение процессов управления проектом по этапам

Процессы и области знаний	**Группы процессов управления проектами по фазам проекта**				
	Инициация	**Планирование**	**Исполнение**	**Мониторинг и управление**	**Завершение**
6 Управление человеческими ресурсами проекта		6.1 Планирование человеческих ресурсов	6.2 Набор команды проекта 6.3 Развитие команды проекта	6.4 Управление командой проекта	
7 Управление коммуникациями проекта		7.1 Планирование коммуникаций	7.2 Распространение информации 7.3 Отчетность по исполнению	7.4 Управление участниками проекта	
8 Управление рисками проекта	8.1 Планирование управления рисками 8.2 Идентификация рисков 8.3 Качественный анализ рисков 8.4 Количественный анализ рисков 8.5 Планирование реагирования на риски		8.6 Мониторинг и управление рисками		
9 Управление поставками проекта		9.1 Планирование покупок и приобретений 9.2 Планирование контрактов	9.3 Запрос информации у продавцов 9.4 Выбор продавцов	9.5 Администрирование контрактов	9.6 Закрытие контракта

На этапе *инициации проекта* необходимо выбрать и обосновать вид (тип) программного продукта, который компания собирается разрабатывать, и определить рыночную нишу его распространения, разработать и утвердить концепцию проекта. Недостаточное внимание этого этапа проекта неизбежно приводит к существенным проблемам при планировании, реализации и завершении проекта.

Планирование программного проекта относится к работам, предпринимаемым для подготовки к успешному ведению программно-инженерной деятельности реализации программного продукта и представляет собой процессы структурного планирования проекта, распределения и назначения ресурсов (материальных и людских) с учетом стоимости и времени выполнения проекта в целом и его отдельных работ. Основополагающими методами планирования, применяемыми на практике, являются: *метод критического пути* — CPM (Critical Path Method) и *метод анализа и оценки программ* PERT (Program Evaluation and Review Technique).

Как правило, редкий проект выполняется в соответствии с первоначальными планами, поэтому важным элементом системы управления проектом является периодический *мониторинг* его состояния, анализ причин расхождения с планом и разработка корректирующих воздействий.

Завершение проекта относится к фиксированию результатов программного проекта после передачи полученного программного продукта в эксплуатацию. На этом этапе проводятся приемо-сдаточные испытания (ПСИ) продукта на предмет соответствия его свойств определенным ранее требованиям. Критерии приемки должны определять числовые значения характеристик системы, которые должны быть продемонстрированы по результатам приемо-сдаточных испытаний или опытной эксплуатации и однозначно свидетельствовать о достижении целей проекта. Для проведения процедуры приемки-сдачи создаются специальные документы — программа и методика испытаний программного продукта. Завершение наступает, когда достигнуты цели проекта; или осознано, что цели проекта не будут или не могут быть достигнуты; или исчезла необходимость в проекте и он прекращается.

2.2 Управление рисками проекта

В методологии по управлению IT-проектами Microsoft Solutions Framework (MSF) компании Microsof [14] под риском проекта понимается событие или условие, которое может оказать как негативное, так и позитивное влияние на итоги проекта, и отмечается, что риски не есть проблемы.

Проблемы — это нечто, имеющее место в настоящее время, в то время как риски относятся к будущему и носят вероятностный характер (могут и не состояться). Однако риски могут стать проблемами, если ими эффективно не управлять.

Цель управления рисками — максимизировать их положительное влияние (открывающиеся возможности), но при этом минимизировать связанные с ними нега-

тивные факторы (убытки). К минимизации рисков стремятся все потенциальные участники проекта: разработчики (поставщики) ПП, заказчики (потребители), а также предполагаемые инвесторы. Управление рисками — это определенная деятельность, которая выполняется в проекте от его начала и до завершения. О положении дел в проекте нужно судить не по количеству рисков, связанных с его выполнением, а по степени проработанности процедуры их выявления, анализа и управления ими.

Процесс управления рисками состоит из логически взаимосвязанных этапов: *идентификация рисков, анализ рисков, планирование рисков, мониторинг и управление рисками* [12]. Необходимо отметить, что описанные этапы являются логическими шагами и не обязательно должны следовать друг за другом в строгом хронологическом порядке. Проектные группы могут циклически повторять шаги выявления-анализа-планирования по мере обнаружения дополнительных факторов, влияющих на проект.

Идентификация рисков — этап, позволяющий определить и вынести на обсуждение команды факты наличия рисков, способных повлиять на проект, и документально оформить их характеристики. Выявление рисков является начальной стадией процесса управления ими. Это интерактивный процесс, который периодически повторяется на всем протяжении проекта, поскольку в рамках его жизненного цикла могут обнаруживаться и новые риски.

Исходные данные для выявления и описания характеристик рисков могут браться из разных источников. В первую очередь, это информация о выполнении прежних проектов. Следует помнить, что проблемы завершенных и выполняемых проектов — это, как правило, риски в новых проектах. Другим источником данных о рисках проекта может служить разнообразная информация из открытых источников, научных работ, маркетинговая аналитика и другие исследовательские работы в данной области. Каждый проект задумывается и разрабатывается на основании ряда гипотез, сценариев и допущений. Неопределенность в допущениях проекта следует также обязательно рассматривать в качестве потенциального источника возникновения рисков проекта.

Результатом идентификации рисков должен стать список рисков с описанием их основных характеристик. Рекомендуется каждый риск формулировать на естественном языке причинно-следственной связи между реально существующим фактором проекта и потенциально возможным, еще не случившимся событием или ситуацией [12]. Первая часть формулировки риска — условие — содержит описание существующего фактора или особенности проекта, которые, по мнению членов проектной группы, могут сделать результат проекта убыточным либо сократить получаемую от проекта прибыль. Вторая часть формулировки риска называется последствием. Она описывает ту нежелательную ситуацию, которой следует избежать. Пример формулировки выявленных рисков представлен в табл. 2.2.

Таблица 2.2 – Описание рисков проекта

Причина	**Условия**	**Последствия**	**Ущерб**
Требования не ясны	Отсутствие описания сценариев использования системы	Задержка начала разработки ППО. Большой объем переработок	Задержки в сроках сдачи готового продукта и дополнительные трудозатраты
Недостаток квалифицированных кадров	Архитектура и код низкого качества	Большое число ошибок. Большие затраты на их исправление	Задержки в сроках сдачи готового продукта и дополнительные трудозатраты
Текучесть кадров	Частая смена участников команды	Низкая производительность при вводе новых участников в проект	Задержки в сроках сдачи готового продукта и дополнительные трудозатраты

Выявление рисков — ответственный и важный этап проекта. Знание о существовании рисков — необходимое условие эффективной работы по предотвращению рисков.

Анализ рисков — этап обработки данных, накопленных при идентификации рисков в формы, позволяющие осуществить качественную и количественную оценки рисков: вероятности наступления риска, его угрозы, ранжирование рисков по степени возможных угроз, ожидаемую величину потерь и т. д.

Качественный анализ рисков проекта включает определение вероятности наступления рисков, тяжести последствий от рисков, степени опасности (ранга) риска, близости наступления риска [12]. Для измерения параметров рисков применяются, как правило, порядковые шкалы либо шкалы интервалов. Определение тяжести последствий от рисков предлагается оценивать в шкале интервалов (табл. 2.3).

Таблица 2.3 – Относительная шкала оценки воздействия рисков

Количественное значение оценки	< 0,4	0,4–0,7	> 0,7
Качественное значение оценки	Умеренные	Критичные	Катастрофические
Потери от наступления риска	Потери менее...	Потери от... до...	Потери более...

Риск может воздействовать и на сроки проекта, и на качество получаемого продукта, но все эти отклонения могут быть оценены в денежном эквиваленте. Например, последствия задержки по срокам могут быть выражены в сумме денежных санкций в контракте.

Похожая шкала может быть применена для оценки вероятности наступления риска (табл. 2.4).

Таблица 2.4 – Относительная шкала измерения вероятности наступления риска

Количественное значение вероятности	< 0,4	0,4–0,7	> 0,7
Качественное значение вероятности	Маловероятно	Возможно	Очень вероятно
Возможность наступления риска	Наступление события весьма сомнительно	Шансы равны	Шансы наступления весьма велики

Ранжирование рисков позволяет проектной группе управлять наиболее важными из них, выделяя для этого необходимые ресурсы. Для определения ранга риска используется информация матриц вероятностей и воздействий (табл. 2.5). Ранг риска определяет его порядковый номер в полной совокупности рисков проекта. Чем выше ранг, тем более опасен риск.

Таблица 2.5 – Матрица рангов выявленных рисков проекта

Причина	**Вероятность**	**Воздействие**	**Ранг**
Требования не ясны	Очень вероятно	Катастрофическое	9
Недостаток квалифицированных кадров	Очень вероятно	Критичные	6
Текучесть кадров	Возможно	Критичные	4

Одной из важных характеристик риска является *близость его наступления.* Естественно, что при прочих равных условиях рискам, которые могут осуществиться уже завтра, следует сегодня уделять больше внимания, чем тем, которые могут произойти не ранее чем через полгода. Возможная шкала оценки близости риска представлена в табл. 2.6.

Таблица 2.6 – Относительная шкала измерения близости наступления риска

Количественное значение близости наступления	Больше чем через . . .	От . . . до	Меньше чем через. . . .
Качественное значение *близости наступления*	Очень нескоро	Не очень скоро	Очень скоро

Итоговые результаты анализа рисков подробно оформляются в виде следующих документов (табл. 2.7).

Таблица 2.7 – Пример карточки с описанием риска

Номер: R-101	Категория: технологический
Причина: недостаток квалифицированных кадров	Симптомы: Разработчики будут использовать новую платформу -J2EE.
Последствия: низкая производительность разработки	Воздействие: Увеличение сроков и трудоемкости разработки
Вероятность: очень вероятно	Степень воздействия: критическая
Близость: очень скоро.	Ранг: 6
Исходные данные: «Содержание проекта», «План обеспечения ресурсами», протоколы совещаний №21 от . . . , №27 от	

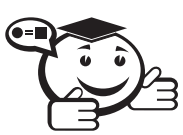
Оценка рисков должна вестись постоянно. Обстоятельства, в которых проектная группа работает над созданием решения, обладают постоянной изменчивостью, следовательно, команда должна регулярно проводить переоценку выявленных рисков и постоянно следить за появлением новых. Управление рисками должно быть интегрировано в общий жизненный цикл проекта.

Результаты качественного анализа используются в ходе последующей количественной оценки рисков и планирования мероприятий по реагированию на риски.

Количественная оценка рисков позволяет определять:

- вероятность достижения конечной цели проекта;
- степень воздействия риска на проект и объемы непредвиденных затрат и материалов, которые могут понадобиться;
- риски, требующие скорейшего реагирования и большего внимания, а также влияние их последствий на проект;
- фактические затраты и предполагаемые сроки окончания работ на проекте.

Количественная и качественная оценки рисков могут применяться в отдельности или вместе в зависимости от времени и бюджета.



Планирование рисков — это процесс определения конкретных действий (мероприятий) по управлению рисками проекта, тщательное и подробное планирование которыми позволяет:

- *определить возможные потери от наступления рисков;*
- *выделить достаточное количество времени и ресурсов для выполнения операций по управлению рисками;*
- *повысить вероятность успешного достижения результатов проекта.*

В соответствии с [8] исходными данными для планирования управления рисками служат:

- отношение и толерантность к риску организаций и лиц, участвующих в проекте, что оказывает влияние на план управления проектом. Это должно быть зафиксировано в основных принципах и подходах к управлению рисками;
- стандарты организации, в которых должны быть изложены подходы к управлению рисками, например категории рисков, общее определение понятий и терминов, стандартные шаблоны, схемы распределения ролей и ответственности, а также определенные уровни полномочий для принятия решений;
- подробное описание содержания проекта;
- план управления проектом, формальный документ, в котором указано, как будет исполняться проект и как будет происходить мониторинг и управление проектом.

План управления рисками обычно включает следующие элементы:

- определение подходов, инструментов и источников данных, которые могут использоваться для управления рисками в данном проекте;
- распределение ролей и ответственности выполнения каждого мероприятия (позиции), включенного в план управления рисками, назначение сотрудников на эти позиции и разъяснение их ответственности;
- оценка стоимости мероприятий и выделение ресурсов, необходимых для управления рисками. Эти данные включаются в базовый план по стоимости проекта;
- структуризация, систематизация и всесторонняя идентификация рисков с нужной степенью детализации;
- определение сроков и частоты выполнения процесса управления рисками на протяжении всего жизненного цикла проекта, а также определение операций по управлению рисками, которые необходимо включить в план реализации проекта;
- определение уровней вероятности наступления рисков, шкалы воздействия и близости рисков на проект.

В общем случае любой риск характеризуется [12] (рис. 2.4):

- причинами, обусловливающими наступление риска;
- симптомами, указывающими на то, что событие риска произошло или вот-вот произойдет;
- последствиями — проблемами, которые могут появиться при реализации проекта в результате произошедшего риска;
- воздействиями на возможность достижения целей проекта: изменения стоимости, графика и технических характеристик разрабатываемого ПП.

Планирование мероприятий по снижению (ликвидации) рисков следует начинать после проведения качественного и количественного анализа рисков, при

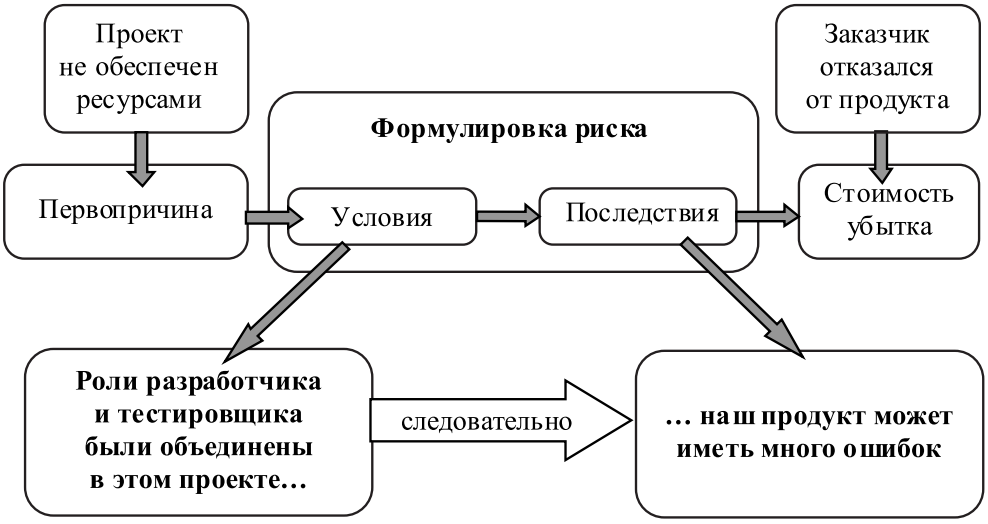


Рис. 2.4 – Основные характеристики риска и их взаимосвязи

этом все мероприятия должны соответствовать серьезности риска, быть экономически эффективными, своевременными, реалистичными и согласованными со всеми участниками проекта.

Согласно [12] возможны четыре вида таких мероприятий: уклонение от риска, передача риска, снижение рисков, принятие риска.

Уклонение от риска предполагает изменение плана управления проектом таким образом, чтобы исключить угрозу, вызванную негативным риском, оградить цели проекта от последствий риска или ослабить цели, находящиеся под угрозой (например, уменьшить содержание проекта). Некоторых рисков, возникающих на ранних стадиях проекта, можно избежать при помощи уточнения требований, например отказаться от реализации рискованного функционального требования или самостоятельно разработать необходимый программный компонент, вместо ожидания поставок продукта от субподрядчика.

Передача риска подразумевает переложение негативных последствий угрозы с ответственностью за реагирование на риск на третью сторону (например, заказ на стороне разработки рискованного компонента). Передача риска просто переносит ответственность за его управление другой стороне, но риск при этом никуда не девается. Передача риска практически всегда предполагает выплату бонусов за риск стороне, принимающей на себя риск.

Снижение рисков предполагает понижение вероятности и/или последствий негативного рискованного события до приемлемых пределов. Принятие предупредительных мер по снижению вероятности наступления риска или его последствий часто оказывается более эффективным, нежели усилия по устранению негативных последствий, предпринимаемые после наступления события риска. Например, если высока вероятность увольнения сотрудников, то введение на начальной стадии в проект дополнительных (избыточных) трудовых ресурсов снижает потери при увольнении членов команды, поскольку не будет затрат на привлечение к выполнению проекта новых участников.

Принятие риска означает, что команда проекта осознанно приняла решение не изменять план управления проектом в связи с появлением рисков или не нашла

подходящей стратегии реагирования на него. Если же компания приняла решение управлять рисками, то необходимо их страхование путем закладывания резерва в оценки срока завершения проекта и/или увеличения трудозатрат, управляемых рисков.

В [8] при разработке мер по предотвращению рисков предлагается объединить их в две категории:

1) «известные неизвестные». Это те риски, которые можно идентифицировать и подвергнуть анализу. В отношении таких рисков можно спланировать ответные действия;
2) «неизвестные неизвестные». Это риски, вызванные непредвиденными обстоятельствами, которые невозможно идентифицировать и, следовательно, спланировать ответные действия. Единственное, что можем в этом случае предпринять, это создать резерв бюджета проекта на случай незапланированных, но потенциально возможных изменений. На расходование этого резерва руководитель проекта, как правило, обязан получать одобрение вышестоящего руководства. Финансовые резервы на непредвиденные обстоятельства не входят в базовый план по стоимости проекта, но включаются в бюджет проекта. Они не распределяются по проекту, как бюджет, и поэтому не учитываются при расчете освоенного объема.

Множество рисков, относящихся к категории «известных неизвестных», в свою очередь, условно можно разбить на определенные группы.

Риски, обусловленные непредвиденными изменениями рыночной ситуации:

1) изменениями нормативного регулирования деятельности компании;
2) изменениями ситуации на рынке программно-аппаратных средств;
3) изменениями ситуации на финансовом рынке;
4) ненадежной работой аутсорсинговых компаний.

