Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Ю. П. Ехлаков

УПРАВЛЕНИЕ ПРОГРАММНЫМИ ПРОЕКТАМИ

Учебное пособие

Томск
«Эль Контент»
2014

УДК 004.413(075.8)
ББК 32.973.26-018.2я73
Е 934

Рецензенты:
Силич В. А., докт. техн. наук, профессор кафедры оптимизации систем управления Национального исследовательского Томского политехнического университета;
Мещеряков Р. В., докт. техн. наук, профессор кафедры комплексной информационной безопасности электронно-вычислительных систем ТУСУРа.

Ехлаков Ю. П.
Е 934 Управление программными проектами : учебное пособие / Ю. П. Ехлаков. — Томск : Эль Контент, 2014. — 140 с.

ISBN 978-5-4332-0163-7

В учебном пособии раскрывается содержание управления программными проектами как специфического вида деятельности по разработке и продвижению на рынок программных продуктов. Предназначено для изучения дисциплины «Управление программными проектами» студентами различных направлений подготовки.

УДК 004.413(075.8)
ББК 32.973.26-018.2я73

ISBN 978-5-4332-0163-7

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время доля IT-услуг составляет 20% в общем обороте IT-отрасли экономики России, а темп ежегодного прироста оценивается экспертами в 19%. Около 26% в общем объеме IT-услуг составляют услуги компаний малого и среднего бизнеса по разработке прикладных программных продуктов (ПП). В то же время только 35% проектов завершились в срок, не превысили запланированный бюджет и реализовали все требуемые функции и возможности; 46% проектов завершились с опозданием, расходы превысили запланированный бюджет, требуемые функции не были реализованы в полном объеме; 19% проектов были полностью неуспешны и не доведены до завершения [1].

В связи с этим для коллектива разработчиков, планирующего выйти на рынок, вопросы управления программными проектами на каждой из фаз жизненного цикла программного продукта (инициации и разработки требований, проектирования и конструирования, вывода на рынок) являются ключевыми.

В пособии приводится понятие программного продукта, как результата деятельности команды разработчиков и программного проекта, как методологии управления процессами создания программного продукта. Отмечается, что регламентация этих видов деятельности описывается в соответствующих международных и отечественных стандартах на жизненные циклы (ЖЦ) программного продукта и программного проекта. Приводится сжатое описание наиболее актуальных стандартов: PMBOK (Project Management Body of Knowledge) — Свод знаний по управлению проектами, Руководство к Своду знаний по программной инженерии (Guide to the Software Engineering Body of Knowledge — SWEBOK), ГОСТ Р ИСО/МЭК 12207–99 «Информационная технология. Процессы жизненного цикла программных средств», «Единая система программной документации (ЕСПД): ГОСТ 19.102–77 ЕСПД «Стадии разработки».

Конкретизация базовых стандартов к особенностям программного продукта проводится при выборе модели ЖЦ ПП. В пособии описываются содержания, достоинства, недостатки и области применения следующих моделей: каскадной, V-образной, прототипирования, быстрой разработки приложений, многопроходной, спиральной.

Проблема командообразования при реализации программных проектов рассматривается, 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.

Наиболее подробно в пособии раскрываются два этапа управления проектами: инициация и планирование. На фазе инициации создается творческое ядро команды по разработке будущего программного продукта, формируются идея и концепция проекта. Задача выбора и оценки перспективности идеи будущего программного продукта является ключевой. Это связано с тем, что ошибки, допущенные на этой стадии, существенно влияют на успех проекта в целом. Описываются две модели оценки перспективности концепции, предлагается совокупность критериев, с помощью которых можно оценить наиболее важную составляющую концепции проекта — ориентацию на потребителя и рыночные условия.

Особенности программного продукта как результата интеллектуальной деятельности предъявляют определенные требования к фазам планирования программных проектов. В пособии последовательно раскрываются вопросы:

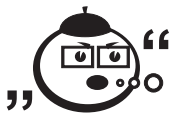
- структурной декомпозиции проекта и определения множества работ, которые необходимо выполнить для получения результатов проекта;
- выявление и документирование зависимостей между работами проекта;
- оценка трудоемкости, определение типов, количества исполнителей (трудовых ресурсов), привлекаемых для выполнения работ, и длительности выполнения работ; разработка календарного плана проекта, плана распределения ресурсов.

