Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Ю.П. Ехлаков

УПРАВЛЕНИЕ ПРОГРАММНЫМИ ПРОЕКТАМИ

Учебник

Томск
Издательство ТУСУРа
2015

УДК 004.413.2(075.8)
ББК 32.973.2-018я73
Е934

Рецензенты:

Силич В.А., д-р техн. наук, профессор,
зав. кафедрой оптимизации систем управления
Национального исследовательского Томского политехнического университета

С.П. Сущенко, д-р техн. наук, профессор,
заведующий кафедрой прикладной информатики
Национального исследовательского Томского государственного университета

Ехлаков, Юрий Поликарпович.

Е934 Управление программными проектами: учебник / Ю.П. Ехлаков. – Томск : Изд-во Томск. гос. ун-та систем управления и радиоэлектроники, 2015. – 216 с.

ISBN 978-5-86889-723-8

Управление программными проектами рассматривается как специфический вид деятельности при создании программных продуктов. Последовательно раскрываются вопросы инициации программного проекта, управления содержанием и сроками, командообразования, управления стоимостью и рисками. Содержание разделов учебника основано на положениях отечественных и зарубежных стандартов на процессы жизненного цикла по разработке программных продуктов.

Для студентов, обучающихся по направлениям подготовки «Программная инженерия» и «Бизнес-информатика», а также менеджеров малых IT-компаний, работающих на рынке прикладных программных продуктов.

УДК 004.413.2(075.8)
ББК 32.973.2-018я73

ISBN 978-5-86889-723-8

Введение

Для коллектива разработчиков, планирующего вывод на рынок тиражных программных продуктов (ПП), вопросы управления программными проектами на каждой фазе жизненного цикла (ЖЦ) программного продукта являются ключевыми. Как отмечается в [1], только 35 % проектов завершаются в срок, не превышают запланированного бюджета и реализуют все требуемые функции и возможности; 46 % проектов завершаются с опозданием, расходы превышают запланированный бюджет, требуемые функции не реализуются в полном объеме; 19 % проектов оказываются полностью неуспешными и не доводятся до завершения. Это связано с особенностями программного продукта как рыночного товара, который, являясь результатом творческого труда, не поддается точному оцениванию ни по времени создания, ни по требуемому бюджету.

Содержание учебника основывается на обобщении и развитии положений теории и практики управления проектами, представленных в монографии Роберта Т. Фартрелла, Дональда Ф. Шафера и Линды И. Шафер, лекциях С.Я. Архипенкова, а также материалах отечественных и зарубежных стандартов на процессы жизненного цикла программных продуктов: ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств»; IEEE-1074-1997 «Процессы и действия жизненного цикла программного обеспечения» (Developing software life cycle processes); Руководства к своду знаний по управлению проектами (Project Management Body of Knowledge — ANSI PMBOK 5-th Edition).

Автором настоящего учебника предложены модели выбора и оценки перспективности концепции программного проекта, приведена математическая постановка задачи оптимального планирования проектных работ, описана многовариантная процедура формирования календарного плана реализации проекта, определены

риски, свойственные программным проектам, предложено множество рискообразующих факторов, влияющих на проявление рисков.

Содержание учебника направлено на формирование у студентов профессиональных знаний и практических навыков по разработке концепции программного проекта и оценке ее перспективности, структурной декомпозиции работ программного проекта; календарному планированию проектных работ на базе математических моделей теории расписаний, созданию проектной команды разработчиков; управлению стоимостью и рисками при реализации программного проекта.

В первом разделе учебника приводится понятие и специфические особенности программного продукта как результата деятельности команды разработчиков и программного проекта как методологии управления процессами создания программного продукта. Рассматриваются специфические свойства рыночного программного продукта, обуславливающие наличие особенностей в процессах управления программными проектами.

Формулируются возможные варианты целей, ограничений и допущений, возникающих при реализации программного проекта. Раскрывается содержание десяти областей знаний по управлению проектами и пяти групп процессов (этапов) ЖЦ проекта, изложенных в международном стандарте PMBOK. Приводится краткое описание процессов (действий и задач) по управлению программными проектами, представленных в ГОСТ Р ИСО/МЭК 12207-2010.

Второй раздел посвящен рассмотрению материалов международного и отечественных стандартов, регламентирующих деятельность команды разработчиков по созданию ПП. Приводится сжатое описание наиболее актуальных стандартов: IEEE-1074-1997 «Процессы и действия жизненного цикла программного обеспечения»; ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Процессы жизненного цикла программных средств»; «Единая система программной документации (ЕСПД): ГОСТ 19.102-77 ЕСПД «Стадии разработки»».

Третий раздел учебника направлен на изучение моделей жизненного цикла разработки программного продукта. Здесь опи-

сываются содержание, достоинства, недостатки и области применения следующих моделей: каскадной, v-образной, инкрементной, спиральной, моделей прототипирования и быстрой разработки приложений. Приводится методика выбора модели ЖЦ разработки ПП, обусловленного особенностями выявления и анализа требований к программному продукту и составу и квалификации команды разработчиков, а также степенью участия коллектива пользователей в реализации программного проекта и сложностью проекта.

Четвертый раздел учебника посвящен описанию действий разработчиков на этапе инициации программного проекта, когда формируется творческое ядро команды по созданию программного продукта, разрабатываются и оцениваются несколько вариантов привлекательных идей и концепция проекта. Описание концепции предлагается проводить в следующем формате: необходимость и/или потребность в программном продукте; цели, ограничения, допущения и основные результаты; основные сегменты рынка и потенциальные пользователи; экономика программного проекта; потенциал команды исполнителей; предварительная оценка рисков. Для оценки перспективности концепции предлагается использовать метод экспертных оценок и гибридную модель функциональных зависимостей, при этом в качестве критериев оценки учитывать совокупность потребностей потенциальных потребителей и рыночные условия.

В пятом разделе последовательно раскрываются вопросы управления содержанием и сроками реализации программного проекта: описываются основные этапы управления проектом; приводится процедура структурной декомпозиции проекта (определение множества работ, которые необходимо выполнить для получения результатов) на основе моделей ЖЦ разработки ПП и материалов международных и отечественных стандартов на ЖЦ ПП; описываются представления множества работ проекта в виде сетевой модели; предлагается математическая постановка задачи оптимального планирования проектных работ и многовариантная процедура формирования календарного плана реализации проекта; описывается алгоритм перераспределения ресурсов.

В шестом разделе рассматривается проблема командообразования при реализации программных проектов как с точки зрения формирования команды, так и создания условий для ее эффективной работы. Описываются основные модели управления командой программного проекта, функциональные роли участников, профессиональные и психологические особенности программиста, роль и задачи руководителя проекта, основы мотивации сотрудников, материальные и моральные стимулы к труду.

В седьмом разделе учебника, посвященном вопросам управления стоимостью программного проекта, описывается последовательность и содержание процесса оценки плановой стоимости проекта с расшифровкой отдельных статей затрат, представлен материал по формированию и исполнению бюджета проекта, дается описание показателей, характеризующих исполнение бюджета и соблюдение календарного плана работ.

В восьмом разделе учебника последовательно рассмотрены вопросы управления рисками при реализации программного проекта: определены риски, свойственные программным проектам; предложено множество рискообразующих факторов, обуславливающих проявление рисков; описаны процедуры идентификации рисков и рискообразующих факторов, проведения качественного и количественного анализа влияния факторов, определения уровня негативных последствий (ущерба, убытков, потерь) от проявления рискообразующих факторов, принятия одного из возможных вариантов решения по реагированию на риски (принятие, уклонение, передача, снижение).

Данный учебник разработан в ходе выполнения государственного задания Министерства образования и науки Российской Федерации № 3653 «Модели, алгоритмы и программное обеспечение поддержки принятия решений по управлению рисками в социально-экономических и производственно-технологических системах».

1. ОСОБЕННОСТИ ПРОЦЕССА УПРАВЛЕНИЯ ПРОГРАММНЫМ ПРОЕКТОМ

1.1. Основные понятия и определения

1.1.1. Программный проект

Классическая теория управления проектами основана на двух базовых понятиях: проекта как совокупности работ по получению уникального результата, оформленных в виде плана действий, и продукта как конечного результата реализации проекта. Остановимся на этих понятиях более подробно.

В литературе приводятся различные определения проекта:

1) ***проект*** — временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов [1, 2];

2) ***проект*** — комплекс уникальных действий, не опирающийся на организационную структуру, имеющий определенные дату начала и окончания, расписание, стоимость и технические задачи [3];

3) ***проект*** — комплекс взаимосвязанных действий, предпринимаемых с целью получения уникальных конкретных результатов при заданных ограничениях по времени, денежным средствам, ресурсам и качеству конечных результатов проекта [4];

4) ***проект*** — произвольный ряд действий или задач, имеющих определенную цель, которая будет достигнута в рамках выполнения некоторых заданий, характеризующихся определенными датами начала и окончания, пределами финансирования и ресурсами [5];

5) ***проект*** — временное усилие, применяемое для того, чтобы создать уникальный продукт или услугу с определенной датой начала и окончания действия, отличающегося от продолжающихся, повторных действий и требующего прогрессивного совершенствования характеристик [5].

Анализ представленных определений позволяет выделить ***специфические характеристики проекта***:

- направленность проекта на достижение конкретного конечного результата, определяемого в терминах требуемых ресурсов, качества и времени реализации;
- уникальность проекта как разового (неповторяющегося) мероприятия, требующего специфической организации управления;
- ограниченность проекта по времени и ресурсам (финансовым, трудовым, материальным) и, как следствие, необходимость нахождения постоянного компромисса между объемом работ, ресурсами, временем, качеством и рисками и их перераспределения в ходе выполнения проекта;
- структурная сложность проекта как комплекса тесно взаимосвязанных мероприятий и его высокая неопределенность, обусловленная возможными изменениями условий реализации, потребности в тех или иных видах ресурсов.

Обобщая вышеизложенное, **определение программного проекта** можно привести в следующей формулировке: это комплекс взаимосвязанных работ, выполняемых командой проекта с целью получения уникального программного продукта или услуги в течение заданного периода при установленном бюджете и потребляемых в ходе реализации проекта ресурсах в условиях повышенного риска, требующих специфического управления.

1.1.2. Программный продукт

Конечным результатом реализации программного проекта является **программный продукт,** который можно определить как совокупность записанных на носителях данных программных компонентов, являющихся продуктом промышленного производства, предназначенных для поставки, передачи или продажи пользователю, снабженных технической документацией, рекламными материалами, инструкциями по обучению пользователей, гарантийными обязательствами по сопровождению и обслуживанию.

Программный продукт обладает множеством специфических особенностей, которые можно разделить на две группы:

1) ***характеристики ПП как объекта промышленного производства, предназначенного для продажи***:

- ПП как товар представляет собой публикацию текста программы/программ на языке программирования или в виде исполняемого кода, зафиксированного на материальном носителе (компьютере, дисковом накопителе и др.), который может быть продан или передан, при этом обладание материальным носителем не делает его владельца уникальным собственником;
- создание ПП связано с постоянными изменениями функционала, сроков разработки и затрат, обусловленных отсутствием у потенциальных потребителей четко сформулированных требований к продукту и слабым представлением о технологии его использования в практической деятельности;
- необходимость адаптации стандартов на процессы ЖЦ программного продукта к конкретным условиям ввиду того, что в существующих документах по регламентации данного вида деятельности процессы ЖЦ разработки ПП описаны в общем виде и прямо не ориентированы на специфику создаваемого продукта;
- в структуре стоимости ПП относительно невысоки затраты на его изготовление (тиражирование), что обусловлено низкой стоимостью производственных операций по созданию копий и высокой стоимостью разработки ПП, в которой основную часть составляют затраты на оплату труда относительно небольшой группы специалистов;
- ПП создается в условиях повышенного риска, поскольку, являясь результатом творческого труда, не поддается точному оцениванию ни по времени создания, ни по требуемому бюджету;
- вовлечение ПП в хозяйственный оборот происходит в процессе его коммерциализации (купли-продажи, переуступки прав собственности) и капитализации (постановки на баланс, инвестирования в уставной капитал);

2) ***характеристики ПП как объекта интеллектуальной собственности***:

- нематериальная природа существования ПП;
- ПП может обмениваться, но при этом не происходит его полное отчуждение;
- ПП может быть неоднократно продан, при этом одновременно выступать объектом нескольких рыночных сделок;
- не исчезает и не изнашивается в процессе использования.

Множество мероприятий (работ) по созданию программного продукта и их взаимосвязей рассматривается как **жизненный цикл ПП,** представляющий собой последовательность различных видов деятельности разработчиков, охватывающей все этапы эволюционного изменения ПП — от установления требований к ПП до полного прекращения его эксплуатации. Виды деятельности по созданию ПП подробно описываются в соответствующих стандартах на процессы жизненного цикла:

1) комплекс государственных стандартов «Единая система программной документации» (ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов. ГОСТ 19.102-77 ЕСПД. Стадии разработки);

2) ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств;

3) IEEE-1074-1997. Процессы и действия жизненного цикла программного обеспечения.

Из представленных определений программного проекта и программного продукта и их специфических особенностей следует, что **программный проект** — это достаточно сложный вид деятельности, которым следует постоянно управлять в силу его уникальности, высокой неопределенности реализации, ограниченности по ресурсам и времени.

1.1.3. Управление программным проектом

Управление программным проектом будем рассматривать как деятельность по управлению всеми этапами ЖЦ программного продукта путем планирования, реализации, монито-

ринга и контроля проектных работ, в ходе которых достигаются цели проекта при нахождении компромисса между объемом работ, ресурсами, временем, качеством и рисками.

Цели программного проекта следует определять в виде желаемого результата, достигаемого командой проекта, при его успешной реализации. При формулировании целей проекта необходимо обеспечить:

конкретность — четкость и однозначность понимания результатов;

измеримость — возможность оценивания степени достижения результата, выраженного как в качественной, так и в количественной форме;

реальность — соизмерение возможностей команды проекта с факторами влияния внешней среды;

согласованность — единство мнений всех участников проекта в понимании конечного результата и координация действий по его достижению;

ограниченность по срокам — установление интервалов времени, по истечении которых необходимо оценивать промежуточные результаты проекта и определять степень достижения конечного результата.

Четкое определение бизнес-целей достаточно важно, поскольку существенно влияет на все процессы и решения в проекте. Проект должен быть закрыт, если обнаружится, что достижение цели невозможно или стало нецелесообразным (например, если реальные затраты на проект будут превосходить будущие доходы от его реализации).

При описании желаемого результата программного проекта необходимо определять:

- конкретные бизнес-выгоды, которые получит заказчик по завершении проекта;
- функционал продукта (услуги), получаемый по окончании проекта;
- краткое описание и при необходимости ключевые свойства и/или характеристики функционала.

Формулирование желаемого результата программного проекта должно отражать конкретные бизнес-цели с учетом ***правила «железного треугольника»*** (рис. 1.1) [1, 5]. Ни один из углов треугольника не может быть изменен без изменения других: например, чтобы уменьшить время, потребуется увеличить бюджет и/или сократить содержание.

С учетом наличия рыночной конкуренции к трем основным характеристикам «железного треугольника» следует добавить четвертую — *приемлемое качество*, которое определяется в виде совокупности нефункциональных требований к ПП.

В этом случае желаемый результат программного проекта можно сформулировать следующим образом: *проект должен быть реализован в нормативные сроки без превышения планового бюджета с заданными заказчиком функциональными и нефункциональными требованиями*.

Рис. 1.1. Характеристики желаемого результата программного проекта по правилу «железного треугольника»

Наличие большой неопределенности в достижении конечных целей проекта объективно требует проведения мероприятий по выявлению и оценке возможных рисков. Под **риском программного проекта** будем понимать наступление события, которое может возникнуть в процессе реализации программного проекта и негативно повлиять на степень достижения его целей.

Ограничения и **допущения** как неотъемлемая часть проекта сокращают возможности проектной команды в выборе решений по его реализации.

В частности, ***ограничения*** могут содержать следующие требования: обязательную сертификация продукта, услуги; использование конкретной заданной программно-аппаратной платформы; специфические требования к защите информации.

Допущения — события или действия, принимаемые как абсолютная истина. Например, оценивая проект по схеме с фиксированной ценой, можно записать в допущении предположение о том, что стоимость лицензий на программное обеспечение (ПО), приобретаемое на стороне, не изменится до завершения проекта. Допущения проекта должны быть оформлены документально и заранее доведены до сведения заказчика.

Оптимальный вариант реализации программного проекта определяется как ***компромисс*** между сформулированными выше характеристиками результатов проекта. В общем случае поиск компромисса состоит в нахождении баланса, приемлемого для всех сторон, связанных с проектом:

1) заказчиков, которым нужна определенная функциональность в конкретные сроки при имеющемся бюджете;

2) исполнителей, которые обладают бюджетом, достаточным для реализации проекта в заданные сроки с расчетным уровнем трудоемкости проекта и соответствующим уровнем качества.

Проект считается успешным, если он выполнен в срок в соответствии со спецификациями функциональных и нефункциональных требований в пределах запланированного бюджета.

1.2. Этапы жизненного цикла программного проекта

Управление процессами разработки уникальных конечных продуктов и/или услуг происходит в рамках проектной деятельности организации. В качестве документов, регламентирующих проектную деятельность, могут быть использованы:

1) Руководство к своду знаний по управлению проектами (Project Management Body of Knowledge — PMBOK) [2];

2) ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств [6].

Стандарт PMBOK был разработан Институтом управления проектами (Project Management Institute — PMI) в 2000 году. Последняя версия — **The Guide[1] to the PMBOK 5-th Edition** — вышла в начале 2013-го года. Стандарт, первоначально принятый в качестве Национального стандарта Америки (ANS) Американским национальным институтом стандартов (ANSI), в настоящее время нашел широкое применение во многих странах в различных сферах деятельности.

Руководство содержит обобщенные принципы и подходы, используемые в области проектного менеджмента, формализованные и структурированные таким образом, чтобы их можно было использовать в большинстве проектов независимо от конкретного применения.

В стандарте PMBOK 5-th Edition описаны пять групп управленческих процессов, соответствующих этапам жизненного цикла проекта:

1) инициация,
2) планирование,
3) исполнение,
4) мониторинг и управление,
5) завершение.

Документ содержит описание десяти областей знаний, которые используются менеджером проекта при определении содержания соответствующих этапов жизненного цикла проекта:

1) управление интеграцией проекта (Project Integration Management);
2) управление содержанием проекта (Project Scope Management);
3) управление сроками проекта (Project Time Management);
4) управление стоимостью проекта (Project Cost Management);

[1] guide (*англ.*) — руководство, путеводитель, ориентир.

5) управление качеством проекта (Project Quality Management);

6) управление человеческими ресурсами проекта (Project Human Resource Management);

7) управление коммуникациями проекта (Project Communications Management);

8) управление рисками проекта (Project Risk Management);

9) управление закупками проекта (Project Procurement Management);

10) управление заинтересованными сторонами проекта (Project Stakeholder Management).

Содержание видов деятельности, необходимых для управления проектом, изложено в Руководстве в описании 47 процессов. Формальная структура описания каждого из процессов представлена в следующих терминах:

вход процесса — информация, данные, документы, технологии, необходимые для эффективного выполнения процесса;

инструменты и методы процесса — способы и технологии, с помощью которых входы процесса наиболее эффективно преобразуются в выходы;

выходы процесса — информация, данные и документы, которые являются результатами выполнения процесса.

Стандарт не обязывает менеджера применять все перечисленные инструменты и реализовывать все 47 описанных процессов. Возможность применения положений PMBOK к каждому проекту определяется индивидуально.

Все пять групп управленческих процессов присущи *любому проекту* и должны выполняться в одной и той же последовательности *независимо от области приложения или специфики жизненного цикла проекта*.

В рамках **процессов инициации** определяются:

- бизнес-потребности, ради удовлетворения которых разрабатывается проект;
- причины, по которым данный проект является лучшей альтернативой для удовлетворения потребностей;
- внутренние и внешние заинтересованные в результатах проекта стороны;

- цели и содержание проекта;
- команда исполнителей проекта;
- финансовые ресурсы.

Кроме того, разрабатывается несколько вариантов Концепции или Устава проекта. Целесообразность и реализуемость Концепции подтверждается в процессе оценки альтернатив.

Группа **процессов планирования** состоит из процессов, определяющих содержание и последовательность работ проекта, необходимых и достаточных для достижения целей проекта.

Под **работой** понимается элемент проекта, выполняемый в процессе его осуществления. Наименьшая самостоятельная единица, используемая для детализации деятельности по достижению поставленной цели и описания логики проекта.

Работа обычно имеет ожидаемую (плановую) продолжительность (длительность), предполагаемую (ожидаемую) стоимость и требуемые ресурсы. В процессах планирования определяются состав и содержание работ проекта и их взаимосвязи; оценивается трудоемкость каждой работы, типы и количество необходимых трудовых ресурсов; разрабатываются календарный план проекта, план распределения ресурсов, бюджетный план, план управления рисками.

В составе группы **процессов исполнения** рассматриваются вопросы организации работ по проекту: распределение обязанностей и ответственности; координация работы исполнителей и использования ресурсов; определение механизмов морального и материального стимулирования исполнителей.

Группа **процессов мониторинга и управления** обеспечивает контроль, анализ и регулирование хода и эффективности выполнения как проекта в целом, так и его отдельных работ; выявление проблем, требующих внесения изменений в план, и инициацию внесения этих изменений; разработку рекомендаций по применению предупреждающих действий в отношении возможных проблем.

Группа **процессов завершения** обеспечивает приемку-сдачу проекта заказчику; документирование и накопление знаний о положительных и отрицательных аспектах практического ис-

пользования результатов проекта; проведение анализа эффективности внедрения; внесение необходимых изменений в историю успеха компании-разработчика.

Каждая из десяти областей знаний включает соответствующие подразделы:

1) ***управление интеграцией проекта*** **—** разработка плана проекта, организация исполнение плана проекта, контроль изменений в проекте;

2) ***управление содержанием проекта*** — формальное принятие решения о начале проекта, планирование объема работ, декомпозиция всего объема работ на мелкие измеримые задачи, подтверждение объема работ, формальная проверка приемлемости результатов работы, изменение объема работ;

3) ***управление сроками проекта*** — определение состава работ и их взаимосвязей, оценка длительностей работ, составление расписания проекта;

4) ***управление стоимостью проекта*** **—** планирование и оценка стоимости ресурсов, необходимых для работ; разработка бюджета, распределение затрат по работам проекта; контроль стоимости и управление изменениями бюджета;

5) ***управление качеством проекта*** **—** определение стандартов качества и средств для их достижения; плановая регулярная оценка исполнения производственных процессов; мониторинг результатов проекта, определение их соответствия стандартам; выявление и устранение причин несоответствия качества;

6) ***управление человеческими ресурсами*** **—** идентификация, документирование и назначение проектных ролей участникам проекта и структуры их отчетности; подбор и получение необходимых для проекта трудовых ресурсов; распределение персонала по работам проекта; формирование планов повышения квалификации исполнителей проекта; стимулирование индивидуальной и командной производительности труда;

7) ***управление коммуникациями*** **—** определение потребностей участников проекта в информации, планирование информационных потоков; регулярное и своевременное обеспечение участников проекта необходимой информацией; сбор и распро-

странение отчетности о текущем состоянии проекта, достигнутом прогрессе и ожидаемых результатах; создание и утверждение отчетов, необходимых для формального завершения проекта или отдельных этапов проекта;

8) ***управление рисками проекта*** **—** разработку плана управления рисками, идентификация рисков, качественный и количественный анализ рисков, планирование реагирования на риски, мониторинг и контроль процессов управления рисками;

9) ***управление закупками проекта*** — определение продуктов и услуг, привлекаемых извне для выполнения проекта; документирование требований к продуктам и услугам от внешних поставщиков; получение предложений, выбор поставщиков; регулирование отношений с поставщиками; подтверждение по выполнению условий контрактов; разрешение споров;

10) ***управление заинтересованными сторонами проекта*** **—** определение состава и ролевых функций участников проекта (инициатора, инвестора, заказчика, руководителя, куратора, соисполнителей); формирование регламентов по взаимодействию участников при реализации этапов жизненного цикла проекта.

Вариант распределения работ по этапам жизненного цикла, рекомендованных стандартом [2], приведен в табл. 1.1.

В **ГОСТ Р ИСО/МЭК 12207-2010** «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств» приведены семь процессов по управлению программными проектами [6]. Каждый процесс представлен в стандарте в виде совокупности действий и задач, предназначенных для достижения одного или более результатов.

В стандарте описываются следующие процессы управления программным проектом:

1) планирование проекта;
2) оценка проекта и процесс управления;
3) менеджмент решений;
4) менеджмент рисков;
5) менеджмент конфигурации;
6) менеджмент информации;
7) процесс измерений.

Таблица 1.1

Распределение процессов управления проектом по областям знаний и этапам жизненного цикла

Область знаний	Процессы управления проектами				
	Инициация	Планирование	Исполнение	Мониторинг и управление	Завершение
1. Управление интеграцией проекта	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 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 работ 3.2. Определение взаимосвязей 3.3. Оценка ресурсов работ 3.4. Оценка длительности 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	

Окончание табл. 1.1

Область знаний	Процессы управления проектами				
	Инициация	Планирование	Исполнение	Мониторинг и управление	Завершение
5. Управление качеством проекта		5.1. Планирование качества	5.2. Процесс обеспечения качества	5.3. Процесс контроля качества	
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	

Ниже приведено краткое содержание каждого из семи процессов, при этом первые четыре процесса, имеющие непосредственное влияние на программный проект, рассмотрены более подробно.

1. Процесс планирования проекта

Основными видами деятельности по реализации процесса планирования проекта являются:

1) ***инициация проекта*** (оценка потребности в результатах проекта, определение цели и ограничений, оценка степени осуществляемости проекта с учетом наличия требуемых ресурсов и сроков завершения проекта);

2) ***планирование проекта*** (разработка календарного плана/графика проведения работ; определение потребности в ресурсах и графика их поставки; распределение задач по исполнителям; определение рисков срыва как отдельных работ, так и проекта в целом; планирование мероприятий по обеспечению качества проекта; определение затрат, связанных с выполнением проекта; выбор типа модели ЖЦ разработки программного продукта);

3) ***активизация проекта*** (определение и распределение полномочий членов проекта; оформление заявок на ресурсы, требуемые для выполнения проекта; организация исполнения проекта).

2. Оценка проекта и процесс управления

Основная цель процесса заключается в определении состояния проекта и гарантии того, что проект выполняется в соответствии с календарным планом в пределах выделенного бюджета и удовлетворяет техническим параметрам качества. В составе процесса оценки проекта и управления проектом выделены следующие виды деятельности:

1) ***мониторинг проекта*** (формирование и представление отчетов о ходе выполнения проекта как руководителем проекта, так и представителем заказчика);

2) ***управление проектом*** (анализ состояния проекта и принятие решений по ликвидации проблем, возникающих в ходе выполнения проекта; формирование отчетов о разрешении возникающих проблемных ситуаций);

3) ***оценка проекта*** (оценка проекта на соответствие выполнению установленных требований; оценка проекта на соответствие календарным планам по сроку завершения, выделенному бюджету);

4) ***завершение проекта*** (оценка соответствия результатов проекта критериям, указанным в контракте, либо установленным внутри организации; формирование отчетов по результатам завершения проекта).

3. Процесс менеджмента решений

Процесс менеджмента решений заключается в выборе из существующих вариантов наиболее предпочтительного направления проектных действий. В соответствии с принятым в организации регламентом в составе процесса выделяются следующие виды деятельности:

1) ***подготовка решений*** (определение условий и потребности (необходимости) в принятии решений; привлечение к принятию решений опытных экспертов; установление и распределение ответственности и полномочий при принятии решения);

2) ***принятие и анализ решений*** (документальное описание проблемных ситуаций и вариантов действий по их ликвидации; оценка последствий альтернативных действий, выбор наиболее предпочтительной стратегии принятого решения; оформление отчетов о результатах принятого решения; ведение историй о возникающих проблемах и принятых решениях по их ликвидации).

4. Процесс менеджмента рисков

Процесс менеджмента рисков заключается в постоянном определении, анализе, обработке, мониторинге и реагировании на риски программного проекта. В составе процесса выделены следующие виды деятельности:

1) ***планирование менеджмента рисков*** (разработка руководящих материалов, регламентирующих процесс менеджмента риска; распределение ролей и обязанностей, распределение ресурсов, оценка состояния, накопление знаний и опыта);

2) ***менеджмент календарного графика рисков*** (описание пороговых значений риска, определяющих условия, при которых риск может быть принят; мониторинг состояния риска по таким

параметрам, как вероятность наступления, последствия проявления порогового значения; рейтинг рисков; мероприятия по реагированию на риск);

3) ***анализ рисков*** (идентификация рисков по характеристикам, описанным в контексте менеджмента рисков; оценка вероятности и последствий проявления; оценка критичности риска относительно пороговых значений характеристик; определение и документирование стратегии управления);

4) ***обработка рисков*** (выбор и реализация одной из альтернативных стратегий управления (принятие, снижение, уклонение, передача) в зависимости от критичности риска);

5) ***мониторинг рисков*** (мониторинг и оценка состояния риска; анализ возможности проявления новых рисков и их потенциальных источников);

6) ***оценка процесса менеджмента риска*** (сбор и анализ информации об источниках и причинах проявления рисков и используемых стратегиях реагирования; пересмотр содержания менеджмента рисков в плане его результативности и эффективности).

5. Процесс менеджмента конфигурации

Процесс предназначен для определения и поддержки в актуальном состоянии целостности всех выходных результатов проекта или процесса и доступа к ним заинтересованных лиц.

6. Процесс менеджмента информации

Процесс менеджмента информации обеспечивает предоставление заинтересованным участникам команды проекта в соответствии с их полномочиями своевременной, полной, достоверной информации о ходе выполнения проекта на всех этапах его жизненного цикла.

7. Процесс измерений

Процесс измерений заключается в сборе, анализе информации и составлении и предоставлении отчетов участникам проекта как о состоянии процессов по разработке продукта, так и технических характеристиках самого продукта.

Контрольные вопросы

1. Дайте понятие программного проекта и перечислите его специфические особенности.

2. Приведите определение программного продукта. Перечислите свойства ПП как объекта интеллектуальной собственности.

3. Дайте понятия цели, результата и ограничений программного проекта. Перечислите и прокомментируйте требования к формулировке целей.

4. Раскройте смысл характеристик «железного треугольника» при управлении программными проектами. В чем состоит процедура достижения компромисса между характеристиками?

5. Приведите понятие жизненного цикла программного продукта и назовите стандарты, регламентирующие этапы ЖЦ.

6. Перечислите и прокомментируйте содержание девяти областей знаний стандарта PMBOK.

7. Перечислите и прокомментируйте содержание пяти этапов жизненного цикла программного проекта стандарта PMBOK.

8. Перечислите и прокомментируйте содержание процессов управления программным проектом стандарта ГОСТ Р ИСО/МЭК 12207-2010.

2. СТАНДАРТИЗАЦИЯ ПРОЦЕССОВ СОЗДАНИЯ ПРОГРАММНОГО ПРОДУКТА

2.1. ГОСТ Р ИСО/МЭК 12207-2010 «Процессы жизненного цикла программных средств»

Одним из основных документов, регламентирующих процессы жизненного цикла создания программного продукта, является ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств».

Основу стандарта составляют следующие базовые понятия [6]:

стадия — период в пределах ЖЦ ПП, который относится к описанию или реализации одного из конкретных состояний;

процесс — совокупность взаимосвязанных или взаимодействующих видов деятельности, преобразующих исходные данные о ПП либо его отдельных компонентах в выходные результаты;

деятельность — совокупность действий, используемых для получения конкретных выходных результатов;

задача — элементарное действие, предназначенное для достижения одного или более выходных результатов процесса, при выполнении которого можно назначить конкретного исполнителя и определить требуемые ресурсы. Задача формулируется в форме требования, рекомендации или допустимого действия, при этом используются глаголы: ***«должен»*** — для выражения условия, требующего проверки на соответствие чему-либо; ***«следует»*** — выражения рекомендации среди других возможностей; ***«может»*** — чтобы отразить направление допустимых действий;

модель жизненного цикла ПП — структура, состоящая из процессов, действий и задач, включающих разработку, эксплуатацию и сопровождение программного продукта, охватывающая

период существования ПП — от установления требований к ПП до полного прекращения его использования.

Каждый процесс в стандарте описывается в следующем виде: наименование, цель, перечень действий, задачи. Последовательность действий или задач в каждом процессе не является жесткой и определяется логическими связями между ними. Результатом выполнения процесса является готовый ПП либо его отдельные компоненты.

Стандарт содержит полный набор описания процессов ЖЦ ПП для некоторого типового проекта с максимально возможным составом процессов, действий и задач, которые используются при приобретении системы, содержащей программные средства, или отдельно поставляемого ПП; при оказании программной услуги; при поставке, разработке, эксплуатации и сопровождении ПП.

В стандарте различные виды деятельности, которые могут выполняться в течение жизненного цикла программных систем, сгруппированы в семь групп процессов:

1) **процессы соглашения:**
   - приобретение;
   - поставка;

2) **процессы организационного обеспечения проекта:**
   - менеджмент модели жизненного цикла;
   - менеджмент инфраструктуры;
   - менеджмент портфеля проектов;
   - менеджмент людских ресурсов;
   - менеджмент качества;

3) процессы проекта:
   - планирование проекта;
   - оценка проекта и процесс управления;
   - менеджмент решений;
   - менеджмент рисков;
   - менеджмент конфигурации;
   - менеджмент информации;
   - измерение;

4) технические процессы:

- определение требований правообладателей;
- анализ системных требований;
- проектирование архитектуры системы;
- реализация;
- комплексирование системы;
- квалификационное тестирование системы;
- инсталляция программных средств;
- поддержка приемки программных средств;
- функционирование программных средств;
- сопровождение программных средств;
- прекращение применения программных средств;

5) **процессы реализации программных средств:**

- реализация программных средств;
- анализ требований к программным средствам;
- проектирование архитектуры программных средств;
- детальное проектирование программных средств;
- конструирование программных средств;
- комплексирование программных средств;
- квалификационное тестирование программных средств;

6) процессы поддержки программных средств:

- менеджмент документации;
- менеджмент конфигурации;
- обеспечение гарантии качества;
- верификация;
- валидация;
- ревизия;
- аудит;
- решение проблем в программных средствах;

7) процессы повторного применения программных средств:

- проектирование доменов;
- менеджмент повторного применения активов;
- менеджмент повторного применения программ.

При решении практических задач необходимо адаптировать состав и содержание процессов стандарта к конкретному проекту. Рассмотрим пример использования стандарта для изложения содержания процессов, определяющих деятельность разработчиков при реализации следующих стадий жизненного цикла создания программных продуктов (рис. 2.1).

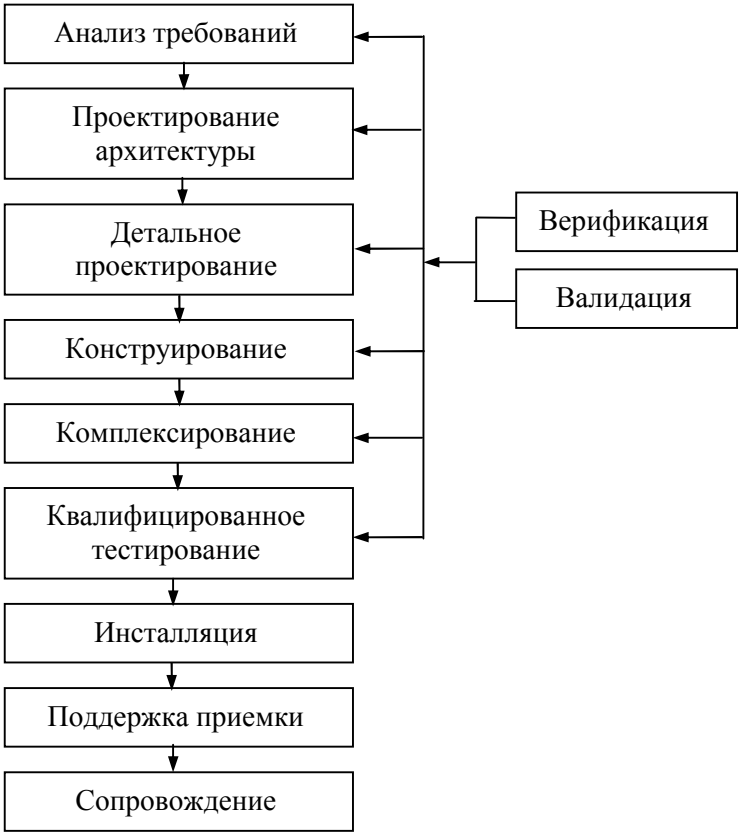

Рис. 2.1. Стадии жизненного цикла разработки программного продукта

Содержание этих процессов должно раскрывать последовательность создания программных средств, выполненных в виде программного продукта, удовлетворяющего как требованиям к архитектурным решениям проекта, что подтверждается посредством его верификации, так и требованиям заказчиков (потребителей), что подтверждается посредством валидации.

Процесс анализа требований к программным средствам заключается в установлении требований к программному продукту и его программным компонентам. При реализации процесса необходимо осуществлять следующие виды деятельности:

1) определение требований к программным компонентам и программному продукту и их интерфейсам;

2) анализ требований к программным компонентам и программному продукту на корректность и тестируемость;

3) определение влияния требований к программным компонентам и программному продукту на среду функционирования;

4) установление совместимости и взаимосвязи между требованиями к программным компонентам и требованиями к программному продукту;

5) определение приоритетов реализации требований к программному продукту и его компонентам;

6) оценку изменения в требованиях к программному продукту и его компонентам по стоимости, времени выполнения работ и воздействиям на технические характеристики;

7) доведение до сведения заинтересованных сторон требований к программным компонентам и программному продукту.

Деятельность по определению требований к программным компонентам, программному продукту и интерфейсам между ними заключается в выявлении и документальном оформлении следующих видов требований:

- функциональных и нефункциональных (эксплуатационных) требований, в том числе производительности, физических характеристик и внешних условий, при которых будет эксплуатироваться программный продукт;
- требований к внешним интерфейсам ПП;

- квалификационных требований к персоналу, в том числе к взаимодействию человека с программно-аппаратным обеспечением, ограничений по персоналу и областям применения, требующим концентрации внимания и чувствительным к ошибкам человека;
- требований к безопасности и защите информации, в том числе к спецификациям, связанным с внешними и внутренними угрозами;
- требований к описанию данных и баз данных (БД), достоверности и допустимой точности информации в БД;
- требований к инсталляции поставляемого ПП;
- требований к документации пользователя, пользовательских требований к сопровождению ПП;
- требований по приемке-сдаче и вводу в эксплуатацию программного продукта на объекте(ах) заказчика;
- требований к условиям эксплуатации, сопровождения, обслуживания и технической поддержки пользователя.

Определение приоритетов требований к программному продукту и его компонентам рекомендуется проводить по следующим критериям: прослеживаемости к системным требованиям и к системному проекту; внешней согласованности с системными требованиями; внутренней согласованности между программными элементами; тестируемости; реализуемости в составе программного проекта; влиянию на процессы функционирования и сопровождения ПП.

Процесс проектирования архитектуры программных средств заключается в разработке архитектурного дизайна ПП, в том числе определении внутренних и внешних интерфейсов.

Реализация процесса включает:

1) разработку и документальное оформление архитектуры ПП, описывающей верхний уровень его структуры и идентифицирующей все программные компоненты последующих уровней;

2) разработку и документальное оформление проекта верхнего уровня для внешних интерфейсов программного продукта и его интерфейсов с программными компонентами;

3) разработку и документальное оформление проекта верхнего уровня для базы данных;

4) разработку и документальное оформление предварительных версий пользовательской документации;

5) определение и документирование требований к предварительному тестированию и графику работ по комплексированию программных компонентов и программного продукта в целом;

6) оценивание и документальное оформление архитектуры программного продукта по интерфейсам и базе данных с учетом следующих критериев: взаимосвязи проекта верхнего уровня базы данных с требованиями программного продукта; внешней согласованности интерфейсов с требованиями к программному продукту; внутренней согласованности интерфейсов между программными компонентами; пригодности методов проектирования и используемых стандартов; возможности дальнейшего детального проектирования; влияния на процессы функционирования и сопровождения программного продукта.

При разработке архитектуры все требования к программному продукту распределяются по программным компонентам и в дальнейшем уточняются для облегчения детального проектирования.