Первые три риска можно снизить путем внесения в договор условий, предусматривающих корректировку порядка его исполнения в случае кризисных и иных явлений, не зависящих от воли сторон, а в последнем случае — путем внесения в договор пункта о штрафных санкциях за нарушение условий поставки продукта аутсорсинговой компанией.

Риски, обусловленные конкурентной борьбой:

1) высокой рыночной конкуренцией и, как следствие, ошибкой в объемах продаж ПП;
2) дискредитацией программного продукта со стороны конкурентов.

Возможные ошибки в объемах продаж непосредственно связаны с непредвиденной конкуренцией. Рекомендации по минимизации рисков в этом направлении сводятся к следующему:

- поставляемый продукт не должен быть полным аналогом существующих на рынке программных систем;
- рекомендуется первым завершить разработку и захватить рынок, даже если это приведет к увеличению затрат на первый экземпляр.

Кроме того, риски сильно зависят от неоправданных увеличений затрат на разработку, рекламу, влияет на риски и компьютерное пиратство. Снижение рисков на этапе разработки может быть достигнуто за счет эффективного использования в проекте готовых компонентов, а также постоянной оценки себестоимости разработки с возможностью внесения корректировок в случае незапланированных отклонений.

Следует принять ряд мер по возможной дискредитации программного продукта со стороны конкурентов, которая может осуществляться в следующих направлениях: нарушение целостности продукта, кражи и присвоение алгоритмов и программных кодов, пиратское распространение копий, необоснованное обвинение в нарушении права интеллектуальной собственности.

Внутренние (собственные) риски проекта:

1) требования заказчика не всегда точны и подвержены частым изменениям;
2) разработка функционально неправильных программных элементов неудачного пользовательского интерфейса;
3) отсутствие эффективного взаимодействия с заказчиком;
4) недостатки планирования проекта, появление «забытых работ»;
5) недооценка сложности проекта, ошибки в оценках трудоемкостей и сроков работ и, как следствие, нереалистичные сроки и бюджет проекта;
6) отсутствие у команды необходимых ресурсов и опыта;
7) недостатки во внутренней организации работ, неумение работать в реальном времени;
8) дефицит специалистов и / или высокая текучесть кадров;
9) разрыв в квалификации специалистов разных областей знаний;
10) нехватка информации о внешних компонентах, определяющих окружение ПП;
11) недостаточная производительность получаемой системы.

К наиболее часто упускаемым требованиям к ПП, как правило, относят:

1) отсутствие функциональных требований к программам установки, настройке, конфигурации; миграция данных; интерфейсам к внешним системам;
2) отсутствие общесистемных требований к производительности; надежность; открытость; масштабируемость; безопасность; кросплатформенность; эргономичность программных продуктов.

Эти требования «всплывают» при подготовке и проведении приемо-сдаточных испытаний и могут сильно задержать завершение проекта и увеличить трудозатраты на его реализацию. Во избежание такой ситуации следует достигать соглашения с заказчиком по всем перечисленным пунктам еще на стадии инициации проекта.

Если вероятность изменений требований проекта высока, то возможны следующие подходы к реагированию на данный риск:

- переоценка проекта при каждом добавлении или изменении требований;
- интеграционная разработка проекта с периодическим уточнением требований, передача части рисков Заказчику;

- учет в оценках трудоемкости и сроков возможности роста требований, например на . . . % (резервирование риска).

Если у нас в проекте недостаточно квалифицированных специалистов, то снизить последствия этого риска можно за счет следующих действий:

- привлечь экспертов-консультантов на начальных этапах;
- учитывать в оценках трудоемкости издержки на обучение сотрудников;
- уменьшать потери от текучести кадров, привлекая на начальном этапе избыточное число участников.

Для установления открытых и доверительных отношений с заказчиком необходимо предпринимать следующие шаги:

- постоянное взаимодействие по вопросам реализации проекта;
- согласование пользовательских интерфейсов и разработка прототипа продукта;
- периодические поставки текущих версий ПП конечным пользователям для их тестирования и оценки.

К числу наиболее часто забываемых работ при планировании проекта относятся: организация обучения пользователей; уточнение требований; управление конфигурациями; разработка и поддержка скриптов автосборки; разработка автотестов; создание тестовых данных; обработка запросов на изменения.

Ошибки в оценках трудоемкости и сроков проекта связаны с принципиальной невозможностью точно определить эти величины на стадии инициации проекта (оценка трудоемкости имеет погрешность от -50% до +100%). Если не прилагать специальных усилий, этот «дамоклов меч» неопределенности будет висеть над проектом на всем его протяжении.

Не стоит надеяться, что участники проекта будут работать только над одним проектом. Есть множество причин, по которым они не смогут тратить на проект 100% своего времени. К списку наиболее распространенных причин этого относятся: сопровождение действующих систем; повышение квалификации; участие в подготовке технико-коммерческих предложений; участие в презентациях; административная работа; отпуска, праздники, больничные и т. д.

Поэтому следует в первую очередь реализовывать ключевые функциональные требования, архитектурно-значимые решения, создавая «представительный» прототип будущей системы. Прототип позволит проверить и оценить общесистемные свойства будущего продукта: доступность, быстродействие, надежность, масштабируемость и т. д. Проработка ключевых функциональных требований и детальное планирование их реализации позволяет уменьшить разброс начальных оценок, детальное проектирование и разработка прототипа — получить более точные оценки трудоемкости.

Может оказаться так, что по результатам прототипирования уточненные оценки суммарной трудоемкости окажутся неприемлемыми. В этом случае проект придется закрыть досрочно, но потери при этом, будут значительно меньше, чем в случае, если то же самое произойдет, когда проект уже в разы превысит первоначальную оценку трудоемкости. Если с заказчиком не удается найти взаимоприемлемое решение при первоначальной оценке проекта, то разумно попытаться договориться о выполнении проекта в несколько этапов с самостоятельным финансированием.

Мониторинг рисков — это процесс наблюдения и контроля за ходом исполнения принятых в отношении рисков планов и инициирование изменений в проекте, если состояние рисков в соответствующих планах влияет на объемы дополнительных работ, требуемые ресурсы или сроки проекта в целом. Другими словами, мониторинг и управление рисками — это процесс идентификации, анализа и планирования реагирования на новые риски, отслеживания ранее идентифицированных рисков, а также проверки и исполнения операций реагирования на риски и оценка эффективности этих операций.

Мониторинг и управление рисками включают в себя следующие задачи: *пересмотр рисков; аудит рисков; анализ отклонений и трендов*.

Пересмотр рисков должен проводиться регулярно, согласно расписанию. Управление рисками проекта должно быть одним из пунктов повестки дня всех совещаний команды проекта.

Аудит рисков предполагает изучение и предоставление в документальном виде результатов оценки эффективности мероприятий по реагированию на риски, относящиеся к идентифицированным рискам, изучение основных причин их возникновения, а также оценку эффективности процесса управления рисками.

На основании анализа отклонений и трендов проекта можно прогнозировать потенциальные отклонения проекта на момент его завершения по показателям стоимости и расписания. Факты отклонения от базового плана могут указывать на последствия, вызванные как угрозами, так и благоприятными возможностями по завершению проекта. Контроль и анализ трендов может повлечь за собой выбор альтернативных стратегий, принятие корректив, перепланировку проекта для достижения базового плана.

2.3 Организация командной работы над проектом

Процесс разработки ПП имеет свою организационную структуру управления, которая определяет распределение ответственности и полномочий среди участников проекта. К участникам проекта относятся все заинтересованные стороны, которые участвуют в проекте или чьи интересы могут быть затронуты при исполнении или завершении проекта. В большинстве литературных источников выделяют следующий основной состав участников проекта:

инициатор проекта — физическое или юридическое лицо (группа лиц), являющееся автором главной идеи проекта, его предварительного обоснования. В качестве инициатора может выступать любой из участников проекта, но деловая инициатива по осуществлению проекта должна исходить либо от инвестора, либо от заказчика;

инвестор — физическое или юридическое лицо (группа лиц), предоставляющие в любой форме финансовые ресурсы для проекта;

заказчик — будущий владелец и пользователь результатов проекта, физическое или юридическое лицо, заинтересованное в осуществлении проекта и достижении его результатов. Следует учитывать, что заказчик и инвестор проекта не всегда совпадают;

куратор проекта — представитель исполнителя, уполномоченный принимать решение о выделении ресурсов и внесении необходимых изменений в проекте;

руководитель проекта — проект-менеджер, физическое лицо, которому заказчик и инвестор делегируют полномочия по руководству работами по осуществлению проекта. Участники также могут влиять на проект и его результаты поставки;

соисполнители проекта — физические или юридические лица, выполняющие на договорной основе отдельные виды работ по проекту;

команда проекта — формируется в зависимости от потребностей, условий проектирования и организационной структуры выполнения проекта.

В соответствии с методологий Microsoft Solutions Framework в команде проекта [11] предлагается выделить пять функциональных групп специалистов:

Группа разработки требований состоит из следующих специалистов, каждый из которых выполняет свойственные только ему роли:

- бизнес-аналитик — разрабатывает модели предметной области (онтологии);
- архитектор — определяет общее видение продукта, его концепцию, интерфейсы, функционал и ограничения;
- системный аналитик — отвечает за перевод требований к продукту в функциональные требования к ПО;

- специалист по требованиям — документирует и сопровождает требования к продукту на всех этапах жизненного цикла ПП;
- менеджер продукта (функциональный заказчик) — представляет в проекте интересы пользователей продукта.

В *группе управления проектом* роли распределяются следующим образом:

- руководитель проекта отвечает за достижение целей проекта при заданных ограничениях (по срокам, бюджету и содержанию), осуществляет управление и контроль реализации проекта и эффективное использование выделенных ресурсов;
- системный архитектор обеспечивает разработку технической концепции системы, принятие ключевых проектных решений относительно внутреннего устройства программного обеспечения и его технических интерфейсов;
- руководитель группы тестирования определяет цели и стратегии тестирования, обеспечивает управление тестированием.

Группа проектирования и разработки ПП состоит из проектировщиков и программистов и обеспечивает:

- проектирование базы данных системы, компонентов и подсистем в соответствии с общей архитектурой, разработку архитектурно значимых модулей, интерфейса пользователя;
- проектирование, реализацию и отладку отдельных модулей системы.

Группа тестирования в проекте выполняет следующее роли: разработку тестовых сценариев и автоматизированных тестов, тестирование продукта, анализ и документирование результатов.

Участники *группы обеспечения реализации проекта*, как правило, не входят в команду проекта. Они выполняют работы в рамках своей профессиональной деятельности. К этой группе можно отнести следующие проектные роли:

- разработчик документации;
- переводчик;
- дизайнер графического интерфейса;
- разработчик учебных курсов;
- специалист по маркетингу и продажам;
- специалист по инструментальным средствам.

За эффективную реализацию программного продукта в целом отвечает руководитель проекта, основная задача которого состоит в том, чтобы:

- найти нужных людей;
- дать им ту работу, для которой они лучше всего подходят;
- не забывать о мотивации;
- помогать им сплотиться в одну команду и работать так дальше.

При этом необходимо учитывать следующие специфические особенности программиста и как личности, и как профессионального участника команды [13].

1) *Высокая самооценка программиста*. Узкая специализация и высокая профессиональная квалификация в конкретной области часто вызывают завышенную самооценку своих возможностей у сотрудников. В связи с этим у руководителя возникают две проблемы:
   - он должен сам точно представлять реальные возможности своих сотрудников, в противном случае неприятные неожиданности неизбежны;
   - сотруднику с высокой самооценкой трудно что-либо приказать, его необходимо убедить, что бывает непросто, в силу того, что сам руководитель вряд ли может быть авторитетом в той области, в которой сотрудник является узким специалистом.
2) *Невысокая трудовая дисциплина*. Программистов часто трудно заставить приходить на работу вовремя, не опаздывать на совещания, своевременно отчитываться о выполненном задании, посылать отчеты. Это связано с индивидуальным характером труда, возможностью выполнять задания вне стен компании, работать во внерабочее время. Руководителю часто приходится доводить до сознания сотрудников тот факта что они работают в команде и разработка программного обеспечения — всегда коллективная деятельность.
3) *Творческий характер программирования*. Разработка программного обеспечения — творческий процесс, разработчики являются креативными личностями и способны привносить энтузиазм, инициативу и собственные нетривиальные решения в общее дело. При наличии сильной мотивации и ясной цели они, как правило, готовы работать с огромной самоотдачей. Это значит, что управление персоналом в программных проектах следует организовывать *по целям*, а не *по заданиям*.
4) *Высокая мобильность сотрудников*. В современных условиях спрос на квалифицированных программистов существенно превышает предложение. Эта тенденция сохранится в обозримом будущем. Из этого следует, что руководитель должен быть в принципе готов к внезапному уходу из команды (и из компании) любого из сотрудников. Процесс разработки следует организовать так, чтобы подобный уход не вызвал катастрофических последствий для проекта. Здесь необходимо учесть два аспекта: возможность утраты необходимой рабочей силы и возможность безвозвратной потери программного кода. Программисты часто не понимают, что программы, разработанные в рамках проекта организации, им не принадлежат.

Каждый специалист при работе над проектом в составе единой команды исполняет как формальные функциональные обязанности (*функциональные роли*), так и определенные неформальные *командные роли*, определяющие его статус и положение в команде.

Распределение функциональных ролей в команде руководитель производит с учетом профессиональных качеств программиста и его типа личности. Одна из первых попыток классификации людей по *типу личности,* которая берет начало с трудов Гиппократа — была сделана на основе разделения по темпераментам [11]:

- *Холерик* — имеет самые скоростные темпоритмы, много и быстро говорит, без промедления отвечает собеседнику, часто перебивает, когда собеседник только начал о чем-то говорить, холерик уже все понял и имеет готовый ответ.
- *Флегматик* — спокойный, миролюбивый и сдержанный человек, никогда не перебивает собеседника, умеет внимательно выслушать и кивает в знак согласия, у него мягкие и неторопливые движения, негромкий голос.
- *Сангвиник* — деловитый, выносливый и работоспособный человек, с хорошим самоконтролем, нередко трудоголик, любит хорошо зарабатывать и делать карьеру.
- *Меланхолик* — чувствительный, обидчивый и очень ранимый человек, легко расстраивается даже при мелких неудачах, любит жаловаться на судьбу, искренне верит, что самая «тяжелая доля» и «самые тяжкие испытания» из всех возможных выпали именно ему.

Наиболее продуктивными специалистами являются *флегматики*, при этом из них получаются как грамотные и настойчивые в реализации программисты, живущие в реальном мире; они конкретны, точны и практичны, стремятся к специализации, так и успешные руководители проектов, рассматривающие широкий спектр возможностей, абстрагирующиеся от технических деталей, склонные обобщать и теоретизировать. В [11] всех программистов — участников работы над проектом с учетом их квалификации и темперамента предлагается условно разбить на девять типов:

1) *Архитектор* мыслит объектами, посвящая себя без остатка решению бизнес-задач, строит абстракции, проводит анализ систем, после чего переходит к кодированию конкретных решений. Зачастую в высшей степени разумные замыслы архитектора воплощаются им в настолько общем и непонятном коде, что людей, могущих разобраться в нем и продолжить начинание, просто не находится. Архитектор любит набросать структуру программы, с тем чтобы впоследствии передать ее дальнейшее кодирование программистам более «низкой» квалификации.
2) *Конструктивисты* получают удовольствие от процесса написания кода и его результата. При этом с написанием кода они справляются быстро, причем в большинстве случаев ошибок в нем не обнаруживается даже на этапе начального тестирования. Основное внимание программисты этого типа уделяют процессу создания кода, поэтому остальное для них не так уж важно. При модернизации ПП конструктивист начинает судорожно искать новые, «заплаточные», решения, отчего надежность кода может резко снизится.
3) *Художник* как тип программиста сконцентрирован на процессе создания кода и искусном сведении объектов пользовательского интерфейса в одну изящную структуру. Работая с компонентами без видимого интерфейса, художники обнаруживают тенденцию к правильной и логичной организации программы. Недостаток художника в том, что очень часто он затягивает кодирование, вдаваясь в излишние украшения и оптимизацию программы. С другой стороны, если программист не культивирует в себе художника, результаты его деятельности зачастую теряют «изюминку».

4) *Инженер* способен на основе готовых программных компонентов создавать множество объектов и сводить их воедино, так что они прекрасно работают в первой же версии программы. Присущая им тяга к усложнению проявляется лишь тогда, когда речь заходит о создании последующих версий.
5) *Ученый* разрабатывает ПП всегда в соответствии с фундаментальными принципами компьютерных наук. Программисты такого типа очень полезны, когда речь заходит об особо трудных задачах кодирования. У инженеров и ученых есть одна общая черта — те и другие очень любят все усложнять. Отдавая должную оценку глубочайшим познаниям ученых, руководитель проекта не должен допускать их полновластия в вопросах написания кода — иначе могут сорваться сроки выполнения проекта.
6) *Лихач* способен разрабатывать ПП в кротчайшие сроки, забывая о комментариях и соглашениях, об поименовании переменных. Тем не менее созданные им продукты вполне успешно работают. Программисты-лихачи незаменимы, если сроки реализации проекта жестко заданы, а качество его исполнения не играет большой роли.
7) *Минималист* создает программы в виде скромного по объему кода, удовлетворяющего обычно всем функциональным требованиям, объекты выстроены четко и однозначно отображают свое назначение. К сожалению, минималисты, решив поставленную задачу, быстро теряют к ней всякий интерес и при обнаружении в ходе тестирования каких-либо проблем выказывают устойчивое нежелание их исправлять.
8) *Трюкач* постоянно осваивает разные новинки, но результат от этого не улучшается. Полагая, что его функции ограничиваются забавами с разными инструментальными средствами, трюкач отказывается учитывать те аспекты программирования, благодаря которым не затрачиваются в дальнейшем титанические усилия на сопровождение программы.
9) *Любителю* не хватает образования, ему нельзя поручать работу над критически важными приложениями. Тщательно изучив какой-нибудь инструментарий, он возводит себя в ранг хакеров. Единственная причина, по которой любители бросают уютные места в отделах тестирования и поддержки пользователей, заключается в том, что, по их мнению, быть программистом — это очень круто.