Вероятностный характер процессов планирования программных проектов увеличивает (усиливает) важность вопросов идентификации и управления рисками. Задачи по управлению рисками состоят в выделении множества рискообразующих факторов, влияющих на результаты проекта; проведении качественного и количественного анализа и оценки влияния факторов; определении вероятности появления факторов и уровня негативных последствий от их проявления (ущерба, убытков, потерь) на результаты проекта; принятии одного из возможных вариантов решения по реагированию на риски (уклонение, передача, снижение).

Соглашения, принятые в книге

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

Эта пиктограмма означает определение или новое понятие.



Эта пиктограмма означает цитату.

Контрольные вопросы по главе

Глава 1

ОСНОВНЫЕ ПОНЯТИЯ ПРОГРАММНОГО ПРОДУКТА И ПРОГРАММНОГО ПРОЕКТА

1.1 Основные понятия программного продукта

В общем случае под *товаром* на рынке понимается, как правило, любой продукт производственно-экономической деятельности в материально-вещественной форме, являющийся объектом купли-продажи и соответственно возникающих между продавцами и покупателями рыночных отношений [1]. С учетом данного определения на рынке прикладных программных продуктов в качестве *товара* следует рассматривать *программный продукт (ПП) и программное изделие*, имеющие следующие специфические формы проявления [2]: программный продукт, программное изделие, коробочный программный продукт.

Программный продукт *— совокупность программных компонентов, предназначенных для поставки, передачи или продажи пользователю, снабженных технической документацией, рекламными материалами, инструкциями по обучению, гарантийными обязательствами по сопровождению и гарантийному обслуживанию.*

Программное изделие *— программа или логически связанная совокупность программ, записанная на носителях данных, являющаяся продуктом промышленного производства, снабженная программной документацией и предназначенная для широкого распространения посредством продажи.*

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

Из представленных определений видно, что продается (поставляется) не только ПП и документация, в которой можно прочитать, как установить программу и как ей пользоваться, данные для установки программы в различных условиях (конфигурационные файлы), но и могут оказываться услуги по внедрению эксплуатации и сопровождения ПП.

ПП как рыночный товар имеет ряд специфических особенностей [3]:

- ПП не материален, его нельзя *увидеть в процессе конструирования* и, следовательно, оперативно повлиять на его реализацию;
- жизненный цикл ПП в существующих стандартах описан в *общем виде* и прямо не ориентирован на специфику конкретного продукта, *необходимо адаптировать* стандарты к конкретным условиям;
- ПП как результат творческого труда *не поддаётся точному оцениванию* как по времени создания, так и по требуемому бюджету и поэтому создаётся в условиях повышенного риска;
- потенциальные потребители не могут *четко сформулировать требования* к ПП и не имеют четкого представления о технологии его использования в практической деятельности;
- ПП является предметом интеллектуального труда и *представляет собой публикацию текста программы/программ* на языке программирования или в виде исполняемого кода;
- ПП вступает в хозяйственный оборот как товар только в случае фиксирования его *на материальном носителе* (компьютере, дисковом накопителе и т. п.), в котором он овеществляется и может быть *сохранен, преобразован или передан*, при этом обладание материальным носителем информации не делает его приобретателя *уникальным собственником* информации;
- вовлечение ПП в хозяйственный оборот происходит в *процессе коммерциализации* (купли-продажи, переуступки прав собственности) и *капитализации* (постановки на баланс, инвестирования в уставной капитал).

Таким образом, ПП как товар по своей сути является объектом интеллектуальной собственности, для которого характерна нематериальная природа существования, и обладает следующими свойствами:

- он может обмениваться, но не отчуждаться полностью;

- он может быть неоднократно продан, являясь при этом одновременно объектом нескольких рыночных сделок;
- не исчезает и не изнашивается в процессе использования;
- состоит из материального носителя и нематериальной части;
- производится в условиях повышенного риска;
- характеризуется ничтожными затратами на тиражирование по сравнению с затратами на разработку продукта;
- большую часть стоимости составляют затраты по созданию данного ПП относительно небольшой группой специалистов;
- характеризуется невысокими затратами на тиражирование, обусловленными ничтожно малой стоимостью производственных операций по изготовлению копий ПП по сравнению со стоимостью самого продукта.

Технология процесса производства любого материального продукта представляется в виде *жизненного цикла (ЖЦ)* как последовательности этапов (фаз, стадий), состоящих из технологических процессов, действий и операций, которые надо выполнить на протяжении всей «жизни» продукта. *Жизненный цикл* ПП как товара определен в стандарте *ГОСТ Р ИСО/МЭК 12207–99 «Информационная технология. Процессы жизненного цикла программных средств»*. В стандарте ЖЦ ПП определен в виде структуры, состоящей из процессов, работ, задач и описывающей этапы создания и развития ПП от установления требований, разработки, эксплуатации, сопровождения программного продукта и до полного прекращения его использования.