Процесс детального проектирования программных средств заключается в декомпозиции его структуры до элементарных программных компонентов, которые могут быть верифицированы относительно установленных требований к архитектуре программного продукта. При реализации процесса для каждого элемента архитектурного дизайна программного продукта необходимо осуществлять:

1) разработку и документальное оформление детального проекта для каждого программного компонента. Программные компоненты должны быть детализированы на более низком уровне, включающем программные модули, которые могут быть закодированы, откомпилированы и протестированы;

2) разработку и документальное оформление детального проекта для внешних интерфейсов программного продукта, между программными компонентами и программными модулями;

3) разработку и документальное оформление детального проекта базы данных;

4) доработку по мере необходимости пользовательской документации;

5) определение и документирование требований к тестированию и графиков работ по тестированию программных модулей;

6) корректировку при необходимости требований к тестированию и определение графиков работ по комплексированию программного продукта;

7) оценивание и документальное оформление детального проекта и требований к его тестированию по следующим критериям: взаимосвязи с требованиями к ПП; внешней согласованности с архитектурным проектом; внутренней согласованности между программными компонентами и программными модулями; соответствию выбранной модели жизненного цикла разработки и используемых стандартов методам проектирования; возможности тестирования программного проекта; влиянию на процессы функционирования и сопровождения ПП.

Процесс конструирования программных средств заключается в разработке исполняемых программных модулей, которые определены и описаны в результате детального проектирования программных средств. При реализации процесса необходимо выполнять следующие виды деятельности в отношении процесса конструирования программных средств:

1) разработку и документальное оформление каждого программного модуля и базы данных, процедуры тестирования и исходных данных для тестирования каждого программного модуля и базы данных;

2) тестирование с использованием набора контрольных наборов тестов, для которых известен результат, и документальное оформление каждого программного модуля и базы данных;

3) внесение при необходимости изменений в документацию пользователя;

4) корректировку при необходимости требований к процедуре тестирования и определение графиков работ по комплексированию программных средств;

5) оценивание и документальное оформление программного кода и результатов тестирования с учетом следующих критериев: взаимосвязи с требованиями к программному продукту и его программным компонентам; внешней согласованности требований к ПП с требованиями к его программным компонентам; внутренней согласованности между требованиями к программным модулям; уровня тестового покрытия программных модулей; соответствия методов кодирования и используемых стандартов; возможности комплексирования и тестирования программных средств; влияния на процессы функционирования и сопровождения ПП.

Процесс комплексирования программных средств заключается в объединении программных модулей и программных компонентов путем создания интегрированных программных элементов, которые демонстрируют, что функциональные и нефункциональные требования к программным средствам удовлетворяются на полностью укомплектованной или эквивалентной ей программно-аппаратной платформе.

При реализации процесса комплексирования программных средств необходимо для каждого программного элемента составной части проекта выполнять:

1) разработку и документальное оформление плана комплексирования по объединению программных компонентов и программных модулей в программный продукт. План должен включать требования к тестированию, процедуры комплексирования, данные для комплексирования, требуемые ресурсы, сведения об ответственных исполнителях и сроках проведения работ;

2) проведение комплексирования программных компонентов и программных модулей в программный продукт в соответствии с планом, тестирование и документальное оформление результатов комплексирования и тестирования;

3) разработку стратегии регрессии для повторной верификации программных элементов в случае изменения кода в программных модулях;

4) внесение изменений по мере необходимости в пользовательскую документацию;

5) разработку и документальное оформление для каждого квалификационного требования ко всем программным элементам проекта комплекта тестов, тестовых примеров и процедур тестирования в целях проведения квалификационного тестирования программных средств;

6) оценку и документальное оформление плана комплексирования, проекта, программного кода, тестов, результатов тестирования и пользовательской документации с учетом взаимосвязи с системными требованиями, внешней согласованности с системными требованиями, внутренней согласованности между программными элементами проекта, тестового покрытия требований программных элементов проекта, пригодности используемых методов и стандартов тестирования, соответствия ожидаемым результатам, реализуемости квалификационного тестирования программных средств, влияния на процессы функционирования и сопровождения программного продукта.

Процесс квалификационного тестирования программных средств заключается в подтверждении того, что скомплексированный программный продукт удовлетворяет установленным требованиям. При реализации процесса необходимо выполнять следующие виды деятельности:

1) определение критериев оценки скомплексированных программных средств с целью демонстрации соответствия их установленным требованиям;

2) верифицирование скомплексированных программных средств с использованием сформулированных критериев;

3) квалификационное тестирование программных элементов на соответствие их квалификационным требованиям. Результаты квалификационного тестирования должны быть документально оформлены;

4) внесение изменений по мере необходимости в пользовательскую документацию;

5) оценивание и документальное оформление проекта, программного кода, тестов, результатов тестирования и пользовательской документации по следующим критериям: тестовому покрытию требований к программной составной части; соответ-

ствию ожидаемым результатам; реализуемости системного комплексирования и тестирования; влиянию на процессы функционирования и сопровождения программного продукта;

6) проведение аудита проекта и его документальное оформление, в случае необходимости внесение изменений в проект;

7) подготовку поставляемого ПП для комплексирования с программно-аппаратной средой применения, системного квалификационного тестирования, инсталляции и приемки-сдачи.

Процесс инсталляции программных средств заключается в установке программного продукта, удовлетворяющего заданным требованиям, в программно-аппаратную среду применения.

При реализации процесса необходимо выполнить:

1) разработку, документальное оформление и согласование с заказчиком плана инсталляции программного продукта в среду его применения. Если инсталлируемый программный продукт заменяет существующую программную систему, то необходимо предусмотреть в плане мероприятия по поддержке необходимых параллельно выполняемых процессов. Важной частью разработки плана инсталляции является возможность действий по проведению повторной установки и возврату к последней рабочей версии системы;

2) инсталляцию ПП в соответствии с планом инсталляции, документальное оформление процедур инсталляции и их результатов. В случае проведения повторной инсталляции последней рабочей версии следует сделать полную резервную копию системы до начала инсталляции. По мере готовности ПП устанавливается в среде применения для проведения приемо-сдаточных испытаний.

Процесс приемки программных средств заключается в подтверждении того, что инсталлированный программный продукт соответствует заданным требованиям. При реализации процесса приемки-сдачи программных средств разработчик должен выполнить следующие виды деятельности:

1) комплектацию и установку программного продукта в программно-аппаратную среду применения;

2) поддержку тестов приемо-сдаточных испытаний, проводимых приобретающей стороной;

3) обеспечение работоспособности программного продукта при тестировании.

4) обучение сотрудников заказчика в процессе проведения тестирования;

Приемочное тестирование программного продукта проводится сотрудниками заказчика. Результаты процесса приемки программных средств должны быть документированы. Проблемы, обнаруженные в течение приемки-сдачи, идентифицируются и передаются разработчику (поставщику) для доработки.

Процесс сопровождения программных средств заключается в обеспечении эффективной по затратам поддержки функционирования и модификации программного продукта, обучения и консультирования пользователей в режиме горячей линии. При реализации процесса сопровождающая сторона в соответствии с принятыми в организации-заказчике регламентами и процедурами в отношении процесса сопровождения программных средств должна выполнить следующие виды деятельности:

1) разработку, документальное оформление и выполнение планов и процедур проведения действий и решения задач в рамках процесса сопровождения программных средств;

2) определение процедур получения, регистрации и мониторинга отчетов о проблемах и заявках на модификацию программных средств от пользователей и обеспечение обратной связи;

3) анализ отчетов о проблемах и/или заявках на модификацию программных средств с учетом масштабов модификации, требуемых финансовых средств, времени на модификацию, критичности модификации, воздействия ее результатов на эксплуатационные характеристики, безопасность или защищенность;

4) определение программных модулей и документации, нуждающихся в модификации;

5) документальное оформление заявки и получение одобрения со стороны заказчика на реализацию выбранного варианта модификации;

6) определение и документальное оформление тестов и критериев оценки тестирования модифицированных и немодифицированных частей системы (программных модулей, компонентов);

7) реализацию и тестирование новых и/или модифицированных требований, обеспечение гарантии того, что исходные немодифицированные требования не были затронуты. Результаты тестирования должны быть документированы;

8) разработку и документальное оформление плана установки (перемещения) модифицированной системы. План должен содержать анализ требований к процедурам перемещения, разработку инструментария перемещения, конверсию программного продукта и данных, сроки выполнения процедур перемещения, верификацию перемещения, сроки поддержки прежней системы;

9) оповещение заинтересованных сторон о планах и действиях по перемещениям. Оповещения должны включать информацию о причинах прекращения поддержки прежней системы, описание новой системы с датой ее готовности, описание доступных вариантов поддержки (при их наличии) прежней системы;

10) установку модифицированной системы в программно-аппаратную среду заказчика для плавного перехода к новой системе. В течение этого периода времени работа может проводиться параллельно и в прежней, и новой системах. Необходимо обеспечить обучение пользователей работе в этих режимах;

11) оповещение всех заинтересованных сторон об окончании процедуры перемещения модифицированной системы. Помещение в архив документации, исходных кодов и других материалов, относящихся к прежней системе;

12) проверку работоспособности модифицированной системы по истечении некоторого периода эксплуатации, отправку результатов проверки соответствующим уполномоченным органам для информации, руководства к действию.

Процесс верификации программных средств заключается в подтверждении (проверке) того, что ПП полностью удовлетворяет предъявляемым к нему требованиям, которые включают непротиворечивость требований, соответствие выбранных параметров условиям технического задания, адекватность стан-

дартам и процедурам разработки, соответствие проектным спецификациям и требованиям, тестируемость и корректность кода, корректность комплексирования в системы, адекватность и полноту документации.

В результате успешного осуществления процесса верификации программных средств разрабатывается и реализуется стратегия верификации, определяются критерии верификации всех программных компонентов и ПП в целом, выполняются требуемые действия и задачи по верификации, определяются и регистрируются дефекты, результаты верификации предоставляются заказчику и другим заинтересованным сторонам.

При реализации процесса необходимо осуществлять:

1) определение и оценку условий верификации, а именно:

- наличие в требованиях к программному продукту ошибок, которые невозможно обнаружить, что приводит к невыполнению задания, финансовым потерям, катастрофической утрате работоспособности программно-аппаратного обеспечения;
- работоспособность программно-аппаратных средств, связанных с появлением ошибок;

2) определение участников процесса верификации для проверки ПП. Если проектом предусматриваются работы по независимой верификации, должна быть выбрана квалифицированная организация, ответственная за проведение верификации;

3) разработку и документальное оформление плана проведения верификации, в котором указываются необходимые действия, требуемые ресурсы, ответственные исполнители и сроки проведения работ;

4) верификацию программного продукта.

При верификации требований должны быть учтены следующие критерии:

- системные требования распределены соответственно по техническим, программным элементам и ручным операциям, требования согласованы, выполнимы и тестируемы;
- требования к программным средствам согласованы, выполнимы, проверяемы и точно отражают системные требования;

• требования к программным средствам, связанные с безопасностью, защищенностью и критичностью работоспособности, корректны и подтверждены соответствующими нормативными документами.

Программный код должен быть верифицирован с учетом следующих условий:

• код как элемент детального проекта соответствует установленным требованиям и стандартам, относящимся к кодированию, тестируемости, корректности работы;

• код реализует надлежащую последовательность событий, согласованные интерфейсы, корректные данные, потоки команд управления и завершения, адекватное распределение времени разработки и размеров финансирования, возможность определения ошибок, их локализацию и восстановление работоспособности;

• код корректно реализует требования по безопасности, защищенности, критичности к работоспособности, изложенные в соответствующих нормативных документах.

Комплексирование программного продукта должно быть верифицировано с учетом следующих критериев:

• полнота и корректность комплектования программных компонентов и модулей в программный элемент;

• выполнение задач комплексирования согласно плану.

Документация должна быть верифицирована с учетом перечисленных ниже критериев:

• адекватность, полнота и согласованность документации;

• своевременность подготовки документации;

• соответствие менеджмента конфигурации документов определенным нормативным регламентам.

Все возникшие в процессе верификации проблемы должны быть решены, а обнаруженные несоответствия устранены. Кроме того, необходимо обеспечить доступность результатов действий по верификации для потенциальных потребителей и заинтересованных организаций.

Процесс валидации программных средств заключается в проверке соответствия спроектированного ПП требованиям и

потребностям заказчика в условиях конкретного применения и предполагает выполнение на этапах ЖЦ разного рода действий для получения корректных результатов.

В процессе валидации программных средств осуществляется разработка стратегии валидации; определение критериев валидации для всех рабочих версий ПП; выполнение требуемых действий и задач процесса валидации; идентификация, регистрация и устранение проблемы; документальное оформление подтверждения работоспособности созданных рабочих версий ПП в условиях конкретного применения; доведение результатов действий по валидации до заказчика и заинтересованных сторон.

При реализации процесса валидации необходимо выполнять следующие виды деятельности:

1) определение и оценку условий реализации процесса валидации и степени организационной независимости работ;

2) подбор участников процесса валидации для подтверждающей проверки работоспособности ПП. Если проект предусматривает независимые работы по валидации, то должна быть выбрана ответственная за проведение работ квалифицированная организация, гарантирующая независимость и подтверждение полномочий при выполнении задач валидации;

3) выбор задач валидации и связанных с ними методов, технологий и инструментария;

4) разработку и документальное оформление плана валидации, в котором должны быть отражены элементы программного продукта, подвергаемые валидации; задачи валидации; ресурсы, исполнители и графики выполнения работ по валидации; процедуры передачи заказчику и другим заинтересованным сторонам отчетов о результатах;

5) осуществление процесса валидации, который включает:

- подготовку требований к процедурам тестирования, тестовых примеров и спецификации для анализа результатов тестирования;
- проверку соответствия требований к тестированию, тестовых примеров и спецификаций частным требованиям для конкретного применения;

• проверку выполнения требований, в том числе:

– тестирование в условиях повышенной нагрузки граничных значений выходных параметров ПП;

– тестирование ПП на его способность к изолированию и минимизации влияния ошибок, возможности обращения к оператору за помощью в условиях повышенной нагрузки;

– тестирование получаемого пользователем требуемого результата при использовании конкретного функционала ПП.

Все проблемы и несоответствия, обнаруженные в процессе валидации, необходимо устранить, кроме того, обеспечить заказчику и другим заинтересованным организациям доступность к информации о результатах действий по валидации.

По завершении процессов верификации и валидации создается комплект материалов, подтверждающих правильность сформированных требований, спецификаций элементов архитектурного и детального проектов, результатов проведения тестирования всех элементов программного продукта.

2.2. Комплекс стандартов «Единая система программной документации»

Стандарт «Единая система программной документации» представляет собой комплект из 23 документов, составляющих систему межгосударственных стандартов стран СНГ (ГОСТ 19), действующих на территории Российской Федерации на основе межгосударственного соглашения по стандартизации. Несмотря на то что большая часть комплекта ЕСПД была разработана в 1970–1980-е годы, стандарт пользуется большой популярностью как у разработчиков ПП, так и у организаций, планирующих участие в конкурсе-тендере) на разработку ПП.

Комплект стандартов ЕСПД включает [7]:

1) ГОСТ 19.001-77 ЕСПД. Общие положения;

2) ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов;

3) ГОСТ 19.102-77 ЕСПД. Стадии разработки;

4) ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов;

5) ГОСТ 19.104-78 ЕСПД. Основные надписи;

6) ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам;

7) ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом;

8) ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению;

9) ГОСТ 19.202-78 ЕСПД. Спецификация. Требования к содержанию и оформлению;

10) ГОСТ 19.301-79 ЕСПД. Программа и методика испытаний. Требования к содержанию и оформлению;

11) ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению;

12) ГОСТ 19.402-78 ЕСПД. Описание программы;

13) ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению;

14) ГОСТ 19.501-78 ЕСПД. Формуляр. Требования к содержанию и оформлению;

15) ГОСТ 19.502-78 ЕСПД. Описание применения. Требования к содержанию и оформлению;

16) ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению;

17) ГОСТ 19.504-79 ЕСПД. Руководство программиста. Требования к содержанию и оформлению;

18) ГОСТ 19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению;

19) ГОСТ 19.506-79 ЕСПД. Описание языка. Требования к содержанию и оформлению;

20) ГОСТ 19.508-79 ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению;

21) ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные документы, выполняемые печатным способом;

22) ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения;

23) ГОСТ 19.781-90 ЕСПД. Обеспечение систем обработки информации программное. Термины и определения.

Далее остановимся только на описании стандартов, регламентирующих состав и содержание стадий жизненного цикла разработки программных продуктов и их документирование.

ГОСТ 19.102-77 ЕСПД. Стадии разработки

В стандарте все действия по разработке программных продуктов и программной документации независимо от их назначения и области применения подразделяются на стадии, этапы и работы (табл. 2.1). При использовании стандарта допускается исключение 2-й стадии разработки, объединение 3-й и 4-й стадий, введение других этапов работ по согласованию с заказчиком. Необходимость изменения стадий разработки указывается в техническом задании (ТЗ).

ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов

В качестве основных видов программ стандартом определены:

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

комплекс — программа, состоящая из двух или более компонентов и/или комплексов, выполняющих взаимосвязанные функции, и применяемая самостоятельно либо в составе другого комплекса.

Виды программных документов и их краткое содержание представлены в стандарте описаниями, приведенными в табл. 2.2.

Перечень эксплуатационных документов, рекомендуемых ЕСПД, представлен в табл. 2.3. Допускается объединение отдельных видов эксплуатационных документов (за исключением ведомости эксплуатационных документов и формуляра), необходимость объединения указывается в ТЗ. Объединенному документу присваивают наименование и обозначение одного из объединяемых документов. В объединенных документах должны быть приведены сведения, которые необходимо включать в каждый объединяемый документ.

Таблица 2.1

Стадии разработки, этапы и содержание работ по разработке программных продуктов и программной документации

Стадии разработки	Этапы работ	Содержание работ
I ТЕХНИЧЕСКОЕ ЗАДАНИЕ	Обоснование необходимости разработки программы	Постановка задачи. Сбор исходных материалов. Выбор и обоснование критериев эффективности и качества разрабатываемой программы. Обоснование необходимости проведения научно-исследовательских работ
	Научно-исследовательские работы	Определение структуры входных и выходных данных. Предварительный выбор методов решения задач. Обоснование целесообразности применения ранее разработанных программ. Определение требований к техническим средствам. Обоснование принципиальной возможности решения поставленной задачи
	Разработка и утверждение технического задания	Определение требований к программе. Разработка технико-экономического обоснования разработки программы. Определение стадий, этапов и сроков разработки программы и документации на нее. Выбор языков программирования. Определение необходимости проведения научно-исследовательских работ на последующих стадиях. Согласование и утверждение технического задания

Продолжение табл. 2.1

Стадии разработки	Этапы работ	Содержание работ
II ЭСКИЗНЫЙ ПРОЕКТ	Разработка эскизного проекта	Предварительная разработка структуры входных и выходных данных. Уточнение методов решения задачи. Разработка общего описания алгоритма решения задачи. Разработка технико-экономического обоснования
	Утверждение эскизного проекта	Разработка пояснительной записки. Согласование и утверждение эскизного проекта
III ТЕХНИ-ЧЕСКИЙ ПРОЕКТ	Разработка технического проекта	Уточнение структуры входных и выходных данных. Разработка алгоритма решения задачи. Определение формы представления входных и выходных данных. Определение семантики и синтаксиса языка. Разработка структуры программы. Окончательное определение конфигурации технических средств
	Утверждение технического проекта	Подготовка плана мероприятий по разработке и внедрению программ. Разработка пояснительной записки. Согласование и утверждение технического проекта

Окончание табл. 2.1

Стадии разработки	Этапы работ	Содержание работ
IV РАБОЧИЙ ПРОЕКТ	Разработка программы	Программирование и отладка программы
	Разработка программной документации	Разработка программных документов в соответствии с требованиями ГОСТ 19.101-77
	Испытания программы	Разработка, согласование и утверждение программы и методики испытаний. Проведение предварительных государственных, межведомственных, приемо-сдаточных и других видов испытаний. Корректировка программы и программной документации по результатам испытаний
V ВНЕДРЕНИЕ	Подготовка и передача программы	Подготовка и передача программы и программной документации для сопровождения и (или) изготовления. Оформление и утверждение акта о передаче программы на сопровождение и (или) изготовление. Передача программы в фонд алгоритмов и программ

Таблица 2.2

Виды программных документов

Вид документа	Содержание документа
Спецификация	Состав программы и документация на нее
Ведомость держателей подлинников	Перечень предприятий, на которых хранятся подлинники программных документов
Текст программы	Запись программы с комментариями
Описание программы	Сведения о логической структуре и функционировании программы
Программа и методика испытаний	Требования, подлежащие проверке при испытании программы, а также порядок проведения испытаний и методы их контроля
Техническое задание	Назначение и область применения программы; технические, технико-экономические и специальные требования к программе; необходимые стадии и сроки разработки; виды испытаний
Пояснительная записка	Схема и общее описание алгоритма и (или) функционирования программы, а также обоснование принятых технических и технико-экономических решений
Эксплуатационные документы	Сведения для обеспечения функционирования и эксплуатации программы

Таблица 2.3

Виды эксплуатационных документов

Вид документа	Содержание документа
Ведомость эксплуатационных документов	Перечень эксплуатационных документов на программу
Формуляр	Основные характеристики программы, комплектность и сведения об эксплуатации программы
Описание применения	Сведения о назначении программы, области применения, используемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств
Руководство системного программиста	Сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения

Окончание табл. 2.3

Вид документа	Содержание документа
Руководство программиста	Сведения для эксплуатации программы
Руководство оператора (пользователя)	Сведения для обеспечения процедуры общения оператора с вычислительной системой в процессе выполнения программы
Описание языка	Описание синтаксиса и семантики языка
Руководство по техническому обслуживанию	Сведения для применения тестовых и диагностических программ при обслуживании технических средств

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

Техническое задание содержит совокупность требований к программным средствам и используется в дальнейшем в качестве основного документа при сдаче-приемке разработанной системы в эксплуатацию. Поэтому достаточно полно составленное (с учетом возможности внесения дополнительных разделов) и принятое заказчиком и разработчиком ТЗ является одним из основополагающих документов проекта.

В состав ТЗ входят следующие разделы:

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

В ТЗ допускается включение приложений. В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

2.3. Международный стандарт «Процессы и действия жизненного цикла программного обеспечения»

Стандарт IEEE[1] 1074-1997 **«**Процессы и действия жизненного цикла программного обеспечения» (Developing a software project life cycle processes) обеспечивает поддержку процессов жизненного цикла разработки программного обеспечения [5]. В документе содержание ЖЦ разработки ПП описывается набором из 6 фаз, 17 процессов и 65 действий (табл. 2.4).

Фаза определяется как группа логически связанных процессов, в ходе осуществления которых выполняется вполне конкретная часть проекта.

Процесс представляет собой ряд действий (работ), выполнение которых приводит к конкретному результату.

Действие (работа) — деятельность, выполняемая в процессе реализации проекта. При этом каждое действие характеризуется продолжительностью, стоимостью и потребностями в ресурсах и может быть детализировано командой разработчиков на более мелкие составляющие (задачи).

Содержание стандарта не отождествляется с конкретной моделью ЖЦ разработки ПП и не предполагает использование определенной методологии разработки.

Практическое использование стандарта начинается **с фазы выбора модели ЖЦ разработки ПП,** наиболее полно отвечающей особенностям программного проекта.

Фаза управления проектом может включать действия по планированию проекта, мониторингу, контролю, анализу хода выполнения проекта и обеспечению качества создаваемого ПП.

---

[1] Читается «ай-трипл-и» [Кулямин В.В. Технологии программирования. Компонентный подход [Электронный ресурс]: лекция 2. – Режим доступа: http://panda.ispras.ru/~RedVerst/RedVerst/Lectures%20and%]

Таблица 2.4

Базовые процессы и действия, выполняемые в рамках ЖЦ разработки ПП

Фаза ЖЦ	Процессы ЖЦ	Действия ЖЦ
1. Выбор модели ЖЦ разработки ПП	1. Установка соответствия между ЖЦ ПП и особенностями проекта	1. Идентификация моделей-«кандидатов» на роль ЖЦ ПП. 2. Выбор модели ЖЦ разработки ПП
2. Управление проектом	2. Планирование проекта	3. Сопоставление действий по выбранной модели ЖЦ ПП. 4. Распределение ресурсов проекта. 5. Установка инструментария по управлению проектом. 6. Управление планом проекта.
	3. Мониторинг и управление проектом	7. Анализ рисков проекта. 8. Планирование непредвиденных ситуаций. 9. Управление проектом. 10. Управление изменениями. 11. Формирование отчетов о ходе выполнения проекта
	4. Управление качеством ПП	12. Планирование управления качеством ПП. 13. Определение метрических показателей качества. 14. Управление качеством ПП. 15. Идентификация потребностей по улучшению качества
3. Предварительная разработка проекта (разработка концепции проекта)	5. Исследование концепции	16. Идентификация идей или потребностей в разработке. 17. Анализ потенциальных потребителей. 18. Проведение исследований по реализуемости проекта. 19. Предварительное планирование ресурсного обеспечения проекта. 20. Уточнение идеи и разработка концепции проекта
	6. Системное проектирование	21. Анализ функционала. 22. Декомпозиция системных требований. 23. Разработка системной архитектуры проекта

Продолжение табл. 2.4

Фаза разработки	Процессы ЖЦ	Действия
4. Разработка проекта	7. Управление требованиями	24. Определение и разработка требований к ПП. 25. Определение требований к интерфейсу. 26. Назначение приоритетов и интеграция требований к ПП
	8. Разработка проекта	27. Разработка проекта архитектуры. 28. Проектирование базы данных (при необходимости). 29. Проектирование интерфейсов. 30. Выбор либо разработка алгоритмов (при необходимости). 31. Выполнение детализированной разработки проекта.
	9. Внедрение	32. Создание тестовых данных. 33. Разработка исходного кода. 34. Генерирование объектного кода. 35. Создание пользовательской документации. 36. Разработка или интеграция ПП. 37. Интеграция программных модулей (компонентов) в ПП
5. Сопровождение проекта	10. Установка	38. Разработка плана установки ПП в операционную среду эксплуатации. 39. Разработка плана интеграции ПП с операционной средой. 40. Установка ПП. 41. Приемка ПП в операционной среде
	11. Эксплуатация и поддержка	42. Регламентная эксплуатация ПП. 43. Обеспечение технической поддержки и консультирования пользователя. 44. Ведение журнала запросов о поддержке
	12. Сопровождение	45. Модернизация и переустановка ПП
	13. Вывод из эксплуатации	46. Извещение пользователей о прекращении эксплуатации.. 47. Согласование мероприятий по выводу из эксплуатации. 48. Вывод системы из эксплуатации

Окончание табл. 2.4

Фаза ЖЦ	Процессы ЖЦ	Действия ЖЦ
6. Интеграция проекта	14. Аттестация и верификация	49. Разработка плана аттестации и верификации. 50. Выполнение задач по аттестации и верификации. 51. Сбор и анализ метрических данных. 52. Разработка плана проведения тестирования. 53. Разработка требований по тестированию. 54. Выполнение тестирования
	15. Менеджмент конфигурации ПО	55. Планирование менеджмента конфигурации. 56. Идентификация конфигурации. 57. Контроль конфигурации. 58. Учет состояния текущей версии ПП
	16. Разработка документации	59. Планирование документации. 60. Применение документации. 61. Производство и распределение документации
	17. Обучение	62. Планирование программы обучения пользователей. 63. Разработка учебно-методических материалов. 64. Аттестация учебной программы. 65. Организация обучения пользователей

Совокупность действий, составляющих содержание **фаз разработки проекта** (третьей и четвертой)**,** охватывает полный жизненный цикл разработки ПП (от исследования концепции до возможного вывода системы из эксплуатации) и включает управление требованиями, разработку проекта на высоком и низком уровнях, кодирование, тестирование, а также другие задачи, выполняемые на протяжении ЖЦ разработки ПП.

Фаза сопровождения обычно занимает наиболее продолжительный период в жизненном цикле разработки ПП и содержит следующий набор действий: установку ПП в операционную среду эксплуатации; техническую поддержку и консультирование пользователей; модификацию программно-аппаратных решений и переустановку ПП; вывод системы из эксплуатации.

Действия по аттестации, верификации, тестированию, администрированию и поддержке задач конфигурации ПО, разработке технической документации и обучению пользователей входят в состав **фазы интеграции**.

Базовые процессы и действия жизненного цикла разработки ПП, представленные в табл. 2.4, приведены без их относительной привязки к конкретной модели ЖЦ. В связи с этим перед менеджером программного проекта возникает задача по установке соответствия между базовыми действиями стандарта и выбранной моделью ЖЦ разработки ПП с последующей адаптацией содержания стандарта под специфику используемой модели ЖЦ. В [5] приведено описание наборов действий и задач для каждой модели ЖЦ ПП, представленной в третьем разделе.

Рассмотрим одну из версий адаптации базовых процессов, действий и задач стандарта IEEE 1074-1997 для модели быстрого прототипирования. Характерная особенность этой модели состоит в том, что процессы разработки представляют собой результат циклической перестановки большинства действий, описанных в базовой модели. При этом следует учитывать, что данная модель применяется согласно линейному закону несколько раз, причем каждый раз создается все более расширенный и качественный функционал ПП. Коль скоро модель ЖЦ разработки уже выбрана, то ***первая фаза*** разработки проекта отсутствует.

Вторая фаза «Управление проектом» заключается в совместной разработке пользователем и разработчиком укрупненного графика выполнения работ и определении характеристик, получаемых на каждом шаге версий ПП. На данной фазе необходимо выполнить следующий набор процессов и действий:

- установку соответствия между действиями и выбранной моделью ЖЦ ПП;
- распределение ресурсов проекта;
- установку инструментария по управлению проектом;
- планирование управления проектом.

На фазе предварительной разработки проекта разработчик совместно с пользователем создает «бумажный прототип» — документ, содержащий требования и спецификации для наиболее сложных частей будущего ПП. Состав процессов и действий фазы предварительной разработки представлен в следующем виде:

- исследование концепции:
  - идентификация идей или потребностей в разработке;
  - анализ потенциальных потребителей;
  - проведение исследований по реализуемости проекта;
  - предварительное планирование ресурсного обеспечения проекта;
  - уточнение идеи и разработка концепции;
- системное проектирование:
  - разработка предварительной системной архитектуры;
  - декомпозиция предварительных системных требований;
  - идентификация предварительных программных требований:
    * проведение предварительного собеседования с пользователями;
    * определение и разработка предварительных требований к программному обеспечению;
    * разработка предварительных требований к создаваемому интерфейсу;
    * расстановка приоритетов и интеграция требований к программному обеспечению.

Фаза разработки состоит из нескольких итераций, на каждой из которых разработчик демонстрирует пользователям текущий вариант прототипа. На основе разработанного «бумажного прототипа» на первой итерации создается первая версия ПП:

- разработка проекта:
  - разработка проекта архитектуры;
  - проектирование базы данных (при необходимости);
  - проектирование интерфейсов;
  - выбор либо разработка алгоритмов (при необходимости);
  - выполнение детализированной разработки проекта;
- создание программного прототипа:
  - создание тестовых данных;
  - разработка исходного кода;
  - генерирование объектного кода;
  - создание пользовательской документации;
  - разработка или интеграция ПП;
  - интеграция программных модулей (компонентов) в ПП;
- установка *текущей версии программного прототипа*:
  - разработка плана установки ПП в операционную среду эксплуатации;
  - разработка плана интеграции ПП с операционной средой;
  - установка ПП;
- приемка функционала программного прототипа;
- оценка соответствия программного прототипа установленным требованиям;
- обновление первоначальных требований и спецификаций;
- переход к следующей итерации либо разработка на основе созданного прототипа промышленной версии ПП.

Процессы и действия фазы разработки продолжаются до тех пор, пока не будет создана промышленная версия ПП, удовлетворяющая всем сформулированным требованиям.

На фазе сопровождения проводится установка *промышленной версии ПП* в операционную среду пользователя, поддерживаются процессы эксплуатации и сопровождения системы.

Перечень процессов и действий данной фазы соответствует процессам и действиям базовой модели (см. табл. 2.4).

Фаза интеграции проекта включает процессы и действия по аттестации и верификации промышленной версии ПП, разработке технической документации, обучению пользователей.

Контрольные вопросы

1. Перечислите и прокомментируйте семь групп процессов, описанных в ГОСТ Р ИСО/МЭК 12207-2010.
2. Прокомментируйте содержание процесса «Анализ требований» стандарта ГОСТ Р ИСО/МЭК 12207-2010.
3. Прокомментируйте содержание процесса «Проектирование архитектуры» стандарта ГОСТ Р ИСО/МЭК 12207-2010.
4. Прокомментируйте содержание процесса «Детальное проектирование» стандарта ГОСТ Р ИСО/МЭК 12207-2010.
5. Прокомментируйте содержание процесса «Конструирование» стандарта ГОСТ Р ИСО/МЭК 12207-2010.
6. Прокомментируйте содержание процесса «Верификация» стандарта ГОСТ Р ИСО/МЭК 12207-2010.
7. Перечислите этапы и прокомментируйте содержание работ стадии «Техническое задание» ГОСТ 19.201-78 ЕСПД.
8. Перечислите этапы и прокомментируйте содержание работ стадии «Технический проект» ГОСТ 19.201-78 ЕСПД.
9. Перечислите этапы и прокомментируйте содержание работ стадии «Рабочий проект» ГОСТ 19.201-78 ЕСПД.
10. Перечислите и раскройте содержание эксплуатационных документов ГОСТ 19.201-78 ЕСПД.
11. Перечислите и прокомментируйте содержание шести фаз жизненного цикла разработки ПП стандарта IEEE 1074-1997.
12. Прокомментируйте содержание фазы «Разработка проекта» стандарта IEEE 1074-1997.
13. Прокомментируйте содержание фазы «Сопровождение проекта» стандарта IEEE 1074-1997.

3. МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

Деятельность по созданию программного продукта описывается в виде **модели жизненного цикла —** последовательности процессов, работ и задач, обеспечивающих разработку, эксплуатацию и сопровождение ПП. Модель ЖЦ отражает эволюцию продукта, начиная от формулирования требований к модели до прекращения ее использования. Такая последовательность может быть линейной, когда фазы следуют друг за другом, либо нелинейной, если фазы повторяются или происходят одновременно. В литературе приводится описание следующих нашедших применение моделей жизненного цикла разработки ПП: каскадной, V-образной, прототипирования, быстрой разработки приложений, инкрементной, спиральной. Особенности названных моделей раскрываются на основе материалов [5, 8].

3.1. Каскадная модель

Каскадная модель («водопад») является одной из первых, применяемых на практике, моделей ЖЦ ПП, в которой каждая работа выполняется один раз в определенной последовательности и с требуемым качеством, после ее завершения и перехода к следующей работе возвращения к предыдущей не требуется (рис. 3.1) [5]. Отличительное свойство каскадной модели состоит в том, что она представляет собой формальный метод (разновидность разработки «сверху-вниз») и состоит из независимых фаз, выполняемых последовательно.

Каскадную модель можно рассматривать как модель ЖЦ, пригодную для создания первой версии ПП, в следующих случаях: требования к ПП максимально конкретизированы, понятны и не изменяются; разрабатывается новая версия уже существующего продукта, при этом вносимые изменения четко определены; автоматизируются типовые бизнес-процессы потребителя, содержание которых закреплено нормативными документами.

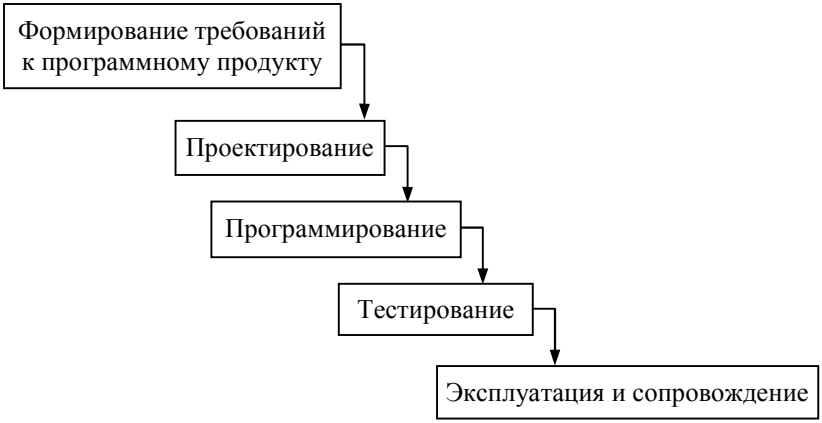

Рис. 3.1. Жизненный цикл каскадной модели

Основные ***положительные моменты применения каскадной модели*** заключаются в следующем: модель проста и понятна заказчикам; каждая последующая фаза начинается только после полного завершения предыдущей фазы; на каждой фазе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности; переход от одной фазы к другой осуществляется после приемки-сдачи работ с участием заказчика; выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты. Каскадные модели на протяжении всего времени их существования используются при выполнении крупных проектов, в которых задействовано несколько больших команд разработчиков.

Недостатки каскадной модели особенно остро проявляются в случаях, когда трудно (или невозможно) четко сформулировать требования либо требования меняются в процессе создания продукта. Кроме того, любая попытка вернуться на одну или две фазы назад, чтобы исправить какую-либо ошибку, приводит к значительному увеличению затрат и нарушению сроков разработки; интеграция программных компонентов, в процессе кото-

рой обычно выявляется бо́льшая часть ошибок, выполняется в конце разработки, что сильно увеличивает стоимость их устранения; происходит большое запаздывание с оценкой качества разрабатываемого ПП. В этих случаях разработка ПП имеет циклический характер, когда результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах. Изменения связаны, как правило, с ошибками разработчиков, допущенными на ранних этапах разработки и выявленными на этапе тестирования. Также они могут быть вызваны изменениями требований в процессе разработки вследствие неготовности заказчиков четко их сформулировать либо введением требований, обусловленных изменением бизнес-процессов предметной области. Таким образом, постоянно возникает потребность в возврате к предыдущим фазам и уточнению либо пересмотру ранее принятых решений, в результате чего ***реальный процесс разработки*** принимает другой вид (рис. 3.2) [5].

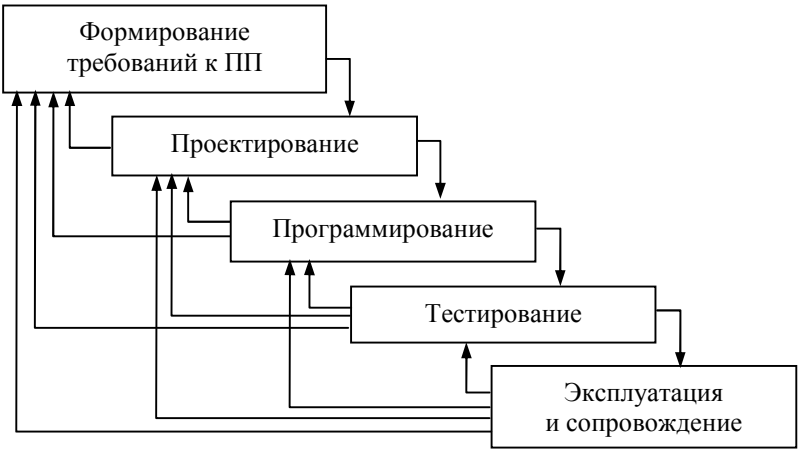

Рис. 3.2. Модифицированная версия каскадной модели

Модифицированная версия каскадной модели является в значительной степени менее жесткой, чем ее первоначальная форма. В этой версии допускается наличие итераций между фазами

(постоянный возврат к предыдущему шагу с целью анализа и проверки на соответствие результатов поставленным задачам), возможность параллельного выполнения фаз и управление изменениями в проекте. Несмотря на то что модифицированная каскадная модель является более гибкой, чем классическая, она также не находит широкого применения, так как на практике не удается выполнить приведенные выше условия применения. Попытки оптимизации каскадной модели привели к возникновению других типов моделей разработки программного продукта (см. подразд. 3.2–3.6).

3.2. V-образная модель

V-образная модель представляет собой итерационную разновидность каскадной модели и предполагает уже на ранних этапах жизненного цикла планирование работ по тестированию программного продукта (рис. 3.3) [5]) Так, например, уже на этапе разработки требований к проекту намечается план проведения заказчиком приемо-сдаточных испытаний, а на этапах анализа и проектирования — план комплексирования и испытания ПП (эти процессы на рис. 3.3 обозначены пунктирными линиями).

От каскадной модели V-образная модель унаследовала последовательную структуру, в соответствии с которой каждая последующая фаза начинается только после успешного завершения предыдущей. **Фазами V-образной модели** являются:

1) разработка требований к проекту и планирование (определение системных требований и планирование работ);

2) разработка требований к продукту и их спецификация (составление полной спецификации требований к ПП);

3) проектирование системной архитектуры, или высокоуровневое проектирование (определение структуры ПП, взаимосвязей между его основными компонентами и реализуемыми функциями);

4) детальное (техническое) проектирование (определение алгоритма работы каждого компонента);

5) кодирование (разработка программного кода, т. е. преобразование алгоритмов в готовый ПП);

6) модульное тестирование (проверка каждого компонента или модуля программного продукта);

7) интеграционное тестирование (интеграция модулей программного продукта и его тестирование);

8) системное тестирование (проверка функционирования программного продукта после его установки в соответствии со спецификацией нефункциональных требований на программно-аппаратную платформу заказчика);

9) эксплуатация и сопровождение (запуск программного продукта в производство). На этой фазе в программный продукт могут быть внесены поправки и выполнена его модернизация.