Командная роль, которую человек исполняет в команде, в основном зависит от состава и состояния команды, при этом люди, выполняющие одни и те же должностные функции в проекте, могут исполнять разные командные роли [11].

1) *Генератор идей* — независимый сотрудник с развитым воображением, оригинальный мыслитель, который решает сложные задачи, дает жизнь новым идеям, но подобно остальным людям может иметь негативные черты характера — чрезмерно чувствителен к критике. Для успеха генератору идей необходимы конструктивные отношения с руководителем или координатором группы.
2) *Координатор* — обычно формальный лидер группы, руководит и направляет группу на качественное выполнение проекта. Может заранее определить, кто из работников хорош для выполнения необходимых задач. Обыч-

но спокойный, уверенный и распорядительный, однако иногда склонен к излишнему доминированию, и группа становится продолжением его сильного «Я».

3) *Аналитик (критик)* — занимает позицию наблюдателя и критически оценивает состояние проекта, не дает группе двигаться неправильным путем. Осмотрительный, бесстрастный, имеет аналитический склад ума, иногда становится чрезмерно критичным.
4) *Вдохновитель команды* — стремится объединять и вносить гармонию в отношения между членами группы, занимает позицию понимающего чужие проблемы, стремится помочь и сглаживает конфликты. По натуре человек добрый, стремится налаживать неформальные отношения, однако бывает нерешительным в сложных или кризисных ситуациях.
5) *Реализатор* — лояльный и честный сотрудник, хороший организатор, методичный и прагматичный специалист, может преобразовать стратегический план в конкретные функциональные задачи, которые доступны для решения. Однако иногда может быть негибким, непреклонным.
6) *Контролер* — отлично умеет создавать отчеты о работе группы, озабочен точным выполнением взятых обязательств и старается не упускать из виду даже мелких деталей Личным примером заставляет сотрудников придерживаться плана работ над проектом, но может становиться излишне тревожным.
7) *Специалист* — профессионал, самостоятелен, стремится стать экспертом в своей области, обладает высокой профессиональной/технической экспертизой и знаниями, гордится своей работой. Привносит вклад только в узкой сфере своей профессиональной деятельности.

Немаловажную роль при командной работе над проектом играет *мотивация* как фактор сознательного и результативного труда любого сотрудника. Для того чтобы человек качественно и в надлежащие сроки исполнял порученную ему работу, он должен испытывать некую потребность и предполагать, что, выполнив эту работу, он в той или иной степени эту потребность удовлетворит. Именно потребности заставляют людей действовать определенным образом. Поэтому управление мотивами осуществляется, как правило, в двух направлениях:

1) формирование правильных потребностей;
2) формирование правильной оценки степени их удовлетворения.

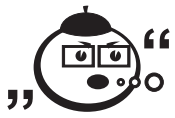
В зависимости от их возраста и опыта работы распределение мотивирующих потребностей у профессиональных разработчиков ПП бывает различным (табл. 2.8).

Для начинающих программистов хорошим стимулом является само участие в успешном проекте, возможность перенимать опыт у более опытных коллег. Для опытных программистов таким стимулом является новизна и востребованность на рынке труда технологий, используемых в проекте, сложность поставленных задач и самостоятельность (потребность самоуважения) в их решении. Для опытного программиста каждая новая задача предоставляет дополнительную возможность доказать свой профессионализм. Пропуск в той или иной графе свидетельствует, что данная потребность не является доминирующей и ее удовлетворение не принесет желаемого результата.

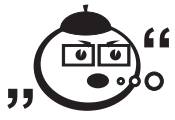
Таблица 2.8 – Зависимость мотивации участника команды от опыта и возраста

Потребности	Профессионализм		
	Начинающий	Опытный	Мастер
Материальные (зарплата, условия труда, социальный пакет)	50%	20%	
Безопасности (стабильность компании, востребованность)		20%	
Принадлежности (возможность учиться у более опытных коллег, опыт участия в успешном проекте, признание в коллективе)	40%	20%	10%
Самоуважения (развиваться, делать что-либо лучше других, повышение в должности, самостоятельность и ответственность в работе)	10%	30%	40%
Самоактуализации (амбициозность целей проекта — сделать то, что никто не делал или не смог сделать)		10%	50%

Ниже приводятся несколько цитат, характеризующих мотивации к труду программистов: [11]

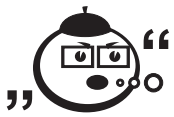


1) Э. Йордан: «Деньги, выгода, комфорт и тому подобное являются факторами «гигиены» — их отсутствие вызывает неудовлетворенность, однако они не могут заставить людей полюбить свою работу и дать им необходимые внутренние стимулы. Что действительно может дать такие стимулы, так это ощущение значительности достигнутых результатов, гордость за хорошо выполненную работу, более высокая ответственность, продвижение по службе и профессиональный рост — все то, что обогащает работу».



2) Программист состоит из четырех компонентов: тело, сердце, разум и душа.

- *Телу необходимы деньги и уверенность в завтрашнем дне.*
- *Сердцу — любовь и признание.*
- *Разуму — развитие и самосовершенствование. Профессионализм в своих глазах и во мнении коллег.*
- *Душе — самореализация [13].*



3) «Программист — это не профессия, это образ мышления». При командной работе над проектом программист должен [11]:

- *Занимать активную позицию, стремиться расширить свою ответственность и увеличивать личный вклад в общее дело.*
- *Постоянно приобретать новые профессиональные знания и опыт, выдвигать новые идеи, направленные на повышение эффективности реализации проекта, добиваться распространения своих знаний, опыта и идей среди коллег.*
- *Получать удовольствие от своей работы, гордиться ее результатами и стремиться, чтобы эти же чувства испытывали все коллеги.*
- *Четко осознавать свои личные и общие цели, понимать их взаимообусловленность, настойчиво стремиться к их достижению.*
- *Быть уверенным в себе и в своих коллегах, объективно оценивать их достижения и успехи, внимательно относиться к их интересам и мнениям, активно искать компромиссные решения в конфликтах.*
- *Всегда оставаться оптимистом, при этом твердо знать, что окружающий мир несовершенен; воспринимать каждую новую проблему, как дополнительную возможность продемонстрировать собственные знания и умения.*

2.4 Практические рекомендации по управлению жизненным циклом разработки программного проекта

Описанные ниже рекомендации предложены и использованы Дж. Маккартни при разработке приложений в среде MS + 4.0 для продукта Microsoft Solution Framework, созданного фирмой Microsoft. Рекомендации объединены в три раздела правил: «Выпустить», «Лучший проект», «Выпустить точно в срок» [15].

Раздел 1. «ВЫПУСТИТЬ»

1) *Вы не знаете того, чего вы не знаете*. В любом проекте всегда существуют неопределенности, но человеку свойственно отвергать свое незнание, подменяя истинное знание предположениями. Неизвестно, что является ошибкой, поэтому старайтесь быть критичными к себе. Найдите для своей команды мотивацию к тому, чтобы она постаралась обнаружить и понять максимум «белых пятен». Не пытайтесь обмануть себя предположениями.

2) *Старайтесь иметь четкое представление о состоянии проекта.* В проекте должны участвовать люди, способные объективно оценить готовность системы. Разработчики всегда более оптимистичны, чем тестировщики. Поощряйте плохие новости, и тогда снизится риск неожиданного провала в самом конце проекта. Самый лучший способ — полагаться на объективные данные, метрики (степень покрытия функциональности тестами, число активных дефектов, динамика их исправления и т. д.). Состояние осведомленности обязательно должно включать знание о всех составляющих проекта и содержать точную информацию о статусе проекта в определенный период времени.

3) *Помните о треугольнике приоритетов.* Существуют три взаимосвязанных компонента любого проекта: ресурсы (люди и деньги), функциональность и сроки. Изменение одного из них всегда влияет на остальные. Можно зафиксировать значения только двух из этих показателей: например, зафиксировать характеристики времени и команды, чтобы получить требуемую функциональность; зафиксировать характеристики времени и команды, чтобы получить нужное время; зафиксировать характеристики времени и команды, чтобы получить требуемую численность команды разработки. Не забывайте, что не все задачи можно выполнять параллельно.

4) *Старайтесь быть на виду.* При планировании надо разбивать задачи на более мелкие, на выполнение которых требуется полдня или день. Тогда о том, что вы начинаете отставать, вы узнаете достаточно рано и сможете предпринять корректирующие шаги. Неделя для проекта разработки — это вечность. Каждый проект, который отстает на месяц, вначале отставал на один день. Узнайте об отставании прежде, чем наступит момент, когда исправить что-то будет уже невозможно. Несомненно, подобный стиль управления весьма опасен и периодически вас будут в нем обвинять. Но если вам удается довести до участников проекта понимание того, что его цель — предоставить готовый продукт в определенное время, ваши коллеги примут этот стиль.

5) *Используйте контрольные точки с отсутствием дефектов.* В каждом приложении можно выделить 20% функциональности, которая будет использоваться 80% времени; в свою очередь, в оставшихся 80% функциональности можно опять выделить 20% и т. д. Реализуйте сначала первые 20% функций, но полностью, а также протестируйте и соберите программу установки с прототипом документации. И только когда все это будет готово, двигайтесь дальше. Этот метод основан на идее выделения контрольных точек, в которых продукт должен содержать некое, небольшое, но четко определенное количество дефектов. Их промежуточный контроль позволяет быстро понять, какие части проекта могут стать причиной проблем, и получить полную картину о проекте, а также дают возможность сфокусироваться на достижении целей каждой контрольной точки.

6) *Бойтесь разработчиков, сидящих в башнях из слоновой кости.* Разработчики должны уметь работать в команде, демонстрировать свой прогресс, делиться опытом и проверять то, что уже сделано. Избегайте «примадонн»,

которые считают себя умнее всех. Несомненно, проект лишь выиграет, если в команде появится пара гениев или просто отличных специалистов, но еще лучше будет, если их талант признает и команда, и все лица, заинтересованные в результатах проекта. Разработка программ осуществляется силами команды, для определения состава которой может пригодиться правило, принятое в Microsoft: шесть разработчиков, три инженера по качеству, один менеджер, два технических писателя. Это правило — результат многочисленных проб и ошибок, через которые прошла корпорация при разработке и масштабных платформ, и совсем небольших утилит.

7) *Плохая дата — не просто плохая дата.* Обычно вы заранее знаете, что опоздаете, знают это и все вокруг. Вы настаиваете на смене даты, но с каждой отсрочкой теряете доверие клиента. Вот хорошее правило: перенос даты должен происходить только в тех случаях, когда точно известны все составляющие и причины задержки. Изменение сроков требует ресурсов, поэтому следующая контрольная точка должна быть реалистичной. В противном случае подобный «проект» никогда не закончится.

8) *Сдвинув сроки, не проваливайте их.* Перенос срока — это симптом, который свидетельствует о выявлении «белых пятен». Такое случается весьма часто, поскольку разработка ИТ-решений является экспериментальным видом деятельности, в котором используются новые технологии. Все это ведет к неопределенности по поводу сроков проекта. Если сдвиг сроков выполнения проекта стал неожиданностью — значит, используемые методы общения с сотрудниками и управления командой проекта надо менять. В то же время задержка дает возможность переоценить ресурсы и возможности продукта; в итоге это приносит положительный эффект. Поэтому, сдвигая сроки, внимательно анализируйте ошибки и старайтесь их больше не допускать.

9) *Чем проще — тем лучше.* Лучше обещать меньше, но сделать больше, чем пообещать больше и не выполнить. Для заказчика важнее результат, а не обещания. А для успеха проекта стоит выбирать максимально простые, но надежные решения.

10) *Время для проектирования — это время для проектирования.* Оценивая способы проектирования, всегда учитывайте временной фактор и выбирайте из альтернативных решений наименее рискованное и оптимальное по времени реализации. Часто в погоне за красотой реализации, проектировщик склонен выбрать вариант, совершенно не укладывающийся в сроки проекта. Помните, что время — весьма ограниченный ресурс, который дан для того, чтобы достичь успеха, а не удивить.

11) *Если вы не можете что-то собрать, значит, вы не сможете это выпустить.* Продукт должен постоянно собираться (компилироваться вместе со всеми компонентами системы, документацией и программой установки). Это необходимо потому, что с самого начала нужно определить все составляющие будущего продукта и адекватно оценить степень их готовности. В противном случае вы обязательно что-нибудь забудете, и это станет весьма неприятной неожиданностью, особенно перед окончанием работ. Пусть

на начальном этапе это будут скромные зачатки будущей системы — придет время, и из них вырастет отличный продукт. С другой стороны, промежуточная версия продукта — хороший способ продемонстрировать заказчику факт готовности того или иного фрагмента работы.

12) *Мысли о многоплатформенности.* Старайтесь «не перебарщивать» с числом платформ, на которых будет работать система. Держите в голове тот факт, что разработка для каждой из платформ зачастую представляет собой переработку большей части функциональности, а также ее тщательное тестирование и последующее исправление ошибок. Соответственно, это увеличивает как стоимость системы, так и затраты на будущую поддержку. Помните, что пользователям чаще всего нужен продукт, а не его удивительная гибкость.

Раздел 2. « ЛУЧШИЙ ПРОДУКТ»

1) *Заказчик — это ваше все.* Старайтесь в следующих версиях учитывать даже весьма странные, нечетко выраженные пожелания клиентов. Часто в этих требованиях скрывается то, что делает продукт уникальным и неотличимым от других, добавляет ему маркетинговой привлекательности и заставляет пользователей использовать его долгие годы. Помните, что заказчик лучше вас знает, что ему нужно.
2) *Самое главное — единство и интеграция.* Единство причины и единство исполнения должны стать девизами команды разработчиков.
3) *Двигайтесь правильным курсом.* Цель — основная идея вашей разработки. Все оценки продукта основываются именно на ней, поэтому она должна быть очень четкой. Старайтесь, как можно раньше наметить цель и сохранить веру в нее вплоть до конца проекта.
4) *Будьте гибким.* Часто по ходу проекта требования к системе могут изменяться — будьте готовы к этому. Старайтесь постоянно проверять, насколько мнение пользователя соответствует поставленной цели. Используйте для этого промежуточные версии продукта, вовлекая заказчика в процесс работы с системой как можно раньше. Однако, собираясь менять курс, помните: цель должна остаться прежней.
5) *Соблюдайте баланс.* Правильно расставьте акценты на разных составляющих проекта. Ни в коем случае не увлекайтесь наращиванием свойств какой-либо из возможностей продукта, это станет причиной дискомфорта пользователей и может привести к серьезным проблемам со сроками реализации продукта.
6) *Развивайте продукт постепенно.* Правильное развитие выглядит так: ранние стадии разработки определяют более поздние, ошибки не повторяются, а результат отвечает потребностям конечного пользователя. Плавное развитие вселяет в вас ощущение предсказуемости и стабильности процесса разработки.
7) *Продукт — это иерархия компонентов.* Следуя этому принципу, элементам проекта уделяют внимание пропорционально их важности, что обеспечи-

вает стабильность и сбалансированное развитие. Иерархию очень удобно использовать как скелет для постепенного расширения системы, сначала вы реализуете основу, а затем наполняете ее будущим содержимым; новые компоненты опираются на уже разработанные.

8) *Все должны разделять общее видение продукта.* Все члены команды должны знать, какие цели должны быть достигнуты, как продукт должен выглядеть, какова стратегия его разработки. Если в команде появляются противники текущей цели, постарайтесь дать им слово и аргументировать свое видение, быть может, оно лишь улучшит будущую систему. Все противоречия должны быть разрешены, а видение предмета приведено к единству.

Раздел 3. «ВЫПУСТИТЬ ТОЧНО В СРОК»

Ваша главная задача — выпустить продукт. Помните:

- команда обязана поставить продукт в срок, а все члены команды должны верить в то, что это возможно;
- каждый должен понимать, что от него для этого требуется, а менеджер должен сделать все от него зависящее, чтобы иметь все шансы сделать то, что требуется;
- любой должен не просто хотеть, а гореть желанием достичь цели.

Выпуск продукта должен стать целью каждого члена команды, самым ожидаемым событием для всех!

Контрольные вопросы по главе 2

1) Приведите возможные определения проекта, его цели, результаты, ограничения.
2) Раскройте смысл «железного треугольника» при управлении программными проектами.
3) Перечислите и прокомментируйте содержание процессов и этапов управления проектами стандарта PMBOK.
4) Приведите основные этапы управления рисками программных проектов.
5) Перечислите и прокомментируйте риски, обусловленные непредвиденными изменениями рыночной ситуации.
6) Перечислите и прокомментируйте риски, обусловленные конкуренцией на рынке.
7) Перечислите и прокомментируйте внутренние риски программного проекта.
8) Перечислите и опишите роли участников проекта.
9) Перечислите и прокомментируйте существующие подходы к выделению функциональных ролевых групп программного проекта.

10) Перечислите и прокомментируйте содержание практических рекомендаций по управлению циклом программного проекта. Раздел: «Выпустить».
11) Перечислите и прокомментируйте содержание практических рекомендаций по управлению циклом программного проекта. Раздел: «Лучший проект».
12) Перечислите и прокомментируйте содержание практических рекомендаций по управлению циклом программного проекта. Раздел: «Выпустить точно в срок».
13) Перечислите и прокомментируйте командные роли участников проекта.
14) Перечислите и прокомментируйте функциональные роли участников проекта.
15) Перечислите и прокомментируйте особенности программиста как участника команды проекта.

Глава 3

ПРОДВИЖЕНИЕ ПРОГРАММНЫХ ПРОДУКТОВ НА ПРОМЫШЛЕННОМ РЫНКЕ

3.1 Основные понятия и особенности промышленного рынка

В настоящее время в литературе имеются самые различные определения и понятия рынка, например:

рынок — это институт или механизм, сводящий вместе покупателей (представителей спроса) и продавцов (поставщиков) товаров и услуг;

рынок — это пакет соглашений, при помощи которых продавцы и покупатели товаров и услуг вступают в контакт по поводу купли-продажи данных товаров или услуг;

рынок — это сложнейшая система взаимоотношений производителей и потребителей, продавцов и покупателей, их хозяйственных связей, включая прямые многозвенные контакты с участием посредников.