Очевидно, что могут существовать различные варианты технологий (как последовательности и способа производства продукта) реализации ЖЦ материального продукта, каждый из которых описывается в виде конкретной *модели ЖЦ*.

Модель является хорошей абстракцией различных методов разработки ПП, позволяя лаконично, сжато и информативно их представить. В моделях ЖЦ, необходимо различать фазы и виды деятельности [4].

Фаза *— это определенный этап процесса ЖЦ, имеющий начало, конец и выходной результат.*

Фазы следуют друг за другом в линейном порядке, характеризуются предоставлением отчетности заказчику и, часто, выплатой денег за выполненную часть работы. Редко какой заказчик согласится первый раз увидеть результаты только после завершения проекта. С другой стороны, разработчики предпочитают получать деньги постепенно, по мере того, как выполняются отдельные части работы. Таким образом, появляются фазы, позволяющие создавать и предъявлять промежуточные результаты проекта. Примерами фаз может служить согласование с заказчиком технического задания, реализация определенной функциональности ПП, этап разработки, оканчивающийся сдачей системы на тестирование или выпуском альфа-версии.

Вид деятельности *— это определенный тип работы, выполняемый в процессе ЖЦ разработки ПП.*

Разные виды деятельности часто требуют разных профессиональных навыков и выполняются разными специалистами. Например, управление проектом выполняется менеджером проекта, кодирование — программистом, тестирование — тестировщиком. В рамках одной фазы может выполняться много различных видов деятельности. Кроме того, один вид деятельности может выполняться на разных фазах — например, тестирование: на фазе анализа и проектирования можно писать тесты и налаживать тестовое окружение, при разработке и перед сдачей производить, собственно, само тестирование.

Наиболее известными *формальными моделями* ЖЦ ПП являются: каскадная модель или водопад, V-образная модель, модель прототипирования, модель быстрой разработки приложений или RAD-модель, многопроходная модель, спиральная модель.

1.2 Основные понятия программного проекта

Управление процессами разработки ПП с использованием одной из формальных моделей ЖЦ происходит в рамках проектной деятельности IT-компании.

В литературе [5–7] *приводятся следующие определения* ***проекта****:*

1) *комплекс взаимосвязанных мероприятий, предназначенных для достижения целевых результатов при решении многокритериальных задач в течение заданного периода времени при установленном бюджете в условиях ограниченных ресурсов;*
2) *ограниченное по времени целенаправленное изменение исследуемой системы с установленными требованиями к качеству результатов, возможными объемами расхода средств, ресурсов и специфической организацией управления. Управление проектом — это управление этими изменениями с высокой степенью уверенности в успешном исходе;*
3) *временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов;*
4) *комплекс усилий, предпринимаемых с целью получения конкретных уникальных результатов в рамках отведенного времени и в пределах утвержденного бюджета, который выделяется на оплату ресурсов, используемых или потребляемых в ходе проекта;*

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

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

. . . . . .

. . . . . .

С этой точки зрения, обобщая представленные в литературе понятия, приведем следующие варианты определения ***программного проекта****:*

1) *комплекс* ***взаимосвязанных мероприятий****, предназначенных для достижения* ***целевых результатов*** *по созданию* ***нового ПП*** *в течение заданного периода* ***времени*** *при установленных* ***бюджете и ресурсах*** *в условиях* ***повышенного риска****;*

2) *комплекс* ***взаимосвязанных работ****, выполняемых командой с целью получения* ***уникального ПП*** *в рамках отведенного* ***времени*** *и в пределах утвержденного* ***бюджета****, который выделяется на* ***оплату ресурсов****, используемых или потребляемых в ходе реализации проекта;*

3) ***уникальная деятельность****, направленная на достижение определённого результата/цели по созданию* ***новых ПП*** *или* ***услуги****, имеющая* ***начало и конец во времени****, при заданных ограничениях по* ***ресурсами и срокам****, а также требованиям* ***к качеству*** *и допустимому уровню* ***риска****.*

. . . . . .

Цели программного проекта следует определять в виде желаемого результата достигаемого командой проекта при его успешной реализации. Формулировки целей должны быть: конкретными, измеримыми, согласованными, реальными, ограниченными по срокам.