Рис. 3.3. Жизненный цикл V-образной модели

Основные преимущества V-образной модели заключаются в следующем:

- особое значение придается верификации и аттестации ПП, планированию всех действий, начиная с ранних стадий его разработки;
- предполагается аттестация и верификация не только самого ПП, но и всех полученных внутренних и внешних данных;
- ход выполнения работы может легко отслеживаться, так как завершение каждой фазы является контрольной точкой.

При использовании V-образной модели в работе над проектом, для которого она не является в достаточной степени приемлемой, становятся очевидными ее ***недостатки***:

- в модели не предусмотрено внесение требования динамических изменений на разных этапах жизненного цикла;
- тестирование требований в жизненном цикле происходит слишком поздно, вследствие чего невозможно внести изменения, не повлияв при этом на график выполнения проекта;
- в модель не входят действия, направленные на анализ и управление рисками.

Данную модель целесообразно использовать при разработке ПП, главным требованием для которых является высокая надежность, когда доступной является информация о методах решения функциональных задач и технологии их реализации, а персонал владеет необходимыми умениями и опытом работы с данной технологией. Подобно своей предшественнице, каскадной модели, V-образная модель лучше всего срабатывает в случаях, когда вся информация о требованиях доступна заранее.

3.3. Модель прототипирования

Основной идеей модели прототипирования является максимальное вовлечение пользователя в процесс разработки с целью извлечения, описания и корректировки реальных требований к будущему ПП. Согласно определению Джона Коннэлла и Линда

Шафера, эволюционным ускоренным **прототипом** является «легко поддающаяся модификации и расширению рабочая модель предполагаемой системы, необязательно представляющая собой все свойства системы, благодаря которым пользователи данного приложения получают физическое представление о ключевых частях системы до ее непосредственной реализации; это — легко создаваемая, без труда поддающаяся модификации, максимально расширяемая, частично заданная рабочая модель основных аспектов предполагаемой системы» [5]. Как правило, под **прототипом** понимается действующий программный компонент, реализующий отдельные функции и внешние интерфейсы разрабатываемого программного продукта.

Использование модели прототипирования позволяет уже на фазе разработки требований создавать работающий программный компонент, реализующий отдельные функции и внешние интерфейсы разрабатываемого программного продукта. Потенциальные пользователи работают с этим прототипом, определяя его сильные и слабые стороны, о результатах сообщают разработчикам программного продукта. Таким образом, обеспечивается ***обратная связь*** между пользователями и разработчиками, которая используется для изменения или корректировки спецификации требований к программному продукту. В результате такой работы продукт будет отражать реальные потребности пользователей. Схема ЖЦ модели прототипирования приведена на рис. 3.4 [5].

Начало жизненного цикла разработки помещено в центре эллипса. Жизненный цикл разработки программного продукта начинается с совместного создания конечными пользователями и разработчиками плана проекта, затем выполняется быстрый анализ предметной области, формулируются предварительные требования, проектируется база данных, пользовательский интерфейс и функционал будущего прототипа программного продукта. В результате этой работы создается *документ*, который содержит частичную спецификацию требований к ПП и в дальнейшем служит основой для итерационного цикла быстрого прототипирования.

Рис. 3.4. Жизненный цикл модели прототипирования

Следующий уровень — создание на основе разработанного документа исходного прототипа будущего программного продукта. Далее на каждой итерации прототипирования разработчик демонстрирует пользователям вариант прототипа, а пользователи оценивают его функциональные возможности и определяют проблемы. Этот процесс продолжается до тех пор, пока пользователи не будут удовлетворены степенью соответствия прототипа программного продукта, поставленным перед ним требованиям. Готовый прототип демонстрируют пользователям, утверждают и на его основе выполняется разработка программного продукта. Именно на этом этапе ускоренный прототип становится промышленным ПП, удовлетворяющим требованиям заказчика. При разработке производственной версии может понадобиться более высокий уровень реализации функциональных возможностей, подключение различных системных сервисов, необходимых в том числе для выполнения нефункциональных требований.

После этого следует тестирование в предельных режимах, а затем, как обычно, техническая поддержка процессов функционирования и сопровождения.

Преимущества модели протипирования состоят в следующем:

- ознакомление заказчика с разрабатываемым ПП начинается на раннем этапе ЖЦ, поэтому снижается вероятность возникновения путаницы, искажения информации или недоразумений при определении требований к программному продукту;
- в процессе разработки всегда можно учесть новые, даже неожиданные требования заказчика, что приводит к созданию более качественного программного продукта;
- прототип представляет собой формальную спецификацию, воплощенную в программный продукт, и позволяет гибко выполнять проектирование и разработку, включая несколько итераций на всех фазах жизненного цикла разработки;
- уменьшается количество доработок, что снижает стоимость разработки;
- возникающие проблемы решаются на ранних стадиях ЖЦ, что резко сокращает расходы на их устранение;
- заказчики принимают участие в процессе разработки на протяжении всего жизненного цикла и в конечном итоге несут ответственность за результаты работы наравне с разработчиками.

Основные ***недостатки модели прототипирования***:

- прототипирование может продолжаться слишком долго и разработчики могут попасть в так называемый цикл «кодирование — устранение ошибок», что приводит к дорогостоящим незапланированным итерациям прототипирования;
- разработчики и пользователи не всегда понимают, когда прототип превращается в конечный продукт, поэтому существует необходимость в традиционном документировании процесса;
- на очередной итерации заказчики могут быть удовлетворены качеством прототипа и требуют его немедленной поставки, вместо того чтобы ждать появления полной, хорошо продуманной версии;

- на разработку системы может быть потрачено слишком много времени, так как итерационный процесс демонстрации прототипа и его пересмотр могут продолжаться бесконечно долго, на заказчиков может оказать негативное влияние тот факт, что они не располагают информацией о точном количестве необходимых итераций;
- при выборе инструментальных средств прототипирования (операционных систем, технологий проектирования, языков программирования, алгоритмов решения функциональных задач) разработчики могут остановить свой выбор на неэффективных решениях, чтобы продемонстрировать свои способности.

Модель прототипирования рекомендуется применять:

- при выполнении новой, не имеющей аналогов разработки;
- когда заказчик неохотно соглашается на фиксированный набор требований, требования к программному продукту заранее неизвестны и могут уточняться в процессе разработки;
- если разработчики не уверены в выбранных решениях относительно пользовательского интерфейса и функционала, оптимальности применяемой архитектуры или алгоритма.

3.4. Модель быстрой разработки приложений

В модели быстрой разработки приложений (Rapid Application Development — RAD) пользователь задействован не только при определении требований, но и на всех остальных фазах жизненного цикла разработки ПП: проектировании, кодировании, тестировании, внедрении (рис. 3.5) [5]. Для этого необходимо использовать специальное ПО — средства разработки графического пользовательского интерфейса и кодогенераторы. Модель основывается на последовательности итераций создания прототипов, критический анализ которых обсуждается с заказчиком.

Характерной чертой RAD-модели является короткое время перехода от определения требований до создания полной систе-

мы. Разработка каждого интегрированного продукта ограничивается четко определенным периодом времени, который, как правило, составляет 60 дней и называется временным блоком.

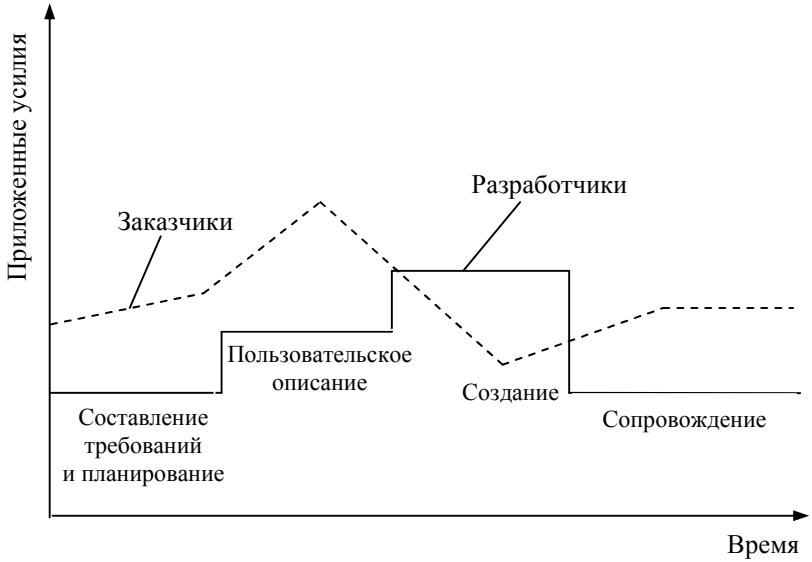

Рис. 3.5. Модель быстрой разработки приложений

В состав каждого временного блока входит анализ, проектирование и внедрение. Факторы, позволяющие создать систему за 60 дней, причем без ущерба качеству, включают в себя применение мощных инструментальных средств разработки, высокий уровень повторного использования программного кода, быстрый и качественный анализ промежуточных результатов, предоставление необходимых ресурсов.

В RAD-модели конечный пользователь играет *решающую роль*. В тесном взаимодействии с разработчиками он участвует в формировании требований и их апробации на работающих прототипах. Таким образом, в начале жизненного цикла на конечного пользователя выпадает большая часть работы, но в результате этого создаваемая система формируется быстрее.

RAD-модель включает следующие фазы:

- составление требований и планирование (сбор требований осуществляется с использованием так называемого метода совместного планирования требований (планирование работ по созданию ПП и составление требований к ПП выполняются одновременно), который заключается в структурном анализе и обсуждении решаемых задач (будущего функционала);
- пользовательское описание (проектирование ПП, выполняемое при непосредственном участии заказчика, при этом работающая над проектом команда зачастую использует специальные инструментальные средства, обеспечивающие сбор пользовательской информации);
- создание (детальное проектирование, кодирование и тестирование ПП и его поставка заказчику за определенное время);
- сопровождение (проведение пользователем приемочных испытаний, установка ПП и обучение пользователей).

Достоинства RAD-модели:

- использование современных инструментальных средств позволяет сократить время ЖЦ разработки;
- постоянное присутствие заказчика сводит до минимума риск неудовлетворения продуктом и гарантирует соответствие системы коммерческим потребностям, а также надёжность программного продукта в эксплуатации;
- основное внимание переносится с разработки документации на создание кода;
- повторно используются компоненты уже существующих программ.

В то же время RAD-модели присущи и ***недостатки***:

- если заказчики не могут постоянно участвовать в процессе разработки, то это может негативно сказаться на качестве программного продукта;
- для создания ПП нужны высококвалифицированные кадры как разработчиков, так и пользователей, умеющих работать с современными инструментальными средствами;

- использование модели может оказаться неудачным при отсутствии пригодных для повторного использования компонентов;
- для реализации модели требуются разработчики и заказчики, которые готовы к быстрому выполнению действий ввиду жестких временных ограничений;
- команды, разрабатывающие коммерческие проекты с помощью модели RAD, могут «затянуть» разработку программного продукта до такой степени, что его поставка конечному пользователю будет под большим вопросом;
- существует риск, что работа над проектом никогда не будет завершена, в связи с этим менеджер проекта должен сотрудничать как с командой разработчиков, так и с заказчиком, что позволит избежать появления замкнутого цикла.

RAD-модель можно применять при создании программных продуктов, хорошо поддающихся моделированию, когда требования к ПП хорошо известны, а заказчик может непосредственно участвовать в процессах разработки ПП на всех этапах ЖЦ.

3.5. Инкрементная модель

Инкрементирование[1] представляет собой процесс поэтапной реализации ПП путем постепенного расширения его функциональных возможностей на основе заранее сформированного полного набора требований. На ранних этапах жизненного цикла (разработки и анализа требований проектирования) выполняется архитектурное проектирование системы в целом. Тогда же определяется и число необходимых инкрементов, и относящихся к ним функций.

Далее на каждой итерации происходит кодирование, тестирование и установка очередного инкремента. При этом сначала выполняется конструирование, тестирование и реализация набора базовых функций, формирующих основу продукта, или/и

[1] Инкрементирование (от англ. increment — увеличение) — операция во многих языках программирования, увеличивающая переменную.

требований первостепенной важности, играющих основную роль для успешного выполнения проекта и снижающих степень риска. Последующие итерации направлены на улучшение функциональных возможностей ПП (рис. 3.6) [5].

Рис. 3.6. Инкрементная модель

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

Итерационная процедура разработки подразумевает не только сборку из инкрементов работающей текущей версии ПП, но и его развертывание в реальной программно-аппаратной платформе. После каждой итерации можно анализировать промежуточные результаты работ и реакцию на них всех заинтересованных лиц, вносить корректирующие изменения на следующих итерациях.

Инкрементные модели целесообразно использовать при наличии следующих предпосылок:

1) если большинство требований и функционал можно сформулировать заранее, но их появление ожидается через определенный период времени;

2) если рыночное окно слишком «узкое» и есть потребность в быстрой поставке на рынок продукта, имеющего функциональные базовые свойства;

3) для проектов, на выполнение которых предусмотрен большой период времени разработки, как правило, более одного года;

4) при разработке программ, связанных с низкой или средней степенью риска;

5) при выполнении проекта с применением новой технологии, позволяющей пользователю адаптироваться к системе путем выполнения более мелких инкрементных шагов без резкого перехода к применению основного нового продукта.

Преимущества инкрементной модели:

- в результате выполнения каждого инкремента получается функциональный продукт, заказчик располагает возможностью высказаться по поводу каждой разработанной версии системы;
- сокращается время и снижаются затраты на первоначальную поставку программного продукта, снижается риск неудачи и изменения требований;
- заказчики могут распознавать самые важные и полезные функциональные возможности продукта на ранних этапах разработки;

- риск распределяется на несколько относительно небольших по размеру инкрементов (не сосредоточен в одном большом проекте разработки);
- требования стабилизируются (посредством включения в процесс пользователей) на момент создания определенного инкремента, поскольку не являющиеся особо важными изменения отодвигаются на момент создания последующих инкрементов;
- инкременты функциональных возможностей несут больше пользы и проще при тестировании, чем продукты промежуточного уровня при поуровневой разработке по принципу «сверху-вниз»;
- улучшается понимание требований для более поздних инкрементов, что обеспечивается благодаря возможности пользователя получить представление о ранее полученных инкрементах на практическом уровне;
- в конце каждой инкрементной поставки существует возможность пересмотреть риски, связанные с затратами и соблюдением установленного графика;
- использование последовательных инкрементов позволяет объединить полученный пользователями опыт в виде усовершенствованного продукта, затратив при этом намного меньше средств, чем требуется для выполнения повторной разработки;
- в процессе разработки можно ограничить количество персонала таким образом, чтобы над поставкой каждого инкремента последовательно работала одна и та же команда.

Недостатки инкрементной модели:

- выделение инкрементов, определение требований и полной функциональной системы должно осуществляться в начале жизненного цикла;
- поскольку создание некоторых модулей завершается значительно раньше других, возникает необходимость в четко определенных интерфейсах;
- использование на этапе анализа общих целей вместо полностью сформулированных требований может оказаться неудобным для руководства;

- для использования инкрементной модели необходимо тщательное планирование работ и подробное архитектурное проектирование;
- может возникнуть тенденция к оттягиванию решений трудных проблем на будущее с целью продемонстрировать руководству успех, достигнутый на ранних этапах разработки.

3.6. Спиральная модель

Спиральная модель была предложена как альтернатива каскадной модели. При этом было предусмотрено использование метода прототипирования либо быстрой разработки приложений.

Модель отображает базовую концепцию разработки, согласно которой набору операций в каждом цикле соответствует такое же количество фаз, как и в модели каскадного процесса, начиная с формулирования требований и заканчивая кодированием каждой отдельной программы (рис. 3.7) [5].

Основные принципы спиральной модели можно сформулировать следующим образом:

- разработка вариантов продукта, соответствующих различным вариантам требований, с возможностью вернуться к более ранним версиям;
- создание прототипов ПП как средства общения с заказчиком для уточнения и выявления требований;
- планирование последующих вариантов с оценкой альтернатив и анализом рисков, связанных с переходом к следующему варианту; переход к разработке следующего варианта до завершения предыдущего в случае, когда риск завершения очередного варианта (прототипа) становится неоправданно высок;
- активное привлечение заказчика к работе над проектом. Заказчик участвует в оценке очередного прототипа ПП, уточнении требований при переходе к следующему, оценке предложенных альтернатив очередного варианта и оценке рисков.

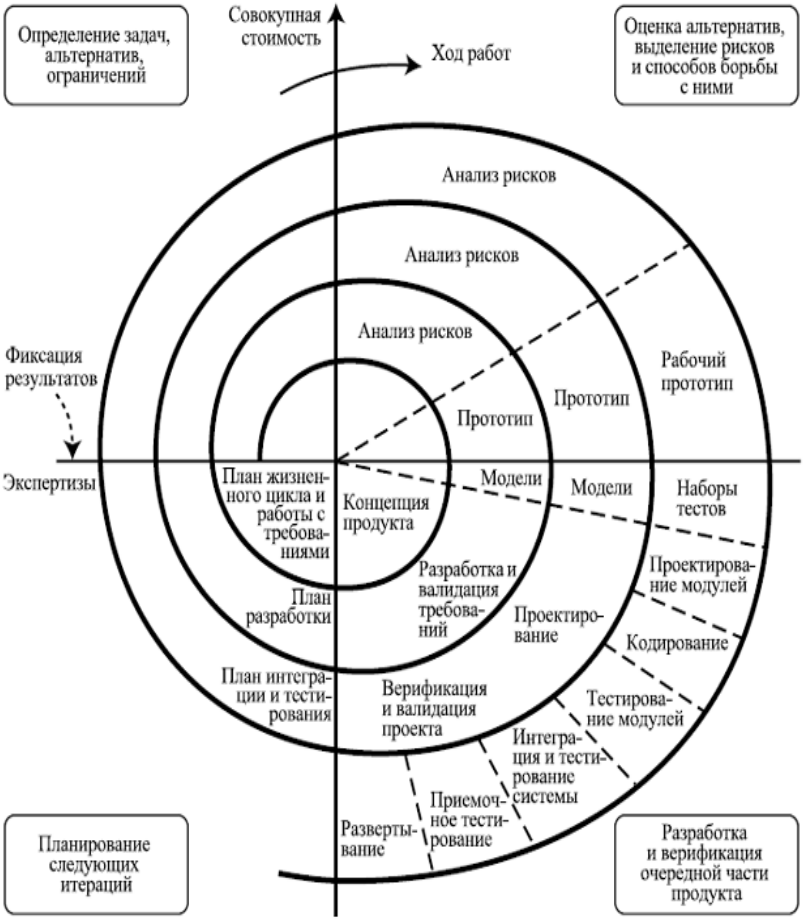

Рис. 3.7. Жизненный цикл спиральной модели

Принципиальная особенность спиральной модели заключается в том, что прикладной программный продукт создается не сразу, как в случае каскадного подхода, а по частям с использованием *метода прототипирования*. Создание прототипов осуществляется за несколько итераций, или витков спирали.

Каждая итерация соответствует созданию фрагмента или версии программного продукта, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируется работа следующей итерации. На каждой итерации производится тщательная оценка риска превышения сроков и стоимости проекта с целью определения необходимости выполнения еще одной итерации, степени полноты и точности понимания требований к системе, а также целесообразности прекращения проекта.

Спиральная модель избавляет пользователей и разработчиков программного продукта от полного и точного формулирования требований к системе на начальной стадии, поскольку они уточняются на каждой итерации. Таким образом, углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

Разработка итерациями отражает объективно существующий спиральный цикл создания системы, позволяя переходить на следующую стадию, не дожидаясь полного завершения работы на текущей стадии, поскольку при итеративном способе разработки недостающую работу можно выполнить на следующей итерации. Главная задача такой разработки — как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Преимущества спиральной модели:

- возможность пользователям «увидеть» систему на ранних этапах, что обеспечивается посредством использования ускоренного прототипирования в жизненном цикле разработки ПП;
- разбиение общего объема работы по созданию продукта на небольшие части, в которых сначала реализуются функции с высокой степенью риска;
- возможность гибкого проектирования, поскольку в ней воплощены преимущества каскадной и инкрементной моделей;
- тесное взаимодействие пользователей и разработчиков выполняется довольно часто и на ранних этапах модели, что обеспечивает создание нужного продукта высокого качества;

• совершенствование административного управления над процессом обеспечения качества, правильностью выполнения процесса разработки, регулирования затрат, соблюдения графика разработки и кадрового обеспечения, которое достигается путем выполнения обзора в конце каждой итерации;

• не нужно распределять заранее все необходимые для выполнения проекта финансовые ресурсы.

Недостатки спиральной модели:

• если проект имеет низкую степень риска или небольшие размеры, модель может оказаться дорогостоящей, так как оценка рисков после прохождения каждой спирали связана с большими затратами;

• модель имеет усложненную структуру, поэтому ее практическое использование требует высокопрофессиональных знаний разработчиков и заказчиков;

• бесконечность модели — на каждом витке заказчик может выдвигать новые требования, которые приводят к необходимости следующего цикла разработки, что в конечном итоге отдаляет окончание работы над проектом;

• использование модели может оказаться дорогостоящим и даже недопустимым по средствам, так как время, затраченное на планирование, повторное определение целей, выполнение анализа рисков и прототипирование, может быть неприемлемым.

Перечисленные недостатки объясняют тот факт, что использование спиральной модели на практике еще не получило такого широкого распространения, как применение других моделей.

Применение спиральной модели целесообразно в следующих случаях:

• пользователи не уверены в своих потребностях или требования слишком сложны и меняются в процессе выполнения проекта, а для их анализа и оценки необходимо прототипирование;

• достижение успеха не гарантировано и необходима оценка рисков продолжения проекта;

• проект является сложным, дорогостоящим и обоснование финансирования возможно только в процессе его выполнения;

- в силу ограниченности ресурсов проект можно выполнять только по частям;
- ПП реализуется с применением новых средств разработки, что связано с риском их освоения и своевременного достижения ожидаемого результата.

3.7. Методика выбора модели жизненного цикла разработки программного продукта

Приведенные выше модели жизненного цикла разработки программного продукта представляют собой логически взаимосвязанную совокупность процессов и работ, описывающих действия разработчиков и пользователей по созданию программных продуктов, начиная с определения требований и заканчивая приемкой-сдачей готового продукта. Каждая модель имеет присущие ей характеристики, определяющие целесообразность ее применения для конкретных проектов.

При выборе модели ЖЦ согласно рекомендациям [5] предлагается использовать следующий набор характеристик:

1) особенности выявления и анализа требований к ПП;

2) квалификация команды разработчиков;

3) участие коллектива пользователей в реализации программного проекта;

4) сложность проекта.

Описание отличительных свойств проекта, определяющих возможность использования каждой модели ЖЦ разработки ПП, приведены в табл. 3.1–3.4.

Выбор конкретной модели жизненного цикла ПП оказывает влияние на состав и содержание работ по созданию программного продукта.

Постановка задачи выбора наиболее эффективной модели ЖЦ программного продукта в зависимости от особенностей конкретного проекта приведена ниже.

Таблица 3.1

Оценка свойств моделей ЖЦ в зависимости от особенностей процесса выявления и анализа требований к ПП

Особенности процесса выявления и анализа требований к ПП	Наличие свойства в модели					
	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
1. Простота и легкость идентификации требований, их известность	Да	Да	Нет	Нет	Да	Нет
2. Возможность предварительного определения требований на ранних этапах разработки	Да	Да	Нет	Нет	Да	Да
3. Необходимость частого внесения изменений в требования в процессе разработки	Нет	Нет	Да	Да	Нет	Нет
4. Обязательная проверка требований	Нет	Нет	Да	Да	Да	Нет
5. Необходимость демонстрации функционала ПП для проверки соответствия концепции проекта	Нет	Нет	Да	Да	Да	Да
6. Зависимость уровня сложности системы от пакета сформированных требований	Нет	Нет	Да	Да	Нет	Да
7. Возможность определения функциональных свойств ПП на раннем этапе формулирования требований	Нет	Нет	Да	Да	Да	Да

Таблица 3.2

Оценка свойств моделей ЖЦ в зависимости от квалификации команды разработчиков

Квалификационные характеристики разработчиков проекта	Наличие свойства в модели					
	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
1. Знание проблем предметной области проекта большинством разработчиков	Нет	Нет	Да	Да	Нет	Нет
2. Знание технологии предметной области большинством разработчиков	Да	Да	Нет	Да	Нет	Да
3. Владение инструментарием, используемым в проекте, большинством разработчиков	Да	Да	Нет	Да	Нет	Нет
4. Постоянство (неизменность) ролей участников проекта в процессе разработки	Нет	Нет	Да	Да	Нет	Да
5. Возможность обучения разработчиков проекта в процессе его выполнения	Нет	Да	Нет	Нет	Да	Да
5. Б*о*льшая значимость для разработчиков проекта структуры ПП по сравнению с его гибкостью	Да	Да	Нет	Нет	Нет	Да
6. Строгое отслеживание менеджером проекта прогресса сотрудников в команде	Да	Да	Нет	Да	Нет	Да
7. Важность оперативного распределения ресурсов	Да	Да	Нет	Нет	Да	Да
8. Приемлемость командой проекта равноправного анализа состояния проекта, допустимость менеджмента со стороны заказчика	Да	Да	Да	Да	Нет	Да

Таблица 3.3

Оценка свойств моделей ЖЦ в зависимости от участия в проекте коллектива пользователей

Участие пользователей в реализации проекта	Наличие свойства в модели					
	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
1. Ограниченность присутствия в жизненном цикле проекта	Да	Да	Нет	Да	Нет	Да
2. Знание возможностей системы	Нет	Нет	Да	Да	Нет	Да
3. Обязательность ознакомления с проблемами предметной области проекта	Нет	Нет	Да	Нет	Да	Да
4. Вовлеченность во все фазы жизненного цикла разработки	Нет	Нет	Да	Нет	Да	Нет
5. Предоставление заказчику возможности отслеживания хода выполнения проекта	Нет	Нет	Да	Да	Нет	Нет

Таблица 3.4

Оценка свойств моделей ЖЦ в зависимости от сложности проекта

Сложность проекта	Наличие свойства в модели					
	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
1. Направленность проекта на разработку нового продукта	Нет	Нет	Да	Да	Нет	Да
2. Необходимость системной интеграции ПП с существующей инфраструктурой	Нет	Да	Да	Да	Да	Да
3. Позиционирование ПП как расширение существующей системы	Нет	Да	Нет	Нет	Да	Да
4. Стабильность финансирования проекта на всем протяжении жизненного цикла	Да	Да	Да	Нет	Да	Нет
5. Длительная эксплуатация продукта в организации	Да	Да	Нет	Да	Нет	Да
6. Необходимость обеспечения высокой надежности ПП	Нет	Да	Нет	Да	Нет	Да
7. Возможность непредвиденного изменения ПП на этапе сопровождения	Нет	Нет	Да	Да	Нет	Да
8. Обязательность соблюдения графика работ	Нет	Нет	Да	Да	Да	Да
9. Обеспечение «прозрачности» интерфейсных модулей	Да	Да	Нет	Нет	Нет	Да
10. Доступность повторно используемых компонентов	Нет	Нет	Да	Да	Да	Нет
11. Достаточность ресурсов (времени, денег, инструментов, персонала)	Нет	Нет	Да	Да	Нет	Нет

Постановка задачи выбора эффективной модели ЖЦ ПП

Пусть известно множество моделей жизненного цикла ПП $R = \{1, ..., \rho, ..., 6\}$. Каждая модель $r \in R$ описывается четырьмя группами характеристик $I = \{1, ..., j, ..., 4\}$, где $j = \{1, ..., j, ..., n\}$ — множество показателей (свойств), входящих в состав каждой группы характеристик.

Тогда $X_r = \{x_{ij}^r\}, i = \overline{1, 4}, j = \overline{1, n}$ — множество нормативных показателей *r*-й модели жизненного цикла ПП. Каждый из показателей x_{ij}^r описывается в качественной шкале оценивания (да, нет) и характеризуется наличием либо отсутствием *j*-го свойства модели в *i*-й группе.

Пусть $X_\text{р} = \{x_{ij}^p\}$ — множество показателей, характеризующих особенности процесса разработки программного продукта.

Мера сходства (близости) показателей жизненного цикла ПП с нормативными показателями каждой из моделей ЖЦ определяется по расстоянию Хемминга $\rho(x_{ij}^r, x_{ij}^p) = |x_{ij}^r - x_{ij}^p|$, которое представляет собой количество совпадений (либо несовпадений) показателей проекта с показателями нормативных моделей ЖЦ ПП. Задача выбора модели жизненного цикла ПП для реализации программного проекта сводится к минимизации выражения (3.1), при этом количество совпадений свойств нормативных показателей ПП и свойств показателей программного проекта будет максимальным:

$$R_\rho = \min_{\rho = R} \sum_{i=1}^{4} \alpha_\rho \sum_{j=1}^{n} |x_{ij}^r - x_{ij}^\rho|, \qquad (3.1)$$

где α_ρ — коэффициент относительной важности ρ-й группы характеристик.

Процедуры выбора модели можно представить в виде следующей последовательности действий:

1) подробное ознакомление с достоинствами, недостатками и областями применения каждой из моделей ЖЦ разработки ПП;

2) определение коэффициента относительной важности каждой группы характеристик α_i, $\sum_{i=1}^{4} \alpha_i = 1$;

3) выбор наиболее существенных параметров по каждой из четырех групп характеристик $X_\rho = \{x_{ij}^\rho\}$, $i = \overline{1,4}$, $j = \overline{1,n}$;

4) оценивание каждого параметра $x_{ij} \in X_\rho$:

$$x_{ij}^\rho = \begin{cases} 1, \text{если } j \text{ - й параметр в } i \text{ - й группе существенен для проекта;} \\ 0, \text{в противном случае;} \end{cases}$$

3) вычисление по формуле (3.1) интегрального показателя соответствия характеристик моделей особенностям проекта. Модель с минимальным значением R_ρ, $\rho = \overline{1,6}$ может быть принята в качестве базовой модели ЖЦ разработки.

Модель, выбранная для какого-либо проекта, должна обеспечивать потребности организации, соответствовать типу выполняемых работ, а также учитывать навыки специалистов и инструментальные средства проектирования и разработки, которые у них имеются.

Контрольные вопросы

1. Раскройте содержание каскадной модели ЖЦ разработки ПП.
2. Раскройте содержание V-образной модели ЖЦ разработки ПП.
3. Раскройте содержание модели прототипирования разработки ПП.
4. Раскройте содержание модели быстрой разработки приложений ПП.
5. Раскройте содержание инкрементной модели ЖЦ разработки ПП.
6. Раскройте содержание спиральной модели ЖЦ разработки ПП.
7. Раскройте содержание методики выбора модели ЖЦ разработки ПП.

4. ИНИЦИАЦИЯ ПРОГРАММНОГО ПРОЕКТА

4.1. Разработка идеи программного проекта и оценка ее привлекательности

Любая IT-компания при создании программного продукта придерживается одной из бизнес-моделей деятельности: разработка и продвижение собственных программных продуктов (продуктовая или тиражная модель) или разработка уникального ПП «под заказ» (заказная модель). Применение каждой бизнес-модели имеет свои особенности.

При использовании заказной модели существует риск разработать «под заказ» программный продукт, работающий с ошибками, непригодный для сопровождения и модификации. Кроме того, возможен риск «затянуть» проект или попасть в опасную зависимость от постоянно меняющихся требований заказчика и т. д.

Использование продуктовой модели предполагает востребованность на рынке предлагаемого к разработке продукта, что обеспечит целесообразность последующего тиражирования ПП и его успешную реализацию. С точки зрения оценки бизнеса компании-разработчика продуктовая модель более перспективна, в силу того что сама компания является непосредственным производителем новых продуктов. При этом малыми ресурсами могут быть созданы инновационные продукты, имеющие большой экономический и коммерческий потенциал.

Практическое использование и продуктовой, и заказной моделей создания ПП связано с решением трудной и важной задачи **разработки идей[1] программного проекта, выбора и оценки наиболее привлекательной идеи,** которая будет реализована командой проекта в виде конкретного программного продукта. Ошибки, допущенные при решении данной задачи, существенно

[1] Идея — замысел, определяющий содержание чего-нибудь; мысль, намерение, план / Толковый словарь под ред. С.И. Ожегова и Н.Ю. Шведовой.

влияют на успех проекта в целом. Поэтому качественный детальный анализ рыночных и финансовых факторов, временных параметров реализации идеи позволит уже на начальном этапе выполнения программного проекта отклонить малоэффективные варианты. Решение этой задачи рассматривается **на фазе инициации программного проекта,** управленческие процессы которой подробно представлены в международном стандарте PMBOK (см. подразд. 1.2): выявление потребностей общества в предлагаемом проекте, формирование целей, ограничений и содержания проекта, определение сроков реализации проекта, потребности в финансовых ресурсах, выделение внутренних и внешних заинтересованных сторон, которые будут взаимодействовать и влиять на общий результат проекта. Недостаточное внимание именно к этой группе управленческих процессов неизбежно приводит к существенным проблемам разработки и дальнейшей коммерциализации продукта.

В процессе инициации проекта осуществляются следующие действия:

1) создается творческое ядро в команде по разработке будущего программного продукта;

2) формируется ряд привлекательных идей программного проекта и производится их оценивание;

3) на основе наиболее привлекательных идей разрабатываются концепции реализации программного проекта, в которых определяется целесообразность и выгодность проекта для компании, выбирается продуктово-рыночное направление продвижения ПП, определяются источники привлечения инвестиций и т. д.;

4) проводится отбор наиболее перспективной концепции разработки будущего программного продукта.

Процедура оценки привлекательности идеи программного проекта содержит следующие этапы: подготовительный, этап генерации идей, этап обсуждения и оценки привлекательных идей.

На подготовительном этапе инициаторами проекта формируется первоначальный состав команды проекта, при этом один из инициаторов принимает на себя роль руководителя проекта.

В состав группы для обсуждения и оценки идеи, кроме членов команды проекта, желательно включить представителей потенциальной целевой аудитории, специалистов-практиков по вопросам продвижения, внедрения и сопровождения ПП. На этом же этапе могут быть определены ограничения и допущения по будущему программному продукту, в частности по типу рынка, сфере применения, категории потенциальных пользователей и т. д.

На этапе генерации идей происходит непосредственная разработка привлекательных для будущих потребителей и инвесторов идей программного продукта, обеспечивающих их потребности и коммерческие интересы. Основная задача группы «генераторов» — «выдать» за отведенное время как можно больше идей (в том числе фантастических, явно ошибочных и шутливых).

Процедура коллективной генерации и оценивания привлекательных идей программного проекта может быть организована с использованием метода мозгового штурма, который представляет собой групповое обсуждение конкретной проблемы с целью выработки вариантов ее разрешения. При использовании метода необходимо соблюдать следующие принципы [9]:

1) ***сознательное генерирование как можно большего количества идей***. При этом рекомендуется вначале определить два крайних варианта решения: например, *первое* — принять в качестве идеи развитие уже существующих в компании программных проектов; *второе* — принять радикальное решение по разработке нового ПП, который будет конкурировать с продуктами ведущих фирм. Далее генерировать варианты решений внутри этого интервала. Не рекомендуется при генерации идей оценивать рынок, конкурентоспособность, возможные проблемы продвижения, а также думать о технологиях реализации идеи;

2) учет существования факторов, как тормозящих работу группы обсуждения, так и способствующих ей (психологическую несовместимость экспертов, инертность мышления, эмоциональное и физическое состояние и др.). Принятие во внимание этих факторов позволяет эксперту не только самому продуктивно генерировать варианты, но и создавать условия для успешной деятельности других участников;

3) проведение предварительного «грубого» отсеивания идей при значительном их количестве и ограниченном времени на принятие решения без сравнения идей по количественным показателям, а лишь с проверкой присутствия в идеях некоторых привлекательных качеств, подчеркивающих их оригинальность;

4) запрещение критики в любом виде, не только явной словесной, но и скрытой в форме скептических улыбок, мимики, жестов и др.;

5) поддержание свободных и доброжелательных отношений между участниками обсуждения. Создание условий обсуждения, при которых идея, выдвинутая одним участником мозгового штурма, подхватывалась бы и развивалась другим.

Отдельным направлением поиска перспективных идей является поиск в Интернете информации о недовольстве пользователей функционалом программных продуктов ведущих компаний. В этом случае необходимо подумать о разработке компонентов, позволяющих решить эти проблемы. Процессом решения задачи управляет руководитель проекта, который обеспечивает соблюдение правил классического мозгового штурма.

Результатом стадии генерации является множество идей, сформулированных в виде нескольких предложений. При описании каждой идеи следует отразить:

- необходимость или потребность в будущем ПП;
- основные результаты проекта;
- тип рынка и потенциальных потребителей, сроки вывода продукта на рынок.

Обсуждение и выбор привлекательных идей могут быть проведены в форме дискуссии — открытого коллективного обсуждения привлекательности сгенерированных идей. Основной задачей дискуссии является всесторонний анализ положительных и отрицательных моментов каждой идеи. Поскольку на этой стадии важно понять, действительно ли существует проблема, которую команда разработчиков собирается решать при помощи своего продукта, обсуждение и оценку идеи желательно проводить с представителями потенциальной целевой аудитории в форме собеседования или анкетирования.

Отбор наиболее привлекательной идеи можно производить как по итогам коллективного обсуждения, так и по результатам индивидуальной оценки идей каждым экспертом. В этом случае эксперту предлагается проранжировать идеи по степени убывания их привлекательности.

Ранжирование заключается в упорядочении объектов по степени их предпочтения [9]. В зависимости от вида отношений между объектами возможны несколько вариантов упорядочения. Если между объектами нет одинаковых по сравнительным показателям элементов, т. е. отсутствует отношение безразличия, то можно говорить, что присутствует отношение строгого порядка $x_1 > x_2 > x_3 \ ... > x_n$.

Для отображения вида отношений могут использоваться действительные числа натурального ряда, связанные между собой отношением неравенства: $C_1 > C_2 > C_3 > C_i > ... > C_n$, где $C_i = f(x_i) = i$. Числа $C_1, C_2, ..., C_n$ в этом случае называются рангами.

Если между рядом объектов существуют отношения безразличия, то будем иметь упорядочение нестрогого порядка $x_1 > x_2 > x_3 \sim x_4 \sim x_5 > ... > x_n$. При ранжировании наиболее предпочтительному объекту присваивается ранг, равный единице, второму по предпочтительности — ранг, равный двум и т. д.

При эквивалентности объектов $x_1 \infty x_2 \infty x_3$ каждому из них назначается одинаковый ранг, равный среднему арифметическому значению номеров объектов в упорядоченной последовательности. В примере ранг объектов x_1, x_2, x_3 определяется как $C_1 = C_2 = C_3 = (1 + 2 + 3)/3 = 2$.

Таким образом, при ранжировании идей каждым экспертом получается матрица $C = \|C_{i,s}\|$ размерности $n \times m$, где m — число экспериментов; n — количество идей. Интегральная оценка привлекательности идеи определяется суммированием рангов, выставленных экспертами.

Результатом стадии отбора является несколько привлекательных идей, жизнеспособность которых обоснована экспертами.

Каждая идея, ориентированная на определенные группы потребителей, должна быть далее воплощена в конкретный вариант концепции программного проекта. Если в ходе дискуссии эксперты не пришли к единому мнению по отбору конкурентоспособных идей программного продукта, то необходимо проведение повторного мозгового штурма, при этом цель проекта должна быть переформулирована, а ограничения изменены.

4.2. Разработка концепции проекта и оценка ее перспективности

Концепция проекта разрабатывается на основе анализа потребностей бизнеса и предназначена для подтверждения и согласования всеми участниками проекта единого видения целей, задач и результатов проекта. Содержание концепции зависит от типа создаваемого программного продукта (заказного либо рыночного). При заказной разработке ПП основными разделами концепции являются цели и ограничения проекта, содержание проекта, требования и ожидания заказчика, бюджет и сроки проекта, ответственность сторон. При рыночной разработке в составе концепции следует подробно привести бизнес-обоснование необходимости проекта. В концепции должны быть изложены в понятной форме вопросы, которые помогут глубже понять содержание идеи и проверить ее на практичность и эффективность реализации [10].

Концепция программного проекта содержит следующие разделы:

1) бизнес-обоснование потребности или необходимости в разработке ПП;

2) цели, ограничения и содержание программного проекта;

3) основные сегменты рынка и потенциальные пользователи;

4) экономика программного проекта;

5) потенциал исполнителей;

6) ожидаемые риски программного проекта.

Бизнес-обоснование потребности или необходимости в разработке программного продукта

При определении необходимости или потребности для общества будущего программного продукта предлагается учитывать четыре аспекта [11]:

1) ***коммерческую привлекательность*** (появление новой потребности рынка в программных продуктах или услугах, изменение потребительского спроса на ПП);

2) ***изменение законодательства,*** регулирующего состав и содержание бизнес-процессов в конкретных предметных областях;

3) ***научно-технический прогресс*** в области развития программно-аппаратных средств информатизации (например, проект по разработке ПП для портативных устройств);

4) ***потребность отраслей социальной сферы*** в информационно-коммуникационных услугах, финансирование которых берет на себя государство.