Анализ представленных определений показывает, что для существования рынка необходимо присутствие производителей, посредников, потребителей, наличие

продуктов и организационно-экономических механизмов перемещения продуктов между участниками рынка. На рынке понятие продукта отождествляется с определенным товаром либо услугой.

При этом под товаром понимается, как правило, любой продукт производственно-экономической деятельности в материально-вещественной форме, являющийся объектом купли-продажи и соответственно возникающих между продавцами и покупателями рыночных отношений.

Понятие услуги трактуется как итоги непосредственного взаимодействия поставщика и потребителя и внутренней деятельности поставщика по удовлетворению потребности потребителя.

Услуга может быть связана в том числе и с производством и поставкой материальной продукции.

Компьютерные программы как товар на рынке могут быть представлены в следующих видах:

программный модуль — отдельно компилируемая часть программного кода (программы);

программный компонент — программы, рассматриваемые как единое целое, выполняющие законченную функцию и применяемые самостоятельно или в составе комплекса;

программный комплекс (программная система) — программы, состоящие из двух или более компонентов, выполняющие взаимосвязанные функции и применяемые самостоятельно или в составе другого комплекса;

программный продукт (изделие) — совокупность отдельных программных средств, их документации, гарантий качества, рекламных материалов, мер по обучению пользователей, распространению и сопровождению готового программного обеспечения;

программный продукт — программное средство, предназначенное для поставки, передачи, продажи пользователю, это самостоятельное, отчуждаемое произведение, представляющее собой публикацию текста программы или программ на языке программирования или в виде исполняемого кода;

программное изделие (программный продукт, программное средство) — программа или логически связанная совокупность программ, записанная на носителях данных, являющаяся продуктом промышленного производства, снабженная программной документацией и предназначенная для широкого распространения посредством продажи;

коробочный программный продукт — программное обеспечение, предназначенное для неопределенного круга покупателей и поставляемое на условиях «как есть» со стандартными для всех покупателей функциями, в отличие от заказного программного продукта, само появление которого обусловлено требованием конкретного заказчика, и в отличие от проектного программного продукта, продажа которого может по требованию заказчика сопровождаться проектной доработкой или разработкой функций, дополняющих стандартные (базовые) возможности.

Анализ приведенных определений показывает, что наиболее полно понятие компьютерных программ как товар раскрывает определение программного продукта (программного изделия).

Под услугой на рынке ПП будем понимать процесс выполнения связанных с программным продуктом работ, заданий или обязанностей (разработка, сопровождение или эксплуатация) как на «свободный» рынок (рыночный ПП), так и под конкретный заказ (заказной ПП).

В отличие от обычных товаров, имеющих материально-вещественную форму, программные продукты являются предметом интеллектуального труда и охраняются авторским правом. С точки зрения правовой охраны и защиты интеллектуальной собственности — это самостоятельное отчуждаемое произведение, представляющее собой публикацию текста программы или программ на языке программирования или в виде исполняемого кода.

В рыночной экономике ПП, являясь объектом авторских прав, выступают в виде принципиально нового продукта, вовлечение которого в хозяйственный оборот происходит в процессе коммерциализации (купли-продажи, переуступки прав собственности) и капитализации (постановки на баланс, инвестирования в уставный капитал).

Таким образом, программный продукт представляет собой интеллектуальный цифровой товар, для которого характерны следующие особенности:

- нематериальная природа существования, его нельзя увидеть в процессе конструирования и, следовательно, оперативно повлиять на его реализацию;
- может быть неоднократно продан, являясь при этом одновременно объектом нескольких рыночных сделок;
- не исчезает и не изнашивается в процессе использования, состоит из материального носителя и нематериальной части;
- как результат творческого труда, производится в условиях повышенного риска, не поддается точному оцениванию как по времени создания, так и бюджету.
- характеризуется ничтожными затратами на тиражирование по сравнению с затратами на разработку продукта.

Низкие затраты на тиражирование обусловлены ничтожно малой стоимостью производственных операций по изготовлению копий программных продуктов по сравнению со стоимостью самого продукта. Большую часть стоимости составляют затраты по созданию данного программного продукта как объекта интеллектуальной собственности относительно небольших групп специалистов, а не гигантских коллектив программистов.

Программный продукт вступает в хозяйственный оборот как товар только в случае фиксирования его на материальном носителе (компьютере, дисковом накопителе и т. п.), в котором он овеществляется и может быть сохранен, преобразован или передан. При этом обладание материальным носителем информации не делает его приобретателя уникальным собственником информации.

Для существования и функционирования рынка программных продуктов необходимо соблюдение следующих условий:

1) наличие реальной потребности у конкретных заказчиков (наличие спроса на ПП);
2) наличие производителей конечных продуктов, обладающих правами на продукт (множество разработчиков/правообладателей);
3) наличие конечных продуктов у производителей, ориентированных на удовлетворение потребностей потребителя (множество предложений ПП);
4) развитая сеть посредников между производителями и потребителями;
5) наличие экономических и организационно-правовых механизмов, регламентирующих цивилизованное взаимодействие участников.

Основными особенностями рынка ПП в настоящее время являются:

- ограниченное количество разработчиков и неограниченное количество покупателей (пользователей);

- продажи ПП происходят либо напрямую (разработчик — пользователь), либо через фирму-посредника;
- ценовая политика продаж определяется монопольным либо олигопольным характером рынка и его конъюнктурой;
- успехи продаж во многом зависят от личных контактов разработчиков и будущих пользователей;
- присутствие на рынке значительного количества продуктов среднего качества, обладающих множеством недокументированных свойств, неизвестных пользователю (а возможно, и разработчику);
- неприхотливость массового пользователя, который имеет неверные ожидания относительно функций и качества приобретаемых программ;
- низкая конкурентоспособность, востребованность и доступность конкретных ПП на фоне широкого ассортимента продукции;
- субъективный выбор потребителем конкретного ПП, осложненный отсутствием аналогов либо наличием у потребителя мнения, сформированного под воздействием PR-кампаний и сильных брендов;
- неблагоприятный налоговый климат;
- сложность получения инвестиций в сфере малого и среднего бизнеса;
- большая часть ПП распространяется с нарушением лицензионных прав;
- пассивная роль государственной политики в развитии рынка;
- сложность доступа на международные рынки;
- отсутствие эффективной системы коммерциализации ПП.

С учетом особенностей ПП как результата интеллектуальной деятельности и требований к условиям рынка выделяются следующие участники рынка программных продуктов (рис. 3.1):

1) государство;
2) потребители ПП;
3) разработчики (правообладатели) ПП;
4) посредники;
5) партнеры и конкуренты.

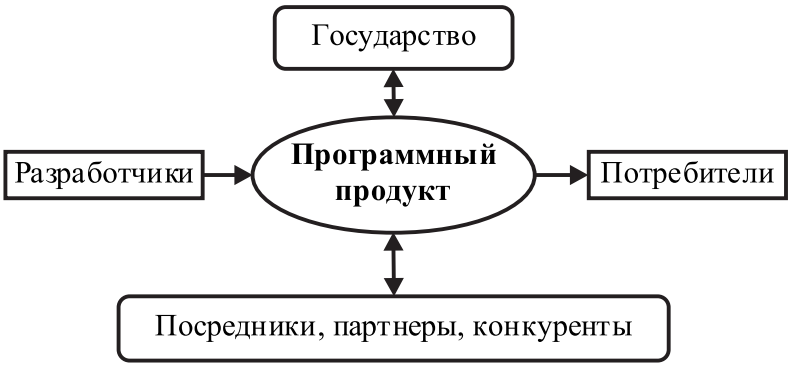


Рис. 3.1 – Участники рынка программных продуктов

Участники официально действуют на рынке, вступают в отношения друг с другом и осуществляют свою деятельность в рамках, предусмотренных законами РФ. При этом для реализации рыночных отношений между субъектами рынка (заказчиком, посредником, пользователем), возникающих по поводу использования либо распоряжения программными продуктами (ПП), посредник должен иметь права на этот продукт.

Разработчики (правообладатели). Представители данной группы участников образуют рынок продуктов, занимаясь проектированием, кодированием, документированием, распространением, сопровождением и модификацией ПП.

При этом конкурентоспособность разработчиков и их положение на рынке определяют пять факторов:

1) цена на программную продукцию либо услугу;
2) качество продукции с точки зрения удовлетворения требуемых потребностей;
3) отличительные особенности продукции, побуждающие покупателя приобретать именно данный программный продукт;
4) гибкость производителя, связанная со способностью реагировать на просьбы покупателя по адаптации либо доработке программных продуктов;
5) время (сроки) реакции производителя на потребности покупателя (например, время адаптации и внедрения ПП, продолжительность обучения пользователей, период гарантийного сопровождения, временные условия по модернизации и поставке новых версий и т. д.).

С учетом вышесказанного наметившиеся тенденции в бизнес-практике разработчиков касаются, прежде всего, стратегии доставки продукта пользователю, ценообразования и системы продаж. Фирмы, занятые разработкой и/или продажей продуктов, обычно работают в одном из трех направлений: разработка приложений, продажа приложений и их обслуживание. Следует отметить, что прибыльность этих направлений неодинакова, поскольку за счет сокращения расходов, прежде всего на маркетинг и распространение, производители получают дополнительную прибыль, а функции маркетинга и распространения стремятся передать другим фирмам.

Если же фирма берет на себя одновременно функции и разработчика и распространителя, то сталкивается с рядом проблем. Производить качественные и эффективные рекламные материалы с привлечением профессиональных рекламных кампаний могут позволить себе только крупные организации, продукция которых и так известна на рынке программных средств. «Самодельная реклама», как правило, описывает функциональные возможности ПП, излагается на «языке» разработчика и мало ориентирована на потребителя. Отсутствие требуемых финансовых средств на проведение рекламной кампании либо их экономия пагубно влияют на продвижение ПП на рынок. Аналогичная ситуация складывается и при участии производителей в выставочно-ярмарочной деятельности. С одной стороны, выставки посещают преимущественно разработчики, а с другой — цены на участие в таких мероприятиях часто бывают неприемлемыми для мелких и средних производителей.

Кроме того, существуют еще ряд причин, препятствующих развитию рынка ПП со стороны разработчиков:

- ориентация производителей на мелкосерийное производство ПП, разрабатываемых, как правило, под конкретный заказ;
- высокая доля фиксированных затрат в структуре издержек и, как следствие, высокие цены на создаваемые ПП (как правило, это цена разработки);
- использование при разработке пиратских инструментальных программных средств, не позволяющее производителю открыто рекламировать свои продукты, участвовать в выставках и пр.;
- отсутствие начального капитала на развитие фирмы, наработку требуемых заделов, приобретение лицензионного программного обеспечения;
- слабое использование индустриальных методов группового проектирования ПП (как правило, разработчик сам находит заказ, разрабатывает, тестирует и документирует программы);
- слабое представление о существующем рынке конкурирующих ПП;
- отсутствие эффективных программных средств защиты от копирования, а также экономических и юридических механизмов, препятствующих этим процессам;
- отсутствие опыта по представлению ПП в виде законченного продукта и организации маркетинга по его тиражированию, неэффективная рекламная кампания, отсутствие профессиональных менеджеров по продвижению продуктов на рынок;
- незнание или несоблюдение отечественных и международных стандартов на управление жизненным циклом, качеством и документированием ПП.

Государство осуществляет регулирование отношений, возникающих в гражданском обороте по поводу использования ПП, посредством экономических, организационных, нормативно-правовых механизмов, обеспечивая цивилизованное взаимодействие участников рынка ПП. Интересы государства, в первую очередь, заключаются в максимальном использовании интеллектуальной собственности в интересах развития отраслей экономики и получении выгод от надлежащей охраны прав интеллектуальной собственности (развитие сегмента рыночного ПП, исключение бесконтрольного распространения ПП, разработка и продажа высококачественных и конкурентоспособных продуктов, укрепление позиций России на мировом рынке). Однако регулирование рынка прикладных программных продуктов со стороны государства в настоящее время практически отсутствует. Имеющиеся законы об охране авторских прав, защите интеллектуальной собственности, информации, информатизации и защите информации не работают, так как нет эффективных механизмов их конкретного применения. В связи с этим процветает компьютерное пиратство, рынок заполнен нелегальными копиями программных продуктов.

Существующая система нормативных документов (ГОСТов), регламентирующих жизненный цикл проектирования и документирования программных средств, морально устарела и носит рекомендательный характер. Сертификация как институт, обязывающий создавать программные продукты с определенными параметрами качества, существует преимущественно в добровольной форме, не но-

сит масштабного характера и, как следствие этого, на рынке зачастую появляется некачественная, плохо документированная программная продукция. Соответствие нормативным документам и сертификатам должно оговариваться в договорах на разработку, адаптацию либо поставку программных систем, о чем пользователь (потребитель) зачастую не информирован. Заказчики и потребители в большинстве своем не знают о существовании таких документов, а государство никак не регулирует эти процессы.

В качестве *потребителей* могут выступать государственные (муниципальные) структуры, юридические (на рынке корпоративных продаж) и физические лица (на потребительском рынке индивидуальных продаж). Экономические интересы потребителей отождествляются с приобретением рыночных преимуществ и доходов от использования ПП либо с удовлетворением в той или иной мере личных потребностей.

Крупные потребители ориентированы в основном на иностранные программные продукты либо на удовлетворение своих информационных потребностей через создание собственных структур, занимающихся разработками программных систем. Основной проблемой при выборе продукта мелкими и средними потребителями является так называемая «некомпетентность» потребителя, который не в состоянии самостоятельно определить нужный ему продукт, так как требуется профессиональное ориентирование в широкой номенклатуре имеющихся аналогов. Кроме того, не имея по экономическим соображениям возможности содержать штат высококвалифицированных программистов, такие потребители при попытке приобретения программных продуктов сталкиваются со следующими проблемами:

- спрос на программное обеспечение, как правило, не сформирован, нет четкого представления о технологии использования программных продуктов в практической деятельности;
- потребители слабо представляют себе рынок предлагаемого программного обеспечения, не способны четко сформулировать требования к приобретаемым программным продуктам, при их выборе по критериям «цена» либо «качество» предпочтение отдается первому;
- незнание, а чаще всего игнорирование экономических и нормативно-правовых механизмов цивилизованной работы на рынке;
- скрытое противостояние программистов потребителям рыночных программных продуктов, связанное с потерей собственного имиджа;
- тяжелое финансовое положение, большое несоответствие между высокими ценами на программное обеспечение и сиюминутными «выгодами» от его использования и, как следствие, массовое использование нелегальных копий;
- ментальность отечественного потребителя, не расценивающего факт использования нелегальных копий как хищение собственности производителя.

Посредниками, которые выступают связующим звеном между разработчиком и потребителем, являются фирмы, берущие на себя функции маркетинга и распространения ПП. Их роль заключается в принятии ПП от разработчика, анализе готовности продукта к продвижению, выполнении мер по продвижению и удовлетво-

рению потребностей потребителей. В рамках этой деятельности разделение между разработчиками и посредниками является наиболее эффективным и дешевым средством быстрого получения пользователем необходимого продукта, поскольку исследование возможностей рынка, а также доставка, реклама и другие вспомогательные действия ложатся на плечи последнего. Однако следует отметить, что ведущие производители в настоящее время являются одновременно и разработчиками, и распространителями, выстраивая свою деятельность через сеть своих филиалов. Сложность (проблемность) самостоятельной реализации функций по продвижению ПП для малого бизнеса заключается в отсутствии для этого необходимых материальных средств, низкой компетентности специалистов в вопросах продвижения и сопровождения процессов продаж. Поэтому небольшим фирмам выгоднее продвигать свои продукты через развитую сеть посредников, не прилагая усилий для создания собственной сети распространения.

Партнерами являются фирмы, производящие аналогичную продукцию и ориентированные на тот же сегмент рынка. Главной движущей силой консолидации усилий партнеров является интеграция в сфере приобретения и совместного использования средств производства ПП, поскольку только крупная фирма в состоянии приобрести базовый ПП, сложную для понимания и использования базу данных и на их основе разработать принципиально новое приложение, а также обеспечить легитимность торговой сделки и расширить занимаемый сектор на рынке.

Кроме того, консолидация фирм-производителей программного обеспечения может проявляться в направлениях освоения каналов распространения и активизации маркетинговой деятельности. На первый взгляд это противоречит сложившимся тенденциям в других сферах создания программных приложений. Но в рамках подобной деятельности наиболее эффективной структурой считается та, в которой создатель продукта отделен от процесса распространения. В этом случае в качестве партнеров выступают фирмы-посредники, являющиеся связующим звеном между продавцом и покупателем.

В зависимости от универсальности товара либо услуги, видов их использования, типа потребителя различают промышленные и потребительские рынки, горизонтальные и вертикальные рынки (рис. 3.2) [16].

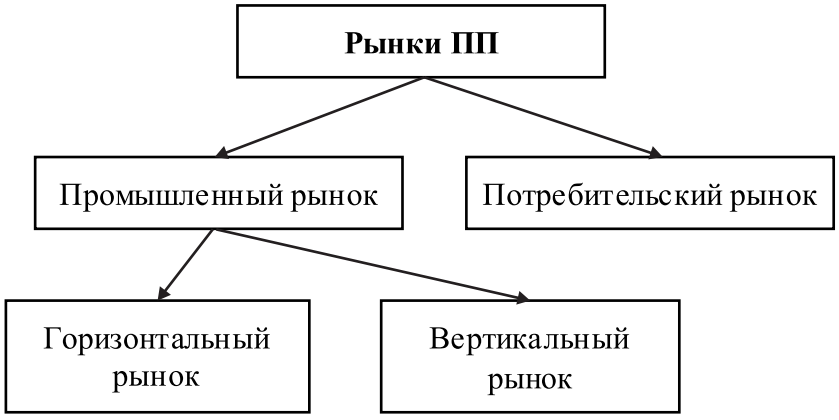


Рис. 3.2 – Виды рынков

Промышленный рынок (рынок корпоративных продаж) характеризуется наличием множества товаров промышленного назначения, которые могут использоваться как самостоятельно, так и для производства других товаров или услуг, продаваемых, сдаваемых в аренду или поставляемых другим потребителям.

Потребительский рынок характеризуется наличием множества товаров и услуг для личного потребления, покупаемых или приобретающихся иным способом отдельными физическими лицами.

На промышленном рынке число покупателей значительно меньше, чем на потребительском. Здесь действуют компании и организации, которые не являются только продавцами или только покупателями, и такие условия сделки, как цена, качество, условия поставки и оплаты, являются объектом переговоров и подписания контракта с каждым конкретным заказчиком в отдельности. Производители и покупатели обладают высокими профессиональными навыками в области ПП, которые продают или покупают, а также навыками коммерческой работы.

На потребительском рынке,в отличие от промышленного, покупателей больше и все они рассматриваются как единая генеральная совокупность. Чаще всего покупатель не осведомлен об истинных характеристиках товара, больше доверяет рекламе и продавцу-консультанту. Для анализа поведения пользователей производится представительная выборка, изучается мнение пользователей и выявляется среднестатистический потребитель.

Рынок корпоративных продаж принято делить на горизонтальные и вертикальные (отраслевые) сегменты.

Горизонтальный рынок представляет собой совокупность различных изделий и/или услуг общего назначения и состоит из широкого спектра отраслей.

Вертикальный/отраслевой рынок представлен продукцией конкретного сегмента рынка, охватывающего организации и предприятия определенного профиля деятельности на всей территории страны или региона. Структура вертикального рынка при анализе обычно выбирается с учетом принятых в статистической отчетности групп отраслей экономики согласно Общероссийскому классификатору видов экономической деятельности (ОКВЭД).

На любом предприятии, независимо от особенностей его деятельности и отраслевой принадлежности, существуют процессы, которые можно назвать общими, типичными для компаний разных отраслей. ПП, реализующие эти процессы, образуют горизонтальные рынки. Помимо таких типовых процессов существует целый ряд процедур, специфичных для конкретной отрасли. Для их автоматизации требуется реализация бизнес-процессов, характерных для данной отрасли или определенного типа предприятий. Именно это отличает вертикальное решение от горизонтального.

На горизонтальном рынке пользователи могут быть охарактеризованы как субъекты, имеющие потребность в решении проблем общего (массового) характера,

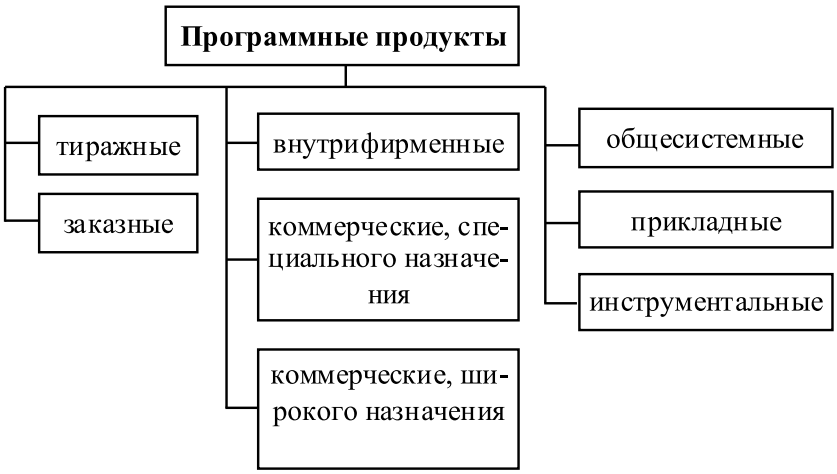


Рис. 3.3 – Классификация программных продуктов

присущих большому количеству таких же субъектов. ПП для горизонтальных рынков создаются с высокой степенью универсальности для охвата самого широкого спектра потребителей. Для данной категории рынка не предполагается разработка ПП под индивидуальные нужды (под заказ), а производится выбор наиболее оптимального варианта среди конкурирующих между собой продуктов.

ПП *для вертикальных рынков* являются достаточно специализированными, чтобы максимально точно соответствовать требованиям компаний выбранной отрасли или подотрасли, и разрабатываются, как правило, под заказ.

Прежде чем сделать выбор компании в пользу занятия вертикального рынка, необходимо определить, способна ли она создавать программное обеспечение для других разработчиков ПП (т. е. развивать индустрию средств производства) либо она будет ориентироваться на создание прикладного программного обеспечения (ППО). В первом случае разработчикам хорошо известна специфика продукции и требования заказчиков. Во втором случае эффективность деятельности на этом рынке зависит от того, насколько быстро и грамотно специалисты компании овладеют спецификой выбранной предметной области.

3.2 Классификация программных продуктов

Любая компания-разработчик при выводе своего ПП на рынок должна представлять, в каком сегменте рынка она предполагает работать, кто является ее основными конкурентами. В этом случае необходимо условно разбить весь рынок ПП на несколько сегментов. В настоящее время в литературе нет единого подхода к классификации рынка ПП. Так, в зависимости от того, для кого разрабатывается ПП – для конкретного заказчика или всего ИТ-рынка, выделяют (рис. 3.3):

тиражные (коробочные) программные продукты – коммерческие программные продукты специального и широкого применения.

заказные (внутрифирменные) программные продукты — разрабатываются под информационную поддержку управления конкретного бизнес-процесса либо адаптируются под требования этих бизнес-процессов.

Процессы разработки тиражного и заказного программного продуктов соответствуют стандартному и индивидуальному подходам к созданию ПП. Проблема выбора заказчиком между стандартным и индивидуальным ПП при информатизации своих бизнес-процессов стояла и будет стоять еще долгое время. Безусловно, любой способ создания (приобретения) прикладного ПП таит в себе свои риски. Всегда есть риск купить типовой пакет прикладных программ, долго пытаться его освоить, но в конечном итоге не получить ожидаемого эффекта. С другой стороны, имеется риск разработать или заказать индивидуальный прикладной программный продукт, работающий с ошибками, непригодный для сопровождения и не соответствующий техническим и отраслевым стандартам. Кроме того, возможен риск затянуть проект или попасть в слишком опасную зависимость от разработчиков (как внешних, так и внутренних).

Внутрифирменные ПП разрабатываются, как правило, по специальным заказам собственными или сторонними программистами. Именно к данной группе программных продуктов относится часто используемый в последнее время термин «заказное ПО».

Коммерческие продукты специального применения предназначены для использования ограниченным кругом пользователей в определенных предметных областях (издательские системы, научные пакеты и пр.).

Принципиальным отличием между программными продуктами двух последних групп является способ их распространения: ПП широкого применения изначально ориентировано на использование разветвленной сетью потребителей, и в этом плане его можно охарактеризовать как «коробочное» ПО. Специальные ПП распространяются, прежде всего, самими разработчиками, и только самые лучшие их образцы — через фирмы-посредники.

Еще одним вариантом классификации множества программных продуктов является разделение на три самостоятельных класса рынка ПП:

1) прикладное программное обеспечение — программный продукт для индивидуальных пользователей, включая программы для развлечений, образования и обработки данных, автоматизации различных бизнес-процессов в экономике, коммерции, бизнесе, индустрии и т. д.;

. . . . . .

2) общесистемное программное обеспечение — это комплекс программ, которые обеспечивают эффективное управление компонентами вычислительной системы. В отличие от прикладного программного обеспечения, системное не решает конкретные прикладные задачи, а лишь обеспечивает работу других программ, управляет аппаратными ресурсами вычислительной системы и т. д. К общесистемному ПО обычно относят: операционные системы; сервисные программные средства, включая программные средства защиты;

. . . . . .

. . . . . .

3) инструментальное программное обеспечение (средства разработки и развертывания) — предназначено для профессионального использования специалистами по проектированию, разработке различных программных продуктов. Это — языки и среды программирования и проектирования; проблемно-ориентированные оболочки; системы управления базами данных.

. . . . . .

Учитывая то, что малый бизнес выходит на рынок с прикладными ПП, рассмотрим один из возможных вариантов классификации этого сегмента рынка (рис.3.4) [17].

Рис. 3.4 – Классификация прикладных программных продуктов

Пользовательские программные продукты — ориентированы на потребительский рынок, к ним можно отнести игровые, развлекательные, обучающие ПП, компьютерные тренажеры, индивидуальные офисные ПП, программные приложения доступа к Интернет-сервисам.

Программные продукты для управления ресурсами предприятия позволяют автоматизировать и оптимизировать бизнес-процессы компании.

При этом имеются в виду те приложения, которые не направлены на решение специфических инженерных задач. Рынок ERP-приложений может включать специальные ПП для конкретных отраслей, а также ПП, которые могут отвечать требованиям нескольких областей деятельности. К ним относятся: финансово-бухгалтерские приложения; приложения для управления персоналом; приложения, связанные с планированием и контролем выполнения основных и вспомогательных и производственных операций, приложения для управления закупками и запасами материальных ресурсов; приложения по управлению проектами и портфелями проектов; приложения для управления основными и оборотными фондами предприятия.

Программные продукты поддержки офисной деятельности компании. Это корпоративные системы, позволяющие автоматизировать и оптимизировать процессы электронного документооборота, нормативно-правового обеспечения деятельности организации, предоставления различных информационно-справочных сервисов. К этому же классу можно отнести также офисные ПП, приложения для организации групповой работы пользователей.

Программные продукты поддержки инженерных расчетов, диагностики, проектирования и конструирования. К этому классу ПП можно отнести:

- *проектирование;*
- *моделирование;*
- *оформление конструкторской документации;*
- *проектирование технических процессов;*
- *оформление технологической и сопроводительной документации;*
- *управление проектами и др.*

Данный рынок включает MCAD-, MCAM-, MCAE-, PIM-, EDA- и AEC-приложения. К данной группе можно отнести также геоинформационные системы.

Системы управления взаимоотношениями с клиентами и партнерами это корпоративные информационные системы, предназначенные для улучшения обслуживания клиентов путем сохранения информации о них и истории взаимоотношений с ними, установления и улучшения бизнес-процедур на основе сохраненной информации и последующей оценки их эффективности.

К такому типу ПП относятся:

- приложения для автоматизации торговли;
- приложения для автоматизации маркетинга;
- приложения для автоматизации службы работы с клиентами;
- приложения для контакт-центров;
- приложения для CRM-аналитики.

Программные продукты разработки и поддержки web-технологий: web-сайты и корпоративные порталы; ПП организации доступа к различным web-сервисам; ПП для создания мультимедийных продуктов и средств доступа к ним и т. д.

Программные продукты поддержки научных исследований: системы математического моделирования; статистического анализа; поддержки принятий решений; экспертные системы и системы искусственного интеллекта и т. д.

3.3 Продвижение программных продуктов в сети Интернет

В зависимости от сложившейся ситуации на рынке, характеристик ПП и целей продвижения определяется возможность применения различных каналов и инструментов распространения информации о продукте.

По способу распространения информации можно выделить два основных канала:

1) реальные каналы. Под реальными каналами продвижения понимаются традиционные инструменты распространения информации: баннеры, печатные издания, телевидение, радиовещание, листовки/буклеты, размещение информации о продукте и образцов ПП в торговых залах, проведение се-

минаров, конференций; организация личных встреч, распространение ПП по OEM-лицензии и т. д.;

2) виртуальные каналы. Под виртуальными каналами продвижения понимаются все формы взаимодействия посредством использования электронных средств, в том числе и Интернета. В качестве инструментов продвижения используются системы электронной коммерции и методы интернет-маркетинга.

Интернет-маркетинг представляет собой «совокупность методов интернет-коммерции, направленных на увеличение экономической эффективности фирмы, включающих в себя: интернет-рекламу..., методы удержания посетителей на сайте, обеспечения приобретения ими товаров или услуг, предлагаемых на сайте..., методы создания постоянной аудитории клиентов...» [18].

Все мероприятия, проводимые в составе интернет-маркетинга, можно условно разделить на два направления. Первое связано с применением Интернета для расширения маркетинговой системы традиционного бизнеса:

- организации активного взаимодействия компании с ее сотрудниками, заказчиками и партнерами;
- формирования имиджа компании и мнения о ней;
- информирования о происходящих внутри компании событиях, изменениях, новостях, акциях и т. п.;
- консультирования клиентов по вопросам товаров и услуг, а также возможностях их приобретения;
- стимулирования потребительского интереса и спроса и пр.

Второе направление связано с появлением новых видов бизнеса, таких, как интернет-магазины, электронные торговые площадки, основные задачи которых заключаются в проведении маркетинговых исследований, удержании уже имеющихся клиентов и привлечении новых.

В этом случае Интернет является не только инструментом, который позволяет повысить эффективность работы компании, минимизируя при этом затраты, но самим бизнесом, основная задача которого приносить доход.

Маркетинговые исследования направлены на изучение:

1) потенциальных потребителей ПП, мотивов их поведения на рынке, потребительских предпочтений к конкретным свойствам ПП (дизайну, техническим характеристикам, сервису и т. п.);
2) состояния рына в целом, его тенденций, конкурентов, их сильных и слабых сторон, занимаемой доли рынка, ценовой политики;
3) инфраструктуры рынка, посредников, действующих на рынке, каналов распространения ПП, роли государства.

В зависимости от применяемых методов и используемых источников информации маркетинговые исследования делятся на кабинетные (вторичные), полевые (первичные) и бенчмаркинг (от англ. benchmarking — эталонное тестирование).

Для удержания посетителей можно использовать следующие инструменты:

1) постоянное обновление информационного наполнения интернет-площадки, добавление новых разделов и т. п., это приводит к тому, что каждый раз пользователь обнаруживает что-то новое и интересное для него;
2) организацию общения пользователей между собой. Гостевые книги, форумы и чаты дают возможность пользователям общаться между собой, делиться впечатлениями о интернет-площадке, программных продуктах и услугах компании, начатая при одном посещении дискуссия стимулирует их к повторному посещению интернет-площадки для ее продолжения;
3) средства общения пользователей и поддержки клиентов — специалисты компании могут выступать в роли экспертов и консультировать посетителей по программным продуктам и услугам, комментировать высказывания посетителей в форумах, чатах и гостевых книгах;
4) предоставление пользователям дополнительных бесплатных, но полезных услуг, например: почтовые (e-mail) услуги, возможность отправки SMS-сообщений, это может существенно увеличить количество постоянных посетителей интернет-площадки.

Для *привлечения новых клиентов* специалист использует практически все инструменты маркетинговых коммуникаций, включая прямую интернет-рекламу, рекламу в традиционных СМИ, PR и т. д.

Учитывая специфику ПП как нематериального (цифрового) продукта, в качестве основного канала распространении информации о программных продуктах целесообразно использовать виртуальные каналы с применением инструментария электронной коммерции, а в качестве основной технологической платформы взаимодействия разработчиков (поставщиков ПП) и потенциальных пользователей при продвижении ПП так называемые интернет-площадки.

Типовая структура интернет-площадки представляется обычно в следующем виде: каталог программных продуктов; каталог разработчиков; кабинет разработчика; кабинет заказчика; информационный блок; подсистема регистрации и авторизации пользователей. Конкретное функциональное наполнение каждого из блоков должно обеспечить:

- поддержку процессов взаимодействия разработчика (производителя) и потенциального пользователя (заказчика);
- информирование потенциальных пользователей о программных продуктах и услугах компании;
- предоставление механизмов публикации, скачивания и тестирования программных продуктов через Интернет;
- информирование посетителей интернет-площадки о последних новостях и событиях в сфере разработки, продвижения и использования программных продуктов;

- предоставление пользователям аналитических обзоров программных продуктов и информационных статей, связанных с жизненным циклом рыночных (тиражных) программных продуктов;
- ведение и поддержку блогов и дискуссионных форумов, раскрывающих историю успеха компании, ее программных продуктов и услуг.

Продвижение программных продуктов, представленных на площадке, напрямую зависит от посещаемости и популярности самой интернет-площадки. Для того чтобы интернет-площадка стала известна целевой аудитории, необходимо после ее создания и размещения в сети осуществить ряд мероприятий по ее продвижению.

На сегодняшний день можно выделить следующие основные методы продвижения: поисковая оптимизация сайта; контекстная реклама; медийная реклама; оптимизация в социальных медиа [19].

Поисковая оптимизация представляет собой комплекс мер для привлечения потенциальных потребителей ПП на интернет-площадку по результатам выдачи поисковых систем по определенным запросам пользователей.

Чаще всего результатом поиска (поисковой выдачей) является огромное количество удовлетворяющих запросу страниц, включающих в том числе и большое количество ссылок на другие сайты. Вместе с тем статистика самих поисковых систем показывает, что в основном пользователи просматривают не более первых трех страниц поисковой выдачи. Соответственно, чтобы обеспечить приток посетителей на сайт с поисковых систем, необходимо, чтобы ссылка на сайт находилась на первых трех страницах поисковой выдачи, причем чем выше, тем больше вероятности, что пользователь перейдет на сайт.

При использовании *поисковой оптимизации* для продвижения информации о ПП важно выделить ключевые слова, характеризующие программный продукт, тогда по целевым запросам в выдаче поисковых систем будут появляться ссылки на страницу интернет-площадки с описанием ПП.

Привлечение посетителей с использованием поисковых систем — один из самых низкозатратных и эффективных способов интернет-маркетинга. Кроме того, посетители, которые привлекаются таким образом, как правило, — целевая аудитория, так как они ищут именно ту информацию или товары, которые есть на сайте.

Контекстная рекламапредставляет собой вид размещения интернет-рекламы, в основе которой лежит принцип соответствия содержания рекламного материала контексту (содержанию) интернет-страницы, на которой размещается данный материал.

При этом по характеру рекламный материал может быть тексто-графическим объявлением либо рекламным баннером. Обычно для определения контекста и от-

бора объявлений используется сервис той или иной системы контекстной рекламы (сервиса контекстной рекламы).

Существуют два основных вида контекстной рекламы в Интернете, отличающихся фактически только видами размещения: контекстная реклама на сайтах (или тематическая реклама) и поисковая реклама.

Тематическая контекстная реклама — один из популярнейших инструментов Интернет-маркетинга, подразумевающий платное размещение различных рекламных элементов (в основном текстовых блоков) на веб-страницах, тематика которых совпадает с рекламируемым продуктом или услугой. Как правило, рекламные блоки привязаны к определенным ключевым словам и словосочетаниям (контексту), выбираемыми рекламодателем для своих рекламных объявлений, поэтому такая реклама и называется контекстной.