Далее в концепции следует показать уникальность, коммерческую привлекательность и стратегию продвижения будущего программного продукта на рынок [12], что потребует рассмотрения ряда вопросов, а именно:

- какие проблемы существуют у потенциального заказчика, насколько значимо для него решение данных проблем;
- зачем нужен данный продукт, какова его основная идея, что собираетесь разрабатывать и какие требования к ПП могут предъявлять потенциальные пользователи;
- кому собираетесь предлагать ваш продукт или услугу;
- какой полезный эффект может извлечь потенциальный потребитель от использования продукта;
- чем отличается ваш продукт от продуктов конкурентов;
- обладает ли продукт какими-либо новыми уникальными особенностями;
- сколько времени уйдет на разработку такого продукта;
- если в вашем продукте нет ничего особенно выдающегося, то что же в нем может привлечь покупателя;
- как вы собираетесь привлечь покупателей;

• с кем вы собираетесь конкурировать в выбранных сегментах рынка, знаете ли вы производителей аналогичных продуктов, за какую цену продаются аналогичные продукты;

• каким образом вы собираетесь продавать продукт;

• возможные каналы поставки продукта, как будут организованы возникающие взаимоотношения с пользователями.

Цели, ограничения и содержание программного проекта

Цели программного проекта представлены в концепции в виде желаемых результатов, достигаемых командой проекта при его успешной реализации. Формулировки целей должны быть конкретными, измеримыми, согласованными, реальными, ограниченными по срокам (см. подробно п. 1.1.3).

Пример

Разработать и вывести на рынок во 2-м квартале ____ года программный продукт _____ при заданном уровне бюджета в размере _____ рублей.

Ограничениями проекта могут быть обязательная сертификация создаваемого продукта или оказываемой услуги; поддержка определенных форматов данных; использование при разработке конкретных стандартов и др.

Содержание концепции на этапе инициации описывается в виде функциональных и нефункциональных требований к программному продукту, определяемых командой проекта на данный момент времени. Функциональные требования должны отражать потребности потенциальных пользователей, а нефункциональные — характеристики качества ПП. Уровень детализации требований должен быть достаточным для представления и краткого описания архитектуры будущего программного продукта как совокупности программных модулей (компонентов) с перечислением их функционала.

Основные сегменты рынка и потенциальные пользователи

В данном разделе концепции определяется тип рынка (промышленный или потребительский) и приводится его описание с выделением множества групп потенциальных потребителей, которым может быть интересен продукт, отвечающий их запросам.

При сегментировании промышленного рынка в качестве переменных сегментирования можно использовать набор характеристик: географическое положение, форму собственности, отраслевую принадлежность, размер организации-потребителя, состав закупаемых услуг, уровень развития информационно-коммуникационных технологий и т. д. На потребительском рынке потенциальных пользователей можно дифференцировать ***по социально-психологическим особенностям*** [13]:

- новаторы, благополучные, оптимисты, рассудительные, престижные, западники, молодые, фаталисты, аутсайдеры;
- независимые, престижные, конформисты, экономные, обычные;
- первопроходцы, прагматики, консерваторы, копуши;
- инноваторы, первопроходцы, активное большинство, опоздавшее большинство, ленивцы.

Экономика программного проекта

В данном разделе концепции приблизительно оцениваются трудозатраты на разработку программных продуктов, определяется бюджет проекта, рыночная цена продажи одной лицензии, минимально допустимый объем продаж, покрывающий расходы бюджета. Методы определения трудозатрат и бюджета проекта описаны в разделе 7.

Минимально допустимое количество продаж, покрывающее все затраты на разработку первой версии программного продукта, его продвижение на рынок и поставку потребителям, не принося при этом ни прибыли, ни убытков, получило название ***«точка безубыточности»***.

Основным методом определения точки безубыточности является **CVP**-анализ (*Cast Value Profit* — затраты, объем, прибыль), основанный на оценке соотношений затрат, выручки и прибыли [14]. В денежном выражении точка безубыточности определяется по формуле

$$t_b = \frac{a}{(s \cdot x - bx) \cdot s \cdot x}, \qquad (4.1)$$

где x — количество продаж;

s — рыночная цена продажи единицы продукции;

a — величина фиксированных расходов;

b — величина переменных издержек на единицу продукции.

Количество продаж, при котором достигается точка безубыточности (прибыль фирмы равна нулю), определяется как

$$x_0 = \frac{a}{s-b}. \tag{4.2}$$

Если объем рынка определен, можно узнать ***рыночную цену продажи одной лицензии ПП*** при нулевом уровне прибыли:

$$s_0 = \frac{a + bx_0}{x_0}. \tag{4.3}$$

Если фирма стремится получить дополнительную прибыль (сверх нормативной) и рыночная цена известна, то количество продаж при заданном уровне прибыли P_0 и рыночной цене s_0 можно определить по формуле

$$x_p = \frac{P_0 + a}{s_0 - b}. \tag{4.4}$$

Чистая прибыль фирмы определяется как разница между выручкой и переменными и постоянными издержками:

$$P = sx - (a + bx) = (s - b)x - a. \tag{4.5}$$

Графическая интерпретация определения и анализа точки безубыточности представлена на рис. 4.1.

Понятие точки безубыточности является одновременно и неким критерием оценки эффективности концепции проекта.

Потенциал исполнителей

В концепции приводится следующая информация:

1) кратко описывается история успеха команды проекта;

2) отмечается наличие сертификатов и других документов, подтверждающих потенциал сотрудников;

3) указывается количество специалистов, требуемых для реализации проекта;

4) определяется в случае необходимости перечень работ для передачи на аутсорсинг.

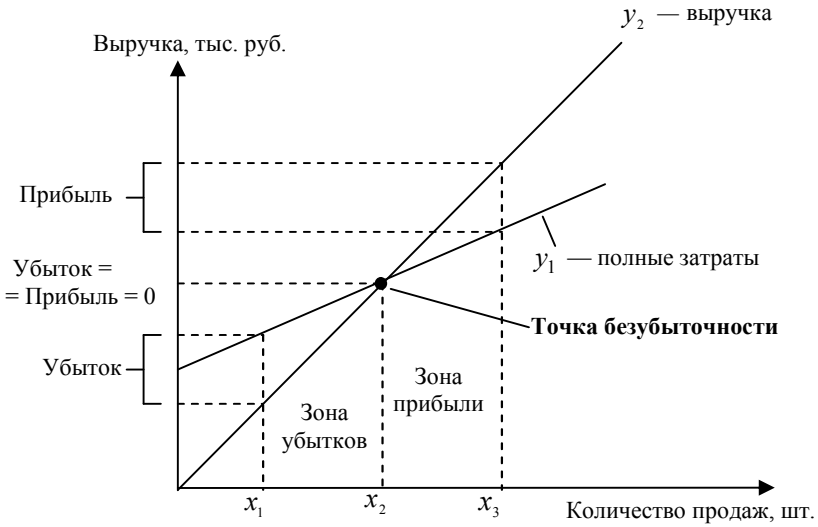

Рис. 4.1. Графическая интерпретация определения
и анализа точки безубыточности

Ожидаемые риски программного проекта

Наличие большой неопределенности в достижении конечных целей проекта объективно требует проведения мероприятий по выявлению и оценке возможных рисков.

Под **риском программного проекта** будем понимать наступление события, которое может возникнуть в процессе реализации программного проекта и негативно повлиять на степень достижения целей проекта.

Оценка риска реализации проекта позволяет определить реальную возможность команды выполнить проект в приемлемые сроки с затребованным потенциальным пользователем функционалом и в пределах имеющихся финансовых ресурсов.

4.3. Выбор перспективной концепции программного проекта

4.3.1 Оценка перспективности концепции методом экспертных оценок

В литературе описывается несколько подходов к выбору и обоснованию наиболее приоритетных для компании концепций проектов. Так, в [11] для решения этой задачи предлагается рассматривать прежде всего коммерческую необходимость проекта, оценивая ее с помощью методов финансового анализа. В качестве критериев отбора концепций проекта рекомендуется рассматривать такие показатели, как *период окупаемости, поток денежных средств, внутренняя норма прибыли,* составляющие основу анализа затрат по проекту и его результатов. Очевидно, что использование этого подхода для оценки перспективности концепции на ранней стадии программного проекта возможно только при достаточно высокой степени достоверности численных значений показателей.

В другом подходе к оценке концепций программных проектов применяется метод экспертных оценок, основная идея которого состоит в использовании интеллекта специалистов в области разработки программных проектов для поиска решений в слабоформализованных задачах при наличии большого количества качественной информации, противоречивых целей, критериев и ограничений.

Постановка задачи оценки перспективности концепции программного проекта может быть представлена в следующем виде.

Пусть разработано множество концепций на создание программного продукта $I = \{1,\ 2,\ ...,\ i, ..., m\}$.

Каждую концепцию предлагается оценивать по множеству критериев $H = \{1, 2, ..., h, ..., n\}$. Важность каждого критерия для оценки перспективности концепции может быть задана в виде множества относительной значимости критериев $Q = \{q_1, q_2, ..., q_h, ..., q_n\}$.

Для оценки перспективности концепции создается экспертная группа, в которой множество экспертов описывается как $S = \{1,\ 2,\ ...,s,...,d\}$. Относительная значимость (важность) каждого эксперта задается в виде множества $K = \{k_1, k_2, ..., k_s, ...,\ k_d\}$.

Требуется определить численные значения перспективности каждой концепции и упорядочить их по убыванию величин с помощью выражения

$$x_i = \sum_{h=1}^{n} \sum_{s=1}^{d} q_h k_s x_{is}^h,\ i = \overline{1,m}. \qquad (4.6)$$

При этом оценивание перспективности концепции проекта x_{is}^h производится в шкале [0–1].

Наиболее сложной задачей при проведении экспертного оценивания перспективности проектов является отбор критериев. В настоящее время в литературе не существует единого подхода к решению этой задачи.

В [1] оценку концепции проекта предлагается определять на основе трех характеристик: ***финансовой ценности, стратегической ценности, уровня рисков.*** Качественная шкала оценки каждой из характеристик имеет следующий вид: высокая, выше среднего, средняя, низкая.

В [15] приводится достаточно подробный обзор подходов к оценке перспективности инвестиционных проектов и научно-технических программ. При этом рекомендуется к использованию следующий набор критериев:

практическая востребованность — направленность проекта на решение реальных первоочередных проблем пользователя;

обозримость — приемлемость сроков реализации проекта;

эффективность — получение максимального социального либо экономического эффекта при разумных затратах на реализацию проекта;

коммерческая привлекательность — принципиальная возможность получения гарантированной прибыли от тиражирования программного продукта;

потенциал исполнителей — наличие в команде проекта достаточного числа специалистов, имеющих опыт создания и внедрения подобных проектов;

реализуемость — соответствие содержания проекта реальным бизнес-процессам, наличие требуемых объемов финансирования и возможности команды проекта;

научно-технический уровень — наличие в проекте новых, претендующих на получение патента технических решений, не имеющих аналогов.

Субъективный характер восприятия экспертами перспективности концепции программного проекта приводит к расхождению их мнений в оценивании проекта. В связи с этим возникает необходимость количественной оценки степени согласованности экспертов. Для этих целей используется ***дисперсионный коэффициент конкордации*** [9], значение которого равно единице, если все оценки экспертов одинаковы, и нулю, если они различны.

Качественная интерпретация результатов экспертизы может быть приведена по шкале оценок, представленной в табл. 4.1.

Таблица 4.1

Интерпретация результатов экспертизы по оцениванию проекта

Значение коэффициента конкордации	Менее 0,3	0,3–0,5	0,5–0,7	0,7–0,9	Более 0,9
Степень согласованности экспертов	Слабая	Умеренная	Заметная	Высокая	Очень высокая

Если степень согласованности экспертов недостаточна, экспертиза может быть повторена. При этом в открытой либо закрытой дискуссии должны участвовать эксперты, имеющие крайние точки зрения. В зависимости от значения оценки перспективности проект может быть отнесен к одной из следующих категорий:

1) проект очень перспективен, если значение $x_i > 0{,}8$;

2) проект перспективен, если значение $x_i > 0{,}6$;

3) перспективность проекта следует уточнить, если значение $x_i > 0{,}4$;

4) проект не перспективен, если значение $x_i > 0{,}2$.

В первых двух случаях проект должен быть продолжен, в третьем — направлен на доработку, в последнем — отклонен.

Очевидно, что качественная и детальная проработка и оценка концепции позволит уже на начальном этапе выполнения программного проекта отклонить малоэффективные варианты. Недостаточное внимание к этой работе существенно влияет на успех проекта в целом и неизбежно приводит к существенным проблемам разработки и коммерциализации продукта. В дальнейшем материалы концепции могут служить основой для разработки бизнес-плана или технико-экономического обоснования на создание и продвижение программного продукта.

4.3.2. Модель функциональных зависимостей оценки перспективности концепции проекта

Модель оценки перспективности концепции рыночного программного проекта в формализованном можно представить в виде ориентированного графа $G = (X, U)$, где $X = \{x_i\}$ — множество вершин графа представлено набором параметров, описывающих различные характеристики концепции; $U = \{u_{ij}\}$ — множество направленных дуг графа, отражающих причинно-следственные связи между параметрами.

В истоках графа находятся первичные параметры. В стоке сети расположен результирующий (целевой) параметр оценки перспективности концепции. Значения остальных параметров зависят от первичных и вычисляются через совокупность функциональных зависимостей между ними [16].

Для четкой фокусировки будущего программного продукта под требования рынка все множество первичных параметров предлагается разбить на три группы характеристик (рис. 4.2): характеристики продукта, рынка и проекта [15].

Характеристики продукта включают параметры, отражающие конкурентоспособность программного продукта, который будет выведен на рынок. Оценка возможности реализации концеп-

ции в виде рыночного программного продукта производится по ***характеристикам проекта***, отражающим оценку процесса создания рыночного продукта. ***Характеристики рынка*** представляют собой набор параметров, показывающих интенсивность конкуренции на рынке, наличие спроса на ПП и направление развития рынка. Каждый из параметров описывается в виде показателя либо в виде атрибута[1]. При этом показатели перспективности проекта описываются в численной шкале, а атрибуты — шкале наименований.

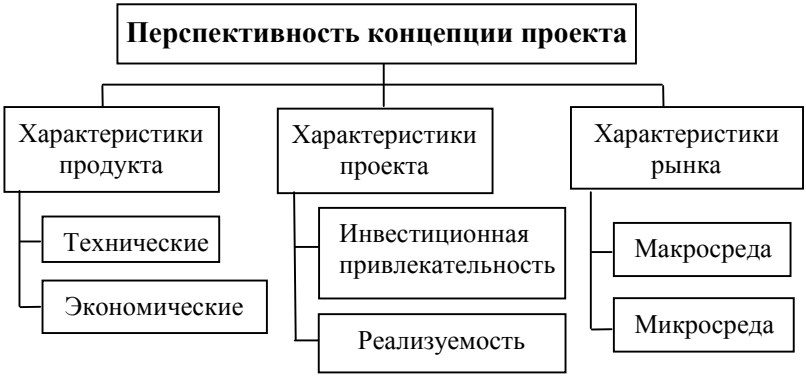

Рис. 4.2. Классификация характеристик оценки перспективности концепции программного проекта

С учетом введенной классификации параметров структуру функциональных зависимостей оценки перспективности концепции можно представить в виде графа (рис. 4.3). В первом слое графа расположены первичные параметры, позволяющие описать перспективность концепции в целом. В состав первичных предлагается включить показатели и атрибуты, представленные в табл. 4.2.

[1] Характеристика «показатель» имеет определенную размерность, т. е. может быть выражена в виде некоторой количественной оценки. Характеристика «атрибут» имеет качественное содержание и может быть описана с использованием балльной либо ранговой шкалы.

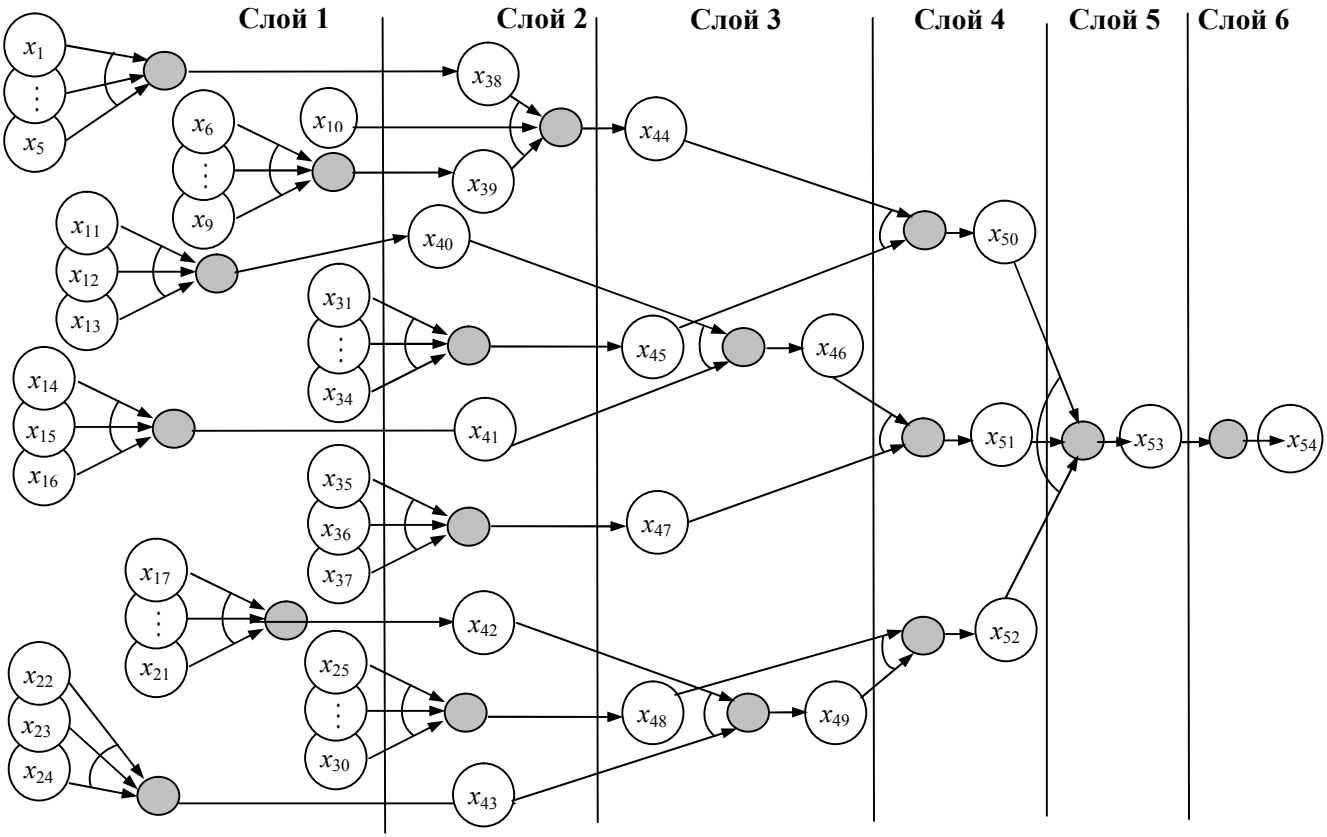

Рис. 4.3. Граф функциональных зависимостей оценки перспективности концепции программного проекта

Таблица 4.2

Состав первичных параметров и правила их оценки

1 слой – истоки		
Параметр	Наименование параметра	Значение параметра и правила оценки
x_1	Затраты на оплату труда разработчиков, тыс. руб.	$x_1 > 0$
x_2	Затраты на аренду (приобретение) офиса, тыс. руб.	> 0
x_3	Затраты на приобретение компьютерной и офисной техники, тыс. руб.	≥ 0
x_4	Затраты на приобретение лицензионного ПО, тыс. руб.	≥ 0
x_5	Затраты на обучение разработчиков, тыс. руб.	≥ 0
x_6	Оценка зрелости команды (*CMM*)	Высокий (1); средний (0,5); простая (0)
x_7	Необходимость дополнительного привлечения специалистов в команду проекта	Да — 1; нет — 0
x_8	Необходимость передачи части работ на аутсорсинг	Да — 1; нет — 0
x_9	Наличие готовых компонентов и модулей	Да — 1; нет — 0
x_{10}	Уровень риска проекта	Высокий (1); средний (0,5); низкий (0)
x_{11}	Количество конкурентов	≥ 0
x_{12}	Интенсивность конкуренции	Высокая (1); средняя (0,5); низкая (0)
x_{13}	Насыщенность рынка	Высокая (1); средняя (0,5); низкая (0)
x_{14}	Общее число потребителей на рынке	> 0
x_{15}	Число потенциальных потребителей	> 0

Продолжение табл. 4.2

Параметр	Наименование параметра	Значение параметра и правила оценки
x_{16}	Уровень платежеспособного спроса	Высокий (1); средний (0,5); низкий (0)
x_{17}	Дополнительные затраты потребителя на реорганизацию бизнес-процессов, тыс. руб.	> 0
x_{18}	Затраты на эксплуатацию ПП, тыс. руб.	≥ 0
x_{19}	Фонд оплаты труда IT-подразделения потребителя, тыс. руб.	≥ 0
x_{20}	Затраты потребителя на приобретение лицензионного программного обеспечения, тыс. руб.	≥ 0
x_{21}	Затраты потребителя на приобретение техники, тыс. руб.	≥ 0
x_{22}	Сокращение затрат потребителя на поддержку бизнес-процессов	> 25 % – «Высокая» (1); ≥ 5 % но ≤ 25 % – «Средняя»(0,5); < 5 % – «Низкая» (0)
x_{23}	Сокращение времени на принятие решений (выпуск продукции)	>25% – «Высокая» (1); ≥ 5 % но ≤ 25 % = «Средняя»(0,5); < 5 % – «Низкая» (0)
x_{24}	Увеличение дохода	>19 % – «Высокая» (1); ≥ 3 % но ≤ 10 % – «Средняя»(0,5); < 3 % – «Низкая» (0)
x_{25}	Функциональные возможности ПП	Высокий (1); средний (0,5); низкий (0)
x_{26}	Надежность ПП	Высокая (1); средняя (0,5); низкая (0)
x_{27}	Практичность ПП	Высокая (1); средняя (0,5); низкая (0)
x_{28}	Эффективность ПП	Высокая (1); средняя (0,5); низкая (0)

Окончание табл. 4.2

Параметр	Наименование параметра	Значение параметра и правила оценки
x_{29}	Сопровождаемость ПП	Высокая (1); средняя (0,5); низкая (0)
x_{30}	Мобильность ПП	Высокая (1); средняя (0,5); низкая (0)
x_{31}	Чистый приведенный доход (*NPV*)	$NPV > 0$ – «прибыльный» (1); $NPV = 0$ – «ни прибыльный, ни убыточный» (0,5); $NPV < 0$ – «убыточный» (0)
x_{32}	Индекс прибыльности (*PI*)	$PI > 1$ – «прибыльный» (1); $PI = 1$ – «ни прибыльный, ни убыточный» (0,5); $PI < 1$ – «убыточный» (0)
x_{33}	Внутренняя норма доходности (*IRR*)	$IRR > K$ – «прибыльный» (1); $IRR = K$ – «ни прибыльный, ни убыточный» (0,5); $IRR < K$ – «убыточный» (0), где K — заданная ставка дисконтирования
x_{34}	Дисконтированный период окупаемости (*DPV*)	$DPV < 12$ мес. – «прибыльный» (1); $DPV \geq 12$ мес., но < 36 – «ни прибыльный, ни убыточный» (0,5); $DPV > 36$ – «убыточный» (0)
x_{35}	Наличие государственной поддержки проекта	Да – 1, Нет – 0
x_{36}	Необходимость сертификации ПП и лицензирования деятельности по его созданию	Да – 1, Нет – 0
x_{37}	Уровень стабильности финансового рынка	Высокий (1); средний (0,5); низкий (0)

Очевидно, что предлагаемый состав первичных параметров можно рассматривать как базовый. В каждом конкретном случае могут возникнуть ситуации, когда часть параметров невозможно объективно оценить, а некоторые параметры окажутся не существенными для рассматриваемого программного проекта.

В первом слое графа расположены первичные параметры, позволяющие наиболее комплексно оценить все характеристики концепции, влияющие на перспективность рыночного ПП; x_1 — затраты на оплату труда разработчиков; x_2 — затраты на аренду (приобретение) офиса; x_4 — затраты на приобретение лицензионного ПО; …, x_8 — необходимость передачи части работ на аутсорсинг; x_{11} — количество конкурентов; x_{12} — интенсивность конкуренции; x_{13} — насыщенность рынка; …, x_{35} — наличие государственной поддержки проекта; …. x_{37} — уровень стабильности финансового рынка.

Второй слой графа составляют оценки параметров, отражающие затраты на разработку ПП, готовность команды проекта, уровень конкуренции на рынке, емкость и объем рынка, совокупные затраты потребителя при эксплуатации и др. (табл. 4.3).

Третий слой графа составляют оценки таких параметров, как вероятность успешной реализации проекта, инвестиционная привлекательность, привлекательность целевого сегмента, привлекательность макросреды и т. д. Значения данных параметров формируются с помощью правил-продукций (табл. 4.4).

В четвертом слое графа производится оценка трех основополагающих характеристик концепции: перспективности проекта, конкурентоспособности ПП, привлекательность рынка (табл. 4.5).

В пятом слое графа определяется интегральная характеристика «Оценка перспективности концепции программного проекта», отражающая количественную оценку уровня перспективности концепции. В качестве математического выражения интегрированной оценки предлагается использовать аддитивную свертку частных критериев с весовыми коэффициентами:

$$x_{53} = a_1 x_{50} + a_2 x_{51} + a_3 x_{52},$$

где a_1, a_2, a_3 — весовые коэффициенты показателей, задаваемые командой экспертов при оценке.

Таблица 4.3

Состав показателей второго слоя и правила их оценки

2-й слой			
Параметр	Наименование параметра	Значение параметра, тыс. руб.	Способ определения значений параметра
x_{38}	Затраты на разработку ПП	Высокая (1); средняя (0,5); низкая (0)	$x_1 + x_2 + x_3 + x_4 + x_5$ ...«Высокая»>200 – (1); «Средняя»≥ 80 но ≤ 200 – (0.5) ; «Низкая» < 80 – (0)
x_{39}	Готовность команды проекта	Высокая (1); средняя (0,5); низкая (0)	Если $x_6 = 0$ или $x_6 = 0{,}25$ и $x_7 = 0$ и $x_8 = 0$ и $x_9 = 0$, то $x_{39} = 0{,}5$; Если $x_6 = 0$ или $x_6 = 0{,}25$ и $x_7 = 0$ и $x_8 = 0$ и $x_9 = 1$, то $x_{39} = 1$; Если $x_6 = 0$ или $x_6 = 0{,}25$ и $x_7 = 0$ и $x_8 = 1$ и $x_9 = 0$, то $x_{39} = 0$; Если $x_6 = 0$ или $x_6 = 0{,}25$ и $x_7 = 0$ и $x_8 = 1$ и $x_9 = 1$, то $x_{39} = 0{,}5$; Если $x_6 = 0$ или $x_6 = 0{,}25$ и $x_7 = 1$ и $x_8 = 0$ и $x_9 = 0$, то $x_{39} = 0$; Если $x_6 = 0$ или $x_6 = 0{,}25$ и $x_7 = 1$ и $x_8 = 1$ и $x_9 = 1$, то $x_{39} = 0{,}5$; Если $x_6 = 0{,}5$ или $x_6 = 0{,}75$ и $x_7 = 0$ и $x_8 = 0$ и $x_9 = 0$, то $x_{39} = 1$; Если $x_6 = 0{,}5$ или $x_6 = 0{,}75$ и $x_7 = 0$ и $x_8 = 0$ и $x_9 = 1$, то $x_{39} = 1$; Если $x_6 = 0{,}5$ или $x_6 = 0{,}75$ и $x_7 = 0$ и $x_8 = 1$ и $x_9 = 0$, то $x_{39} = 0$; Если $x_6 = 0{,}5$ или $x_6 = 0{,}75$ и $x_7 = 0$ и $x_8 = 1$ и $x_9 = 1$, то $x_{39} = 0{,}5$; Если $x_6 = 0{,}5$ или $x_6 = 0{,}75$ и $x_7 = 1$ и $x_8 = 0$ и $x_9 = 0$, то $x_{39} = 0{,}5$; Если $x_6 = 0{,}5$ или $x_6 = 0{,}75$ и $x_7 = 1$ и $x_8 = 1$ и $x_9 = 1$, то $x_{39} = 1$; Если $x_6 = 1$ и $x_7 = 0$ и $x_8 = 0$ и $x_9 = 0$, то $x_{39} = 1$; Если $x_6 = 1$ и $x_7 = 0$ и $x_8 = 0$ и $x_9 = 1$, то $x_{39} = 1$;

Продолжение табл. 4.3

Параметр	Наименование параметра	Значение параметра, тыс. руб.	Способ определения значений параметра
x_{39}	Готовность команды проекта	Высокая (1); средняя (0,5); низкая (0)	Если $x_6 = 1$ и $x_7 = 0$ и $x_8 = 1$ и $x_9 = 0$, то $x_{39} = 0{,}5$; Если $x_6 = 1$ и $x_7 = 0$ и $x_8 = 1$ и $x_9 = 1$, то $x_{39} = 1$; Если $x_6 = 1$ и $x_7 = 1$ и $x_8 = 0$ и $x_9 = 0$, то $x_{39} = 0{,}5$; Если $x_6 = 1$ и $x_7 = 1$ и $x_8 = 1$ и $x_9 = 1$, то $x_{39} = 0{,}5$
x_{40}	Уровень конкуренции на рынке	Высокий (1); средний (0,5); низкий (0)	Если $x_{11} = 1$ и $x_{12} = 0{,}5$ и $x_{13} = 1$, то $x_{40} = 1$; Если $x_{11} = 1$ и $x_{12} = 0$ и $x_{13} = 0$, то $x_{40} = 0$; Если $x_{11} = 0{,}5$ или $x_{11} = 0$ и $x_{12} = 1$ и $x_{13} = 0$, то $x_{40} = 0{,}5$; Если $x_{11} = 0$ или $x_{11} = 0{,}5$ и $x_{12} = 0$ и $x_{13} = 1$, то $x_{40} = 0{,}5$; Если $x_{11} = 0{,}5$ или $x_{11} = 0$ и $x_{12} = 0{,}5$ и $x_{13} = 0$, то $x_{40} = 0$; Если $x_{11} = 1$ и $x_{12} = 1$ или $x_{12} = 0{,}5$ и $x_{13} = 0$, то $x_{40} = 0{,}5$; Если $x_{11} = 1$ и $x_{12} = 0$ или $x_{12} = 0{,}5$ и $x_{13} = 1$ или $x_{13} = 0{,}5$, то $x_{40} = 0{,}5$; Если $x_{11} = 0$ и $x_{12} = 1$ и $x_{13} = 1$ или $x_{13} = 0{,}5$, то $x_{40} = 1$; Если $x_{11} = 0$ или $x_{11} = 0{,}5$ и $x_{12} = 0$ и $x_{13} = 0{,}5$ или $x_{13} = 0$, $x_{40} = 0$; Если $x_{11} = 1$ или $x_{11} = 0{,}5$ и $x_{12} = 1$ и $x_{13} = 0{,}5$ или $x_{13} = 1$, то $x_{40} = 1$; Если $x_{11} = 0$ или $x_{11} = 0{,}5$ и $x_{12} = 0{,}5$ и $x_{13} = 0{,}5$ или $x_{13} = 1$, то $x_{40} = 0{,}5$
x_{41}	Емкость и объем рынка	Высокая (1); средняя (0,5); низкая (0)	Если $x_{14} = 1$ или $x_{14} = 0{,}5$ и $x_{15} = 1$ и $x_{16} = 1$ или $x_{16} = 0{,}5$, то $x_{41} = 1$; Если $x_{14} = 1$ и $x_{15} = 0$ или $x_{15} = 0{,}5$ и $x_{16} = 1$, то $x_{41} = 1$; Если $x_{14} = 0{,}5$ и $x_{15} = 0$ или $x_{15} = 0{,}5$ и $x_{16} = 1$, то $x_{41} = 1$;

Окончание табл. 4.3

Параметр	Наименование параметра	Значение параметра, тыс. руб.	Способ определения значений параметра
x_{41}	Емкость и объем рынка	Высокая (1); средняя (0,5); низкая (0)	Если $x_{14}=0{,}5$ и $x_{15}=0$ или $x_{15}=0{,}5$ и $x_{16}=0$, то $x_{41}=0$; Если $x_{14}=0{,}5$ и $x_{15}=0$ или $x_{15}=0{,}5$ и $x_{16}=0{,}5$ то $x_{41}=0{,}5$; Если $x_{14}=0$ и $x_{15}=1$ или $x_{15}=0{,}5$ и $x_{16}=0$, то $x_{41}=0$; Если $x_{14}=0$ и $x_{15}=1$ или $x_{15}=0{,}5$ и $x_{16}=1$, то $x_{41}=1$; Если $x_{14}=1$ и $x_{15}=0$ и $x_{16}=0{,}5$, то $x_{41}=0{,}5$; Если $x_{14}=1$ и $x_{15}=0$ и $x_{16}=0$, то $x_{41}=0$; Если $x_{14}=0$ и $x_{15}=1$ и $x_{16}=0$, то $x_{41}=0{,}5$; Если $x_{14}=0$ и $x_{15}=0{,}5$ и $x_{16}=0{,}5$ то $x_{41}=0{,}5$; Если $x_{14}=0$ и $x_{15}=0$ и $x_{16}=1$, то $x_{41}=0{,}5$; Если $x_{14}=0$ и $x_{15}=0$ и $x_{16}=0{,}5$ или $x_{16}=0$, то $x_{41}=0$
x_{42}	Совокупные затраты потребителя при эксплуатации ПП	Высокая >20000 – (1); средняя ≥ 50000, но ≤ 200000 – (0.5); низкая <50000 – (0)	$x_{17}+x_{18}+x_{19}+x_{20}+x_{21}$
x_{43}	Эффект от внедрения	Высокий >2 –(1); средний ≥ 1, но ≤ 2 – (0,5); низкий <1 – (0)	$x_{22}+x_{23}+x_{24}$

Таблица 4.4

Состав третьего слоя показателей и правила их оценки

3 слой			
Параметр	Наименование параметра	Значение параметра	Способ определения значений параметра
x_{44}	Вероятность успешной реализации	Высокая (1); средняя (0,5); низкая (0)	Если $x_{38}=1$ и $x_{39}=1$ и $x_{10}=1$, то $x_{44}=0,5$; Если $x_{38}=0,5$ и $x_{39}=0,5$ и $x_{10}=0$, то $x_{44}=1$; Если $x_{38}=1$ или $x_{38}=0,5$ и $x_{39}=0,5$ и $x_{10}=0,5$, то $x_{44}=0,5$; Если $x_{38}=1$ или $x_{38}=0,5$ и $x_{39}=0$ и $x_{10}=0,5$, то $x_{44}=0$; Если $x_{38}=1$ и $x_{39}=0$ или $x_{39}=0,5$ и $x_{10}=1$, то $x_{44}=0$; Если $x_{38}=1$ и $x_{39}=0,5$ или $x_{39}=1$ и $x_{10}=0,5$ или $x_{10}=0$, то $x_{44}=1$; Если $x_{38}=0,5$ или $x_{38}=0$ и $x_{39}=0$ и $x_{10}=0,5$ или $x_{10}=1$, то $x_{44}=0$; Если $x_{38}=0,5$ или $x_{38}=0$ и $x_{39}=0,5$ или $x_{39}=1$ и $x_{10}=1$, то $x_{44}=0,5$; Если $x_{38}=0,5$ или $x_{38}=1$ и $x_{39}=1$ и $x_{10}=0,5$ или $x_{10}=0$, то $x_{44}=1$
x_{45}	Инвестиционная привлекатель-ность ПП	Высокая $>2-(1)$; средняя ≥ 1, но $\leq 2-(0,5)$; низкая $<1-(0)$	$x_{31}+x_{32}+x_{33}+x_{34}$
x_{46}	Привлекатель-ность целевого сегмента	Высокая (1); средняя (0,5); низкая (0)	Если $x_{42}=1$ и $x_{43}=1$ или $x_{43}=0,5$, то $x_{49}=0,5$; Если $x_{42}=0,5$ или $x_{42}=0$ и $x_{43}=1$, то $x_{49}=1$; Если $x_{42}=1$ и $x_{43}=0$, то $x_{49}=0,5$; Если $x_{42}=0,5$ или $x_{42}=0$ и $x_{43}=0,5$, то $x_{49}=0,5$; Если $x_{42}=0,5$ или $x_{42}=0$ и $x_{43}=0$, то $x_{49}=0$

Окончание табл. 4.4

Параметр	Наименование параметра	Значение параметра	Способ определения значений параметра
x_{47}	Привлекательность рыночной макросреды	Высокая (1); средняя (0,5); низкая (0)	Если $x_{35}=1$ и $x_{36}=1$ или $x_{36}=0$ и $x_{37}=1$, то $x_{47}=1$; Если $x_{35}=1$ или $x_{35}=0$ или $x_{36}=1$ и $x_{37}=0$, то $x_{47}=0$; Если $x_{35}=0$ и $x_{36}=0$ и $x_{37}=1$ или $x_{37}=0{,}5$, то $x_{47}=0{,}5$; Если $x_{35}=1$ и $x_{36}=1$ и $x_{37}=0{,}5$, то $x_{47}=0{,}5$; Если $x_{35}=1$ и $x_{36}=0$ и $x_{37}=0{,}5$, то $x_{47}=1$; Если $x_{35}=1$ и $x_{36}=0$ и $x_{37}=0$, то $x_{47}=0{,}5$; Если $x_{35}=0$ и $x_{36}=1$ и $x_{37}=1$, то $x_{47}=0{,}5$; Если $x_{35}=0$ и $x_{36}=1$ и $x_{37}=0{,}5$, то $x_{47}=0$; Если $x_{35}=0$ и $x_{36}=0$ и $x_{37}=0$, то $x_{47}=0$
x_{48}	Технические характеристики качества ПП	Высокий > 4,5 – (1); Средний ≥ 1,5, но ≤ 4,5 —(0.5) ; «Низкий»< 1,5 – (0)	$x_{25}+x_{26}+x_{27}+x_{28}+x_{29}+x_{30}$
x_{49}	Экономическая привлекательность ПП	Высокая (1); средняя (0,5); низкая (0)	Если $x_{42}=1$ и $x_{43}=1$ или $x_{43}=0{,}5$, то $x_{49}=0{,}5$; Если $x_{42}=0{,}5$ или $x_{42}=0$ и $x_{43}=1$, то $x_{49}=1$; Если $x_{42}=1$ и $x_{43}=0$, то $x_{49}=0{,}5$; Если $x_{42}=0{,}5$ или $x_{42}=0$ и $x_{43}=0{,}5$, то $x_{49}=0{,}5$; Если $x_{42}=0{,}5$ или $x_{42}=0$ и $x_{43}=0$, то $x_{49}=0$

Таблица 4.5

Состав показателей четвертого слоя и правила их оценки

4 слой			
Параметр	Наименование параметра	Значение параметра	Способ определения значений параметра
x_{50}	Перспективность проекта	Высокая (1); средняя (0,5); низкая (0)	Если $x_{44}=1$ или $x_{44}=0,5$ и $x_{45}=1$, то $x_{50}=1$; Если $x_{44}=1$ или $x_{44}=0,5$ и $x_{45}=0,5$, то $x_{50}=0,5$; Если $x_{44}=1$ или $x_{44}=0,5$ и $x_{45}=0$, то $x_{50}=0$; Если $x_{44}=0$ или $x_{45}=0,5$ и $x_{45}=0$, то $x_{50}=0$; Если $x_{44}=0$ и $x_{45}=1$, то $x_{50}=0,5$
x_{51}	Конкурентоспособность ПП	Высокая (1); средняя (0,5); низкая (0)	Если $x_{44}=1$ или $x_{44}=0,5$ и $x_{45}=1$, то $x_{50}=1$; Если $x_{44}=1$ или $x_{44}=0,5$ и $x_{45}=0,5$, то $x_{50}=0,5$; Если $x_{44}=1$ или $x_{44}=0,5$ и $x_{45}=0$, то $x_{50}=0$; Если $x_{44}=0$ или $x_{45}=0,5$ и $x_{45}=0$, то $x_{50}=0$; Если $x_{44}=0$ и $x_{45}=1$, то $x_{50}=0,5$
x_{52}	Привлекательность рынка	Высокая (1); средняя (0,5); низкая (0)	Если $x_{44}=1$ или $x_{44}=0,5$ и $x_{45}=1$, то $x_{50}=1$; Если $x_{44}=1$ или $x_{44}=0,5$ и $x_{45}=0,5$, то $x_{50}=0,5$; Если $x_{44}=1$ или $x_{44}=0,5$ и $x_{45}=0$, то $x_{50}=0$; Если $x_{44}=0$ или $x_{45}=0,5$ и $x_{45}=0$, то $x_{50}=0$; Если $x_{44}=0$ и $x_{45}=1$, то $x_{50}=0,5$

Стоком сети является параметр «Перспективность концепции», значения которого отражают качественную (лингвистическую) оценку перспективности концепции программного проекта. Величина данного параметра определяется через значения параметра x_{53} с помощью следующих продукционных правил:

ЕСЛИ $0{,}8 < x_{53} < 1$, ТО x_{54} = «очень высокая»;

ЕСЛИ $0{,}6 \le x_{53} < 0{,}8$, ТО x_{54} = «высокая»;

ЕСЛИ $0{,}4 \le x_{53} < 0{,}6$, ТО x_{54} = «средняя»;

ЕСЛИ $0{,}2 \le x_{53} < 0{,}4$, ТО x_{54} = «ниже среднего»;

ЕСЛИ $0 \le x_{53} \le 0{,}2$, ТО x_{54} = «низкая».