Поисковая контекстная реклама — частный случай контекстной рекламы, применяемый в поисковых системах. В этом случае демонстрация рекламных сообщений осуществляется на страницах выдачи результатов поиска с учетом поискового запроса пользователя. При этом система поисковой рекламы также может размещать рекламные объявления не только на своем основном сайте, но и на других популярных сайтах, имеющих функцию поиска.

Поисковая контекстная реклама считается более эффективной, чем просто контекстная реклама на сайтах, по следующим причинам:

1) в распоряжении поисковой рекламы сразу же имеется поисковый запрос, сформулированный пользователем, что лишает необходимости определения системой тематики и основных ключевых слов в содержании страницы, а следовательно, увеличивается и релевантность рекламных объявлений, показываемых пользователю;
2) поисковая реклама показывается пользователям, которые уже выразили заинтересованность в рекламируемых товарах/услугах, т. е. производят поиск соответствующей информации в поисковой системе, а не просто осуществляют навигацию по Интернету;
3) психологически поисковая реклама не воспринимается в качестве рекламной информации, так как она показывается наравне с результатами поиска. Таким образом, высокий уровень доверия к поисковым системам переносится и на рекламные объявления.

Контекстную рекламу размещают такие системы, как Яндекс (direct.yandex.ru/), Google (adwords.google.com), Бегун (www.begun.ru/) и многие другие системы.

Использование контекстной рекламы также предполагает выделение ключевых слов, характеризующих программный продукт, по запросу которых будет показываться реклама продукта. Для подбора ключевых слов можно воспользоваться сервисами вышеперечисленных систем.

Медийная рекламаосновывается на размещении текстово-графических рекламных материалов (баннеров) на сайтах, представляющих собой рекламные площадки. По многим признакам этот вид рекламы аналогичен рекламе в печатных СМИ. Однако наличие у баннера гиперссылки и возможность анимированного изображения значительно расширяют возможности воздействия медийной рекламы.

Медийная реклама может быть размещена следующими способами:

1) прямое размещение рекламы на рекламных площадках;
2) прямой обмен ссылками и баннерами между Интернет-площадкой разработчика ПП и рекламной площадкой;
3) баннерные сети и сети обмена ссылками.

Размещение медийной рекламы может осуществляться как по прямой договоренности с владельцами рекламных площадок, так и через специализированные рекламные агентства, которые занимаются оптовыми закупками ссылок с различных ресурсов и через специальные специализированные сервисы — биржи ссылок. Одним из таких сервисов является сайт sape.ru.

Баннерообменные сети обеспечивают показ баннеров на web-страницах сайтов, являющихся участниками сети, учет количества таких показов и переходов на интернет-площадку разработчика. Показ баннеров, размещенных на интернет-площадке, будет осуществляться при условии показа чужих баннеров или путем покупки определенного количества показов на web-стра-ницах сайтов участников сети.

Баннерная реклама до определенного момента была наиболее эффективным методом интернет-рекламы, однако со временем посетители привыкают к изобилию баннеров настолько, что фактически перестают на них реагировать. Кроме того, современные интернет-обозреватели предоставляют возможность отключения показа баннеров, что также негативно сказывается на эффективности медийной рекламы. С другой стороны, основное преимущество и отличие медийной рекламы от других методов продвижения — имиджевая составляющая рекламного воздействия. Медийная реклама лучше всего подходит для кратковременного масштабного привлечения внимания к разовой акции по продвижению продукта, услуги и бренда компании.

Оптимизация в социальных медиапредставляет собой комплекс мер, направленных на привлечение на сайт посетителей из социальных медиа: блогов, социальных сетей и условно подразделяется на две ветви:

1) оптимизация под социальные медиа (social media optimization, SMO) — комплекс технических мероприятий, направленных на преобразование контента интернет-площадки разработчика таким образом, чтобы его можно было максимально просто использовать в сетевых сообществах (форумах, блогах);
2) маркетинг в социальных медиа (Social media marketing, SMM) — продвижение или пиар какой-либо информации в социальных медиа (блогах, форумах, сетевых сообществах).

Основной целью SMO является создание условий для цитирования сайта в большем количестве в социальных медиа со ссылками на сам сайт. Это может быть достигнуто за счет проведения следующих мероприятий:

- публикации интересных исследований, тематических статей, фото и видеоматериалов;
- быстрого добавления записи в сервисы закладок и социальные сервисы;
- проведения конкурсов среди пользователей и поощрения тех, чьи сообщения в социальных медиа содержат больше ссылок на сайт;
- предоставления контента сайта в различных форматах, например наличие, кроме HTML-версии, PDF-версии сайта;
- поощрения использования контента через организацию ссылок с другого сайта.

Описанные мероприятия по оптимизации сайта для социальных медиа ведут к увеличению индекса цитируемости сайта, что дает преимущество перед сайтами конкурентов при поиске по одинаковым поисковым запросам. Таким образом, при использовании SMO процесс продвижения идет сразу по двум направлениям — сайт оптимизируется как для поисковых систем, так и для пользователей Интернета.

Основными задачами SMM является самостоятельное размещение разработчиками ПП в социальных медиа скрытой рекламы о компании, ее продуктах и услугах, а также поощрение размещения информации об этом другими пользователями. При этом можно выделить три основных способа социального маркетинга:

1) создание и развитие собственных *аккаунтов* в социальных медиа;
2) заказ платных публикаций пользователям социальных медиа;
3) использование услуг специализированных сервисов.

По своей сути методы SMO и SMM дополняют друг друга, являясь двумя направлениями маркетинга в социальных медиа, и наибольшая эффективность продвижения продуктов и услуг может быть достигнута при их совместном использовании.

Информационное наполнение каждого из описанных методов продвижения может быть представлено в виде: текстовых рекламных блоков; графических рекламных блоков (баннеров); рекламных заставок.

Текстовые рекламные блоки представляют собой текстовые ссылки, ведущие на сайт рекламодателя. Такие ссылки обычно располагаются непосредственно в текстах рекламной площадки или результатах выдачи поисковых систем и имеют большое значение в поисковой оптимизации сайта, а также являются основным рекламным носителем в контекстной рекламе и продвижении социальных медиа.

Графические рекламные блоки (баннеры) представляют собой чаще всего анимированные изображения, размещенные в специально выделенных местах рекламной площадки. Безусловно, баннер более заметен для посетителя рекламной площадки по сравнению с текстовым рекламным блоком. Баннеры являются основным рекламным носителем в медийной рекламе.

Рекламные заставки, как правило, также представляют собой графические анимированные изображения, но в отличие от баннеров загружаются одновременно с сайта и располагаются поверх него, закрывая часть информации. Пользователь вынужденно просматривает рекламную заставку, однако такой вид рекламного воздействия часто вызывает негативную реакцию пользователя.

Кроме методов продвижения программных продуктов и услуг через интернет-площадки, существуют и ряд методов прямого продвижения программных продуктов через Интернет.

Регистрация в каталогах программных продуктов. Размещение информации о новом продукте в каталогах программных продуктов, как правило, является первым шагом в популяризации продукта (после размещения о нем информации на интернет-площадке). Эта информация содержит в себе ссылку на страницу интернет-площадки, где непосредственно размещен программный продукт. При использовании данного метода важно, чтобы информация, размещенная в каталогах, со временем не потеряла своей актуальности.

Публикация обзоров, сравнений, новостей. Метод представляет собой SMO-продвижение только в тематических статьях, публикуемых в социальных медиа. Представляется информация о продукте, сравнении продукта с аналогичными продуктами и т. д. Кроме того, использование данного метода позволит получить обратную связь от пользователей социальных медиа, которая может быть полезна для уточнения стратегии развития продукта и стратегии его продвижения.

Использование электронной почты. Использование электронной почты позволяет пользователям программного продукта организовать адресную рассылку о выходе новых версий или новых модулей к ПП. Использование этого метода позволит периодически напоминать пользователям о продукте и, одновременно, о своих услугах, а также обеспечить продвижение новых версий или модулей ПП среди целевой аудитории. Чтобы воспользоваться данным методом, необходимо знать контактную информацию пользователей ПП.

В заключение следует отметить, что регулярное проведение мероприятий по продвижению интернет-площадки обуславливает постоянный приток новых потенциальных клиентов. Преимущество продвижения состоит и в том, что интернет-площадка выходит со временем на некий «непотопляемый» уровень и может считаться брэндом в Интернете, попадает в популярные каталоги и т. д., и т. п., что ведет к повышению доверия пользователей и соответственно росту их числа. То есть эффект от рекламной кампании как бы аккумулируется со временем.

3.4 Основы ценообразования на тиражные программные продукты

При формировании стратегии ценообразования на ПП следует различать вопросы формирования договорной цены между разработчиком и заказчиком при разработке ПП под конкретные потребности последнего и вопросы определения рыночной цены продаж при продвижении на рынок тиражного ПП.

При разработке ПП под конкретный заказ проблема сводится к взаимосогласованности мнений между разработчиком и заказчиком по определению договорной цены на разработку ПП.

В этом случае очевидно стремление заказчика снизить договорную цену проекта, от разработчика же требуется корректное обоснование ее величины. Проблемы определения договорной цены заключаются в том, что процесс создания ПП, по сути, является процессом преобразования спецификаций, требований и пожеланий заказчика в код и документацию программного продукта. Сложность управления таким процессом вызвана сложностью получения количественных оценок входной и выходной информации, а также операций по ее преобразованию. Он отчасти является творческим и тяжело поддается формализации и моделированию. Таким образом, можно утверждать, что наличие эффективной методики оценивания

проектов позволит устранить значительную часть причин срыва проектов и будет способствовать повышению конкурентоспособности организации-разработчику на рынке.

В основе существующих методов расчета договорной цены программного продукта используются, как правило, следующие показатели: размеры программной системы, трудозатраты на разработку и внедрение, длительность разработки, численность и квалификация специалистов разработчика, нормативы оплаты труда при создании программного кода, величина накладных расходов разработчика, включая желаемую норму прибыли.

Следует отметить, что возможность получения объективных количественных оценок характеристик ПП полезна не только разработчикам, но и заказчикам. Обе заинтересованные стороны могут видеть, какой вклад в стоимость проекта вносит та или иная функциональность, и сопоставить со степенью ее важности. По обоюдному согласию из проекта по разработке ПП может быть оперативно исключена не оправдывающая своей цены функциональность или ее реализация перенесена на следующую версию или даже в другой проект.

Основная проблема разработчика (производителя) при оценке рыночной стоимости ПП сводится к определению цены, обеспечивающей максимальный доход фирмы. При этом очевидно, что один и тот же доход можно получить, продавая товар по низкой цене многим пользователям либо по высокой цене ограниченному количеству пользователей, очевидно, что цена продажи и количество продаж непосредственно влияют друг на друга. Первая — стратегия ценообразования рассчитана на охват максимального количества возможных пользователей благодаря низкой цене продукта. Вторая — направлена на реализацию продукта наиболее платежеспособным пользователям по соответственно высокой цене. В чистом виде данные стратегии почти не встречаются, чаще всего применяются их модификации. При второй стратегии — цены после того, как программный продукт был внедрен на наиболее платежеспособном целевом рынке пользователей, как правило, снижаются с учетом платежеспособности следующей потенциальной группы покупателей. И наоборот, фирма долгое время может продавать свои пакеты прикладных программ по низкой цене, а в последующем повысить цены, например с помощью искусственного разбиения продуктов на две или более составные части.

Можно предложить, что при достаточно большой цене на продукты доходы компаний могут постепенно уменьшаться. Здравый смысл подсказывает, что, как правило, после продажи и установки ПП продолжается техническая поддержка пользователя и с каждым новым пользователем трудозатраты разработчика на поддержку увеличиваются. Поэтому при прочих равных условиях необходимо стремиться получить максимальный доход при меньшем количестве пользователей. Вместе с тем, чем больше пользователей пользуются продуктом фирмы, тем больше о нем говорят и опосредованно рекламируют его, снижая тем самим издержки фирмы на рекламу и продвижение. Таким образом, вопрос о том, что лучше: «много продаж по малой цене либо мало продаж по большой цене» не имеет однозначного ответа.

С точки зрения политики ценообразования выделяются четыре типа рынка: рынок чистой конкуренции; монополистический рынок; олигополистический рынок; рынок чистой монополии [20].

Рынок чистой конкуренции состоит из множества производителей и потребителей какого-либо программного продукта, реализующего выполнение конкретной функции, например электронный документооборот.

Ни один отдельный производитель или потребитель не оказывает большого влияния на уровень текущих рыночных цен. Производитель не в состоянии запросить цену выше рыночной, поскольку потребители могут свободно приобрести программный продукт у конкурента. Не будут производители запрашивать и цену ниже рыночной, поскольку могут поставить программный продукт по существующей рыночной цене. На рынке чистой конкуренции роль маркетинговых исследований, политики цен, рекламы, стимулирования сбыта и прочих мероприятий минимальна.

На рынке монополистической конкуренции доминирует дифференцированный подход к ценообразованию программного продукта при этом предполагается, что цены на него могут устанавливаться в широком диапазоне.

Наличие диапазона цен объясняется способностью производителей предложить потребителям разные варианты услуг по продаже и сопровождению программных продуктов. Различия могут заключаться в предложении потребителю сопутствующих программных продуктов, дополнительных сервисов по поддержке пользователя и т. д. Чтобы выделиться чем-то, помимо цены на программный продукт и перечня услуг, разработчики ориентируются на разные потребительские сегменты, широко пользуются рекламой и другими каналами продвижения.

Олигополистический рынок состоит из небольшого числа производителей, весьма чувствительных к политике ценообразования и маркетинговым стратегиям друг друга.

Небольшое количество разработчиков объясняется тем, что новым претендентам трудно проникнуть на этот рынок; если какая-то компания снизит свои цены на продукт, потребители могут отдать предпочтение программным продуктам именно этой компании. В этом случае другим производителям придется реагировать либо тоже снижением цен, либо предложением большего числа сервисов.

На *рынке чистой монополии* всего один продавец. В случае регулируемой монополии государство может устанавливать компании цены, обеспечивающие получение «справедливой нормы прибыли», которая даст организации возможность поддерживать производство, а при необходимости и расширять его. В случае нерегулируемой монополии фирма сама вправе устанавливать любую цену, которую только выдержит рынок.

При *постановке задачи ценообразования* прежде всего необходимо определиться с целями ценовой политики, при этом необходимо выбрать один из нижеперечисленных вариантов.

1) *Обеспечение выживаемости ПП на рынках.* Проблемы могут возникнуть из-за конкуренции или изменившихся запросов потребителей. Чтобы обеспечить разработку и продажу ПП, фирмы вынуждены устанавливать низкие цены в надежде на благожелательную ответную реакцию потребителей, понимая, что в данный момент выживание важнее прибыли. До тех пор пока снижение цены на ПП покрывает его издержки, программный продукт следует продвигать, можно продолжать коммерческую деятельность.

2) *Максимизация прибыли.* При реализации данной цели фирмы ориентируются на варианты цен, обеспечивающие на краткосрочном периоде получение максимальной прибыли и не рассматривают долгосрочные перспективы, определяемые использованием всех других элементов маркетинга, политикой конкурентов, регулирующей деятельностью государства.

3) *Максимальное увеличение объема продаж.* Увеличение объема сбыта приведет к снижению издержек на единицу продукции и к увеличению прибыли. Исходя из возможностей рынка выбирается «ценовая политика наступления на рынок». Фирма снижает цены на свою продукцию до минимально допустимого уровня, повышая долю своего рынка, добивается снижения издержек на единицу продукции и на этой основе может и дальше снижать цены. Но такая политика приносит успех только, если чувствительность рынка к ценам велика, если реально уменьшить издержки производства в результате расширения объемов производства и, наконец, если цены конкурентов не будут скорректированы в сторону уменьшения.

4) *«Снятие сливок», благодаря установлению высоких цен.* Фирма устанавливает на новый продукт, продвигаемый на рынок, максимально возможную цену благодаря сравнительным преимуществам новинки. Когда сбыт по данной цене сокращается, фирма снижает цену, привлекая к себе следующий слой потребителей, достигая в каждом сегменте целевого рынка максимально возможного оборота.

5) *Лидерство по качеству.* Фирма, которая способна закрепить за собой такую репутацию, устанавливает высокую цену, чтобы покрыть большие издержки, связанные с обеспечением высокого качества и необходимыми для этого затратами.

Определившись с целями ценовой политики, компания должна выбрать одну либо несколько методик определения цены на ПП. Классическое ценообразование основывается на определении фактических расходов (себестоимости) на производство и реализацию продукта и прибавление к ним желаемой прибыли, выраженной в денежной форме. При выборе методов ценообразования фирма исходит из того, что в любом случае минимальная цена определяется себестоимостью продукта, а максимальная — наличием каких-то уникальных потребительских ценностей или монопольным положением фирмы на соответствующем сегменте рынка.

Методы ценообразования, ориентированные на затраты

Методы ценообразования, ориентированные на затраты, основаны на том, что производитель понес расходы на разработку и продажу продукта. В этом случае при определении цены продажи необходимо определить диапазон цен, обеспечивающих покрытие переменных и постоянных затрат и получение желаемой прибыли.

При тиражном ПП затраты разработчика определяются расходами на разработку и тестирование первой версии ПП, распространение, обслуживание, модернизацию и поддержку пользователей в течение всего срока существования программного продукта. Модернизация ПП и поддержка заказчика являются долговременной финансовой нагрузкой, которую разработчик принимает на себя в момент поставки первой версии продукта, и это обязательно следует учитывать при оценке затрат. Модернизация должна поддерживать приемлемую обратную совместимость (например, допускать применение как старых, так и новых форматов файлов), причем так, чтобы возможность перейти на конкурирующие продукты не стала для заказчиков более привлекательной, чем обновление существующего решения. Таким образом, модернизация порождает сложную дилемму выбора оптимального соотношения затрат и цены на ПП.