Концепция проекта, получившая наибольшую оценку привлекательности, оформляется в виде оригинального документа, согласуется с членами команды и утверждается руководителем.

Контрольные вопросы

1. Раскройте различие между тиражной и заказной бизнес-моделями разработки ПП.
2. Раскройте содержание подготовительного этапа инициации проекта.
3. Раскройте содержание этапа обсуждения и оценки привлекательности идеи.
4. Раскройте содержание концепции проекта.
5. Поясните технологию оценки привлекательности программного проекта методом экспертных оценок.
6. Поясните содержание математической модели оценки перспективности концепции программного проекта.

5. УПРАВЛЕНИЕ СОДЕРЖАНИЕМ И СРОКАМИ ПРОГРАММНОГО ПРОЕКТА

5.1. Основные этапы управления программным проектом

В общем случае управление содержанием и сроками реализации программного проекта описывается в виде взаимосвязанного комплекса работ, которые необходимо выполнить для достижения заданных целей проекта. Исходными данными для определения содержания и сроков являются концепция проекта, функциональные и нефункциональные требования.

С точки зрения классической теории управления состав бизнес-процессов по управлению содержанием и сроками реализации можно представить в виде схемы (рис. 5.1).

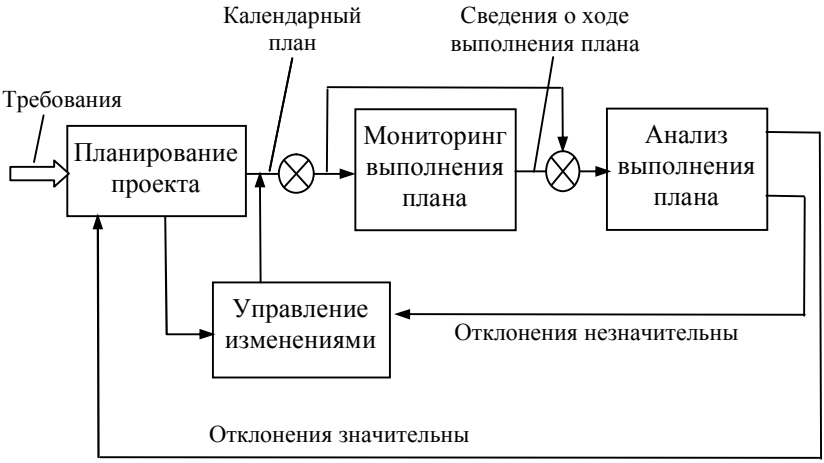

Рис. 5.1. Схема управления проектом

Процесс планирования содержит стандартный набор действий:

- структурную декомпозицию проекта и определение множества работ, которые необходимо выполнить для получения результатов проекта;
- выявление и документирование зависимостей между работами проекта;
- определение типов и количества исполнителей (трудовых ресурсов), привлекаемых для выполнения работ;
- оценку трудоемкости и длительности выполнения работ исполнителями;
- оценку рисков, связанных с выполнением работ;
- разработку календарного плана проекта, плана распределения ресурсов, бюджетного плана, плана управления рисками.

В процессе календарного планирования проекта необходимо определить длительность, сроки начала и окончания выполнения работ. Круг вопросов, связанных с решением этой задачи, рассматривается в рамках прикладной науки — теории расписаний.

Процесс мониторинга календарного плана обеспечивает функции контроля, анализа плановых заданий и при необходимости корректировки первоначального варианта календарного плана. Контроль и анализ выполнения календарного плана предполагает сбор фактических данных о ходе выполнения работ, сравнение плановых заданий и фактических показателей выполнения проекта, определение и анализ отклонений и причин их проявления. При этом контролировать необходимо все четыре составляющие эффективности программного проекта: 1) функционал, 2) сроки, 3) бюджет, 4) качество.

В литературе принято выделять следующие виды контроля:

традиционный контроль (контроль по фактическим отклонениям на конкретную дату);

упреждающий контроль, основанный на применении процедур экстраполяции значений контролируемых параметров на ближайшую перспективу и прогнозирование хода предстоящих работ.

Использование методов упреждающего контроля позволяет руководителю проекта предвидеть ситуацию, которая может возникнуть при условии сохранения наметившейся тенденции по реализации проекта.

Если размеры отклонений достигли некоторой критической величины, необходима выработка управленческих решений по их устранению. При этом возможны два варианта изменений первоначального плана проекта:

1) корректировка плановых заданий за счет привлечения дополнительных ресурсов;

2) разработка нового варианта календарного плана.

Модифицированный план в обязательном порядке доводится до сведения всех членов команды и с этого момента становится основным документом, регламентирующим порядок выполнения проекта.

Периодичность процессов контроля, анализа и корректировки плана программного проекта объективно требует организации **процесса управления изменениями**. Под изменением понимается замещение одного проектного решения другим вследствие возникновения влияющих на ход выполнения проекта негативных факторов. Причины проявления таких факторов:

- необходимость корректировки функциональных и нефункциональных требований к ПП;
- изменение рыночной конъюнктуры реализации проекта (появление продуктов-конкурентов, резкое изменение финансовых условий выполнения проекта и т. д.);
- ошибки при оценке трудозатрат и стоимости проекта;
- критические отклонения плановых и фактических показателей реализации проекта.

Процедура управления изменениями включает следующие этапы:

1) идентификацию и описание причин внесения изменений в проект и содержание самих изменений;

2) анализ влияния и оценку последствий внесения изменения на целевые показатели программного проекта (функционал, сроки, стоимость, качество);

3) принятие решения об отклонении либо внесении изменения в проект;

4) разработку и реализацию плана внесения изменений в проект (в содержание проекта, проектную документацию, программный код, календарный план проекта и т. д.);

5) организацию мониторинга хода внесения изменений в проект. В случае подтверждения успешной реализации изменение снимается с контроля.

Информационной основой процесса управления изменениями программных проектов является хранилище данных о текущем состоянии рабочего варианта проекта. Очевидно, что первыми элементами хранилища должны быть техническое задание и календарный план проекта. В связи с возможными нарушениями целостности данных при многочисленных синхронных изменениях доступ к хранилищу должен быть строго ограничен.

5.2. Структурная декомпозиция работ

Структурная декомпозиция работ проекта (*Work Breakdown Structure — WBS*) является ключевым элементом управления содержанием проекта и представляет собой описание перечня необходимых действий команды проекта для успешного достижения целей проекта. В этот перечень должны быть включены производственные, управленческие и административные действия, обеспечивающие разработку ПП и управление программным проектом. Сложные проекты выполняются коллективами разработчиков, что вызывает необходимость использования определенных методик декомпозиции проекта на отдельные работы, выполняемые командой проекта, распределения работ между членами коллектива.

В качестве оснований для декомпозиции проекта можно использовать фазы ЖЦ разработки ПП, элементы архитектурного дизайна ПП, функциональные ролевые группы организационной структуры проекта, этапы финансирования проекта. В данном случае предлагается декомпозировать работы программного проекта

с использованием модели жизненного цикла разработки ПП и состава архитектурного дизайна программного продукта. В качестве элементов архитектуры программного продукта будем выделять программный комплекс, программу, программный модуль (рис. 5.2):

- **программный комплекс** — совокупность двух и более взаимосвязанных программ, в которой функционирование одной из них зависит от результатов функционирования другой;
- **программа** — совокупность программных модулей, реализующих конкретный бизнес-процесс;
- **программный модуль** — совокупность программных кодов, реализующих элементарную функцию бизнес-процесса.

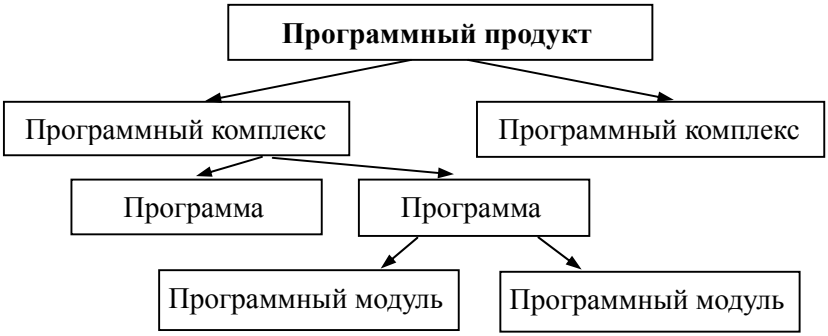

Рис. 5.2. Структура программного продукта

Состав и последовательность разработки каждого элемента архитектурного дизайна ПП зависит от используемых отечественных и международных стандартов, регламентирующих процессы ЖЦ ПП (см. раздел 3). В разных стандартах приводится собственная классификация структурных элементов декомпозиции ЖЦ ПП. В рассматриваемом случае примем вариант декомпозиции, представленный в ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Процессы жизненного цикла программных средств», где последовательность разработки ПП представлена в виде процессов, работ и задач. Множество **про-**

цессов соответствует **этапам жизненного цикла ПП**. Процессы декомпозируются далее на совокупность взаимосвязанных ***работ,*** представляющих собой деятельность по преобразованию объектов планирования — конкретных элементов архитектуры ПП на определенном этапе ЖЦ — из исходных (входных) данных в выходные результаты. Если после очередного этапа декомпозиции работа не удовлетворяет требованию элементарности, ее следует разложить на совокупность задач. В этом случае работа определяет достижение промежуточного результата, а ***задача*** является конкретным действием по достижению этого результата и рассматривается как элементарная единица планирования, поддающаяся достоверной оценке и управлению. В дальнейшем будем рассматривать термины «работа» и «задача» как синонимы, если для них можно адекватно определить трудоемкость, потребности в ресурсах и назначить исполнителей.

Вариант распределения работ по элементам архитектурного дизайна программного продукта при использовании ***каскадной модели ЖЦ*** представлен в табл. 5.1.

В состав WBS, кроме работ, непосредственно связанных с созданием программного проекта, могут быть включены и работы, описывающие деятельность сотрудников по управлению проектом. Например, в стандарте IEEE 1074-1997 выделен процесс (фаза) <управление проектом> и в качестве работ этого процесса определены: выполнение проекта, отслеживание и контроль проекта, управление качеством ПП. Кроме того, учитывая вероятностный характер выполнения работы, в модель структурной декомпозиции можно вводить буферные работы (например, «Планирование непредвиденных ситуаций») с соответствующей оценкой их длительности и привлекаемых специалистов.

Структурная декомпозиция работ может выполняться с разной степенью детализации. Если имеется некоторая неопределенность относительно длительности выполнения отдельных работ программного проекта, то работы отдаленной перспективы целесообразно планировать укрупненно, а ближайшего будущего — детально.

Таблица 5.1

Дорожная карта распределения работ по объектам планирования

Работы	Объекты планирования						
	ПП	Программный комплекс		Прог-рама		Программ-ный модуль	
		1	2	1	2	1	2
1. Разработка функциональных требований к ПП	+						
2. Разработка системных требований к ПП	+						
3. Разработка ТЗ на ПП							
4. Проектирование архитектурного дизайна как многоуровневой структуры ПП	+						
6. Проектирование архитектуры программных комплексов		+	+				
7. Программирование программных модулей						+	+
8. Разработка технической документации на модуль						+	+
9. Модульное тестирование						+	+
10. Сборка программы				+	+		
11. Разработка пользовательской документации на программу				+	+		
12. Тестирование программ				+	+		
13. Интеграция программ в программный комплекс		+	+				
14. Разработка эксплуатационной документации на программный комплекс		+	+				
15. Тестирование программного комплекса		+	+				
16. Сборка ПП	+						
17. Разработка эксплуатации онной документации на ПП	+						
18. Организация приемки-сдачи ПП	+						
19. Ввод в эксплуатацию ПП	+						

Например, на поздней стадии проекта в качестве единицы планирования можно рассматривать работу «Тестирование программных комплексов», а далее по мере выполнения проекта детализировать ее на две отдельные элементарные работы: тестирование программного комплекса 1 и тестирование программного комплекса 2. Такой метод планирования получил название «метод набегающей волны» [2].

В целом глубина детализации зависит от размера и сложности программного продукта, культуры управления проектом, используемых стандартов жизненного цикла и т. д. С одной стороны, проект должен быть рассмотрен максимально всесторонне и полно; с другой стороны, по мере увеличения глубины декомпозиции, расширяющей возможности планирования, управления и контроля работ, полученные результаты могут быть недоступны для понимания и анализа. Чрезмерная декомпозиция приводит к непродуктивным затратам труда управленческого персонала, неэффективному использованию ресурсов и, как итог, — к снижению эффективности реализации проекта в целом.

В этом случае структурная декомпозиция должна производиться до получения работ, которые понятны исполнителю и могут быть достаточно адекватно оценены по срокам исполнения и требуемым ресурсам. При этом продолжительность каждой работы на данном уровне декомпозиции не должна быть больше периода контроля выполняемых работ проекта.

Корректность процедуры декомпозиции подтверждается проверкой необходимости и достаточности низкоуровневых элементов для получения соответствующих результатов более высокого уровня.

Структура зависимостей между работами и задачами проекта может быть представлена как в виде ***иерархического вложенного списка задач***, так и в виде ***дерева задач*** [5]. В иерархической структуре WBS множество работ и последовательность их выполнения определяются полным перебором элементов архитектурного дизайна ПП и используемого набора работ над их преобразованием (рис. 5.3).

1 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

1.1 Разработка функциональных требований к ПП

……..

1.4 Проектирование архитектурного дизайна ПП

1.4.1 РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА 1

1.4.1.1 Проектирование архитектуры программного комплекса 1

1.4.1.1.1 Разработка программы 1

1.4.1.1.1.1 Проектирование архитектуры программы 1

1.4.1.1.1.1.1 Разработка программного модуля 1

1.4.1.1.1.1.1.1 Программирование модуля 1

1.4.1.1.1.1.1.2 Разработка документации на модуль 1

1.4.1.1.1.1.1.3 Тестирование модуля 1

1.4.1.1.1.1.2 Разработка программного модуля 2

1.4.1.1.1.1.2.1 Программирование модуля 2

…….

1.4.1.1.1.2 Сборка программы 1

……..

1.4.2 РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА 2

1.4.2.1 Проектирование архитектуры программного комплекса 2

1.4.2.1.1 Разработка программы 1

1.4.2.1.1.1 Проектирование архитектуры программы 2

……

1.19. Ввод в эксплуатацию ПП

Рис. 5.3. Иерархический список работ программного проекта

На каждом уровне иерархии «дерева» отбираются работы, соответствующие определенным элементам архитектуры ПП. Подобный тип описания структуры WBS рекомендуется для сложных программных проектов с большим количеством задач. Списки задач целесообразно создавать с помощью электронных таблиц, что позволяет руководителю проекта формировать различные сортировки и произвольные выборки заданий. Структура зависимостей между работами в виде «дерева задач» представлена на рис. 5.4.

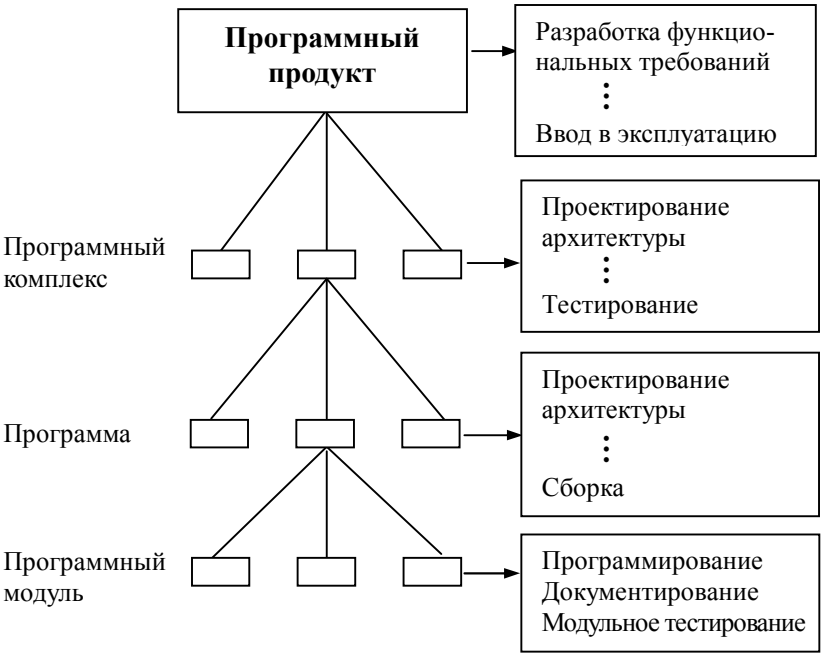

Рис. 5.4. Древовидная структура зависимостей между работами и задачами

Такое описание структуры позволяет визуально представить сложность программного продукта и провести изменение архитектурного дизайна (добавить либо удалить его элементы). Однако размеры графического представления дерева велики, и эти процедуры достаточно трудоемки.

5.3. Управление сроками реализации проекта

5.3.1. Формальное представление проекта в виде сетевой модели

Сформированный состав работ проекта является основой для разработки календарных планов реализации проекта, управления сроками выполнения отдельных работ, определения потребности в трудовых ресурсах. Согласно стандарту PMBOK процессы управления сроками реализации проекта предназначены для составления базового календарного плана (расписания) проекта без учета ограничений на трудовые ресурсы.

Основой для разработки календарных планов являются сетевые модели (графики), описывающие логические зависимости между работами. Существует два способа графического представления сетевой модели: на «языке событий» и на «языке работ». Описание сетевой модели на «языке событий» предполагает, что каждая работа отображается на графике в виде «дуги», соединяющей два кружка (события): событие, фиксирующее факт начала работы, и событие, фиксирующее факт окончания работы. Оба события имеют нулевую длительность. Этот способ построения сетевой модели описан в стандарте PMBOK.

Однако в большинстве рыночных пакетов прикладных программ по управлению проектами реализован способ описания модели на «языке работ» (рис. 5.5). В этом случае каждый кружок на сетевой модели описывает конкретную работу, а дуги определяют логическую последовательность выполнения работ. В верхней части кружка отображается номер работы, в нижней — указывается ее длительность.

Каждая работа сетевой модели описывается в виде следующих характеристик: номер работы; длительность выполнения; ранняя дата начала; ранняя дата окончания; поздняя дата начала; поздняя дата окончания; полный резерв времени; свободный резерв времени.

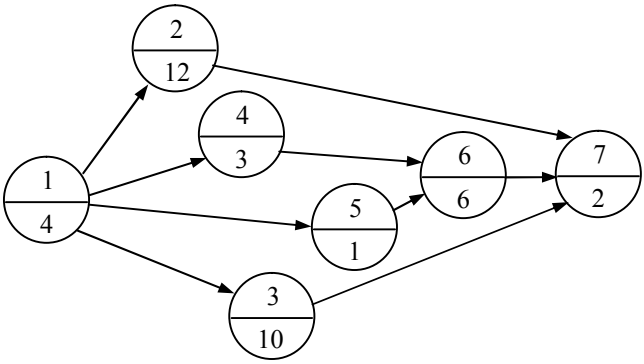

Рис. 5.5. Пример сетевого графика на «языке работ»

Ранней датой начала выполнения работы называется наиболее раннее время ее начала, не противоречащее взаимосвязям между работами и длительности их выполнения. Соответственно ***ранняя дата окончания*** выполнения работы отличается от ранней даты начала на величину длительности работы.

С точки зрения расчета времени раннего начала и окончания работ сетевой модели существуют два типа логических зависимостей между ними [2, 10]:

1) взаимосвязь «финиш – старт», определяемая следующим правилом: предшествующая работа должна заканчиваться до того, как последующая работа может начаться. Такие две работы должны выполняться последовательно друг за другом, при этом возможно наличие некоторого *времени задержки* начала выполнения последующей работы: *последующая работа должна начинаться не ранее чем через t единиц времени после окончания предшествующей работы*;

2) взаимосвязь «финиш – финиш», которая определяется следующим образом: *предшествующая работа должна заканчиваться не ранее чем за t единиц времени до окончания последующей работы.* Этот вид взаимосвязи предполагает наличие параллельно-последовательного порядка выполнения работ. При этом опережение начала выполнения последующей работы в зависи-

мости от длительности этих работ определяется вычитанием от момента начала либо окончания последующей работы определенного количества периодов времени (*t единиц времени опережения*).

Поздняя дата начала выполнения работы — это самое позднее время ее начала, при котором сохраняется общая длительность выполнения проекта и выполняются условия взаимосвязей между работами. Соответственно, ***поздняя дата окончания*** работы также отличается от ранней на величину длительности.

Полный резерв времени — это максимально допустимое время, на которое можно отложить момент окончания выполнения работы (сдвинуть время ее начала), при этом общая длительность выполнения проекта остается неизменной.

Величина полного резерва времени вычисляется как *разность между поздним и ранним временем начала выполнения работы*.

Свободный резерв времени работы — это показатель, характеризующий период времени, на который работа может быть отложена без изменения сроков выполнения проекта.

Свободный резерв времени вычисляется как *разность между минимальным ранним временем начала всех работ, следующих за данной, и ранним сроком окончания данной работы*. Свободный резерв времени работы всегда меньше или равен ее полному резерву времени. Увеличение длительности выполнения работы на величину свободного резерва времени либо сдвиг ее раннего начала на эту же величину никак не влияет на выполнение последующих работ проекта.

Очевидно, что величина резерва времени работы является индикатором, по которому менеджер проекта распределяет свои усилия в управлении программным проектом. Работы с большим резервом времени не требуют особого внимания, поскольку начало выполнения таких работ можно отложить на некоторый период времени, одновременно направив ресурсы для выполнения работ, лежащих на критическом пути. И наоборот, работы с малым резервом времени необходимо выполнять в первую очередь.

Расчет приведенных характеристик сетевой модели производится методом критического пути [17].

На первом шаге для каждой работы вычисляются ранние даты начала и окончания ее выполнения:

$$t_i^{\text{рн}} = 0,\ t_i^{\text{ро}} = t_i^{\text{рн}} + t_i,\ \ i = \overline{1,n}.$$

Вычисления начинаются с начальной работы и продолжаются до тех пор, пока не будет определено время окончания последней работы проекта.

На втором шаге вычисления начинаются с завершающей работы проекта, для каждой работы, начиная с последней, определяются поздние даты окончания и начала ее выполнения:

$$t_i^{\text{по}} = T, t_i^{\text{пн}} = t_i^{\text{по}} - t_i,\quad i = \overline{1,n}.$$

На третьем шаге для всех работ проекта вычисляются резервы времени:

1) полный — $R_i^{\text{п}} = t_i^{\text{пн}} - t_i^{\text{рн}},\quad i = \overline{1,n}$;

2) свободный — $R_i^{\text{с}} = \min\limits_{j \in \Gamma_i}\left(t_j^{\text{рн}} - t_i^{\text{ро}}\right),\quad i = \overline{1,n}.$

На четвертом шаге с учетом результатов вычислений на предыдущих шагах определяется совокупность взаимосвязанных работ, имеющих нулевой резерв времени, т. е. работ, у которых совпадают раннее и позднее время начала и окончания:

$$\left\{ t_i^{\text{рн}} = t_i^{\text{пн}}, t_i^{\text{ро}} = t_i^{\text{по}} \right\}.$$

Совокупность таких работ образует ***критический путь*** программного проекта.

Длина критического пути (продолжительность реализации программного проекта) определяется суммированием длительностей всех работ, находящихся на этом пути.

Совокупность работ с нулевым и относительно малым резервом времени (меньше некоторой заданной величины) называется ***критической зоной*** календарного плана проекта.

Пример расчета характеристик сетевой модели (рис. 5.5) приведен в табл. 5.2. Расчеты проводились при условии наличия взаимосвязи «финиш – старт» и времени задержки, равном нулю. Критический путь составляют работы 1, 2 и 7, длина критического пути равна 18 единицам.

Очевидно, что любое увеличение длительности выполнения критических работ приводит к срыву календарного плана, а изменение времени начала и окончания некритических работ должно происходить только в пределах резерва времени этих работ.

Таблица 5.2

Числовые характеристики сетевой модели

Номер работы	Параметры работ						
	Длительность	Ранние сроки		Поздние сроки		Резервы времени	
		начала	окончания	начала	окончания	полные	свободные
1	4	0	4	0	4	0	0
2	12	4	16	4	16	0	0
3	10	4	14	6	16	2	2
4	3	4	7	7	10	3	0
5	1	4	5	9	10	5	2
6	6	7	13	10	16	3	3
7	2	16	18	16	18	0	0

Расчет критического пути позволяет определить теоретические даты раннего и позднего начала и окончания работ проекта. Эти даты указывают периоды времени, в рамках которых возможно выполнение работ с учетом их логических взаимосвязей, нормативной трудоемкости и запланированных (выделяемых) трудовых ресурсов.

Если длина критического пути превышает плановый срок реализации проекта, то необходимо корректировать длительность выполнения работ, входящих в критический путь, за счет выделения дополнительных трудовых ресурсов, введения сверхурочных работ, использования дополнительных методов материального поощрения либо переноса некоторых работ в «буферные зоны».

С учетом вышеизложенного процедура формирования календарного плана программного проекта, представленного в виде сетевой модели, может содержать следующую шаговую последовательность действий.

Шаг 1. Определить множество работ программного проекта, необходимых для его реализации, и их взаимосвязь.

Шаг 2. Определить длительность (время выполнения) каждой работы.

Шаг 3. Сформировать сетевую модель реализации программного проекта.

Шаг 4. Вычислить ранние времена начала и окончания каждой работы.

Шаг 5. Вычислить поздние времена начала и окончания каждой работы.

Шаг 6. Вычислить полный и свободный резервы времени для каждой работы.

Шаг 7. Определить множество работ критического пути и его длительность.

Шаг 8. Сравнить время окончания выполнения последней работы критического пути $t(j)$ с плановым сроком окончания проекта T. Если $T > t(j)$, перейти к шагу 10, иначе — к шагу 9.

Шаг 9. Скорректировать за счет привлечения дополнительных ресурсов длительности выполнения ряда работ, входящих в критический путь, либо увеличить плановый срок окончания проекта T. Перейти к шагу 4.

Шаг 10. Утвердить календарный план программного проекта, определить объем требуемых ресурсов в каждом интервале времени.

Сформированные календарные планы проекта могут быть представлены:

- в виде таблиц, где по строкам перечисляются наименования работ, а по столбцам — исполнители, длительность, время начала и окончания;
- в форме линейной диаграммы Ганта (рис. 5.6), где слева показаны наименования работ и исполнители, а справа — длительность работ, соответствующая длине горизонтальных линий согласно принятой шкале временных параметров планирования;
- в виде сетевых диаграмм (моделей), изображенных в виде совокупности узлов (работ) и дуг (взаимосвязей между ними) с указанием всех параметров работ (времени начала и окончания, величины полного и свободного резерва времени).

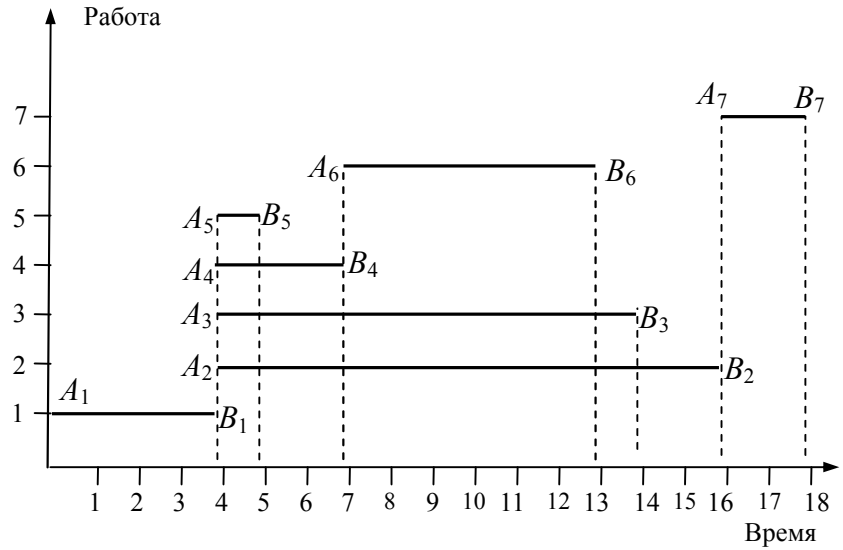

Рис. 5.6. Календарный план проекта,
сформированный в виде диаграммы Ганта

5.3.2. Модель и алгоритмы формирования календарного плана проекта

Пусть задано множество работ программного проекта $I = \{1, 2, ..., i, ..., n\}$. Для выполнения каждой работы потребуется один или несколько типов специалистов (архитекторов, бизнес-аналитиков, разработчиков, тестеров, технических писателей и т. д.) $Z = \{1, 2, ..., z, ..., m\}$. Каждый специалист в заданный момент времени может выполнять только одну работу, при этом для выполнения одной работы может потребоваться несколько специалистов разного профиля. Закрепление специалистов по работам может быть оформлено в виде матрицы распределения полномочий (ответственности).

Первичными оценочными характеристиками работы являются:

$T = \{t(i,z)\}$ — трудозатраты, количество человекомесяцев (дней), необходимых для выполнения работы;

$d(i,z) = f\big(i,k(z)\big)$ — длительность работы (в часах, днях, неделях, месяцах);

$s(i,z)$ — стоимость выполнения работы конкретным специалистом.

Для определения трудозатрат на ранних этапах разработки программного проекта целесообразно использовать метод PERT-анализа (Project Evaluation and Review Technique) [5], суть которого заключается в том, что для каждой работы проекта указываются три оценки трудоемкости — оптимистическая *o*, пессимистическая *p* и реалистическая *b*.

При ***оптимистическом оценивании*** трудозатрат предполагается, что все работы будут выполнены в срок, команда проекта состоит из высококвалифицированных специалистов, возникновение каких-либо рисков маловероятно.

Пессимистическая оценка трудозатрат производится при наличии множества проблем. Исходная предпосылка заключается в том, что если что-то плохое может произойти, оно обязательно произойдет.

Реалистическая оценка трудозатрат определяется как наиболее вероятная трудоемкость выполнения работы.

Средняя оценка трудозатрат определяется путем умножения реалистической оценки на 4, добавлением оптимистической и пессимистической оценок и делением полученного результата на 6: $t(i,z) = \left[t^o(i,z) + 4t^b(i,z) + t^p(i,z) \right] / 6$.

Данный способ оценки трудозатрат рекомендуется использовать для работ с высокой степенью риска.

В иных случаях оценку трудозатрат можно определять как среднее арифметическое от оптимистической, пессимистической и реалистической оценок.

Длительность выполнения работы обратно пропорциональна количеству привлекаемых специалистов:

$$d(i,z) = d(i,z) / k(z).$$

Например: «Разработка программного модуля 1» оценивается экспертами в 48 человекодней, для выполнения работы привлекаются 4 специалиста, тогда длительность работы составит 12 календарных дней.

Стоимость выполнения работы конкретным специалистом зависит от норматива его оплаты труда в единицу времени (*N* у.е в час, день) и доли заработной платы в общей стоимости работы.

По аналогии с [18] представим процесс реализации программного проекта графически в виде точек на плоскости. Каждая работа как точка на плоскости может быть описана в виде следующего кортежа $< i,\ z,\ d(i,z),\ t^{\text{н}}(i,z),\ t^{\text{к}}(i,z) >$.

Две точки (начальная и конечная) соответствуют началу и окончанию проекта. Каждые две точки могут иметь технологическую и ресурсную зависимости.

Технологическая зависимость между точками характеризуется существующими взаимосвязями между работами программного проекта и определяется условием: последующая работа начинается после завершения всех предшествующих ей работ. Если $i_1 \neq i_2$ и $z_1 \neq z_2$, то эти две точки — и (i_1, z_1) и (i_2, z_2) (две работы) — взаимосвязаны в том смысле, что время начала выполнения одной зависит от времени окончания другой и графически их следует соединить связью типа «дуга», направленной от одной точки к другой.

Ресурсная зависимость между различными работами определяет факт выполнения работ одним и тем же специалистом (использованием однотипных ресурсов): $i_1 \neq i_2$, но $z_1 = z_2$. Такие работы не могут выполняться одновременно, однако очередность их выполнения заранее не задана. Эти точки (задания) следует соединить связью типа «ребро». В результате получаем смешанный граф $G = (X, \overline{U}, U)$, представленный в терминах <событий>, где X — множество работ (событий), $\overline{U}$ и U — множество взаимосвязей между работами, описанных в виде дуг и ребер соответственно.

Рассмотрим процесс построения смешанного графа на следующем примере. Пусть $I = \{i_1, i_2, i_3\}$ — множество работ программного проекта. Для выполнения работ привлекаются четыре типа специалистов $Z = \{z_1, z_2, z_3, z_4\}$. Последовательность выполнения работ специалистами представлена в табл. 5.3. Каждая работа состоит из определенного количества задач.

Таблица 5.3

Последовательность выполнения работ

Работы	Задача		
	Программирование	Тестирование	Документирование
i_1	i_1, z_1	i_1, z_2	i_1, z_4
i_2	i_2, z_1	i_2, z_2	i_2, z_3
i_3	i_3, z_1	i_3, z_3	i_3, z_4

Все задачи каждой работы связаны между собой технологической зависимостью и соединяются связью типа «дуга». Задания $(i_1z_1,\ i_2z_1, i_3z_1), (i_1z_2, i_2z_2), (i_3z_3, i_2z_3), (i_1z_4, i_3z_4)$, связаны ресурсной зависимостью и соединяются связью типа «ребро» (рис. 5.7).

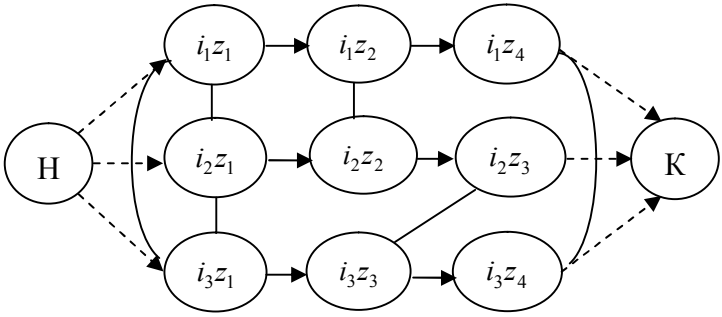

Рис. 5.7. Смешанный граф

Очевидно, что изменяя порядок замены ребер на направленные дуги, можно получить различные варианты ориентированного графа, описывающего взаимосвязи между работами проекта.

В этом случае задача определения оптимального календарного плана состоит в формировании из графа $G=(X,\overline{U},U)$ ориентированного графа $G'=(X',U')$ путем замены по определенному правилу ребер на направленные дуги и определения в графе G' критического пути. В теории расписаний такие задачи относятся к классическим задачам распределения ограниченных ресурсов на сетевых моделях и для их решения используются эвристические алгоритмы, основанные на использовании различных *функций предпочтения, или приоритетов* [18].

Ниже приводится интерпретация нескольких классических правил предпочтения применительно к данной задаче.

1. ***Правило «кратчайшей операции»*** (***SIO***): первой выбирается работа с наименьшей длительностью выполнения.

2. ***Правило «первым пришел — первым обслуживается»*** (***FIFO***): первой выбирается работа с наименьшим резервом времени.

3. ***Правило «последним пришел — первым обслуживается»*** (***LIFO***): первой выбирается работа с наибольшим резервом времени.

Использование различных правил предпочтения позволяет создать программный генератор формирования различных вариантов ориентированных графов.

Алгоритм выделения ориентированного графа и расчета ранних времен начала и окончания работ может быть представлен в виде последовательности этапов:

1) выделение множества вершин (работ), в которые не входит ни одна дуга, как множества ожидаемых работ

$$X_0 = \left\{ x_i \mid x_i \in X, \Gamma^{-1}(x_i) = \varnothing \right\};$$

2) если множество ожидаемых работ пусто $X_0 = \varnothing$, то переход к шагу 7;

3) выбор из сформированного множества ожидаемых работ X_0 работы $x_i' \Rightarrow \underset{x_i \in X_0}{\text{ext}} P(x_i)$ по определенному правилу предпочтения $P(x)$;

4) определение множества работ

$$X_1 = \left\{ x_j \mid \forall x_i \in X_0,\, x_j \in \Gamma^1(x_i),\, x_j \notin X_0 \right\},$$

смежных с вершиной x_i' и связанных с ней технологической зависимостью;

5) замена всех ребер, соединяющих работу x_i' с работами $x_j,\ x_j \in \Gamma^1(x_i),$ на исходящие из x_i' дуги. Удаление работы x_i' из множества ожидаемых работ;

6) определение времени начала и окончания исполнения всех работ множества X_1:

$$t^H(x_j) = t^K(x'),\ t^K(x_j) = t^H(x_j) + t(x_j),\ x_j \in \Gamma^1(x_i).$$

Переход к шагу 1.

В результате работы алгоритма получим из смешанного графа $G = (X, \overline{U}, U)$ ориентированный граф $G' = (X', U')$ (рис. 5.8).

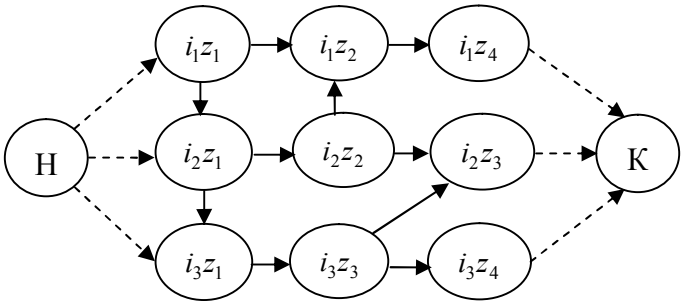

Рис. 5.8. Ориентированный граф $G' = (X', U')$

Использование различных правил предпочтения позволяет создать программный генератор формирования различных вариантов ориентированных графов.

Задача заключается в нахождении графа G', в котором длина критического пути окажется наименьшей. Иными словами, необходимо выбрать работы в такой последовательности, чтобы получаемый в результате календарный план позволял реализовать программный проект в кратчайшие сроки.

Рассчитанный по методу критического пути календарный план проекта может содержать неравномерную нагрузку (перегрузку либо простой) различных типов исполнителей в определенных интервалах планового периода проекта. Очевидно, что это негативно сказывается на ритмичности использования трудовых ресурсов. Данную проблему можно частично решить путем перепланирования работ, имеющих ненулевые резервы времени. В основу процедуры перераспределения ресурсов могут быть положены следующие действия:

- сдвиг некритических работ вперед либо назад — изменение дат начала и окончания в пределах резерва времени;
- разбиение некритических работ на части и сдвиг отдельных частей назад либо вперед;
- увеличение либо уменьшение количества исполнителей, привлекаемых для выполнения работ;
- увеличение либо уменьшение длительности выполнения работ;
- сдвиг критических работ в «буферные зоны».

Ниже приводится описание одного из возможных **алгоритмов выравнивания ресурсов**, основанного на методе критической цепи [2], в котором предполагается, что оптимизация календарного плана должна быть направлена не только на сокращение времени выполнения проекта, но и на рациональное использование трудовых ресурсов. В первую очередь необходимо оптимизировать ключевые (дефицитные) ресурсы, лежащие на критическом пути. В этом случае процесс перераспределения ресурсов начинается с конца планового периода.

Календарный план реализации программного проекта представим в виде линейной диаграммы Ганта (рис. 5.9). Каждая работа, выполняемая одним или несколькими типами специалистов, характеризуется продолжительностью выполнения и плановой трудоемкостью. Интенсивность потребления ресурсов при выполнении работы в каждом интервале времени равномерна. Возможно прерывание процесса выполнения работы. Объемы трудовых ресурсов в каждом интервале времени ограничены.

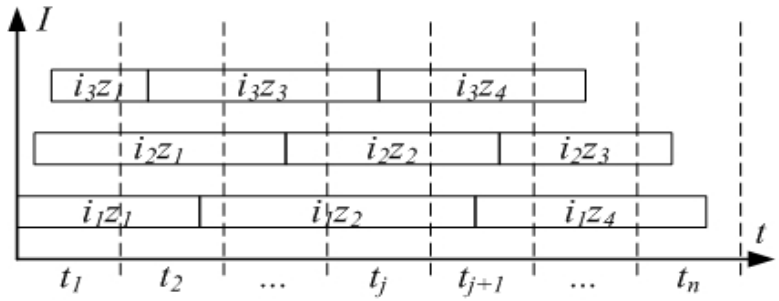

Рис. 5.9. Линейная диаграмма Ганта

Сущность алгоритма выравнивания ресурсов при заданном времени выполнения работ заключается в следующем. Интервал планирования $[0, T]$ разбивается на $(T-1)$ равных частей. Интервалы планирования рассматриваются справа налево, т. е. с последнего интервала планируемого периода. Рассматривается совокупность работ, которые должны выполняться в интервале $[t_j, t_{j+1}]$, все работы ранжируются в порядке убывания полного резерва времени и последовательно в соответствии с рангом включаются в план. При этом должны быть выполнены следующие условия:

- если работа может быть осуществлена разными специалистами, то предпочтение отдается исполнителю с наименьшей на данный момент времени загрузкой;
- объем имеющихся ресурсов в данном интервале планирования соответствует установленным ограничениям. Если ограничение по ресурсам не выполняется, работа двигается влево на один интервал времени либо принимается решение об увеличении количества исполнителей в данном интервале планирования.

Работа алгоритма заканчивается, если просмотрены все интервалы планируемого периода. Полученный план в общем случае не является оптимальным, однако такие алгоритмы просты в реализации, время работы их сравнительно невелико, степень точности достаточно высока.

Использование алгоритма особенно актуально в тех случаях, когда наблюдается перегрузка специалистов в определенные периоды планирования и/или имеется дефицит определенных типов специалистов.

Контрольные вопросы

1. Приведите и прокомментируйте схему взаимосвязи основных этапов ЖЦ управления программным проектом.
2. Поясните технологию и раскройте содержание и представление структурной декомпозиции работ программного проекта.
3. Представьте процесс разработки проекта в виде сетевого графика, перечислите и прокомментируйте основные характеристики сетевой модели.
4. Раскройте содержание и приведите пример алгоритма определения критического пути.
5. Приведите алгоритм формирования календарного плана проекта, представленного в виде сетевой модели.
6. Приведите содержательную и математическую постановку задачи формирования календарного плана проекта, представленного в виде смешанного графа.
7. Приведите алгоритм формирования календарного плана проекта, представленного в виде смешанного графа.
8. Раскройте содержание и приведите пример алгоритма выравнивания ресурсов.

6. УПРАВЛЕНИЕ ЧЕЛОВЕЧЕСКИМИ РЕСУРСАМИ

6.1. Организация командной работы над проектом

6.1.1. Основные участники и ролевые группы команды проекта

Управление человеческими ресурсами включает в себя ряд процессов по организации и управлению командой проекта, в том числе подбор команды проекта, назначение проектных ролей участникам команды, распределение участников по работам проекта, управление командой в процессе реализации проекта [2].

К участникам команды проекта относятся все заинтересованные стороны, которые заняты в проекте или интересы которых 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/юридическое лицо (группа лиц), предоставляющее 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 проект;

руководитель проекта — менеджер проекта (физическое лицо), которому заказчик и инвестор делегируют полномочия по руководству работами при осуществлении проекта;

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

команда проекта — группа разработчиков программного проекта, формируемая в зависимости от потребностей, условий проектирования и организационной структуры управления проектом.

Сложные программные проекты не могут быть выполнены индивидуально, их разработка ведется коллективом разноплановых специалистов. Введение специализации, распределение функциональных обязанностей и ответственности между членами команды проекта в зависимости от их квалификации приводит к тому, что специалисты, владеющие однотипными компетенциями, работают над проектом в одной функциональной группе.

В соответствии с методологией Microsoft Solutions Framework в команде проекта рекомендуется выделять [19]:

функциональные ролевые группы:

1) группа управления проектом;
2) группа проектирования архитектуры;
3) группа разработки программного продукта;
4) группа тестирования;
5) группа управления выпуском;
6) группа обеспечения связи с заказчиком;
7) группа управления продуктом;

виды специализации сотрудников:

1) менеджер проекта,
2) архитектор,
3) бизнес-аналитик,
4) разработчик,
5) тестер,
6) менеджер по работе с заказчиками.

Действия функциональных ролевых групп конкретизированы следующим образом:

1) ***группа управления проектом*** — управление процессом разработки с целью получения готового продукта в отведенные сроки; регулирование взаимоотношений и коммуникаций внутри проектной группы; контроль временного графика проекта и подготовка отчетности о его состоянии; разработка, поддержка и исполнение сводного плана и календарного графика проекта; организация управления рисками;

2) ***группа проектирования архитектуры*** — формулирование спецификации решения и разработка его архитектуры, определение структуры развертывания (внедрения) решения;

3) ***группа разработки ПП*** — определение деталей физического дизайна; оценивание необходимого времени и ресурсов на реализацию каждого элемента дизайна; разработка или контроль разработки элементов; подготовка продукта к внедрению; консультирование команды по технологическим вопросам;

4) ***группа тестирования*** — поиск и обнаружение дефектов; разработка стратегии и планов тестирования; тестирование;

5) ***группа управления выпуском*** — представление интересов отделов поставки и обслуживания продукта; организация снабжения проектной группы; организация внедрения продукта; выработка компромиссов в управляемости и удобстве сопровождения продукта; организация сопровождения и инфраструктуры поставки;

6) ***группа обеспечения связи с заказчиком —*** представление интересов потребителя в команде; организация работы с требованиями пользователя; нахождение компромиссов, относящихся к удобству использования и потребительским качествам продукта; определение требований к системе помощи и ее содержанию; разработка учебных материалов и обучение пользователей;

7) ***группа управления продуктом*** — осуществление функций по представлению интересов заказчика; организация работы с требованиями заказчика; формирование ожиданий заказчика; формирование общего видения и рамок проекта; поиск компромиссов между параметрами «возможности продукта», «время» и «ресурсы»; организация маркетинга.

В [20] предлагается все роли и ответственность участников команды проекта разработки ПП условно разделить на пять

групп: группа разработки требований; группа управления проектом; группа проектирования и разработки ПП; группа тестирования; группа обеспечения реализации проекта.

Группа разработки требований состоит из 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 программному обеспечению. *Специалист по требованиям* документирует и сопровождает требования к продукту. *Менеджер продукта* (функциональный заказчик) представляет в проекте интересы пользователей продукта.

В группе управления проектом *руководитель проекта* отвечает за достижение целей проекта при заданных ограничениях по срокам, бюджету и содержанию, осуществляет управление разработкой проекта, а также контроль за реализацией проекта и эффективным использованием выделенных ресурсов. *Системный архитектор* обеспечивает разработку технической концепции системы, принятие ключевых проектных решений относительно внутреннего устройства программной системы и ее технических интерфейсов. *Руководитель группы тестирования* определяет цели и стратегии тестирования, обеспечивает управление тестированием. *Ответственный за управление изменениями* обеспечивает процессы конфигурации, сборки и поставки ПП.

Группа проектирования и разработки ПП обеспечивает проектирование БД системы, компонентов и подсистем в соответствии с общей архитектурой, разработку архитектурно значимых модулей и интерфейса пользователя, проектирование, реализацию и отладку модулей системы. В состав группы входят проектировщики архитектурного дизайна, баз данных, интерфейса пользователя и разработчик программного кода.

Группа тестирования, включающая проектировщика тестов**,** разработчика автоматизированных тестов и тестировщика, создает тестовые сценарии и автоматизированные тесты, тестирует продукты, занимается анализом и документированием результатов.

Участники ***группы обеспечения реализации проекта*** выполняют работы в рамках своей профессиональной деятельности, и, как правило, не входят в команду проекта. К ним относятся *разработчик документации, переводчик, дизайнер графического интерфейса, разработчик учебных курсов, специалист по маркетингу и продажам, специалист по инструментальным средствам.*

В [21] приводятся несколько другие роли и обязанности участников проекта: *менеджер проекта* — главное действующее лицо, обладающее знаниями и навыками, необходимыми для успешного управления проектом; *проектировщик* — сотрудник, отвечающий за проектирование архитектуры высокого уровня и контроль ее выполнения; *разработчик* — специалист, ответственный за создание качественного программного кода; *тестировщик* — сотрудник, отвечающий за удовлетворение функциональных и нефункциональных требований к ПП; *инженер по качеству*, обязанностями которого является управление качеством конечного продукта, процессов разработки и организации работ по проекту; *технический писатель* — разработчик технической документации на ПП; *технолог разработки* — специалист, отвечающий за поддержку работоспособности принятой модели ЖЦ ПП и управление версиями.

Руководителю проекта необходимо с учетом особенностей проекта выбрать и распределить функциональные обязанности сотрудников в команде с учетом профессиональных качеств программиста и его типа личности. Следует помнить, что в процессе реализации проекта команда не остается неизменной, она проходит определенные стадии формирования и, как правило, количественно растет по мере развития проекта.

6.1.2. Организационные структуры управления проектом

Процесс управления программными проектами характеризуется организационной структурой управления, которая определяет обязанности, ответственность и полномочия сотрудников, входящих в проектную команду. Речь идет о наличии объективно существующих процессов специализации и разделения труда, что

приводит, с одной стороны, к разукрупнению функций и повышению квалификации исполнителей и, как следствие, к улучшению качества управления, с другой стороны, увеличивает количество исполнителей и взаимосвязей между ними, что объективно вызывает ухудшение качества управления.

Под **организационной структурой управления программными проектами** будем понимать совокупность взаимодействующих функциональных групп, выполняющих определенные ролевые функции, направленные на достижение конечных результатов программного проекта.

При проектировании организационной структуры требуется решить следующие принципиальные вопросы:

- определить состав подразделений (функциональные ролевые группы) и виды специализации отдельных сотрудников;
- определить количество уровней управления и состав подразделений на каждом уровне;
- определить горизонтальные (функциональные) и вертикальные (административные) взаимосвязи между подразделениями.

В качестве основных видов специализации выделяются:

1) специализация по видам деятельности (функциональным ролям подразделений);

2) продуктовая специализация по разрабатываемым в компании программным продуктам либо оказываемым услугам.

Первый вид специализации порождает функциональную структуру управления, второй — проектную структуру управления. Подробное описание каждой структуры, их достоинств и недостатков приведено в [1, 13].

Функциональная структура управления (рис. 6.1) предполагает, что каждое подразделение специализируется на выполнении определенных видов работ и имеет в своем составе специалистов высокой квалификации в определенной предметной области. В одном функциональном подразделении работают специалисты, обладающие однотипными профессиональными навыками. Выполнение указаний руководителя подразделения в пределах его компетенции обязательно для всех сотрудников подразделения.

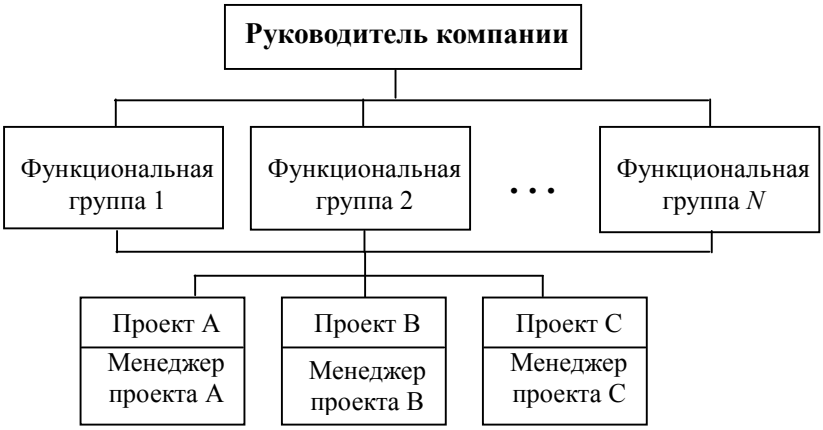

Рис. 6.1. Функциональная структура управления

Функциональные группы могут специализироваться либо в соответствии с этапами жизненного цикла создания ПП (например, этапом разработки требований, проектирования и разработки ПП, тестирования, управления проектом, обеспечения реализации проекта), либо в соответствии с используемыми при реализации проекта инструментальными средствами разработки.

Преимущества функциональной структуры управления:

- потенциальная возможность качественной разработки продукта за счет четкого распределения обязанностей, высокой профессиональной компетентности специалистов, отвечающих за реализацию конкретных функций;
- возможность совершенствования профессиональных навыков за счет взаимовыгодного обмена опытом между сотрудниками;
- возможность карьерного роста наиболее квалифицированных сотрудников.

Недостатки функциональных структур управления:

- трудности поддержания постоянных взаимосвязей между различными функциональными группами при разработке конкретного программного продукта;

- отсутствие целостной картины и ответственности отдельных функциональных групп за состояние разработки в целом, длительная процедура принятия решений при корректировке планов реализации проекта;
- отсутствие формализованных взаимосвязей между работниками разных функциональных групп;
- возможность возникновения конфликтов между функциональными группами подразделений, поскольку каждый функциональный руководитель может ставить интересы своей группы на первое место.

Функциональные структуры целесообразно применять при реализации небольших проектов, когда роль менеджера проекта целиком ложится на руководителя компании. Для мониторинга и координации за ходом выполнения проекта при руководителе может создаваться группа управления проектом.

Проектная структура управления (рис. 6.2) основана на том, что каждый программный проект организуется как самостоятельная IT-компания, персонал набирается по временным контрактам, после завершения проекта все сотрудники могут быть уволены.

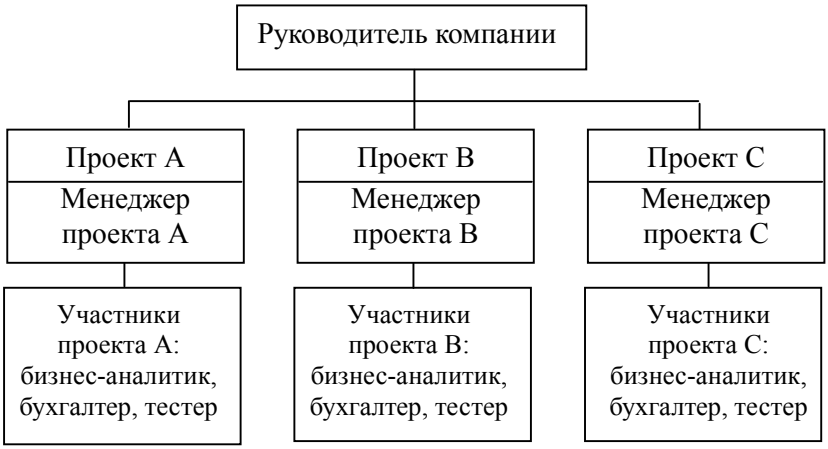

Рис. 6.2. Линейная (проектная) структура управления организацией

Проект возглавляется менеджером проекта, осуществляющим единоличное руководство подчиненными ему подразделениями (сотрудниками) и сосредотачивающим в своих руках все ресурсы по реализации проекта.

Преимущества линейной структуры управления:

- личная ответственность менеджера за конечные результаты;
- единство команды исполнителей, согласованность их действий;
- оперативность в принятии решений.

Недостатки линейной структуры управления:

- высокие требования к менеджеру проекта, который должен иметь хорошие профессиональные знания;
- неэффективное использование трудовых ресурсов (сотрудники функциональных подразделений, закончившие работу над проектом, остаются «не у дел»);
- полученный при реализации проекта опыт не сохраняется.

Проектную структуру целесообразно применять при территориальном распределении участников проекта.

Матричная структура управления — наиболее распространенная при выполнении программных проектов — создается посредством совмещения функциональной и проектной структур при достижении баланса между функциональными руководителями и менеджерами проектов (рис. 6.3). В соответствии с функциональной структурой по вертикали происходит управление отдельными видами работ над проектом, а по горизонтали организуется управление проектом в целом.

Матричные структуры управления имеют три типа исполнения: слабые, сбалансированные и сильные [1]. Типы структур отличаются друг от друга соотношением полномочий между руководителями функциональных подразделений и менеджерами проектов. Так, ***в слабой матрице*** роль и полномочия менеджера проекта сильно ограничены. Реальное руководство проектом осуществляет один из функциональных руководителей, а менеджер помогает руководителю собирать информацию о статусе выполняемых проектных работ, учитывает затраты, составляет отчеты.

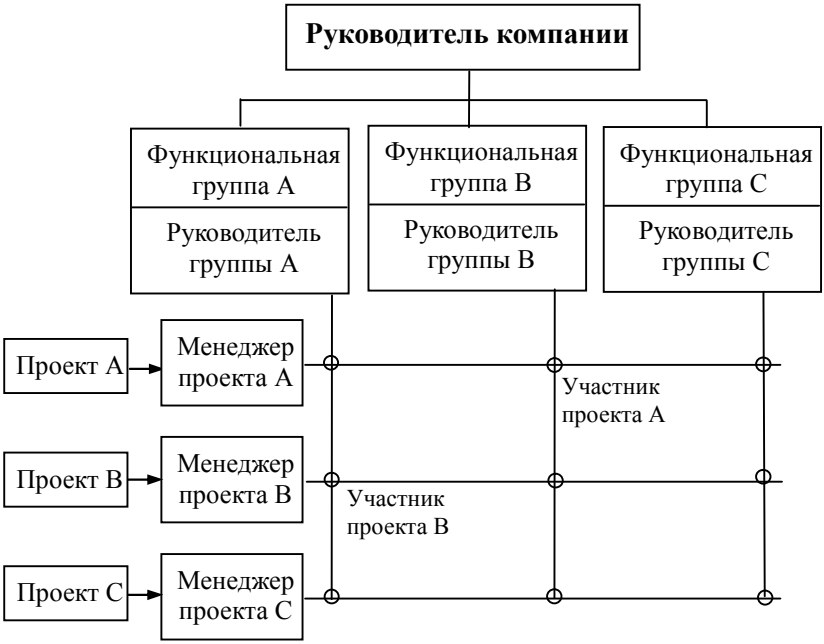

Рис. 6.3. Матричная структура управления организацией

Сбалансированная матрица характеризуется тем, что менеджер проекта реально управляет выделенными на проект ресурсами. Он планирует работы, распределяет задачи среди исполнителей, контролирует сроки и результаты, несет полную ответственность за достижение целей проекта при соблюдении ограничений. Руководитель функционального подразделения отвечает за организацию и качество выполнения работ.

В сбалансированных матрицах наиболее ярко проявляется проблема двойного подчинения. Руководитель функционального подразделения и менеджер проекта имеют примерно равное влияние на материальное обеспечение и профессиональный рост разработчиков.

В сильной матрице менеджер проекта имеет всю полноту власти, получает необходимые ресурсы, ставит задачи, определяет, что и когда должно быть сделано по каждому разделу проекта. Роль начальника функционального подразделения в производственном процессе заметно снижается. В его компетенции остаются вопросы стратегического развития функционального направления, планирования и развития карьеры сотрудников, материально-технического обеспечения. Следует учитывать, что такое перераспределение полномочий и ответственности от функциональных руководителей к менеджерам проектов часто служит источником конфликтов в компаниях. В сильной матрице менеджеры проектов объединяются в самостоятельную функциональную группу — офис управления проектами.

Преимущества матричной структуры управления:

- сокращение нагрузки на руководителей IT-компаний путем передачи полномочий по принятию решений по менеджерам проекта при сохранении координации и контроля за выполнением всех проектов;
- разделение функций управления между менеджерами, ответственными за выполнение сроков реализации проекта и обеспечение высокого качества разработок, и руководителями, ответственными за эффективное использование трудовых ресурсов;
- значительная активизация деятельности руководителей и работников за счет формирования временных коллективов, получение исполнителями увязанных между собой заданий, обеспеченных ресурсами и материальными стимулами, гибкость и оперативность маневрирования ресурсами при завершении проектов.

Недостатки матричной структуры управления:

- возможные конфликты между менеджерами проектов (распределителями финансовых ресурсов) и руководителями функциональных подразделений (распределителями трудовых ресурсов);
- двойное подчинение исполнителей конкретных заданий, необходимость отчитываться перед двумя руководителями;
- увеличение накладных расходов на содержание подразделений по управлению проектами.

Важно понимать, что выбор организационной структуры управления зависит от сложности проекта, условий его реализации, квалификации исполнителей и является задачей нетривиальной. Поэтому перед руководителем IT-компании, реализующей несколько проектов, встает проблема выбора эффективной организационной структуры управления. Основываясь на материалах [5], приведем рекомендации по выбору организационной структуры управления в зависимости от характеристик программного проекта (табл. 6.1). При этом необходимо помнить, что для достижения наивысшей эффективности использования человеческих ресурсов выбирать структуру управления организацией следует в соответствии с особеннос-тями проекта и возможностями команды, а не заниматься подбором участников команды под выбранную структуру управления.

Обобщая вышеизложенное, отметим, что проектирование организационной структуры управления не является одноразовым мероприятием. Изменения законодательной базы, рыночной ситуации, развитие организации объективно требуют пересмотра и структуры управления. В связи с этим приведем несколько практических рекомендаций по решению данной проблемы:

1) проведение постоянной корректировки структуры управления в соответствии с целями и задачами компании, обусловленное динамическим характером процесса распределения функций по подразделениям;

2) закрепление всех функций за соответствующими подразделениями, при этом данная задача должна быть обязательно доведена до логического конца;

3) закрепление каждой функции только за одним подразделением во избежание дублирования и возникновения просчетов в управлении;

4) разработка регламентов работы функциональных подразделений и их взаимодействие, направленных на достижение конечных результатов программного проекта, что обеспечивает завершенность процесса управления командой проекта.

Таблица 6.1

Взаимосвязь характеристик проекта и типов организационной структуры управления

Характеристики проекта	Типы организационных структур			
	Функциональная	Матричная слабая	Матричная сильная	Проектная
1. Степень неопределенности	Низкая	Умеренная	Высокая	Высокая
2. Технология разработки	Стандартная	Стандартная	Сложная	Новая
3. Сложность предметной области	Низкая	Низкая	Средняя	Высокая
4. Длительность проекта	Короткая	Средняя	Средняя	Длинная
5. Степень риска	Низкая	Средняя	Средняя	Высокая
6. Количество кураторов проекта от заказчика	Много	Много	3–4	Один
7. Уровень зависимости внутри функциональной группы	Низкий	Средний	Средний	Высокий
8. Уровень зависимости между функциональными группами	Высокий	Средний	Средний	Низкий
9. Частота внесения изменений в проект	Низкая	Низкая	Высокая	Средняя

Оценку качества организации процессов управления программными проектами можно проводить на основе международного стандарта **CMM (Capability Maturity Model for Software — модель зрелости процессов разработки программного обеспечения)**, созданного в Институте программной инженерии (Software Engineering Institute — SEI) при университете Карнеги-Меллона в Питтсбурге [5]. В стандарте приведены признаки, которыми должна обладать профессиональная организация по разработке ПП. В соответствии со стандартом **модель технологической зрелости** — это описание стадий эволюции организации-разработчика по мере определения, реализации, измерения, контроля и совершенствования процессов разработки программных проектов.

В стандарте определено пять уровней технологической зрелости, по которым заказчики могут оценивать потенциальных претендентов на получение контракта на разработку ПП, а компании-разработчики совершенствовать процессы управления программными проектами. Каждый уровень технологической зрелости соответствует определенному этапу развития компании по управлению и непрерывному совершенствованию процесса разработки ПП:

1) начальный. Технология управления разработкой ПП характеризуется как произвольная, пригодная только для некоторых случаев, скорее, даже хаотическая. Лишь некоторые процессы ЖЦ определены, успех проекта в основном зависит от компетенции отдельных сотрудников;

2) повторяемый. Базовые процессы управления программным проектом определены и позволяют отслеживать затраты, график работы и функциональности программного продукта. Соблюдается необходимый порядок выполнения процессов и обеспечивается возможность повторения достижений, полученных ранее при выполнении подобных проектов;

3) определенный. Базовые, вспомогательные и организационные процессы документированы, стандартизованы и интегрированы в унифицированную для данной компании технологию управления программными проектами, которая и используется при разработке ПП;

4) управляемый. Управление всеми процессами по разработке ПП осуществляется по количественным оценкам. Детальные и объективные показатели качества исполнения процессов разработки ПП и характеристики самого продукта в соответствии с регламентом процессов разработки собираются и накапливаются;

5) оптимизируемый. Совершенствование процессов разработки ПП осуществляется непрерывно как на основе количественного анализа эффективности процессов, так и использования инновационных методов и технологий.

Заметим, что модель CMM не содержит численных критериев оценки, рекомендаций и не указывает, как конкретно оценить продукт, а лишь показывает, что необходимо сделать для достижения требуемого качества программного обеспечения. CMM содержит также способы контроля за правильностью выполнения ключевых действий и методы их корректировки. Для сертификации по CMM необходимо выполнить несколько крупных заказов, чтобы показать на их примере полный цикл разработки. Применение CMM наиболее эффективно в компаниях, где большинство процессов управления жестко формализовано.

6.1.3. Основные модели управления командой проекта

В общем случае принятая в организации модель управления определяет роль и место руководителя и подчиненных, правила выработки и реализации управленческих решений. С этой точки зрения в [22] описываются две модели управления: административная (бюрократическая) модель и модель участия.

Административная модель управления, возникшая в начале XX столетия, характеризуется четкой иерархией правил, жесткой регламентацией должностных инструкций для каждого подразделения и исполнителя. Личные качества исполнителей используются слабо, а зачастую игнорируются вообще. Для этого типа организации управления характерны следующие предположения:

- «средний» человек не любит работу и, насколько возможно, ее избегает;
- большинство людей следует принуждать к исполнению ролей, необходимых для достижения целей проекта;
- «средний» человек в основном пассивен, предпочитает скорее быть ведомым, чем принимать на себя ответственность и риск.

В этом случае, очевидно, необходима жесткая регламентация и контроль деятельности сотрудников, которые автоматически приводят к эффективной работе всей организации.

Основные отличительные черты бюрократической модели управления могут быть определены следующим образом:

- организация проектной деятельности по достижению целей фирмы в виде зафиксированных формальных ролей и обязанностей сотрудников и четкое распределение ответственности;
- властная пирамида управления проектом, построенная по принципам иерархии;
- выполнение процессов и работ по проекту в соответствии с утвержденным регламентом;
- строго формализованное управление проектом со стороны менеджеров, исключающее эмоциональные факторы при принятии решений.

Несомненным ***достоинством административной модели*** является четкая организация труда, разграничение ролей, обязанностей и ответственности, хорошо организованная система контроля. Ее рекомендуется использовать при создании ПП по каскадной модели жизненного цикла.

Недостатки административной модели связаны с тем, что при жесткой регламентации работы подразделений и исполнителей игнорируется активная роль сотрудников при выполнении проекта, отсутствует мотивация к творческому труду. Модель плохо восприимчива к изменениям ситуации (появление новых типов проектов, применение новых технологий, необходимость оперативного реагирования на изменение требований пользователей и поведения рынка и т. д.).

Модель участия ориентирована на развитие и использование творческих способностей сотрудников при выполнении своих обязанностей. В качестве исходных предположений в модели участия рассматриваются следующие положения:

- работа естественна для человека, как игра или отдых, поэтому он ее не избегает и склонен принимать и даже искать ответственность, которую ему предоставляет коллектив;
- стремление к труду и удовлетворенность результатами наступают, когда сотрудник причастен к достижениям целей проекта, следовательно, важным фактором мотивации активного участия сотрудника является степень его вовлеченности в проект.

Главный принцип использования модели участия можно сформулировать так: «Работаем спокойно, работаем вместе» [8]. Основу данного принципа составляют следующие положения:

- если разрабатываем независимые модули, то расходимся и делаем, если проектируем архитектуру ПП, то собираемся вместе и обсуждаем идеи;
- с принятым вариантом решения не все могут согласиться, но принятое решение является коллективным и в силу этого — обязательным для всех;
- обязательное участие менеджера проекта при коллегиальной выработке решений;
- контроль конструктивности обсуждений, обеспечение возможности активного участия всех сотрудников;
- распределенная ответственность за качество своих разработок и коллективная — за принятие коллегиальных решений (отвечают все, кто обсуждал, вырабатывал, принимал);
- явное отсутствие специализации — сотрудники могут при необходимости заменить друг друга.

Наиболее полно модель участия нашла свое отражение в теории японского менеджмента, основой которого являются нижеперечисленные принципы [23]:

- работник достаточно разумен, он самостоятельно повышает производительность и качество своего труда. Три административных метода помогают претворить этот принцип в жизнь:

постоянное участие в работе кружков качества, практика стимулирования всех работников к совершенствованию профессиональных умений и навыков, практика перевода работников с одного места на другое;

• работник сам стремится сделать свою работу лучше. Реализация принципа основана на пожизненном найме работников, премировании рабочих в случае повышения прибыли фирмы, причем в неблагоприятной ситуации руководящий состав получает значительно более урезанную зарплату, чем рабочие;

• работники фирмы образуют единую «семью», оказывая но-вичку помощь, сочувствие и поддержку, ожидая от него в дальнейшем проявления такого же поведения по отношению к себе. Свободное время работники фирмы проводят вместе. Отношения между руководством фирмы и работниками формируются на основе взаимных обязательств;

• интересы коллектива важнее интересов отдельной личности. Этот принцип основан на традиционной японской ценности — никто не должен быть эгоистичным и думать только о себе.

Использование модели участия особенно эффективно в случае, когда для решения проблемы требуется поиск новых подходов, методов, идей и средств. Недостатки модели связаны с тем, что сотрудники, способные к генерации идей, редко обладают терпением для доведения идей до полной реализации, а творческая активность переходит в конкуренцию — сначала идей, а потом личностей. В этом случае во избежание конфликтов необходима разумная система поиска компромиссных решений и эффективные методы мотивации к достижению общих целей команды проекта.

Очевидно, что не может быть общих рецептов по выбору той или иной модели организации. Кроме того, в различных структурных подразделениях одной фирмы могут использоваться и различные модели. Например, производственные подразделения с регламентным характером выполняемых функций могут использовать бюрократические модели; подразделения маркетинга более динамичны, постоянно сталкиваются с конкуренцией и поэтому для них эффективнее будет модель участия.

Отметим лишь несколько общих характеристик, определяющих выбор модели организации управления:

1) ***используемые производственные технологии.*** При разработке ПП под конкретный заказ, где работа носит в основном творческий характер, целесообразно использовать модель участия. При тиражном производстве программных продуктов, где бизнес-процессы жестко заданы и заранее определены, более эффективной будет бюрократическая модель;

2) ***характер поведения внешней среды.*** Если потребности пользователей неизменны и рынок предсказуем, то фирма может уверенно использовать бюрократическую модель организации управления. Для большинства же предприятий динамичные, быстро изменяющиеся потребности внешней среды требуют гибкой, быстро приспосабливаемой организации управления разработками, которой соответствует модель участия;

3) ***размер организации***. Условно, в зависимости от количества сотрудников, можно выделить следующие виды организаций: малые; средние; крупные. Как правило, чем крупнее фирма, тем сильнее проявляется тенденция к использованию формальной бюрократической модели управления;

4) ***используемые персональные и корпоративные стратегии компании.*** На выбор структуры управления сильно влияют и положение компании на рынке, и управленческий «темперамент» ее служащих. Например, компания намерена добиться наибольшего успеха на рынке агрессивным управлением, она будет стремиться создать нестабильность на рынке и захватить себе большую его часть. Очевидно, в этом случае целесообразнее всего использовать модель участия.

С другой стороны, лидер на рынке испытывает большое давление со стороны конкурентов и стремится к относительной стабильности. В этом случае управление должно быть в большей мере ориентировано на сохранение статус-кво, что может быть обеспечено только четкой регламентацией действий всех сотрудников и организации в целом.

6.2. Специфические особенности командной работы над программным проектом

6.2.1. Роль руководителя в команде проекта

Командный процесс разработки ПП предполагает, что основная задача руководителя группы состоит в следующем: найти нужных людей, подобрать наиболее подходящую для них работу, помнить о мотивации; помогать сотрудникам сплотиться в одну команду для дальнейшей совместной работы. Управление командой разработчиков ПП имеет ряд особенностей [4]:

1) ***узкая специализация сотрудников***. Менеджер проекта подбирает команду, в которой каждый из участников максимально эффективен в конкретной области. При этом никто из них не может видеть проблему в целом. В результате нагрузка по оптимизации и интеграции системы в значительной мере переносится на системного архитектора, который, в свою очередь, не может быть специалистом во всех областях сразу;

2) ***постоянное повышение квалификации***, поскольку технологии разработки ПП весьма быстро морально устаревают, а качественно новые версии инструментальных средств появляются достаточно часто, иногда с периодичностью раз в полгода;

3) ***высокая самооценка программиста***. Узкая специализация и высокая профессиональная квалификация в конкретной области часто вызывают завышенную самооценку своих возможностей у сотрудников, что обусловливает возникновение у руководителя следующих проблем:

- необходимости постоянного изучения реальных возможности своих сотрудников и корректировка их самооценки;
- формирования способности убеждать и мотивировать членов команды, поскольку 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 области, где сотрудник является узким специалистом;

4) ***невысокая трудовая дисциплина***. Программистов часто трудно заставить приходить на работу в назначенное время, не опаздывать на совещания, своевременно представлять отчеты о выполнении задания. Это связано с индивидуальным характером труда, возможностью выполнять задания вне стен фирмы, работать во внерабочее время. Руководителю часто приходится доводить до сознания сотрудников тот факт, что они работают в команде, и разработка ПП — всегда коллективная деятельность;

5) ***творческий характер труда программиста***. Разработка ПП — 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*;

6) ***высокая мобильность сотрудников***. В современных условиях спрос на квалифицированных программистов существенно превышает предложение. Эта тенденция сохранится в обозримом будущем, из чего следует, что руководитель должен быть готов к внезапному уходу из команды (фирмы) любого из сотрудников. Процесс разработки следует организовать так, чтобы это не вызвало 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 ПП, разработанные в рамках проекта организации, им не принадлежат.

В этих условиях особая роль при командной работе над проектом отводится руководителю коллектива, которому должны быть присущи два качества: умение управлять и быть лидером. Нельзя быть лидером материальных ресурсов, денежных потоков, планов, графиков и рисков — ими необходимо управлять, потому что у вещей нет права и свободы выбора, присущих только человеку. Интеллектуальными людьми управлять невозможно. Творческие команды можно только направлять и вести за собой.

Не существует единственной наилучшей стратегии руководства. В зависимости от готовности участников рабочей группы выполнять порученные им задания руководитель может использовать одну из четырех предлагаемых в [5, 20] стратегий:

1) «Директивное управление», при которой руководитель указывает, направляет, устанавливает. Характеризуется жестким распределением работ, строгим контролем сроков и результатов;

2) «Объяснения», при которой лидер объясняет и проясняет свои решения, убеждает. Характеризуется сочетанием директивного и коллективного управления;

3) «Участие», при которой лидер участвует, поощряет, сотрудничает, проявляет преданность. Характеризуется преобладанием коллективного принятия решений, обмена идеями, поддержки инициативы подчиненных;

4) «Делегирование», при которой лидер передает часть своих полномочий, наблюдает, обслуживает, определяет цели, обеспечивает ресурсами и не мешает (пассивное управление сформировавшегося лидера).

При реализации каждой из этих стратегий руководитель должен уметь эффективно выполнять четыре функции:

1) ***направлять.*** Если сотрудник не понимает, что делать, задача руководителя — обеспечить общее видение целей и стратегии их достижения;

2) ***обучать.*** Если сотрудник не умеет, задача руководителя — «обучать», быть наставником и образцом для подражания;

3) ***помогать.*** Если сотрудник не может выполнить работу, задача руководителя — «помогать», обеспечивать исполнителя всем необходимым, убирать препятствия с его пути;

4) ***вдохновлять.*** Если у сотрудника недостаточно желания выполнить работу, задача руководителя — «вдохновить», обеспечить адекватную мотивацию участника на протяжении проекта.

Эффективные команды не образуются сами по себе, они интегрируются вокруг признанного лидера. Как не бывает лидеров без последователей, так и не бывает команд без лидеров. Поэтому первый шаг руководителя при создании эффективной команды — это стать лидером, вокруг которого сможет сплотиться коллектив.

6.2.2. Мотивация программиста как участника проекта

В общем случае понятие **«мотивация»** определяется как методы воздействия на людей с целью получения желаемого результата. Для того чтобы человек совершил какое-либо действие, он должен испытывать некую потребность и предполагать, что выполнив это действие, он в той или иной степени эту потребность удовлетворит. Поэтому управление мотивами осуществляется, как правило, в двух направлениях: 1) формирование правильных потребностей; 2) формирование правильной оценки степени их удовлетворения. Именно потребности заставляют людей действовать определенным образом.

Американский психолог Абрахам Маслоу в 40-х годах XX в. представил все потребности человека в виде возрастающей пятиуровневой иерархии [4, 20]:

1) **физиологические потребности**, являющиеся необходимыми для выживания (еда, жилище, одежда);

2) **потребности в безопасности**, включающие потребности в защите от физических и психологических опасностей и уверенность в удовлетворении физиологических потребностей (пенсия) в будущем;

3) **социальные потребности**, заключающиеся в привязанности, принадлежности к какой-либо общности, дружбе;

4) **потребности в уважении**, включающие внутренние (самоуважение, личные достижения) и внешние (статус, признание, одобрение со стороны окружающих) факторы уважения;

5) **потребность в самовыражении**, заключающаяся в самореализации и духовном росте личности.

При этом уровни 1-й и 2-й отнесены к низшим потребностям, а уровни 3-го по 5-й — к высшим потребностям.

В каждый момент времени поведение человека определяется самой сильной из неудовлетворенных потребностей. Однако потребности высших уровней не мотивируют человека, пока не удовлетворены хотя бы частично потребности низших уровней. В основе теории мотивации лежат следующие понятия: ***побуж-***

дение — порождаемое потребностью ощущение недостатка в чем-либо, имеющее определенную направленность; ***вознаграждение (поощрение)*** — все, что человек считает ценным для себя.

Вознаграждения принято подразделять на следующие виды:

- *внутренние* — ценности, существующие в сознании человека (чувство самоуважения, удовлетворенность результатом, ощущение значимости и ответственности своего труда, комфорт неформального общения в коллективе);
- *внешние* — ценности, предоставляемые организацией за выполненную работу (зарплата, премии, продвижение по службе, символы статуса и престижа, похвалы и признания, дополнительные льготы).

Основными факторами, определяющими мотивацию человека, являются:

- характеристики рабочей атмосферы в команде — дизайн помещения, оборудование рабочего места и предоставляемые сервисы, уровень шума, чистота, режим работы и др.;
- виды вознаграждений — оплата труда и другие выплаты, система медобслуживания, социальное обеспечение, дополнительные материальные выгоды;
- микроклимат коллектива — причастность к качественному выполнению проекта, уважение и одобрение коллектива, стиль общения с руководством, принятые в компании отношения между сотрудниками.

Конкретные потребности, которые необходимо удовлетворять, чтобы мотивировать сотрудников, определяются личностными качествами конкретного человека, особенностями характера (темперамента), профессиональным опытом, внешними обстоятельствами, жизненной ситуацией.

Так, в [20] отмечается, что при организации работы программиста-флегматика, принимая во внимание особенности его характера, необходимо:

1) формулировать конкретно поставленные задачи, требующие глубокой и качественной проработки, допускающие неспешность и постепенность при их решении;

2) привлекать для решения текущих, повседневных, технологически отработанных заданий, не требующих частых и глубоких контактов с клиентами, с гарантированной материальной отдачей;

3) поручать управление, организацию и контроль производственной деятельности, в том числе составление и анализ планграфиков разработки программ;

4) учитывать мотивацию данной категории специалистов на планомерные, регулярные, предсказуемые по срокам задачи, требующие систематичности и упорядоченной последовательности действий.

Распределение мотивирующих потребностей для профессиональных разработчиков ПП в зависимости от опыта работы приведено в табл. 6.2 [20].

Таблица 6.2

Зависимость мотивации участника команды от опыта

Потребности	Предпочтения сотрудников в зависимости от профессионального уровня, %		
	Начинающий	Опытный	Мастер
1. Материальные (зарплата, условия труда, социальный пакет)	50	20	—
2. Безопасность (стабильность компании, востребованность)	—	20	—
3. Принадлежность к команде (возможность учиться у более опытных коллег, опыт участия в успешном проекте, признание в коллективе)	40	20	10
4. Самоуважение (профессиональный и карьерный рост, возможность «выделиться» за счет индивидуальных качеств, позволяющих выполнять работу лучше других, самостоятельность в работе)	10	30	40
5. Самоактуализация (амбициозность целей проекта — сделать то, что никто не делал или не смог сделать)	—	10	50

Для начинающих программистов стимулом к эффективной работе 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 в их решении, что позволяет реализовать потребность в самоуважении. 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.