Немаловажное значение при установлении цены, ориентированной на затраты, имеют *вопросы снижения совокупных затрат заказчика при внедрении и эксплуатации ПО*: прямых текущих затрат на эксплуатацию (сопровождение); затрат на реорганизацию производства, связанных с модернизацией бизнес-процессов, например при внедрении ERP-систем (управление ресурсами предприятия); затрат на обучение персонала; затрат на модернизацию ПО, в том числе по переходу на другие программные продукты. Заказчики хорошо представляют себе последствия «привязки» к производителю. Их беспокоит не только отсутствие возможности использовать конкурирующие продукты, но и перспектива того, что производитель либо прекратит поддержку и модернизацию своего продукта, либо вообще выйдет из бизнеса.

Затраты на создание ПП могут значительно увеличиться, если заказчик и разработчики допустили при обследовании существенные ошибки, вследствие которых представленная версия не будет соответствовать требованиям существующих бизнес-процессов.

При заказном ПП затраты на разработку и внедрение ПП согласуются на основе прямых переговоров между разработчиком и заказчиком, при этом иногда разработчик берет на себя авторское сопровождение ПП. В этом случае затраты разработчика складываются из себестоимости первой версии ПП, сопровождения и создания новых версий, отвечающих изменившимся потребностям заказчика.

Недостатком определения цены по методу «издержки плюс прибыль» является произвольность накладных расходов при подсчете полных издержек и игнорирование фактора спроса.

Один из вариантов метода ценообразования, ориентированного на затраты, является *метод установления «целевой» цены,* основанной на определении точки безубыточности. В этом случае выбранная ценовая политика должна обеспечить получение «целевой» нормы прибыли на произведенные затраты при установленных объемах продаж. Недостаток этого метода связан с трудностями определения

величины объема продаж, так как сама цена может существенно влиять на этот объем.

Методы ценообразования, ориентированные на спрос

Методы ценообразования, ориентированные на спрос, предусматривают готовность потребителей оплачивать ПП по определенной цене (верхней границе цены). При этом необходимо учитывать реакции потребителей на изменения цен и возможности их дифференцирования. При использовании данных методов не прослеживается непосредственная связь между затратами и установлением цен, за исключением случаев необходимости устанавливать цены ниже нижней границы цены. Если потребители твердо представляют себе «справедливую цену», то ценообразование должно учитывать это обстоятельство. Проблема в процессе использования данных методов состоит в том, что спрос значительно труднее определить и выразить в количественных величинах, чем издержки. Проблема оценки спроса особенно усложняется применительно к новым ПП продвигаемым на рынок в силу отсутствия статистики за прошлые периоды. Возможное частичное решение этой проблемы — изучение спроса на аналогичные продукты, если таковые существуют. Возможные ошибки при прогнозировании спроса непосредственно связаны с непредвиденной конкуренцией. Практические рекомендации по минимизации рисков в этом направлении сводятся к следующему: поставляемый продукт не должен быть полным аналогом существующих на рынке программных систем; рекомендуется первым завершить разработку и захватить рынок, даже если это приведет к увеличению затрат.

Методы ценообразования, ориентированные на конкуренцию (конкурентов)

Методы ценообразования, ориентированные на конкуренцию (конкурентов), основываются на сравнениях потребительской ценности программного продукта с продуктами конкурентов. Проблема осложняется тем, что индустрия производства программного обеспечения достаточно молода, и у потенциальных пользователей нет навыков объективного анализа потребительских свойств ПП. Кроме того, потребительские свойства ПП с точки зрения пользователя зачастую отличаются от точки зрения разработчика. В этом случае при описании продукта разработчику необходимо смотреть на него глазами клиента. Главную роль при сравнении программного продукта с конкурентами-аналогами играют потребительские свойства продукта, при этом не обязательно, чтобы он превосходил конкурентов по всем параметрам. Необходимо выбрать главное конкурентное преимущество и на нем делать акцент.

При равной маркетинговой активности продавцов заказчики, как правило, отдают предпочтение программным продуктам, позволяющим получить *максимальную выгоду* при их использовании, определяемую как разность между *потребительской ценностью и общей стоимостью продукта*, включающей единовременные на приобретение и внедрение ПП и текущие расходы на эксплуатацию. В этом случае очевидно стратегия разработчика должна быть направлена на увеличение потребительской ценности и снижение общей стоимости.

Потребительская ценность продукта зависит от ряда факторов, таких как качество ПП, возможность подключения других приложений, платформонезависимая реализация и т. д. *Качество программного обеспечения* должно максимально соответствовать требованиям стандарта ГОСТ Р ИСО/МЭК9126-93, который рекомендует оценивать программное обеспечение по следующим критериям: функциональная пригодность, надежность, практичность, эффективность, сопровождаемость, переносимость. Особенно большое значение имеют для пользователя функциональные возможности программного продукта.

Рекламируя свою программную продукцию, фирме следует максимально использовать эти критерии, обращая внимание клиентов, прежде всего, на *уникальность* созданной системы. В этом случае при выборе новых направлений разработки программных систем следует избегать создания продуктов, напрямую альтернативных решениям других производителей, поскольку это «играет на руку» тому поставщику, который контролирует большую долю рынка, так как в этом случае они имеют более высокий доход при сопоставимых затратах. Очевидно, что серьезное отличие от конкурентов по максимально большему числу критериев имеет решающее значение. С другой стороны, компания, первой на рынке предложившая новую категорию продуктов или значительно улучшившая характеристики ПП из уже имеющейся категории, получает важное преимущество. Таким образом, при проектировании ПП в первую очередь следует уделять внимание полезным и выгодным для заказчиков новаторским решениям. Тезис «мы делаем лучше других» должен быть заменен на тезис «мы делаем отличное от других».

Дополнительным фактором, увеличивающим практическую полезность продукта, является *возможность использования заказчиком приложений других производителей*. В этом случае ПП должно соответствовать некоторым стандартам на взаимодействие различных приложений. Открытый интерфейс является, как правило, двусторонним, он создает взаимозависимость и поддерживает конкурентоспособность обеих сторон. Поддержка стандартов делает успех на рынке более предсказуемым, в частности потому, что гарантирует интероперабельность с дополняющими решениями и популярность у пользователей.

При распространении любого программного продукта круг потенциальных заказчиков определяется выбором инфраструктурной *платформы.* Возможность переноса и поддержка программного обеспечения для многих платформ помогают увеличить доходы, но при этом возрастают расходы на разработку, обслуживание, тестирование и поддержку ПП.

Следующим фактором, увеличивающим потребительскую ценность ПП, является комплексирование и продажа различных программных продуктов в одном пакете. Заказчику предоставляется линейка различных вариантов поставки программного обеспечения и выбор одного из них в зависимости от реальных потребностей и платежеспособности. Чтобы такой подход стал возможным, архитектура программного продукта должна позволять формировать различные комбинации модулей в зависимости от предполагаемых требований к производительности и функциональности при условии, что пользователи не могут самостоятельно менять конфигурацию. При этом разработчики должны установить, оправдана ли подобная гибкость с точки зрения затрат на поддержку и обслуживание, которые будут расти с увеличением числа вариантов и даже числа комбинаций конфигурационных возможностей.

С учетом вышеизложенного в условиях *рынка монополистической конкуренции* фирма может выбрать одну из трех ценовых стратегий: приспособление к рыночной цене, последовательное снижение цены, последовательное повышение цены. Политика последовательного снижения цены, ориентированная на цены конкурентов, может использоваться при выводе на рынок нового программного продукта, что должно обеспечить быстрое привлечение клиентов и резкое увеличения объемов продаж. С помощью низкой цены воздвигаются барьеры для контрафактных подделок ПП.

Базовая (исходная) цена, установленная по любому из перечисленных методов ценообразования, может корректироваться в сторону уменьшения с учетом различных факторов:

- наличия скидок при большом заказе количества лицензий;
- предлагаемой формы оплаты: предоплата, оплата по факту получения продукта, оплата с рассрочкой и т. д.;
- возможности установления скидки определенным группам клиентов;
- включения в цену продажи сервисов по техническому сопровождению и поддержке пользователя;
- набора функциональностей программного продукта и способа поставки: стандартная, профессиональная, промышленная версии.

3.5 Управление лицензиями на программное обеспечение

Эффективно управлять лицензиями на ПО — значит, быть уверенным в завтрашнем дне, избегать большого числа рисков и, в итоге, иметь более предсказуемый и более устойчивый, а следовательно, и более прибыльный бизнес.

Лицензирование программ — это процедура, позволяющая организации приобрести, установить и использовать программное обеспечение на отдельном компьютере или в сети соответственно лицензионному соглашению с производителем этого программного обеспечения. Цель лицензирования — защитить как инвестиции компании разработчика, минимизировав вероятность его взлома пиратами, так и инвестиции предприятия, снизив риск наказания за использование пиратского программного обеспечения. Как правило, разработчик реализует лицензионное соглашение путем встраивания в продукт специальных механизмов, не позволяющих использовать программу в случае нарушения пользователем каких-либо пунктов этого соглашения.

Нормативным документом, регламентирующим правила и условия взаимодействия разработчика (продавца) и пользователя (покупателя), является лицензионное соглашение.

Процесс управления лицензиями включает в себя несколько этапов [21].

3.5.1 Проведение инвентаризации установленного программного обеспечения

Инвентаризация имеющегося программного обеспечения может проходить по следующим сценариям:

- если компьютеры компании *не объединены в сеть*, инвентаризацию каждого компьютера придется производить вручную, сканируя жесткий диск и занося полученную информацию в отчет;
- если компьютеры клиента *объединены в сеть*, то необходимо воспользоваться программами для инвентаризации ПО, которые автоматически просканируют жесткие диски рабочих станций и серверов. Программы создают отчеты обо всем найденном ПО, которые можно взять за основу при выборе модели лицензирования и разработке регламентов по управлению лицензиями [19].

3.5.2 Выбор моделей лицензирования

Основные модели лицензирования представлены в табл. 3.1.

Таблица 3.1 – Модели лицензирования

Наименование модели	**Основные особенности модели**
1. Пакетная	Лицензия выдается одному пользователю либо на один компьютер
2. Сетевая	Лицензия выдается на определенное количество пользователей (компьютеров) в сети
3. По подписке	Лицензия приобретается в сетевой версии модели на определенный период времени
4. Повременная	Лицензия приобретается в сетевой версии модели с оплатой за время фактического использования программы

Традиционно защита программ от пиратского копирования при пакетной модели обеспечивалась различными механизмами блокировки, реализуемыми через IP-адреса, адреса хостингов компьютеров, серийные номера системных дисков, специальные аппаратные ключи.

В настоящее время интенсивно используется технология распространения лицензий через Internet, которая позволяет самим пользователям при запуске пользовательской программы и указании серийного номера устройства или кода активации активизировать лицензионные ключи и получать информацию о разрешении на использование программы.

Сетевая модель лицензирования выросла из необходимости развертывания множества программных продуктов на настольных и мобильных компьютерах сотрудников. Наиболее известны две модели — лицензия на одновременный доступ поль-

зователей (concurrent), называемая также «плавающей», и лицензирование по сетевым именам (network-named).

В сетевой модели ответственность за управление лицензированием программ возлагается на серверные приложения. Каждый клиент в определенные интервалы времени выходит на связь с сервером лицензий (так называемый «пульс»). Если лимит доступных лицензий исчерпан, клиенту будет отказано в праве использовать лицензию, а следовательно, и сам продукт.

Лицензия с одновременным доступом обеспечивает определенную гибкость, позволяя предприятию купить пул лицензий, которые оно само может впоследствии распределять между пользователями компьютеров, на которых применяется данный продукт.

Лицензии на сетевые имена, в общем, менее дорогостоящи, чем лицензии с одновременным доступом, но дают меньшую гибкость, так как каждая лицензия выделяется для одного пользователя и не предназначена для совместного доступа. Вместо этого их работа основана на списках, отражающих связь отдельных пользователей с доступом к лицензируемому продукту.

Сетевая модель наиболее предпочтительна для крупных организаций и больше соответствует требованиям разработчиков, чем традиционные пакетные модели. Последние обычно не справляются с реалиями больших организаций, где сотрудники устраиваются на работу и увольняются и становится все труднее обеспечивать мониторинг и поддержку программ на каждом компьютере. На передний план выходят механизмы, обеспечивающие автоматическое восстановление сетевых лицензий, так как они предлагают более щадящий метод учета установленных программных продуктов.

Обратная сторона сетевого лицензирования — дополнительный сетевой трафик, который является результатом информационного обмена между клиентом и сервером. Сетевая модель требует постоянного обмена данными между клиентским приложением и сервером лицензий. При отсутствии правильного регулирования этот обмен может стать настолько интенсивным, что вызовет отказ сети.

Пакетное сетевое лицензирование осуществляется в двух видах: пожизненная лицензия и пробная лицензия. Пожизненные лицензионные соглашения, как правило, включают техническую поддержку, доработки и обновления версий на какой-то определенный период. Пробные лицензии обычно не включают никакой технической поддержки и делают программу недоступной по истечении 30–60 дней с момента установки либо до тех пор, пока покупатель и разработчик не перейдут к пожизненному лицензированию.

В [22] отмечается, что пожизненное лицензирование в настоящее время доминирует на рынке. Вместе с тем набирают популярность новые модели: повременная модель и модель по подписке.

При повременном моделировании организации покупают необходимое количество лицензий с оплатой за время использования, а компания-разработчик назначает плату за каждое использование соответственно тому, как это понятие определено в лицензионном соглашении. Такая модель лицензирования хорошо интегрируется с сетевой моделью с одновременным доступом, поскольку система управления сетевыми лицензиями может регистрировать каждый факт использования продукта и обновлять внутренние параметры, чтобы вовремя заметить, что поль-

зователи превысили лимит использования. Предприятие может настроить систему с одновременным доступом таким образом, чтобы она регистрировала каждый факт использования программного обеспечения и представляла отчеты уполномоченным сотрудникам, а они, в свою очередь, периодически направляли эти отчеты компании-разработчику.

В рамках модели по подписке организация приобретает лицензии на определенный срок (обычно на год), в течение которого автоматически получает обновления и дополнительные возможности. Плата, как правило, вносится за год. Когда срок действия лицензии истекает, пользователь перестает получать обновления до тех пор, пока предприятие не возобновит подписку. Модель лицензирования по подписке основана на предположении, что непрерывные автоматические обновления выгодны пользователям и разработчикам. Стоимость дополнительных ресурсов, как правило, невысока, и это несет в себе еще больше возможностей для максимизации возврата инвестиций в программное обеспечение по сравнению с традиционными моделями.

Пример

В качестве примера перечислим основные модели лицензирования фирмы Microsoft [21]:

1) *OEM-версия программного обеспечения*, которая может быть приобретена с новым компьютером. Лицензионность ПО подтверждают:
   - лицензионное соглашение Microsoft с конечным пользователем;
   - сертификат подлинности — COA (сертификат подлинности ОС Windows должен быть наклеен на корпус компьютера);
   - исходные носители;
   - руководство пользователя (если прилагается);
   - квитанция или счет;
2) *«коробочные» продукты*, которые можно покупать в розницу в магазинах и у партнеров Microsoft. Лицензионность ПО подтверждают:
   - индивидуальное лицензионное соглашение с конечным пользователем (EULA);
   - исходные носители;
   - руководство пользователя (если прилагается);
   - квитанция или счет;
3) *Open License*, предназначенная для организаций, которые приобретают не менее пяти лицензий;
4) *Microsoft Multi — Year Open License* — программа корпоративного лицензирования, позволяющая организациям, имеющим пять и более компьютеров, приобрести лицензии и ключевые продукты Microsoft в рассрочку;

5) *Microsoft Open Subscription License* — корпоративное лицензионное соглашение, предоставляющее организациям, имеющим пять и более ПК, уникальную возможность лицензировать основные продукты Microsoft для настольных компьютеров с минимальными единовременными затратами;
6) *Microsoft Enterprise Agreement* и *Microsoft Enterprise Agreement Subscription* — корпоративное лицензирование, предназначенное для организаций, выбравших платформу Microsoft в качестве корпоративного стандарта, с числом компьютеров от 250.

. . . . . .

3.5.3 Разработка регламента по управлению лицензиями организации

При разработке регламента следует подробно описать следующие процедуры: порядок приобретения лицензионного ПО, порядок использования ПО, процедуру автоматизированной установки ПО, процедуру хранения и обновления лицензионного программного обеспечения и документации на него, права и обязанности ответственных лиц за управление лицензиями компании, технологию ведения инвентаризационной базы данных для контроля за лицензиями, процедуру периодического мониторинга (официального и неофициального) используемого ПО в подразделениях компании.

Грамотное управление лицензиями на программное обеспечение обеспечивает:

1) эффективное использование финансовых ресурсов на приобретение ПО.

   В организациях, где отсутствует системный взгляд на управление лицензиями, часто происходит дублирование в приобретении лицензионного ПО, отсутствует возможность приобретения корпоративных лицензий, что не позволяет получить определенные скидки в ценовой политике, возможность получения рассрочки в платежах при оптовом заказчике;

2) сведение до минимума юридических и технических рисков при эксплуатации ПО.

   Своевременный мониторинг состояния ПО в структурных подразделениях компании позволит: препятствовать появлению нелицензионных копий; обеспечить соблюдение авторских прав на программы для ЭВМ, свести до минимума вероятность сбоев в программном обеспечении, несовместимость форматов данных различных версий одноименных программных продуктов;

3) возможность легального и законного бизнеса.

   Это касается как компаний, использующих информационные технологии в качестве инструментария по поддержке и принятию решений, так и компаний, работающих на рынке информационных технологий. Последние, используя в своей деятельности лицензионное ПО, могут активно включаться в российский и международный бизнес и систему проектирования, активно участвовать в различных выставочно-ярмарочных мероприятиях, пройти сертификацию по стандартам ISO9000;

4) централизованное управление лицензиями, которое позволяет:
   - хранить все соглашения и лицензии в одном месте (это облегчит поиск);
   - сократить расходы (приобретая только нужные виды лицензий);
   - проверять соответствие бюджета и реальных расходов на ПО (это позволит более эффективно использовать имеющиеся средства);
   - повторно использовать ПО, передавая его в другие отделы (предварительно проверив, допускают ли это условия лицензионных соглашений).