В зависимости от поведения человека в команде описываются четыре типа программистов, имеющих различные побудительные мотивы к работе [20]:

1) **хороший парень.** Общительный, стремится всем нравиться, оптимист, особенно в оценках проекта, готов выполнять любое поручение начальника, никогда не говорит «нет», старательно избегает конфликтов, не настаивает на своем мнении, озабочен только тем, «чтобы угадать и угодить». Такой специалист сильно зависим от мнения окружающих, не имеет четких личных целей, мотивирован на комфортные взаимоотношения, а не на успех. Для определения мотивирующих воздействий необходимо помочь человеку выявить личные ценности и выработать стратегию их достижения;

2) **тихоня.** Ведет себя сдержанно, стремится «не высовываться», ожидает подробных инструкций, старается действовать строго в пределах своих функциональных обязанностей, редко высказывает свое мнение, никогда не настаивает на нем, избегает любых ситуаций, связанных с возможными конфликтами, замалчивает «неприятную» информацию. Как правило, такое поведение обусловлено прошлым отрицательным опытом работы с агрессивным руководством, этот программист мотивирован не на успех, а на избежание неудачи. Такому специалисту надо помочь поверить в себя, всячески развивать и поощрять его инициативу и самостоятельность;

3) **ведущий программист.** Имеет хорошую квалификацию и большой опыт работы, неплохо справляется с порученными задачами, но делает это не всегда по-настоящему хорошо, инициативу особо не проявляет, стремится получать задачи попроще. Повышение оклада дает только кратковременный результат, премии не вызывают какого-либо энтузиазма к работе. Очевидно, что у этого программиста отсутствует мотивация к эффективной работе. Возможно, это связано с тем, что человек не видит перспектив личностного роста, необходимо помочь ему выявить личные цели и спланировать карьерное развитие;

4) **суперпрограммист.** Стремится решать задачи, которые до него еще никто не решал, играет роль технического лидера, ведущего за собой остальных участников под лозунгом: «Делай как я!», всегда готов продемонстрировать, как можно эффективно решить любую задачу, склонен к обучению коллег и передаче им своего опыта. Для суперпрограммиста следует находить достойные задачи, которые его заинтересуют, поручать ему системную интеграцию проекта, реализацию архитектурно значимых компонентов («скелета» системы).

Основные рекомендации по разработке подходов к мотивации сотрудников сводятся к следующему [7, 20]:

- разумная мотивация — это возможность дать каждому сотруднику шанс расти и развиваться на своем рабочем месте через совместную работу над проектом;
- мотивировать или стимулировать только отдельных сотрудников нецелесообразно. Более эффективный путь — создание климата, в котором большинство работников сами создадут себе стимулы для того, чтобы помочь организации добиться намеченных результатов;
- оплата труда является мотивирующим фактором только в случае устойчивой связи между ее размером и результатами труда. При этом оптимально разбиение зарплаты на три части: часть, определяемая должностью (постоянна и равна у всех сотрудников с равными должностями); часть, связанная с выслугой лет (равна у всех с одинаковым стажем работы); основная часть, зависящая от результатов конкретного труда. Начиная с определенного уровня

благосостояния (или в определенных социальных ситуациях), роль денег как мотиватора уменьшается, необходимо использование нематериальных вознаграждений и льгот.

В [20] рекомендуется избегать упрощенных подходов и типичных ошибок мотивации, к которым следует отнести следующие:

- «То, что является мотивацией для одного сотрудника, будет являться мотивацией и для других». Необходимо помнить, что все люди разные и находятся в разных жизненных условиях;
- «Мотивацию для человека, прежде всего, составляют деньги». Это утверждение верно в случае, когда денег нет. Если материальные потребности удовлетворены процентов на семьдесят, то дальнейшее повышение доходов не заставит сотрудника работать более эффективно. Более того, слишком высокие, относительно рыночных, зарплаты могут породить в компании эффект «самодостаточности сотрудника»;
- «Самый лучший лидер проекта — это умелый вдохновитель». Трудовой энтузиазм, основанный на «идеологической обработке», быстро угасает и перестает мотивировать. Лидер просто обязан быть наставником, который помогает участникам команды выявлять свои *личные* цели и способствует их достижению;
- «Эти люди — профессионалы. Им не нужна никакая мотивация». Невозможно уговорить суперпрограммиста перейти на сопровождение системы и поддержку пользователя, даже если вы пообещаете ему в два раза большую зарплату;
- «Если правильно мотивировать, все в людях можно изменить». Взрослые люди меняются очень медленно. Порой на это требуются годы, порой — десятилетия.

Обобщение вышеизложенных положений к мотивации труда программиста может быть представлено в виде следующих высказываний [4, 20]:

1) «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) личность программиста раскрывается через четыре компонента — тело, сердце, разум и душу:

«Телу необходимы деньги и уверенность в завтрашнем дне;

Сердцу — любовь и признание;

Разуму — развитие и самосовершенствование;

Душе — самореализация»;

3) программист должен обладать определенными, особенно необходимыми при командной работе над проектом, качествами. Программист как участник команды должен:

- 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 подтвердить собственный профессионализм.

6.2.3. Прием, аттестация и увольнение программиста

Прием сотрудника на вакантную должность — ответственное и важное мероприятие. Правила, которым должен следовать руководитель при поиске специалистов и приеме их на работу, состоят в следующем [12, 20]:

- начинайте поиск специалиста только в случае, когда в нем возникла объективная потребность (поиск только при большой необходимости);
- составьте письменное описание предполагаемых функций кандидата и попросите его прокомментировать их непосредственно во время собеседования. В отсутствие такого списка кандидат может подумать, что вы не знаете, что вам нужно, и будет говорить лишь то, что вы ожидаете от него услышать (обязательность комментариев самого кандидата);
- предложите кандидату выполнить тестовое задание, которое он должен сделать сразу либо взять на дом и представить к установленному сроку;
- обязательно проводите устную проверку навыков кандидата. Если у него есть сертификаты, протестируйте его хотя бы по одному и это позволит оценить полученные кандидатом знания, а заодно и его способность решать задачи в стрессовой ситуации;
- при приеме на работу специалиста даже с высоким уровнем профессионализма, достойными человеческими качествами, хорошими рекомендациями возможны ошибки, поэтому принимайте на работу с условием прохождения испытательного срока;
- соблюдайте трудовой и гражданский кодексы при оформлении специалиста на работу. Это позволит вам избавиться от потерь при возникновении конфликтных ситуаций;
- прислушивайтесь к мнению других сотрудников компании и учитывайте их точку зрения. Это позволит сохранить в коллективе хороший микроклимат. При выборе специалиста отдавайте предпочтение тому, кто превосходит вас хотя бы по од-

ному из профессиональных качеств, в противном случае хорошую команду собрать будет трудно;

- принимайте на работу специалистов, которые способны адекватно оценивать себя. Перспективный специалист должен знать свои профессиональные недостатки и постоянно совершенствоваться (самокритичность специалиста как одно из необходимых качеств);
- при найме на работу отдавайте предпочтение специалистам с широким кругом возможностей, а не создателям «феноменального» программного кода, т. е. специалистам, способным качественно выполнять все виды работ (проектирование, кодирование, тестирование, документирование, общение с пользователями), их достаточно широкий кругозор позволит создавать более качественные продукты, отвечающие потребностям пользователей. Диплом о престижном высшем образовании немаловажен для хорошего программиста, хотя и не является необходимым условием.

Общепринятой процедурой при приеме кандидата на работу является **собеседование**, позволяющее определить уровень способностей кандидата, соответствие способностей требованиям должности; личные особенности кандидата, т. е. получить ответ на три вопроса: знает ли кандидат дело; умеет ли его делать; хочет ли это делать. При проведении собеседования рекомендуется придерживаться следующих принципов:

1) помните, что вы ведете переговоры с потенциальным партнером по бизнесу, а не пытаетесь «купить на рынке товар подешевле»;

2) имейте в виду, что несправедливо обиженный кандидат — пятно на бренде компании. Сегодня это значит не меньше, чем несправедливо обиженный клиент;

3) люди не рождаются победителями, они ими становятся. Кандидата стоит нанимать только в случае, если ему будет предоставлена возможность полностью раскрыть свои способности.

В литературе нет однозначных рекомендаций по проведению собеседования при приеме на работу программистов. В [20] сценарий проведения собеседования выглядит следующим образом.

В начале беседы следует задать следующие вопросы: что можете рассказать о вашем самом успешном проекте; что считаете своей наибольшей профессиональной удачей; что меньше всего нравилось делать на прежней работе; почему хотите поменять место работы. Ответы на эти вопросы позволят оценить:

- эмоциональность и заинтересованность кандидата при ответах. Если кандидат инертен при ответах, то можно предположить, что и к своей будущей работе он будет относиться также;
- логическую последовательность, лаконичность и ясность повествования. Умение четко формулировать и объяснять свою позицию — необходимое качество программиста;
- зрелость личности. Если кандидат винит в своих неудачах коллег, окружение, начальство — это явно несамостоятельная личность. Если не вспоминает коллег при рассказе о своих достижениях, возможно, недооценивает важность работы в команде.

Следующая группа вопросов ориентирована на выяснение желаний кандидата. Если кандидат не знает, что он хочет, его не стоит брать, скорее всего, это недостаточно зрелая личность. Кроме того, обязательно следует попытаться уточнить: чем бы он определенно не хотел заниматься; если бы работу выбирал сам, что бы он выбрал; какие качества он ценит/порицает в коллегах; как видит развитие своей карьеры.

Если кандидат объясняет, к чему он стремится, следует уточнить, адекватно ли он оценивает свои возможности. Анализ ответов на эти вопросы позволит оценить:

- целеустремленность. Если у человека есть ясное видение своих профессиональных целей и стратегии их достижения, скорее всего, это «ваш человек», даже если знаний и умений пока недостаточно. Желания «просто работать» мало. В этом случае человек мотивирован на избегание неудач, а не достижение успеха, от такого трудно ожидать инициативы, поиска новых возможностей и творчества;
- умение анализировать проблему (проводить декомпозицию на более простые компоненты, определять последовательность действий, синтезировать и обосновывать предлагаемое решение);

• способность к диалогу и эффективному взаимодействию (умение рассуждать, задавать вопросы, анализировать ответы, искать взаимовыгодное решение в конфликтных ситуациях).

В конце беседы рекомендуется предоставить возможность кандидату задать вопросы. Факт наличия вопросов говорит об активной позиции и заинтересованности кандидата в работе. К «правильным вопросам», свидетельствующим об опытности специалиста, можно отнести следующие: как организован процесс разработки ПО; используемые методы и технологии и подходы; виды отчетности и оценка результатов; наличие авралов, сверхурочных работ; система мотивации.

Аттестация сотрудников. Успешно работая в компании, сотрудник приобретает новые знания и опыт, следовательно, стоимость его услуг растет, поэтому необходима постоянная мотивация работника, пока это не сделали конкуренты. С другой стороны, сотрудник сам может переоценить стоимость своего труда, исходя из своих реальных физических и психологических затрат, которые определяются качеством обеспеченности рабочего места, атмосферой в коллективе, отношением руководства и т. п. В силу своей интроверсионности программист зачастую предпочтет поиск более высокой зарплаты в другой компании, вместо того чтобы просить о повышении руководство. Поэтому желательно ***периодически проводить аттестацию сотрудников*** и в случае необходимости пересматривать условия оплаты их труда.

В ходе аттестации сотрудников необходимо давать экспертную оценку следующих характеристик [20]: профессиональные знания и опыт, самостоятельность и лидерство, инициативность и творчество, способность четко формулировать и объяснять, умение конструктивно разрешать конфликты, сопереживание, взаимопомощь, результативность работы. В состав комиссии по аттестации необходимо включать не только руководство компании, но и людей, взаимодействующих с этим сотрудником по разным вопросам профессиональной деятельности. По результатам аттестации должен быть сделан один из следующих выводов:

• соответствует или не соответствует сотрудник занимаемой должности;

- рекомендуется ли он на повышение в должности или в резерв на эту должность;
- следует ли повышать зарплату сотруднику и насколько.

Высокая мобильность программистов и специфика рынка труда объективно приводят к высокой текучести кадров, и ***любой руководитель должен быть готов к увольнению любого сотрудника в любой момент времени***. В [4, 12] приводятся некоторые рекомендации руководителю при увольнении сотрудника.

Всегда просчитайте возможные последствия для компании от увольнения сотрудника. Какие проблемы могут появиться от разглашения конфиденциальных сведений в связи с увольнением? Быть может, программист, от которого вы намерены избавиться, обладает обширными знаниями и просто не успел их проявить?

Если сотрудника необходимо уволить, то чем раньше вы это сделаете, тем меньшими будут издержки. Если вы подумываете о том, чтобы уволить какого-то сотрудника, спланируйте свои действия заранее. Зафиксируйте в письменном виде неверные действия, которые предпринял сотрудник, и трудности, причиной которых он стал. Предложите ему один, последний, шанс исправиться. Чем дольше в вашем окружении остается плохой программист, тем хуже для вашей команды, поскольку сотрудники рискуют заразиться его вредными привычками.

Увольнение — стрессовая ситуация для сотрудника, менеджера и компании, даже если речь идет просто об уходе сотрудника по собственному желанию, не сопровождающемся никаким конфликтом. При уходе любого сотрудника (если, конечно, речь не идет об откровенных бездельниках) возникает проблема замены. Обязательно обдумайте влияние сокращения отдельного программиста на всех остальных сотрудников отдела, постарайтесь объяснить сотрудникам, по какой причине одному из их сослуживцев предложено уволиться.

Помните, что увольнение, помимо прочего, приводит к положительному эффекту — вы избавляетесь от проблемных сотрудников, а все остальные понимают, что присутствие подобных деятелей в отделе вы не потерпите.

Увольнение должно проводиться в соответствии с действующим трудовым законодательством. Так как увольнение (если оно только не происходит по собственному желанию работника) является конфликтной ситуацией, следует учитывать возможность обращения уволенного сотрудника в суд с жалобой на незаконное увольнение. Прописывайте в условиях договора найма обязательное прохождение кандидатом испытательного срока, в течение которого он должен показать свои способности к решению поставленных задач.

Чтобы избежать высокой текучести, оценивайте своих сотрудников по достоинству: платите столько, сколько они заслуживают. Чрезмерная экономия может привести к потере специалиста. Классический треугольник применительно к процессу разработки ПП выглядит следующим образом: «дешево – быстро – качественно». Из этих трех качеств сочетаться могут только любые два: если придерживаться практики привлечения низкооплачиваемых сотрудников, то в результате получите дешевое (во всех смыслах) программное обеспечение; если хотите получить качественный конечный продукт, то за качество нужно платить.

Чем моложе сотрудник, тем большее рвение он проявляет в погоне за высокой должностью. У людей постарше другие приоритеты — им нужна интересная работа и хорошие деньги. Если сотрудник хочет всего вместе, значит он либо действительно этого заслуживает, либо несколько неадекватно оценивает свои способности. Продвигая сотрудников по должностной лестнице, делайте это осторожно. Обязательно оценивайте, по силам ли им новые обязанности. Многие программисты готовы всю жизнь довольствоваться интересной и творческой работой, даже не задумываясь о том, чтобы принять на себя руководящие функции.

Контрольные вопросы

1. Перечислите и опишите роли участников проекта.
2. Прокомментируйте существующие подходы к выделению функциональных ролевых групп в команде программного проекта.
3. Приведите и прокомментируйте классификацию людей по темпераменту. Поясните, какие качества присущи программисту-флегматику и почему.

4. Перечислите и раскройте функциональные (должностные) обязанности участников проекта.

5. Приведите и опишите функциональную организационную структуру управления программным проектом.

6. Опишите и прокомментируйте роль руководителя программного проекта.

7. Дайте описание понятия «мотивация» и раскройте содержание методов мотивации.

8. Приведите конкретные примеры побуждения и мотивации участников проекта к результативной работе.

9. Приведите сравнительный анализ бюрократической модели и модели участия при управлении программным проектом.

10. Перечислите и прокомментируйте специфические особенности управления командой программистов и отличительные качества программиста как сотрудника.

7. УПРАВЛЕНИЕ СТОИМОСТЬЮ ПРОГРАММНОГО ПРОЕКТА

7.1. Оценка плановой стоимости проекта

Объективная необходимость управления стоимостью программного проекта объясняется ограниченностью финансовых, материальных и трудовых ресурсов, выделяемых для его выполнения. **Стоимость проекта** определяется совокупной стоимостью всех ресурсов и выполненных работ, необходимых для получения планируемого результата. Следует отличать стоимость проекта от его цены. ***Цена*** — это определенное количество денежных средств, которые заказчик проекта планирует оплатить исполнителю при условии успешного выполнения проекта.

Исходными данными для определения стоимости проекта являются календарный план проекта; объемы ресурсов, необходимых для выполнения работ, входящих в календарный план; рыночная стоимость единицы каждого вида ресурсов. **Управление стоимостью (затратами) проекта** включает процессы, обеспечивающие завершение проекта в рамках утвержденного бюджета:

- оценку плановой стоимости (сметы затрат) проекта;
- формирование бюджета проекта;
- мониторинг исполнения бюджета проекта (учет фактических затрат по проекту, контроль стоимостных параметров, корректировку стоимости проекта).

Оценка плановой стоимости — это процесс планирования ориентировочного объема затрат на выполнение всех работ программного проекта. Затраты в проекте разделяются на три типа [24]:

1) ***обязательства*** — плановые будущие затраты, которые возникают при заключении контрактов на разработку ПП, договоров на поставку ПП и сопутствующих услуг;

2) ***бюджетные затраты*** — распределенная во времени сметная стоимость работ, которые необходимо выполнить в процессе реализации проекта;

3) ***фактические затраты*** — реальные расходы по проекту, которые осуществляются во время выполнения работ проекта, в момент выплаты денежных средств или момент списания денежных средств со счетов компании.

Затраты отражают все расходы команды, связанные с реализацией программного проекта, и подразделяются на основные, накладные и прочие**.**

Основными являются затраты, непосредственно связанные с технологическим процессом разработки ПП. Большую часть основных затрат составляют затраты на оплату труда и материальные затраты, связанные с разработкой ПП.

В основу определения ***затрат на оплату труда*** должны быть положены:

1) иерархическая структура работ;

2) трудозатраты на выполнение работ и их распределение по этапам жизненного цикла разработки ПП;

3) количество и качественный состав специалистов, привлекаемых на каждом этапе разработки;

4) базовая месячная ставка заработной платы программиста.

В качестве универсального ***измерителя трудозатрат*** используется показатель «*человекомесяц*». Каждый человекомесяц содержит 160 человекочасов (четыре недели → пять рабочих дней → восьмичасовой рабочий день). Все методы, нормативы и статистические данные, используемые в методиках технико-экономического обоснования трудозатрат и стоимости проекта, основываются на материалах, обобщающих зарубежный и российский опыт разработки программных систем, а конкретное значение применяемых показателей зависит от типа создаваемого продукта.

В [2, 10] приводятся четыре метода оценки трудозатрат: «сверху вниз», «снизу вверх», «по аналогии» и «по исходным параметрам» проекта.

Оценка трудозатрат «*сверху вниз*» используется для определения трудозатрат на ранних стадиях разработки концепции программного проекта, когда требование к ПП сформулировано в самом общем виде. Оценка производится для проекта в целом.

Оценка трудозатрат «*снизу вверх*» может применяться в случае, когда достаточно подробно описана архитектура будущего программного продукта и иерархическая структура работ проекта. В этом случае имеется возможность оценить трудозатраты на разработку каждого отдельного программного компонента и путем суммирования вычислить окончательные трудозатраты проекта в целом. Несомненно, этот метод является более точным, так как ошибки, допущенные при оценке одного программного компонента, могут быть компенсированы при определении трудозатрат другого компонента. Преимущество метода заключается в достаточно высокой точности расчетов. Однако использование метода связано и с большими трудностями, что вызывает увеличение времени для определения трудозатрат.

В качестве исходной методики оценки трудозатрат по методам «сверху вниз» и «снизу вверх» целесообразно использовать метод экспертных оценок в комбинации с методом PERT-анализа (Project Evaluation and Review Technique) [5]. Его суть заключается в том, что для каждой работы проекта указываются три оценки трудоемкости t: оптимистическая o, пессимистическая p и реалистическая b, а итоговые трудозатраты определяются по формуле $t(i,z) = \left[t^o(i,z) + 4t^b(i,z) + t^p(i,z) \right] / 6$.

Оценка трудозатрат «*по аналогии*» используется в случаях, когда у разработчика имеется опыт создания аналогичных (подобных) программных проектов. Точность оценки прямо пропорциональна отношению трудозатрат нового проекта с проектом-аналогом. Метод оценки трудозатрат по аналогам, как правило, обходится дешевле и занимает меньше времени, чем другие методы. Оценку по аналогам целесообразно применять в случаях, когда предыдущий проект подобен текущему не только по внешним признакам, но и по сути, а у членов команды проекта, занятых подготовкой оценки, есть необходимые знания.

Методы оценки трудозатрат «*по исходным параметрам*» (*параметрические*) основываются на определении совокупности параметров, характеризующих сложность ПП, и построении на их основе математической модели оценки трудозатрат. Наиболь-

шее распространение при построении подобных математических моделей оценки трудозатрат получил метод функциональных точек [5].

Сравнительная характеристика различных методов оценки трудозатрат проекта приведена в табл. 7.1 [25].

Распределение трудозатрат и длительности разработки проекта по основным этапам жизненного цикла создания программного продукта представлено в табл. 7.2 [26].

Распределение трудозатрат различных типов специалистов по этапам жизненного цикла ПП представлено в табл. 7.3 [26].

Размер месячной базовой ставки программиста может быть определен на основе сложности ПП и сложившейся рыночной базовой ставки программиста в данном регионе. Соотношение месячной ставки специалиста-программиста к месячной ставке системного аналитика составляет 1:1,3, а к месячной ставке технического специалиста — 1:0,7.

Материальные затраты при разработке программного проекта включают:

- стоимость вычислительной и офисной техники;
- стоимость лицензий на инструментальные средства проектирования и разработки ПП;
- стоимость приобретенных со стороны компонентов, которые входят в состав разрабатываемого ПП;
- стоимость работ и услуг, выполняемых сторонними организациями.

Накладные расходы образуются в связи с организацией управления программным проектом и зависят от структуры управления проектом, эффективности менеджмента и других факторов. Величина этих расходов не зависит от объемов выполненных работ и определяется, как правило, в виде процентов от величины основных затрат.

Основными статьями накладных расходов являются фонд оплаты труда (ФОТ) аппарата управления и обслуживающего персонала, начисления на ФОТ, представительские расходы, налоги и иные платежи в бюджет.

Таблица 7.1

Сравнительная характеристика методов оценки трудозатрат по проекту

Метод оценки	Основания для использования и этап применения	Необходимые условия применения
Параметрическая оценка	Наличие методик оценки трудозатрат по проекту в целом и по отдельным работам. Применяется на любых этапах проекта. Точность зависит от четких оценок объемов работ и их нормативной длительности	Наличие возможностей для нормирования длительности работ и расчета оценок исходя из объемных параметров работ. Наличие нормативов длительности отдельных типовых операций
Оценка по аналогам	Недостаток детальной информации. Применяется на ранних этапах проекта	Схожесть работ по содержанию и типу. Наличие информации о фактической длительности работ-аналогов. Наличие опыта у участников проекта
Оценка «снизу вверх»	Необходимость в уточненной либо повторной оценке длительности проекта. Рекомендуется для фазы детального планирования	Невысокая трудоемкость и объема работ в целом по проекту, и отдельных операций. Наличие достаточно точных оценок необходимых ресурсов для отдельных операций. Наличие ретроспективной информации о длительности отдельных типовых операций. Наличие нормативов затрат. Тщательно проработанная ИСР
Оценка «сверху вниз»	Необходимость быстрой укрупненной оценки длительности проекта. Применяется для фазы замысла (разработки идеи) проекта	Возможность укрупненной оценки длительности всего проекта

Таблица 7.2

Распределение трудозатрат на разработку проекта по основным этапам жизненного цикла создания ПП

Этапы жизненного цикла	Трудозатраты α, %
1. Анализ предметной области и разработка требований	8–10
2. Проектирование	14–22
3. Программирование	41–46
4. Тестирование и комплексные испытания	24–27

Таблица 7.3

Распределение специалистов по этапам жизненного цикла ПП

Этапы жизненного цикла	Типы специалистов, %		
	Аналитики	Программисты	Технические специалисты
1. Анализ предметной области и разработка требований	40	20	40
2. Проектирование	35	35	30
3. Программирование	10	65	25
4. Тестирование и комплексные испытания	15	60	25

Прочие затраты включают расходы на техническую литературу, сменные носители, картриджи, канцелярские товары, командировки, оплату телефонных переговоров, услуг связи, подписку и др.

Итоговая плановая смета затрат проекта по статьям калькуляции и распределение их по плановым периодам представляется в форме таблицы (табл. 7.4) с последующим описанием каждой статьи затрат.

Очевидно, что определить реальную стоимость проекта практически невозможно, поэтому процедура планирования затрат носит итеративный характер и основывается на информации, известной в конкретный момент времени. По ходу выполнения проекта стоимость его последовательно уточняется. При этом точность оценки затрат повышается по мере приближения срока окончания проекта.

Таблица 7.4

Структура затрат на разработку проекта

Наименования статей затрат	Плановые периоды			
	I кв.	II кв.	III кв.	IV кв.
1. Фонд оплаты труда исполнителей (ФОТ)				
2. Начисления на ФОТ				
3. Приобретение готовых компонентов, расходных материалов				
4. Увеличение стоимости основных фондов (приобретение компьютерной и офисной техники, комплектующих)				
5. Командировочные расходы				
6. Оплата услуг сторонних организаций				
7. Затраты на подготовку и переподготовку персонала				
8. Оплата коммунальных услуг				
9. Амортизация				
10. Накладные расходы				
11. Налог на добавленную стоимость 12. Налог на прибыль				
13. Оплата услуг связи				

В проектном менеджменте предлагается использовать четыре типа оценки затрат по этапам проекта [25] (рис. 7.1):

1) ***грубый порядок оценки*** — ожидания по стоимости проекта на этапе оценки привлекательности идеи;

2) ***предварительный порядок оценки*** — предположения по стоимости проекта, рассчитанные на этапе технико-экономического обоснования проекта или разработки бизнес-плана;

3) ***бюджетная оценка*** — оценка стоимости проекта, полученная на основе данных, предоставленных исполнителями работ на этапе планирования работ;

4) ***точная оценка*** — оценка стоимости, включаемая в бюджет при определении окончательной плановой стоимости проекта перед переходом к этапу реализации.

Диапазоны изменения оценки точности стоимости проекта по этапам представлены на рис. 7.1 [25].

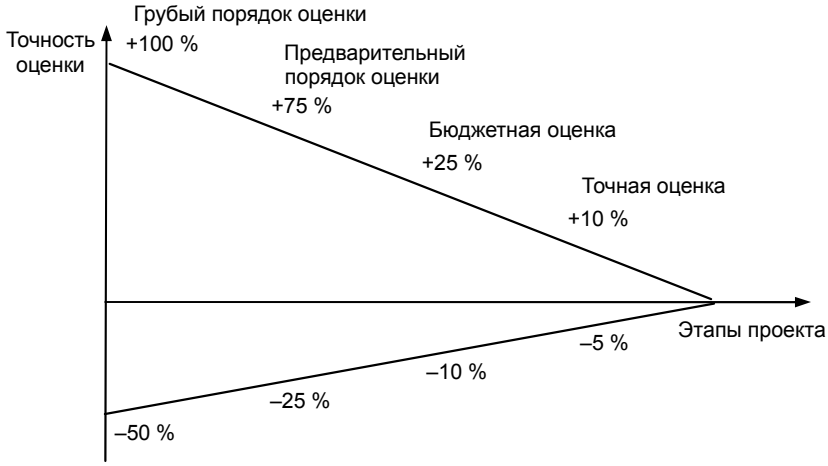

Рис. 7.1. Диапазоны изменения оценки точности стоимости проекта

7.2. Формирование бюджета программного проекта

Основным плановым документом, определяющим структуру затрат по проекту и источников их покрытия, является **бюджет проекта.** Разработать точный, полный и реальный бюджет с первой попытки практически невозможно, поэтому по ходу реализации проекта бюджет уточняется и корректируется.

Структура бюджета состоит из расходной и доходной части. Основу расходной части составляет плановая смета затрат проекта по статьям калькуляции. В доходной части рассматриваются:

1) ***собственные средства*** участников проекта;

2) ***заемные средства*** (кредиты банков);

3) ***привлеченные средства*** (средства инвесторов проекта и различных инвестиционных фондов);

4) ***гранты, субсидии*** федеральных целевых программ;

5) ***средства заказчиков,*** полученные за выполненные работы по проекту.

Если при составлении бюджета проекта доходная часть отсутствует, то бюджетом считается плановая смета затрат — документ, содержащий обоснование распределения затрат по видам работ, статьям расходов и времени. Таким образом, принципиальным отличием бюджета проекта от сметы затрат является наличие в бюджете календарного графика будущих расходов и графика поступления финансовых ресурсов на их покрытие.

Доходы следует планировать в разрезе каждого планового периода от всех источников поступления денежных средств (табл. 7.5). Периодом поступления доходов следует считать зачисление денежных средств на расчетные счета организации.

Таблица 7.5

Динамика поступления финансовых ресурсов

Виды доходов	Плановые периоды			
	I кв.	II кв.	III кв.	IV кв.
1. Кредиты банков				
2. Средства инвесторов проекта				
3. Средства инвестиционных фондов				
4. Средства, полученные по грантам				
5. Субсидии федеральных целевых программ				
6. Средства заказчиков, полученные за выполненные работы				
7. Собственные средства команды проекта				

Заключительный этап бюджетного планирования — составление баланса поступления и расходования денежных средств в каждом плановом периоде. Прогноз осуществляется на определенный период в разрезе подпериодов: год по кварталам, год по месяцам и т. п. (табл. 7.6)

Таблица 7.6

Движение денежных средств

Финансовые средства	Плановые периоды			
	I кв.	II кв.	III кв.	IV кв.
1. Остаток на начало периода	150	30	80	30
2. Поступления	200	250	250	350
3. Платежи	320	200	300	330
4. Баланс денежных средств	30	80	30	50

Расчеты по движению денежных средств выполняются в следующей последовательности:

1) прогнозирование денежных поступлений;

2) прогнозирование финансовых затрат по проекту;

3) формирование текущего баланса денежных средств;

4) определение финансовой потребности в краткосрочном финансировании.

Анализ баланса денежных средств покажет, достаточно ли их у команды проекта для обеспечения текущей деятельности, понадобится ли привлечение денежной массы из других источников, например получение краткосрочного кредита.

7.3. Мониторинг исполнения бюджета проекта

Разработка бюджета проекта важная, но далеко не окончательная процедура управления стоимостью программного проекта. Основные усилия менеджеров проекта по управлению стоимостью направлены на осуществление контроля и анализа взаимосвязей между расходованием финансовых средств и конкретными работами, на выполнение которых они потрачены. Обязательность контроля и анализа расходования средств на выполнение проекта при наличии высоких рисков обусловливают необходимость создания эффективной системы постоянного контроля и анализа за состоянием характеристик «железного треугольника» (бюджета, времени, содержания) и проведением корректирующих действий, если отклонения фактических характеристик проекта от плановых достигают критического порога.

Основной метод решения этих задач — **метод освоенного объема —** позволяет менеджеру проекта одновременно оценивать и анализировать текущее состояние проекта как по освоению бюджета (фактическим затратам), так и выполнению календарного плана проекта [10]. При этом и фактическое исполнение календарного плана, и фактические затраты измеряются в денежном выражении. Стоимость отставания календарного плана работ по проекту определяется на основе следующих показате-

лей: количества периодов отставания, численности привлеченных исполнителей, средней ставки оплаты труда исполнителя. В качестве исходных ключевых параметров мониторинга состояния проекта методом освоенного объема используются [10, 27] (рис. 7.2):

1) ***плановые бюджетные затраты*** (ПБЗ) — плановая стоимость (объем) запланированных работ проекта, т. е. утвержденная расходная часть бюджета проекта. Учитывая, что весь комплекс работ ЖЦ проекта распределен по времени, плановый бюджет проекта следует представлять нарастающим итогом;

2***) освоенный бюджет*** (ОБ) — плановая стоимость объема выполненных работ проекта, представленных в бюджете проекта нарастающим итогом;

3) ***фактический бюджет*** (ФБ) — фактическая стоимость выполненных работ проекта, т. е. фактически зарегистрированная стоимость выполнения работ в соответствии с календарным планом проекта. По мере реализации проекта фактическая стоимость также приводится нарастающим итогом.

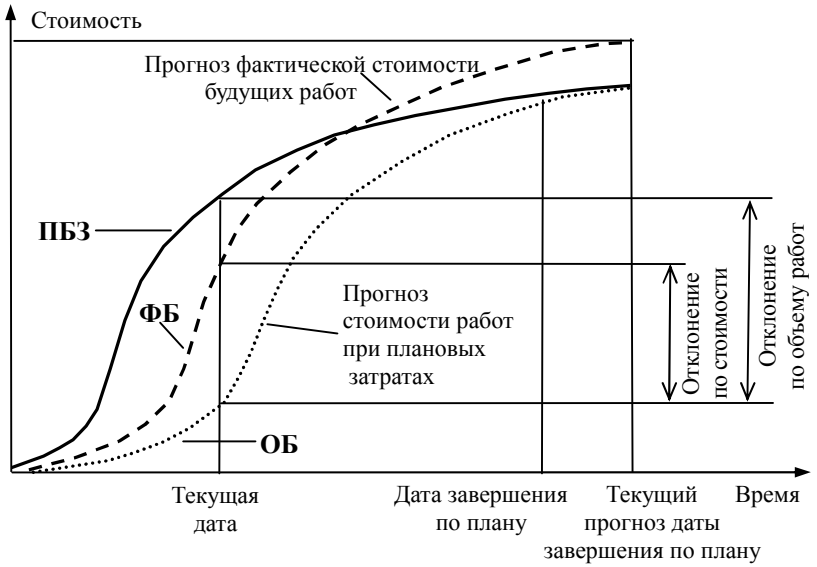

Рис. 7.2. Динамика параметров бюджета проекта:
ПБЗ — плановые бюджетные затраты; ФБ — фактический бюджет;
ОБ — освоенный бюджет

Очевидно, что если проект выполняется в соответствии с календарным планом, то в каждый момент времени все три параметра будут одинаковыми. Для повышения объективности и точности организации мониторинга стоимости проекта необходимо заранее определить правила измерения параметров фактического и освоенного бюджета: как производить оплату работ (предоплату, оплату по факту, стимулирующие выплаты и т. д.); как учитывать и списывать единовременные затраты (приобретение компьютерной и офисной техники, лицензионного ПО, оплата аренды и т. д.); как производить оплату работ, выставленных на аутсорсинг.

Перечисленные параметры положены в основу расчета показателей, характеризующих исполнение бюджета проекта и соблюдение календарного плана работ [10, 27]:

1. Отклонение по стоимости на данный период времени ($CV = \text{ОБ} - \text{ФБ}$) характеризует выполнение плана по стоимости проекта и вычисляется как разница между стоимостью выполненных работ и затратами на их выполнение. Физический смысл расчета показателя CV — сравнение реально выполненных работ по плановым (бюджетным) и фактическим расходам.

Если $CV > 0$, в проекте имеет место экономия бюджета, в противном случае имеется проблемная ситуация в виде перерасхода бюджета. Вместе с тем анализ только данного показателя не позволяет сделать вывод о причинах перерасхода. По величине CV не видно, как выполняется календарный план проекта: может быть, перерасход средств обусловлен перевыполнением плана; возможно, что работы были выполнены в соответствии с планом, а перерасход связан с предоплатой будущих работ. По окончании проекта величина CV будет равна разнице между объемом бюджета по завершении проекта и фактическими затратами на выполнение работ.

2. Отклонение по срокам ($SV = \text{ОБ} - \text{ПБЗ}$) характеризует выполнение календарного плана проекта и вычисляется как разница между стоимостью выполненных работ и стоимостью запланированных работ на данный период времени. Физический

смысл расчета показателя *SV* — сравнение в плановых (бюджетных) деньгах объема работ, который реально выполнен, и объема работ, который должен быть выполнен согласно календарному плану проекта. Если $SV > 0$, то в проекте имеет место опережение календарного плана работ на данный период времени, в противном случае имеется проблемная ситуация в виде срыва календарного плана. По окончании проекта величина *SV* будет равна нулю, так как все плановые объемы запланированных работ проекта должны быть освоены.

Анализ абсолютных показателей *CV* и *SV* дает возможность сделать выводы *только о текущем состоянии проекта*. Для определения тенденций изменения значений показателей *CV* и *SV в будущих периодах времени* в методе освоенного объема предлагается вычислять два относительных показателя: индекс выполнения стоимости и индекс выполнения сроков проекта.

3. Индекс выполнения стоимости (CPI = ОБ/ФБ) — относительный показатель, характеризующий эффективность расходования денежных средств в проекте. Индекс определяется как отношение показателей объема выполненных работ и фактической стоимости выполнения проекта. Если $CPI > 1$, имеет место недоосвоение бюджета проекта, в противном случае имеется проблемная ситуация, указывающая на перерасход финансовых ресурсов при выполнении работ проекта.

4. Индекс выполнения сроков (SPI = ОБ/ПБ) — относительный показатель, характеризующий эффективность выполнения календарного плана проекта. Индекс определяется как отношение показателей достигнутых и запланированных объемов выполнения проекта. Значение индекса выполнения сроков большее 1 ($SPI > 1$) указывает на то, что на данный момент времени выполнено работ больше, чем было запланировано, т. е. в проекте имеет место опережение календарного графика работ, в противном случае имеется отставание от графика.

Полученные значения индексов CPI и SPI используются менеджером проекта для прогнозирования будущих стоимостных показателей проекта. Именно прогноз оставшейся стоимости

проекта и является главной задачей по управлению стоимостью проекта. Если значения индексов выполнения стоимости и сроков значительно меньше единицы, то становится очевидным, что плановый бюджет проекта будет превышен. В этом случае менеджеру проекта необходимо определять величину ***фактической стоимости (бюджета) проекта к моменту его завершения*** *FAC*:

$$EAC = \text{ПБЗ} / CPI,$$

где ПБЗ — величина плановых бюджетных затрат нарастающим итогом на момент окончания проекта.

Эта оценка характеризуется пессимистическим подходом к управлению проектами, показатель *EAC* рассчитывается из предположения, что отставания (как по бюджету, так и по срокам), допущенные при выполнении проекта на момент текущей контрольной даты, будут продолжаться с такой же интенсивностью и в оставшейся части проекта.

При оптимистическом взгляде на управление проектом предполагается, что допущенные отставания не будут повторяться и оставшаяся часть проекта станет исполняться в соответствии с планом. В этом случае ***фактическая стоимость (бюджет) проекта к моменту его завершения*** будет определяться по формуле $EAC = \text{ФБ} - \text{ПБ}$, где ПБ — остаток планового бюджета проекта на текущий момент времени.

Полученное значение *EAC* используется в дальнейшем для определения суммы денежных средств *ETC*, которые предстоит дополнительно израсходовать для завершения проекта, по формуле $ETC = EAC - \text{ФБ}$. Таким образом, *ETC* представляет собой ***прогнозное значение стоимости выполнения оставшихся работ проекта от момента анализа до окончания проекта***.

Вычисление и анализ вышеперечисленных показателей позволяют менеджеру проекта:

- определять текущее состояние проекта (перерасход бюджета и/или отставание по срокам);
- прогнозировать ход выполнения проекта в будущем;
- при необходимости вносить предложения по корректировке стоимости и сроков выполнения проекта.

Для принятия решений по выработке корректирующих воздействий необходимо определить контрольные точки мониторинга хода выполнения проекта и величины пороговых (критических) отклонений показателей от плановых значений. Процесс корректировки плана исполнения программного проекта может проходить по следующим направлениям:

- изменение директивных (плановых) сроков работ, входящих в критический путь, что может автоматически привести к пересмотру сроков окончания проекта;
- привлечение дополнительных трудовых, а значит и финансовых ресурсов для выполнения работ, входящих в критический путь, что автоматически приводит к увеличению бюджета проекта;
- перераспределение трудовых и финансовых ресурсов для ликвидации отставания как по срокам, так и по стоимости проекта, что возможно только за счет изменения механизмов управления проектом.

Все перечисленные подходы к корректировке плана исполнения проекта реализуются в рамках процесса управления изменениями проекта.

Контрольные вопросы

1. Перечислите и прокомментируйте процессы управления стоимостью проекта.
2. Перечислите и прокомментируйте методы оценки трудозатрат по проекту.
3. Проведите сравнительный анализ методов оценки трудозатрат по проекту.
4. Перечислите и прокомментируйте структуру затрат (расходов) на разработку программного проекта.
5. Перечислите и прокомментируйте содержание доходной и расходной части бюджета программного проекта.
6. Раскройте содержание работ и ключевые параметры мониторинга исполнения бюджета проекта.
7. Перечислите и прокомментируйте основные показатели, характеризующие исполнение бюджета и календарного плана работ по выполнению программного проекта.