Контрольные вопросы по главе 3

1) Дайте понятие рынка программных продуктов, товара и услуги, перечислите условия существования рынка.
2) Перечислите и прокомментируйте определения компьютерных программ как товара на рынке.
3) Отметьте роль государства при регулировании рынка ПП.
4) Основные проблемы разработчика на рынке ПП.
5) Основные проблемы потребителей на рынке ПП.
6) Перечислите и прокомментируйте основные виды рынков ПП.
7) Дайте понятия тиражного и заказного программного обеспечения.
8) Приведите классификацию прикладных программных продуктов.
9) Дайте понятие и перечислите основные задачи интернет-маркетинга.
10) Раскройте содержание поисковой оптимизации при продвижении ПП.
11) Раскройте содержание контекстной рекламы при продвижении ПП.
12) Раскройте содержание медийной рекламы при продвижении ПП.
13) Раскройте содержание оптимизации в социальных медиа.
14) Основные проблемы ценообразования на ПП.
15) Перечислите основные типы рынков с позиции ценообразования и раскройте их содержание.
16) Перечислите возможные ценовые политики и раскройте их содержание.
17) Раскройте содержание метода ценообразования, ориентированного на затраты.
18) Раскройте содержание метода ценообразования, ориентированного на спрос.
19) Раскройте содержание метода ценообразования, ориентированного на конкурентов.

20) Перечислите и прокомментируйте основные факторы, влияющие на договорную цену.
21) Определите, к каким классическим моделям лицензирования относятся модели лицензирования на фирме Microsoft.
22) Что дает компании грамотный подход к управлению лицензиями?

ЗАКЛЮЧЕНИЕ

В представленном учебном пособии изложены лишь основные (фундаментальные) основы программной инженерии, позволяющие студенту в будущем понять место и назначение специальных дисциплин учебного плана направления подготовки «Программная инженерия». Научные направления программной инженерии включают в себя несколько больших разделов. Это прежде всего:

- Методы и инструментальные средства проектирования ПП;
- Инструментальные средства и технологии разработки программ;
- Языки и системы программирования, качество, верификация и валидация ПП;
- Методы, средства и системы управления базами данных и знаний;
- Человеко-машинные интерфейсы, средства визуализации, системы виртуальной реальности и мультимедийного взаимодействия;
- Методы и средства обеспечения информационной и функциональной безопасности ПП;
- Нормативно-правовые, организационные и экономические механизмы программной инженерии;
- Управление программными проектами, риски и командообразование;
- Маркетинг и продвижение программных продуктов на потребительские и промышленные рынки;
- Управление жизненным циклом создания, внедрения и сопровождения ПП.

В той или иной степени эти вопросы найдут отражение в специальных дисциплинах, учебно-исследовательской работе студента при выполнении им курсовых и дипломных проектов.

ЛИТЕРАТУРА

[1] Введение в программную инженерию и управление жизненным циклом программного обеспечения = Guide to Software Engineering Base of Knowledge (SWEBOK): пер. с англ. С. Орлик [Электронный ресурс]. — Режим доступа: http://sorlik.blogspot.com/

[2] Рекомендации по преподаванию программной инженерии и информатики в университетах = Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering; Computing Curricula 2001: Computer Science: пер. с англ. — М.: ИНТУИТ.РУ «Интернет-Университет Информационных Технологий», 2007. — 462.

[3] Лаврищева Е. М., Методы и средства инженерии программного обеспечения [Электронный ресурс] : учебник / Е. А. Лаврищева, В. А. Петрухин. — М.: МФТИ (ГУ), 2006. — Режим доступа: http://www.arkhipenkov.ru/resources/sw_team_management.pdf

[4] Силич В. А. Моделирование и анализ бизнес-процессов: учеб. пособие / В. А. Силич, М. П. Силич. — Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 2010. — 212 с.

[5] Методология IDEF0. Стандарт. Русская версия. — М.: Метатехнология, 1993. — 107 с.

[6] Соловьев Д. А. Проектирование автоматизированных систем обработки информации и управления : учеб. пособие / Д. А. Соловьев. — Томск: ТМЦДО, 2007. — Часть 1: Унифицированный язык моделирования. — 170 с.

[7] Мытник С. А. Проектирование информационных систем : учеб. пособие / С. А. Мытник. — Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 2008. — 100 с.

[8] Руководство к своду знаний по программной инженерии. The Guide to the Software Engineering Body of Knowledge [Электронный ресурс]. — SWEBOK. IEEE Computer Society Professional Practices Committee, 2004. — Режим доступа: http://swebok.sorlik.ru/pdf/software_engineering.pdf

[9] ГОСТ Р ИСО/МЭК 12207-99. Информационная технология. Процессы жизненного цикла программных средств [Электронный ресурс]. — Режим доступа: http://vsegost.com/Catalog/38/38119.shtml

[10] ГОСТ 28806-90. Качество программных средств. Термины и определения [Электронный ресурс]. — Режим доступа: http://vsegost.com/Catalog/10/10605.shtml

[11] Архипенков С. Я. Руководство командой разработчиков программного обеспечения. Прикладные мысли [Электронный ресурс] / С. Я. Архипенков. М., 2008 — 79с. — Режим доступа: http://www.arkhipenkov.ru/resources/sw_project_management.pdf

[12] Архипенков С. Я. Лекции по управлению программными проектами [Электронный ресурс] / С. Я. Архипенков. — М., 2009. — Режим доступа: http://www.arkhipenkov.ru/resources/sw_project_management.pdf

[13] Новиков Ф. А. Учебно-методическое пособие по дисциплине «Управление проектами и разработкой ПО» / Ф. А. Новиков, Э. А. Опалева, Е. О. Степанов. — СПб.: СПбГУ ИТМО, 2008. — 256 с.

[14] Microsoft Solutions Framework. Дисциплина управления рисками MSF, вер. 1.1, 2009, [Электронный ресурс]. — Режим доступа: http://vernikov.ru/informacionnye-tehnologii/item/287-disciplina-upravlenija-riskami-msf.html

[15] Смольянинов А. Некоторые секреты командной разработки / А. Смольянинов, А. Ложечкин // Открытые системы. — 2005. — №7, 8. — С. 49– 56.

[16] Шив Ч. Д. Курс MBA по маркетингу / Ч. Д. Шив, А. Хайем . — М.: Альпина Бизнес Букс, 2007. — 718 с.

[17] Харатишвили Д. Рынок прикладного ПО и его сегментация [Электронный ресурс] // КомпьютерПресс. — 2008. — №1. — Режим доступа: http://www.compress.ru/article.aspx?id=18558&iid=858

[18] Успенский И. В. Интернет-маркетинг : учебник / И. В. Успенмкий. — СПб.: Изд-во СПГУЭиФ, 2003.

[19] Ашманов И. Оптимизация и продвижение сайтов в поисковых системах (+CD) / И. Ашманов, А. Иванов. — СпбЖ : Питер, 2008 — 400 с.

[20] Котлер Ф. Основы маркетинга / Ф. Котлер. — М.: Ростинтэр, 1996. — 704 с.

[21] Гайдаманчук Р. Управление лицензиями на программное обеспечение // Корпоративные системы. — 2004. — №2. — С. 38–40.

[22] Ферранте Дэниел. Модели лицензирования программ: что дальше? // Открытые системы. — 2007. — №1. — С. 40–45.

ГЛОССАРИЙ

Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов, и таким образом четко определяет его концептуальные границы с точки зрения наблюдателя.

Анализ требований — отображения функций системы и ее ограничений в модели предметной области.

Артефакт — любой продукт деятельности специалистов по разработке ПО.

Архитектура программной системы — определение системы в терминах подсистем и интерфейсов между ними, отображающая правила декомпозиции проблемы.

Ассоциация — наиболее общее и существенное отношение, которое утверждает наличие связи между понятиями без уточнения их содержания и размеров.

Бизнес-процесс — множество внутренних упорядоченных видов деятельности организации по преобразованию исходных ресурсов в готовую продукцию (услугу).

Валидация — проверка соответствия разработки программной системы требованиям заказчика.

Верификация — проверка правильности реализации системы заданным требованиям на каждом этапе жизненного цикла.

Водопадная (каскадная) модель — схема работ, в которой каждая из работ выполняется один раз и в том порядке, который указан в модели жизненного цикла.

Гарантия качества программного обеспечения — действия на каждом этапе жизненного цикла по проверке и подтверждению достигаемого качества соответственно стандартам и процедурам.

Дефект — это ошибочное событие в результате неверного описания спецификаций требований, исходных или проектных спецификаций, документации и т. п.

Диаграмма — графическое представление моделирования системы с помощью классов, сценариев, состояний и т. п.

Динамическое тестирование — выполнение программ для обнаружения ошибок, установления их причины и устранения.

Жизненный цикл системы (ЖЦ) — непрерывный процесс, который начинается с момента принятия решения о необходимости ее создания и заканчивается в момент ее полного изъятия из эксплуатации.

Иерархия — это упорядочение абстракций, расположение их по уровням.

Инженерия — применение научных результатов и дисциплины управления программированием задач в целях получения пользы от свойств продуктов, способов взаимосвязи и выполнения.

Инженерия качества — процесс управления предоставлением продуктам программного обеспечения свойств качества (надежность, сопровождаемость и т. п.).

Инженерия требований — сбор, анализ, оформление условий и ограничений на разработку системы в виде спецификации, согласованной как заказчиком, так и исполнителем.

Инкапсуляция — это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение; инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации.

Инспекция кода — формальная проверка описания используемых типов и структур данных в проекте системы на их соответствие требованиям.

Информационное обеспечение — набор средств для предоставления информации пользователям о содержании и условиях ее применения.

Интерфейсные объекты — связка (стыковка) программ, представленная в виде описания передаваемых через сообщения параметров для выполнения.

Качество программного обеспечения — совокупность свойств, которое определяют пригодность программного обеспечения удовлетворить заказчика в соответствии его требований к разработке.

Компонент — тип, класс, проектное решение, документация или иной продукт программной инженерии приспособленный для практического использования.

Компонентная разработка — конструирование программного обеспечения путем композиции готовых компонентов, сохраняемых в каталогах.

Конечные пользователи системы — профессиональные лица, для потребностей которых заказывается компьютерная система.

Конфигурация — вариант (версия) изготовленной программной системы из отдельных экземпляров компонентов и подсистем.

Критерий — количественная или качественная характеристика состояния системы, позволяющая оценить степень достижения цели и сформулировать решающие правила выбора средств (способов, технологий) достижения цели.

Критерий эффективности — критерий, позволяющий оценить степень достижения цели с учетом произведенных затрат различных ресурсов.

Менеджмент — профессиональное управление коллективами работников (персоналом).

Метод белого ящика — исследование внутренний структуры программы в целях выявления ошибок путем исчерпывающего тестирования всех путей и потоков передач управления.

Метод черного ящика — тестирование реализованных функций путем проверки соответствия реального поведения функций с ожидаемым поведением исходя из спецификаций требований.

Метрика — количественная мера и шкалы измерения характеристик программы.

Модель жизненного цикла — типичная схема последовательности работ на процессах разработки программного продукта.

Модель процессов — определенная последовательность действий, сопровождающая изменение состояния программных объектов.

Модульность — это свойство системы, которая была разложена на внутренне связные, но слабо связанные между собой модули.

Надежность программной системы — это способность системы сохранять свои свойства (безотказность, устойчивость и др.) в процессе преобразования исходных данных в результаты в течение определенного промежутка времени при определенных условиях эксплуатации.

Нефункциональные требования — требования, которое характеризуют организационные, исполнительские, операционные аспекты работы программной системы в среде реализации.

Отладка — проверка программы на наличие в ней ошибок и их устранение без внесения новых.

Отказ — переход программы из работающего состояния в неработающее в связи с ошибками или дефектами в ней.

Ошибка — недостатки в операторах программы или в технологическом процессе ее разработки, которые приводят к неправильной интерпретации исходной информации и к неверному решению.

Объектно-ориентированная модель — структура из совокупности объектов, которые взаимодействуют между собою, обладают свойствами и поведением.

Онтология — совокупность элементарных понятий, терминологии и парадигмы их интерпретации в среде проблемы, которую требуется разработать.

Оценивание качества — действия, направленные на определение степени удовлетворения программного обеспечения требованиям, соответствующим его предназначению.

Пакет — программная структура с общим механизмом организации элементов (объектов, классов) в группы, начиная от системы (стереотип «система») и к ее подсистемам различного уровня детализации.

Параллелизм — это свойство, отличающее активные объекты от пассивных.

Переносимость системы — возможность изменять сервис системы (ОС, связи, сетевые коммуникации, данные СУБД и т. п.) путем настройки модулей на новые условия среды или платформы.

План тестирования — описание стратегии, ресурсов и график тестирования отдельных компонентов и системы в целом.

Повторное использование — использование в качестве готовой порции любых формализованных знаний, полученных при реализации программных систем.

Повторно используемый компонент (ПИК) — фрагмент знаний о минувшем опыте программирования системы, представленный так, чтобы его можно было использовать не только его разработчиками, но и пользователями после соответствующей адаптации к новой среде.

Предметная область представляет собой совокупность бизнес-процессов организации, адекватно описывающих деятельность организации по удовлетворению общества в определенных продуктах и услугах.

Прикладная система — продукт программной инженерии, предназначенный для выполнения конкретных задач конечного пользователя.

Принципы — базовые концепции, лежащие в основе всей области программирования.

Приложение — область применения, в которых принципы и практика находят свое наилучшее выражение.

Программная инженерия — система методов, средств и дисциплины планирования, разработки, эксплуатации и сопровождения программного обеспечения, способного к массовому воспроизводству.

Процесс разработки — действия разработчика по инженерии требований, проектированию, кодированию и тестированию программного продукта.

Процесс сдачи — действия по передаче разработанного продукта покупателю.

Процесс эксплуатации — действия по обслуживанию системы пользователем.

Процесс сопровождения — действия по управлению модификациями и поддержкой системы в актуальном состоянии при выполнении функций системы или изъятие системы из употребления.

Проектирование — преобразования требований в последовательность проектных решений и их в архитектуру из программных компонентов.

Проектирование концептуальное — уточнение понимания и согласование деталей требований к системе.

Проектирование архитектурное — определение структурных особенностей строящейся системы.

Проектирование техническое — отображение требований среды функционирования и разработки системы путем определения всех конструктивных элементов и их композиций.

Проектирование детальное — определение подробностей реализации функций для заданной среды и связей между соответствующими компонентами системы.

Реализация программной системы — преобразования проектных решений в работающую систему (синонимы: кодирование, конструирование).

Сертификация программного продукта — процесс для установления соответствия программной продукции (процесса или услуг) конкретному стандарту или техническим условиям со специальным знаком или свидетельством.

Спецификация — описание алгоритма, правил, ограничений действий объектов с учетом стандартов, критериев качества и др.

Спиральная модель ЖЦ — модель процессов в жизненном цикле разработки системы, позволяющей возвращаться к любому предыдущему процессу с целью переработки элементов сделанного продукта.

Средства проектирования — создание моделей программного продукта на основе моделей предметной области (например, IBM Rational Rose, Sybase Power Designer).

Событие — явление, которое провоцирует смену определенного состояния и переход к другому состоянию в системе.

Состояние (системы, объекта и тому подобных) — фиксация определенных свойств на определенный момент или интервал времени.

Сохраняемость — способность объекта существовать во времени, переживая породивший его процесс, и (или) в пространстве, перемещаясь из своего первоначального адресного пространства.

Статическое тестирование — анализ и рассмотрение спецификаций компонентов на правильность представления без их выполнения на компьютере.

Стереотип — указатель элемента моделирования UML.

Сопровождение — работы по внесению изменений в программную систему после того, как она передана пользователю для эксплуатации.

Структура системы — множество элементов и отношений между ними.

Субъект (экземпляр) — кто-то или что-то, вне системы, что взаимодействует с системой.

Сущность (Entity) — реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению.

Сценарий — конкретная последовательность действий, которая иллюстрирует поведение и выполнение экземпляра прецедента.

Тест — некоторая программа, предназначенная для проверки правильности ее работы и выявления в ней ошибочных ситуаций.

Тестовые данные — данные, которые готовятся на основе документов программы или спецификаций для проверки работы программной системы.

Тестирование — способ семантической отладки (проверки) программы, который состоит в выполнении последовательности различных контрольных наборов тестов и сверке с известным результатом.

Технология разработки программного обеспечения — это упорядоченная совокупность взаимосвязанных этапов создания программного продукта и набор инструментальных средств их реализации.

Типизация — это способ защититься от использования объектов одного класса вместо другого или, по крайней мере, управлять таким использованием.

Требование — соглашение или договор между заказчиком и исполнителем системы относительно ее работы.

Унаследованная система — существующая действующая система, созданная любыми методами и технологиями для поддержки некоторых процессов бизнеса.

Управление качеством — комплекс способов и системной деятельности по планированию, управлению и оценке качества программного обеспечения.

Функция — содержание действий, выполнение которых возлагается на элемент системы при заданных требованиях, условиях и ограничениях.

Функциональные требования — требования, которые определяют цели и функции системы и принципы их выполнения на компьютере.

Функциональная полнота — атрибут, показывающий степень достаточности основных функций для решения специальных задач соответственно назначению программного обеспечения.

Функциональная структура — структура, элементами которой являются функции, реализуемые подразделениями предприятия, а отношениями — связи, обеспечивающие передачу предметов труда.

Характеристики качества — функциональность, надежность, удобство, эффективность, сопровождаемость, переносимость и тому подобное.

Эксплуатация — действия по выполнению готовой программной системы.

UML (Unified Modeling Language) — диаграммный способ (язык) для спецификации, визуализации, конструирования и документирования продуктов на процессах ЖЦ.

Учебное издание

Ехлаков Юрий Поликарпович

ВВЕДЕНИЕ В ПРОГРАММНУЮ ИНЖЕНЕРИЮ

Учебное пособие

Корректор Осипова Е. А.
Компьютерная верстка Хомич С. Л.

Подписано в печать 17.10.11. Формат 60x84/8.
Усл. печ. л. 17,21. Тираж 300 экз. Заказ

Издано в ООО «Эль Контент»
634029, г. Томск, ул. Кузнецова д. 11 оф. 17
Отпечатано в Томском государственном университете
систем управления и радиоэлектроники.
634050, г. Томск, пр. Ленина, 40
Тел. (3822) 533018.