8. УПРАВЛЕНИЕ РИСКАМИ ПРОГРАММНОГО ПРОЕКТА

8.1. Основные понятия риска и рискообразующих факторов

В литературе под **риском проекта** понимается событие или условие, которое может произойти либо не произойти в будущем при реализации программного проекта и негативно повлиять на степень достижения одной или нескольких характеристик целей проекта.

Учитывая явную логическую взаимосвязь между целями проекта и возможными рисками, можно предположить, что при разработке программного проекта могут возникнуть четыре типа (категории) рисков [25, 28, 29]:

1) срыв плановых сроков проекта;

2) превышение стоимости (бюджета) проекта;

3) критические отклонения по составу и содержанию проекта (невыполнение функциональных требований);

4) критическое отклонение по показателям качества проекта (невыполнение нефункциональных требований).

Нельзя сказать заранее, произойдет ли это событие, но точно известно, что его возникновение скажется на функционале, качестве, стоимости или сроках выполнения программного проекта.

В [28] предлагается рассматривать риск с трех точек зрения:

1) риск как возможность угрозы бизнесу;

2) риск как негативное событие, не позволяющее достичь в полной мере цели проекта;

3) риск как неопределенность между возникающими неблагоприятными ситуациями и возможными действиями по их устранению.

В табл. 8.1 приведены возможные для каждого этапа ЖЦ ПП варианты формулировок целей и рисков проекта.

Таблица 8.1

Соотношение целей и рисков программного проекта

Этапы ЖЦ	Цели	Риски
Инициация	Выбор продуктово-рыночного направления и разработка концепции будущего коммерческого ПП	Ошибки в выборе функционала и, как следствие, невостребованность ПП, нарушение сроков окупаемости проекта
Разработка	Разработка коммерческого ПП с требуемым функционалом при ограничениях на сроки и бюджет проекта	Критические отклонения по срокам и бюджету проекта
Продвижение	Обеспечение в определенном интервале времени заданного уровня объема продаж ПП при ограничении на бюджет рекламной компании	Несоответствие между желаемыми и фактическими объемами продаж ПП. Наличие критических отклонений по бюджету программы продвижения
Внедрение	Обеспечение процесса поставки и внедрения ПП в соответствии с договорными отношениями между разработчиком и заказчиком	Критические отклонения по срокам и бюджету внедрения ПП

Появление каждого из рисков возможно при наличии причин (процессов или явлений), способствующих его возникновению и поясняющих, почему наступление риска неизбежно. Такие явления принято называть **рискообразующими факторами** [28].

Для стандартизации и унификации терминологии рискообразующих факторов, оценки их влияния на конкретные цели и фазы программного проекта необходима систематизация и классификация факторов по определенным признакам.

В данном случае для классификации факторов риска используется иерархический метод классификации, при котором

множество рискообразующих факторов последовательно в соответствии с выбранными основаниями (признаками) классификации разбивается на подмножества (рис. 8.1) [30].

На первом уровне классификатора в качестве основания классификации используется модель жизненного цикла программного проекта: инициация – разработка – продвижение – внедрение.

На втором уровне для каждого этапа жизненного цикла ПП можно выделить внешние и внутренние факторы. *Внешние факторы* — это события, которые лежат за пределами контроля и влияния команды проекта. *Внутренние факторы* (специфичные для конкретной компании) определяют способность самой организации успешно реализовать проект.

На третьем уровне проявление внешних факторов обусловливается как политикой государства в отношении бизнеса малых IT-компаний, так и различными ситуациями на продуктовом, финансовом рынках и рынке труда. Набор внутренних факторов определяется составом системной модели деятельности: средствами деятельности, предметами деятельности, кадрами, технологией.

Четвертый уровень представляет собой набор первичных факторов риска. При этом допускается возможность принадлежности одного и того же фактора разным основаниям классификации.

На основе предложенного классификатора и обобщения литературы [1, 5, 28, 29, 31, 32] в табл. 8.2 представлено множество рискообразующих факторов, свойственных программным проектам, без их относительного распределения по этапам ЖЦ ПП. Очевидно, что приведенный перечень рискообразующих факторов не претендует на полноту и может быть дополнен. Вместе с тем эти сведения будут полезны менеджерам проектов при первичном отборе факторов, влияющих на достижение целей конкретного проекта.

Рис. 8.1. Классификация факторов риска программного проекта

Таблица 8.2

Состав внутренних и внешних первичных факторов риска программных проектов

Основания классификации	Первичные факторы риска
Внутренние факторы риска	
1. Продукт	1. Недостаток финансирования проекта. 2. Нестабильное финансирование работ по проекту. 3. Высокие невозвратные издержки проекта. 4. Нереальные сроки выполнения проекта. 5. Частые изменения требований к проекту у заказчика. 6. Неполные или нечеткие требования к программному продукту. 7. Недостаточная поддержка проекта руководством заказчика. 8. Возможная смена руководства у заказчика. 9. Низкая готовность заказчиков к внедрению ПП. 10. Низкий уровень сервисов интеграции ПП с информационными системами заказчиков
2. Персонал	1. Отсутствие у команды необходимых трудовых ресурсов. 2. Высокая текучесть кадров. 3. Отсутствие у команды опыта, необходимого для реализации проекта. 4. Разрыв в квалификации специалистов разных областей знаний. 5. Саботаж отдельных членов команды проекта
3. Технология реализации продукта	1. Недостаточная зрелость технологий, применяемых в процессе реализации проекта. 2. Высокая скорость устаревания применяемых технологий. 3. Ошибки при выборе программно-аппаратной платформы и средств реализации продукта. 4. Недостаточные навыки владения новыми инструментальными средствами разработки ПП у исполнителей

Продолжение табл. 8.2

Основания классификации	Первичные факторы риска
4. Технология управления программным проектом	1. Отсутствие у разработчика эффективной методологии управления программным проектом. 2. Отсутствие опыта управления командой разработчиков. 3. Отсутствие адекватной методологии управления требованиями и изменениями. 4. Ошибки в расчетах финансовых затрат на разработку (продвижение) ПП. 5. Ошибки в оценках трудоемкости и сроков работ. 6. Несоблюдение стандартов при разработке (продвижении) ПП. 7. Недостатки в планировании проекта, появление «забытых» работ. 8. Недостатки во внутренней организации работ, неумение работать в реальном времени. 9. Недооценка взаимосвязи между работами по проекту. 10. Ошибки при выборе потребительских предпочтений пользователей. 11. Ошибочный выбор целевого сегмента. 12. Ошибки выбора каналов и инструментов коммуникаций. 13. Недостаточная проработка коммуникационных сообщений. 14. Отсутствие эффективного взаимодействия с заказчиком
Внешние первичные факторы риска	
1. Государство	1. Изменение нормативно-правовых механизмов ведения бизнеса. 2. Изменение нормативного регулирования бизнес-процессов предметной области. 3. Отсутствие устоявшейся законотворческой практики по защите авторских и имущественных прав. 4. Изменение экономической ситуации в государстве, отрасли, регионе.

Окончание табл. 8.2

Основания классификации	Первичные факторы риска
2. Финансовый рынок	1. Колебания курса валют. 2. Изменение ставок по кредитам
3. Рынок труда	1. Отсутствие на рынке труда узкопрофильных специалистов
4. Продуктовый рынок	
4.1. Потребители	1. Неполнота и неточность оценки потребностей потенциального рынка. 2. Несоответствие функциональных характеристик ПП потребностям потребителей. 3. Слабое влияние внедрения ПП на совершенствование бизнес-процессов компаний-потребителей. 4. Несовместимость предлагаемого продукта с ПП компаний-потребителей. 5. Несоответствие нефункциональных характеристик ПП программно-аппаратным средствам и коммуникациям потребителей. 6. Ошибочные прогнозы объема продаж. 7. Несоответствие рыночной цены возможностям потенциальных потребителей. 8. Ухудшение финансовой ситуации компаний, являющихся потенциальными потребителями. 9. Невостребованность ПП рынком. 10. Скрытое противостояние специалистов-потребителей внедрению ПП. 11. Низкий уровень подготовки пользователей ПП
4.2. Партнеры	1. Появление на рынке новых аналогичных продуктов. 2. Непредсказуемое поведение конкурентов. 3. Дискредитация ПП со стороны конкурентов. 4. Пиратское распространение копий ПП. 5. Ненадежная работа аутсорсинговых компаний. 6. Изменение цен на услуги связи. 7. Изменение цен на размещение рекламы

8.2. Содержание этапов управления рисками

8.2.1. Идентификация и анализ рисков и рискообразующих факторов

Идентификация — этап, позволяющий выявить и коллективно обсудить возможность проявления риска и рискообразующих факторов, способных повлиять на цели проекта, документально описать результаты в виде логически увязанных их характеристик.

Выявление рисков — ответственный и важный этап проекта. Знание о существовании рисков — необходимое условие эффективной работы по их предотвращению Исходными данными для выявления возможных рисков могут служить базы знаний о рисках в прежних выполненных проектах компании; информация из научно-технических журналов, аналитические обзоры в открытой печати и другие источники в данной области. Каждый проект задумывается и разрабатывается на основании ряда ограничений и допущений. Неопределенность в них следует также рассматривать в качестве потенциального источника возникновения рисков.

Описание факторов риска следует проводить с разъяснением причинно-следственной связи между реально существующей причиной и потенциально возможным, еще не случившимся событием или ситуацией (рис. 8.2) [1]. ***Условие*** содержит описание причины, которая может сделать ре-зультат проекта убыточным либо же сократить получаемую от проекта прибыль. ***Последствие*** описывает ту нежелательную ситуацию при наступлении рискоообразующего фактора, которую следует избежать. ***Воздействие на цели*** отражают негативные изменения характеристик целей проекта.

Последовательность действий команды проекта по выявлению и описанию рисков может быть определена на основе предлагаемого классификатора рискообразующих факторов, а в качестве

методов и инструментов использованы метод мозгового штурма, опросы экспертов, SWOT-анализ. Результатом идентификации рисков должен стать список рисков с описанием их основных характеристик (табл. 8.3).

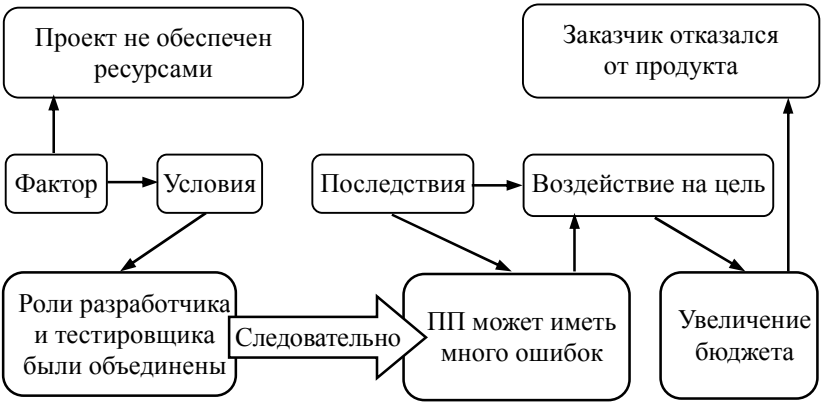

Рис. 8.2. Основные характеристики риска и их взаимосвязи

На этапе анализа рисков необходимо определять следующие качественные и количественные оценки рисков и рискообразующих факторов:

- вероятность появления рискообразующих факторов и уровень их влияния на цели проекта;
- временной диапазон проявления рискообразующих факторов;
- множество рискообразующих факторов, оказывающих критическое влияние на результаты проекта и требующие скорейшего реагирования;
- вероятность достижения целей проекта.

Термин **«вероятность»** означает меру возможности того, что появление рискообразующего фактора, описанное в его формулировке, действительно наступит. Вероятность определяется как соотношение количества проектов, когда рискообразующие факторы имели место, к общему числу проектов.

Таблица 8.3

Фрагмент описания схемы рискообразующих факторов

Факторы	Описание фактора		
	Условие	Последствия	Воздействие на цели
1. Изменение нормативного регулирования бизнес-процессов предметной области	Принятие законов по регулированию бизнес-процессов в области применения ПП	Необходимость доработок функционала ПП	Увеличение бюджета проекта
2. Изменение экономической ситуации при продвижении ПП	Экономический кризис	Изменение платежеспособности потребителей	Сокращение объемов продаж
3. Появление новых аналогичных продуктов	Выход на рынок новых аналогичных продуктов	Усиление конкуренции	Сокращение объемов продаж
4. Отсутствие у команды проекта необходимого опыта по разработке ПП	Ошибки при реализации проекта	Необходимость доработок проекта	Срыв сроков разработки. Увеличение бюджета проекта
5. Ошибки выбора каналов и инструментов коммуникаций	Снижение необходимого уровня информирования целевой аудитории	Несоответствие плановых и фактических показателей результативности программы продвижения	Сокращение объемов продаж

Влияние рискообразующих факторов на цели проекта (стоимость, сроки, содержание, качество) отражает меру **негативных последствий** (**ожидаемых потерь**) команды проекта при реагировании на конкретный рискообразующий фактор. Потери могут оцениваться в виде возможных убытков в объемах продаж, увеличении бюджета проекта, дополнительных затрат на предотвращение рисков и т. д.

Оценка вероятности и воздействия может быть проведена по каждому рискообразующему фактору отдельно для каждой цели проекта: стоимости, времени, содержания и качества. При этом вероятность должна быть больше нуля (иначе рискообразующий фактор просто не влияет на цели) и меньше единицы (иначе проявление фактора не содержит неопределенности и представляет собой известную проблему).

Оценка может производиться на основании результатов опросов членов команды проекта и экспертов (специалистов, имеющих широкие познания в оцениваемой области).

Значения показателей вероятности и воздействия могут оцениваться как в количественных, так и в качественных шкалах. Среди количественных методов оценки вероятности рискообразующих факторов и их влияния на цели проекта наиболее часто используется метод PERT-анализа (Project Evaluation and Review Technique) [5]. Суть его заключается в том, что для каждой характеристики эксперту необходимо указывать три оценки — оптимистическую, наиболее вероятную (реалистическую) и пессимистическую.

Тогда вероятность наступления рискообразующих факторов можно вычислять по формуле

$$P(x_j) = [p_1(x_j) + 4p_2(x_j) + p_3(x_j)]/6, \qquad (8.1)$$

где $p_1(x_j)$, $p_2(x_j)$, $p_3(x_j)$ — соответственно оптимистическая, реалистическая и пессимистическая вероятности наступления фактора.

Однако, учитывая сформулированные в разделе 1 особенности ПП, оценки рискообразующих факторов не всегда можно описать с помощью числовых значений. В таком случае для этих целей необходимо использовать качественную шкалу с несколь-

кими градациями, например <низкая, средняя, высокая> вероятности. Кроме того, при отсутствии достоверных статистических данных о проекте, оценки рискообразующих факторов формируются, как правило, путем проведения опроса множества специалистов (экспертов), что тоже обуславливает применение интервальных оценок, а не конкретных числовых значений. В этом случае целесообразно использовать математический аппарат нечеткой логики [33].

Для оценки вероятности проявления рисков и рискообразующих факторов и степени их влияния на цели проекта применяются шкалы различной градации. Так, в [1] оценку вероятности предлагается проводить в четырех градациях:

1) *низкая*. Цели проекта и требования хорошо поняты и документированы, масштаб и рамки заданы четко, ресурсы доступны в полном объеме, при реализации проекта не требуется освоения новых инструментальных средств разработки;

2) *средняя*. Цели проекта определены более-менее четко, масштаб и рамки заданы хорошо, ресурсы в основном доступны, при реализации проекта используются новые, но хорошо освоенные командой проекта инструментальные средства;

3) *выше среднего*. Цели проекта недостаточно четки, требования изложены нечетко и могут изменяться, масштаб и рамки проекта определены недостаточно четко, ресурсы сильно ограничены, проект реализуется с использованием новых для команды инструментальных средств;

4) *высокая.* Цели проекта нечетки, требования не определены, масштаб и рамки непонятны, ресурсы практически отсутствуют, при реализации проекта используются новые, но недостаточно хорошо освоенные командой проекта инструментальные средства.

Эксперты при оценке вероятности должны анализировать причины и условия, которые могут привести к проявлению риска либо рискообразующего фактора.

В качестве примера приведем шкалу оценки вероятности проявления риска программного проекта и качественную интерпретацию условий его проявления [1] (табл. 8.4).

Таблица 8.4

Шкала оценки вероятности проявления рисков
и рискообразующих факторов

Качественная оценка	Низкая	Средняя	Выше среднего	Высокая
Интервал количественной оценки	0,01–0,24	0,25–0,49	0,5–0,74	0,75–1,0

В таблице 8.5 с учетом рекомендаций [28] представлена возможная шкала оценки влияния рискообразующих факторов на цели проекта. Ввиду присутствия неопределенности интервалы оценки могут пересекаться.

По результатам идентификации и анализа описанных выше показателей необходимо выделить множество рискообразующих факторов, оказывающих критическое влияние на результаты проекта и требующих скорейшего реагирования на них. Эта процедура может быть реализована путем построения и анализа матрицы <вероятность-воздействие> (табл. 8.6) [28].

Для построения матрицы используются полученные ранее оценки вероятности появления рискообразующих факторов и уровень их влияния на цели проекта. Произведение этих величин определяет единую интегральную оценку критичности рискообразующего фактора.

Одной из важных характеристик рисков и рискообразующих факторов является ***близость их 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. 8.7.

Интегральная оценка критичности и характеристика близости наступления рискообразующего фактора являются основой для его ранжирования. Ранг определяет его порядковый номер в полной совокупности рисков проекта. Чем выше ранг, тем более опасен риск (табл. 8.8).

Таблица 8.5

Шкала оценки влияния факторов на цели проекта

Цель проекта	Влияние фактора				
	Незначительное менее 0,15	Умеренное 0,1–0,4	Высокое 0,2–0,6	Критичное 0,5–0,9	Катастрофическое более 0,8
1. Стоимость	Незначительное увеличение стоимости	Увеличение стоимости < 10 %	Увеличение стоимости 10–20 %	Увеличение стоимости 20–40 %	Увеличение стоимости >40 %
2. Сроки	Незначительное увеличение времени	Увеличение времени < 5 %	Увеличение времени 5–10 %	Увеличение времени 10–20 %	Увеличение стоимости >20 %
3. Содержание	Едва заметное уменьшение содержание	Затронуты второстепенные области содержания	Затронуты основные области содержания	Уменьшение содержания неприемлемо для заказчика	Конечный продукт проекта практически бесполезен
4. Качество	Едва заметное понижение качества	Затронуты только самые трудоемкие приложения	Для понижения качества требуется одобрение заказчика	Понижение качества неприемлемо для заказчика	Конечный продукт проекта практически бесполезен

Таблица 8.6

Матрица вероятности и воздействий рискообразующих факторов

Вероятность	Воздействия				
	0,01–0,15	0,1–0,4	0,2–0,6	0,5–0,9	0,8–0,99
0,75–1,0	0,0075–0,15	0,075–0,4	0,15–0,6	0,375–0,9	0,6–0,99
0,5–0,74	0,005–0,111				
0,25–0,49					
0,01–0,244					

Таблица 8.7

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

Таблица 8.8

Матрица рангов выявленных рисков проекта

Факторы	Вероятность	Воздействие	Ранг
Отсутствие у команды проекта необходимого опыта по разработке ПП	Высокая	Катастрофическое	9
Появление новых аналогичных продуктов	Высокая	Критичное	6
Изменение нормативного регулирования бизнес-процессов предметной области	Низкая	Критичное	4

Итоговые результаты анализа рисков подробно оформляются в виде документа, представленного в табл. 8.9.

Таблица 8.9

Пример карточки с описанием риска

Номер: R-101	Категория
Фактор: отсутствие у команды проекта необходимого опыта по разработке ПП	Условия: ошибки при реализации проекта
Последствия: необходима доработка проекта	Воздействие: увеличение бюджета проекта
Вероятность: высокая	Степень воздействия:
Близость: очень скоро	Ранг: 6
Исходные данные: «Содержание проекта», «План обеспечения ресурсами», протоколы совещаний № 21 от …, № 27 от …….	

Ранжирование факторов риска позволяет команде проекта распределить их по категориям опасности последствий [1]:

- рискообразующие факторы, требующие немедленного реагирования;
- рискообразующие факторы, реагирование на которые можно выполнить позже;
- рискообразующие факторы, требующие дополнительного рассмотрения (включая количественный анализ);
- рискообразующие факторы, за которыми в дальнейшем должно проводиться наблюдение.

8.2.2. Определение интегральной оценки риска программного проекта

Влияние отдельных групп рискообразующих факторов на цели проекта и определение вероятности достижения целей проекта и категории интегрального риска можно определить на основе описанной в разделе 4 модели функциональных зависимостей. С учетом этого и предложенной системы классификации первичных рискообразующих факторов структуру модели сети функциональных зависимостей можно представить в виде ориентированного графа (рис. 8.3).

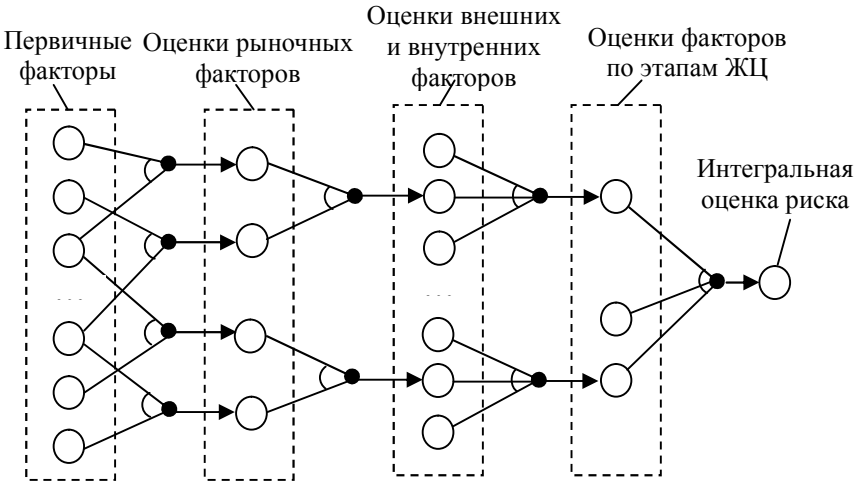

Рис. 8.3. Структура модели оценки риска программного проекта

В первом слое модели описываются первичные рискообразующие факторы, на втором, третьем и четвертом слоях вычисляются промежуточные риски программного проекта, в последнем слое определяется интегрального оценка риска.

Вероятность наступления промежуточного и интегрального рисков определяется по двум выражениям:

1) вероятность наступления x_i риска есть некоторая функция первичных факторов, равная произведению вероятностей независимых событий, в которых реализуется хотя бы один из факторов: $P(x_i) = 1 - \prod_{j=1}^{n} \overline{p(x_i)}$;

2) вероятность наступления x_i-го риска, которая представляет собой некоторую функцию, равную произведению вероятностей независимых событий при условии, что все факторы имеют место: $P(x_i) = \prod_{j=1}^{n} \overline{p(x_j)}$.

Негативные последствия — ***ожидаемые убытки проекта*** — можно описать в виде двух показателей:

1) дополнительные затраты ресурсов (времени, финансов) на снижение влияния на проект x_i-го рискообразующего фактора: $Z(x_i) = \sum_{i=1}^{n} p(x_j) z(x_j)$;

2) возможные потери команды проекта при наступлении рискообразующего x_i-го фактора: $W(x_i) = \sum_{i=1}^{n} p(x_j) w(x_j)$, где $x_i \in \Gamma^{-1} x_j, j = 1, ..., n.$

Очевидно, что задав в качестве исходных значений вероятности и убытки от наступления первичных факторов риска, можно вычислить аналогичные оценки и от наступления промежуточных факторов и интегрального риска в целом.

В зависимости от значения интегральной оценки рисков программный проект в соответствии с табл. 8.4 может быть отнесен к следующим категориям: высокорискованный, рискованный, среднерискованный, низкорискованный.

8.2.3. Планирование мероприятий по реагированию на риски и их мониторинг

Процессы планирования мероприятий по реагированию на риски предполагают выбор стратегии по снижению угроз для каждой из целей проекта и разработку планов мероприятий по реализации стратегий. Согласно [2] возможны четыре вида таких стратегий: уклонение от риска, передача риска, снижение риска, принятие риска.

Уклонение от риска предполагает разработку комплекса мероприятий по нейтрализации критических рискообразующих факторов, т. е. изменение плана управления проектом таким образом, чтобы исключить влияние негативных факторов на цели

проекта или скорректировать целевые показатели, находящиеся под угрозой, например, отказаться от реализации рискованного функционального требования.

Передача риска подразумевает переложение негативных последствий от проявления рискообразующего фактора на третью сторону (но риск при этом остается), например, заказать разработку рискованного компонента «на стороне». Данная стратегия эффективна для нейтрализации критических рискообразующих факторов, влияющих на бюджет проекта. Условия передачи ответственности третьей стороне должны определяться в контракте (гарантии выполнения контракта, гарантийные обязательства).

Снижение риска предполагает понижение вероятности и/или последствий негативного проявления рискообразующего фактора до приемлемых пределов, например, увеличить сроки выполнения проекта, понизить значения ряда показателей качества ПП. Принятие предупредительных мер по снижению вероятности наступления фактора или его последствий зачастую оказывается более эффективным, чем действия по устранению негативных последствий, предпринимаемые после наступления события.

Принятие риска предполагается в тех случаях, когда избежать проявления рискообразующих факторов маловероятно и команда проекта не нашла эффективных мероприятий реагирования на риски. Реализация данной стратегии возможна в двух вариантах: активной либо пассивной.

Пассивное принятие данной стратегии не предполагает проведения каких-либо предупредительных мероприятий, оставляя команде проекта право действовать по собственному усмотрению в случае наступления негативных событий. Наиболее распространенной формой активного принятия данной стратегии является создание резерва на непредвиденные обстоятельства в виде возможности привлечения дополнительных финансовых и/или трудовых ресурсов либо корректировки сроков реализации проекта.

Одним из возможных математических аппаратов принятия решений по выбору стратегии снижения угроз является аппарат таблиц решений.

При составлении планов мероприятий качественный и количественный анализ рискообразующих факторов проводится в соответствии с их рангами: риски, требующие немедленного реагирования; риски, нуждающиеся в проведении дополнительных анализов; риски, обладающие низким приоритетом, за которыми следует вести наблюдение. По каждому из мероприятий первой группы назначают одного или несколько ответственных лиц («ответственных за реагирование на риски»), определяются бюджет и сроки выполнения мероприятия. Прошедший экспертизу и утверждений план мероприятий должен быть включен в общий процесс управления изменениями программного проекта.

Ниже приводится описание мероприятий, направленных на снижение ряда рискообразующих факторов. Так, например, негативное влияние фактора ***<неполные или нечеткие требования к программному продукту>*** можно понизить за счет изменения стоимости и сроков реализации проекта при каждом добавлении или корректировке требований; согласования с заказчиком подробного перечня требований и включение этого списка в контракт на разработку ПП; использования моделей ЖЦ, позволяющих периодическое уточнение требований; введения буферных работ с соответствующими ресурсами и длительностью выполнения.

Риски, связанные с ***изменениями ситуации на финансовом рынке***, можно снизить путем внесения в контракт условий, предусматривающих корректировку стоимости проекта в случае кризисных явлений, не зависящих от воли сторон (изменение курса валют). Потерь, связанных с ***ненадежной работой аутсорсинговых компаний***, можно избежать путем внесения в контракт пункта о штрафных санкциях за нарушение условий поставки продукта.

Негативные последствия, связанные с ***недостаточными навыками владения исполнителями новыми инструментальными средствами разработки***, можно уменьшить за счет следующих

действий: привлечения экспертов-консультантов на начальных этапах проекта; учета при оценках трудоемкости дополнительного времени на обучение сотрудников; введения специальных тренингов по освоению новых средств разработки.

Для установления ***открытых и доверительных отношений с заказчиком*** необходимо осуществлять следующие мероприятия:

- постоянное взаимодействие по вопросам поиска взаимоприемлемых решений о выполнении проекта в несколько этапов с самостоятельным финансированием;
- согласование пользовательских интерфейсов и разработку прототипа продукта;
- периодические поставки текущих версий ПП конечным пользователям для их тестирования и оценки.

Мониторинг и управление рисками представляют собой процесс идентификации, анализа рисков и рискообразующих факторов и планирования мероприятий по реагированию на новые риски, отслеживания ранее идентифицированных рисков, а также проверки и исполнения мероприятий по реагированию на риски и оценки эффективности их выполнения. При этом необходимо постоянно решать следующие задачи: пересмотр рисков, аудит рисков, анализ отклонений и трендов.

Пересмотр рисков предполагает регулярную, согласно принятым регламентам, идентификацию, анализ и планирование реагирования на новые риски. Управление рисками проекта должно быть одним из пунктов повестки дня всех совещаний команды проекта.

Аудит рисков — это изучение и предоставление в документальном виде результатов оценки эффективности выполнения мероприятий по реагированию на риски, требующие немедленного реагирования, анализ основных причин их возникновения.

На основании ***анализа отклонений и трендов*** проекта можно прогнозировать на очередной плановый период влияние негативных последствий проявления рискообразующих факторов на цели проекта. Контроль и анализ трендов может повлечь за собой выбор альтернативных стратегий, принятие корректив, перепланировку проекта для достижения базового плана.

Контрольные вопросы

1. Дайте определение и приведите пример понятий «риск» и «рискообразующий фактор».

2. Приведите пример и прокомментируйте по схеме «условие» — «последствие» — «воздействие» описание внутренних факторов риска программного проекта.

3. Приведите пример по схеме «условие» — «последствие» — «воздействие» описания внешних факторов риска программного проекта.

4. Раскройте содержание и методы описания показателей вероятности и негативных последствий рискообразующих факторов.

5. Поясните процедуру ранжирования рискообразующих факторов по степени опасности последствий от их наступления.

6. Раскройте содержание модели функциональных зависимостей определения рисков программного проекта.

7. Раскройте содержание стратегий по управлению рисками, приведите примеры конкретных мероприятий по каждой из стратегий.

8. Раскройте содержание этапа мониторинга и управления рисками.

Литература

1. Архипенков С.Я. Лекции по управлению программными проектами [Электронный ресурс]: персональный сайт. – 2009. – Режим доступа: http://www.arkhipenkov.ru/resources/sw_project_management.pdf (дата обращения: 16.04.2014).

2. Руководство к своду знаний по управлению проектами (PMBOK). – 4-е изд. – М.: Project Management Institute, 2010. – 496 с.

3. Кознов Д.В. Введение в программную инженерию: курс лекций / Д.В. Кознов. – М.: Национальный Открытый университет «ИНТУИТ», 2009. – 283 с.

4. Новиков Ф.А. Управление проектами и разработкой ПО: учеб.-метод. пособие по дисциплине / Ф.А. Новиков, Э.А. Опалева, Е.О. Степанов. – СПб.: СПбГУ ИТМО, 2008. — 256 с. [Электронный ресурс]. – Режим доступа: http://books.ifmo.ru/file/pdf/430.pdf (дата обращения: 23.10.2014).

5. Фатрелл Роберт Т. Управление программными проектами. Достижение оптимального качества при минимуме затрат / Роберт Т. Фатрелл, Дональд Ф. Шафер, Линда И. Шафер. – М.: Вильямс, 2004. – 1136 с.

6. ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. – М.: Стандартинформ, 2011. – 106 с. [Электронный ресурс]: текст документа. – Режим доступа: http://protect.gost.ru/v.aspx?control=8&baseC=-1&page=0&month=-1&year=-1&search=&RegNum=1&DocOnPageCount=15&id=16909 (дата обращения: 11.02.2015).

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

8. Карпенко С.Н. Введение в программную инженерию: учеб.-метод. пособие / С.Н. Карпенко. – Н. Новгород: Изд-во ННГУ, 2007. – 103 с.

9. Евланов Л.Г. Теория и практика принятия решений / Л.Г. Евланов. – М.: Экономика, 1984. – 176 с.

10. Ньюэлл Майкл В. Управление проектами для профессионалов. Руководство по подготовке к сдаче сертификационного экзамена /Майкл В. Ньюэлл; пер. с англ. А.К. Казаков. – 3-е изд. – М.: КУДИЦ-ОБРАЗ, 2006. – 416 с.

11. Хэлдман Ким. Управление проектами. Быстрый старт / Ким Хелдман; пер. с англ. Ю. Шпаковой; под ред. С.И. Неизвестного. – М.: ДМК Пресс; Академия АйТи, 2008. – 352 с.

12. Эинк Э. Бизнес для программистов. Как начать свое дело / Э. Эинк. – СПб.: Питер, 2008. – 256 с.

13. Ехлаков Ю.П. Организация бизнеса на рынке программных продуктов: учебник / Ю.П. Ехлаков. – Томск: Изд-во Томск. гос. ун-та систем управления и радиоэлектроники, 2012. – 312 с.

14. Хьюберт К. Рамперсад. Универсальная система показателей деятельности / К. Рамперсад Хьюберт. – М.: Альпина Бизнес Букс, 2004. – 360 с.

15. Ехлаков Ю.П. Модели и алгоритмы управления жизненным циклом программного продукта: монография / Ю.П. Ехлаков, Д.Н. Бараксанов, Е.А. Янченко. – Томск: Изд-во Томск. гос. ун-та систем управления и радиоэлектроники, 2013. – 212 с.

16. Силич М.П. Системная технология: объектно-ориенти-рованный подход: моногр. / М.П. Силич. – Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 2002. – 224 с.

17. Математические основы управления проектами: учеб. пособие для вузов / С.А. Баркалов [и др.]; ред. В.Н. Бурков. — М.: Высшая школа, 2005. — 421 с.

18. Танаев В.С. Введение в теорию расписаний / В.С. Танаев, В.В. Шкурба. — М.: Наука, 1975. — 256 с.

19. Microsoft Solutions Framework. Модель процессов MSF. Версия 3.1; пер с англ. – 2003. – 41 с. [Электронный ресурс]: текст документа. – Режим доступа: https://www.google.ru/url?sa=t&rct=j&q=&esrc=s&source=web&cd= 10 &cad=rja&uact=8&ved=0ahUKEwin8PaZkdvJAhUEEiwKHcWRBQoQFghYMAk&url=http%3A%2F%2Fcs.karelia.ru%2F~kulakov%2Ftools%2FMSF_process_model_rus.doc&usg=AFQjCNHCx2FqDOOIqfgiB8twMtBpr1ohqQ (дата обращения: 15.12.2015).

20. Архипенков С.Я. Руководство командой разработчиков программного обеспечения. Прикладные мысли [Электронный ресурс]: персональный сайт. – М., 2008. – Режим доступа: http://www. arkhipenkov. ru/resources/sw_team_management.pdf (дата обращения: 26.05.2015)

21. Салливан Эд. Время – деньги. Создание команды разработчиков программного обеспечения: пер. с англ. / Эд. Салливан. – М.: Русская редакция, 2002. – 364 с.

22. Фалмер Роберт М. Энциклопедия современного управления / Роберт М. Фалмер. – М.: ВИПКэнерго, 1992. – Т. 2. – 142 с.

23. Гончаров В.В. Руководство для высшего управленческого персонала В.В. Гончаров. – М.: Изд-во МНИИПУ, 1996. – Т. 1. – 752 с.

24. Библиотека управления [Электронный ресурс]: Корпоративный менеджмент. – Режим доступа: http://www.cfin.ru (дата обращения: 24.03.015).

25. Основы управления проектами: [Электронный ресурс]. – Режим доступа: http://www.e-college.ru. – (дата обращения: 1.04.2015).

26. Липаев В.В. Экономика производства программных продуктов / В.В. Липаев. – 2-е изд. – М.: СИНТЕГ, 2011. – 352 с.

27. Романова М.В. Управление проектами: учеб. пособие для вузов / М.В. Романова. – М.: ФОРУМ; Инфра-М, 2007. – 253 с.

28. Авдошин С.М. Информатизация бизнеса. Управление рисками / С.М. Авдошин, Е.Ю. Песоцкая. – М.: ДМК Пресс, 2011. – 176 с.

29. Липаев В.В. Анализ и сокращение рисков проектов сложных программных средств / В.В. Липаев. – М.: СИНТЕГ, 2003. – 224 с.

30. Ехлаков Ю.П. Классификация и описание рискообразующих факторов при создании программных продуктов / Ю.П. Ехлаков // Доклады ТУСУРа. – 2013. – № 4(30). – С. 142–147.

31. Шапкин А.С. Экономические и финансовые риски / А.С. Шапкин, В.А. Шапкин. – М.: Дашков и К°, 2008. – 543 с.

32. Фатхутдинов Р.А. Инновационный менеджмент: учеб. для вузов / Р.А. Фатхутдинов. – 6-е изд. — СПб.: Питер, 2008. – 448 с.

33. Заде Л. Понятие лингвистической переменной и ее роль в принятии приближенных решений / Л. Заде. – М.: Мир, 1976. – 168 с.

Оглавление

Введение ……... 3

1. ОСОБЕННОСТИ ПРОЦЕССА УПРАВЛЕНИЯ ПРОГРАММНЫМ ПРОЕКТОМ

1.1. Основные понятия и определения ……... 7
1.1.1. Программный проект …….. 7
1.1.2. Программный продукт ……... 8
1.1.3. Управление программным проектом …… 10
1.2. Этапы жизненного цикла программного проекта …. 13

2. СТАНДАРТИЗАЦИЯ ПРОЦЕССОВ СОЗДАНИЯ ПРОГРАММНОГО ПРОДУКТА

2.1. ГОСТ Р ИСО/МЭК 12207-2010 «Процессы жизненного цикла программных средств» ……. 25
2.2. Комплекс стандартов «Единая система программной документации» …….. 41
2.3. Международный стандарт «Процессы и действия жизненного цикла программного обеспечения» …… 49

3. МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

3.1. Каскадная модель …….. 57
3.2. V-образная модель ……..... 60
3.3. Модель прототипирования ……... 62
3.4. Модель быстрой разработки приложений …….. 66
3.5. Инкрементная модель ……... 69
3.6. Спиральная модель ……... 73
3.7. Методика выбора модели жизненного цикла разработки программного продукта …… 77

4. ИНИЦИАЦИЯ ПРОГРАММНОГО ПРОЕКТА

4.1. Разработка идеи программного проекта и оценка ее привлекательности ……... 84
4.2. Разработка концепции проекта и оценка ее перспективности ……... 89

4.3. Выбор перспективной концепции программного проекта …….. 95
4.3.1 Оценка перспективности концепции методом экспертных оценок ……... 95
4.3.2. Модель функциональных зависимостей оценки перспективности концепции проекта …… 98

5. УПРАВЛЕНИЕ СОДЕРЖАНИЕМ И СРОКАМИ ПРОГРАММНОГО ПРОЕКТА
5.1. Основные этапы управления программным проектом 112
5.2. Структурная декомпозиция работ …… 115
5.3. Управление сроками реализации проекта 122
5.3.1. Формальное представление проекта в виде сетевой модели …… 122
5.3.2. Модель и алгоритмы формирования календарного плана проекта …….. 128

6. УПРАВЛЕНИЕ ЧЕЛОВЕЧЕСКИМИ РЕСУРСАМИ
6.1. Организация командной работы над проектом …….. 137
6.1.1. Основные участники и ролевые группы команды проекта ……. 137
6.1.2. Организационные структуры управления проектом …….. 141
6.1.3. Основные модели управления командой проекта …… 151
6.2. Специфические особенности командной работы над программным проектом …… 156
6.2.1. Роль руководителя в команде проекта …… 156
6.2.2. Мотивация программиста как участника проекта 159
6.2.3. Прием, аттестация и увольнение программиста 166

7. УПРАВЛЕНИЕ СТОИМОСТЬЮ ПРОГРАММНОГО ПРОЕКТА
7.1. Оценка плановой стоимости проекта …….. 173
7.2. Формирование бюджета программного проекта ….. 180
7.3. Мониторинг исполнения бюджета проекта …… 182

8. УПРАВЛЕНИЕ РИСКАМИ ПРОГРАММНОГО ПРОЕКТА

8.1. Основные понятия риска и рискообразующих факторов …….. 188

8.2. Содержание этапов управления рисками ……... 195

8.2.1. Идентификация и анализ рисков и рискообразующих факторов …… 195

8.2.2. Определение интегральной оценки риска программного проекта ……. 203

8.2.3. Планирование мероприятий по реагированию на риски и их мониторинг ……..... 205

Литература …… 210

Учебное издание

Ехлаков Юрий Поликарпович

УПРАВЛЕНИЕ ПРОГРАММНЫМИ ПРОЕКТАМИ

Учебник

Редактор Коновалова Н.В.

Подписано в печать 22.12.15. Формат 60x84/16.
Усл. печ. л. 12,56. Тираж 100 экз. Заказ 955.

Томский государственный университет систем управления и радиоэлектроники
634050, г. Томск, пр. Ленина, 40. Тел. (3822) 53-30-18.