Конкретность цели должна обеспечивать четкость и однозначность понимания результатов; *измеримость* позволяет впоследствии оценить степень достижения результата и может выражаться как в качественной, так и в количественной формах; *реальность* рассматривается с точки зрения собственных возможностей команды проекта и влияния внешней среды; *согласованность* обеспечивает единство мнений всех участников проекта в понимании конечного результата и согласованность действий между ними для достижения целей; *ограниченность по срокам* определяет интервал времени, когда необходимо оценить степень достижения конечных результатов.

Четкое определение бизнес-целей достаточно важно, поскольку существенно влияет на все процессы и решения в проекте. Проект должен быть закрыт, если признается, что достижение цели невозможно или стало нецелесообразным (например, если реальные затраты на проект будут превосходить будущие доходы от его реализации). При описании желаемого результата программного проекта необходимо определять: какие именно бизнес-выгоды получит заказчик по завершении проекта; какой функционал продукта или услуги будут получены по окончании проекта; краткое описание и при необходимости ключевые свойства и/или характеристики функционала.

Конкретные формулировки *желаемого результата программного проекта* можно приводить с учетом правила «железного треугольника» (рис. 1.1) [5]. В проекте должен быть реализован заданный функционал, необходимо выполнить его в нормативный срок и не превышать плановый бюджет.

Рис. 1.1 – «Железный треугольник» ограничений проекта

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

С учетом наличия рыночной конкуренции к трем основным характеристикам «железного треугольника» следует добавить четвертую — *приемлемое качество*, параметры которого задаются в виде совокупности нефункциональных требований к ПП (рис. 1.2).

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.

Ограничения и *допущения* являются неотъемлемой частью проекта и сокращают возможности проектной команды в выборе решений по его реализации. В частности, *ограничения* могут содержать: требования обязательной сертификация продукта, услуги на соответствие определенным стандартам; требования на использование конкретной заданной программно-аппаратной платформы; специфические требования к защите информации. *Допущения* — 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, приобретаемое на стороне, не изменится до завершения проекта. О допущениях проекта заказчик должен знать заранее, все они должны всегда оформляться документально.

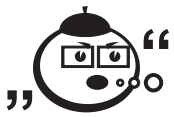
Рис. 1.2 – Возможные варианты системы ограничений проекта по имеющимся характеристикам: *а* — отсутствует ограничение на качество; *б* — отсутствует ограничение на бюджет; *в* — отсутствует ограничение на время реализации

Оптимальный вариант реализации программного проекта необходимо определять на *компромиссе* между сформулированными выше характеристиками результатов проекта. В общем случае поиск компромисса состоит в нахождении баланса, приемлемого для всех сторон, связанных с проектом: заказчиков, которым нужна определенная функциональность в конкретные сроки при имеющемся бюджете; исполнителей, которые обладают бюджетом, достаточным для 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 не функциональных требований в пределах запланированного бюджета.

С точки зрения актуальности и проблемности задачи управления программными проектами следует выделить следующие определяющие характеристики:

- *направленность* проекта на достижение конкретного конечного результата, определяемого в терминах затрат, качества и времени реализации;
- *уникальность* проекта как разового мероприятия, которое не будет повторяться;
- *ограниченность проекта по времени и ресурсам* (финансовым, трудовым, материальным) и как следствие, необходимость нахождения постоянного компромисса между объемом работ, ресурсами, временем, качеством и рисками, их перераспределения в ходе выполнения проекта;
- *структурная сложность* проекта как комплекса тесно взаимосвязанных мероприятий и его *высокая неопределенность, связанная с возможными* изменениями условий реализации, потребности в тех или иных видах ресурсов, невозможностью достижения цели в указанные сроки и т. д.

Иными словами, проект — это достаточно сложный вид деятельности, которым следует управлять в силу его уникальности и ограниченности ресурсов и времени. Среди различных определений *управления проектами* наиболее полным следует считать определение, сформулированное в международном стандарте PMBOK (Project Management Body of Knowledge — Свод знаний по управлению проектами) [8].



«Управление проектами — это наука и искусство руководства и координации людских и материальных ресурсов на протяжении жизненного цикла проекта путем применения современных методов и техники управления для достижения определенных в проекте результатов по составу и объему работ, стоимости, времени, качеству и удовлетворению участников проекта».

С учетом вышеизложенного, ***управление программным проектом*** *— процессы выполнения требований к программному продукту путем планирования, реализации, мониторинга и контроля проектных работ, в ходе которых достигаются цели проекта при нахождении компромисса между объемом работ, ресурсами, временем, качеством и рисками.*

В стандарте PMBOK приведены 9 процессов (областей знаний) по управлению проектами и пять этапов (фаз) жизненного цикла проекта: *инициация, планирование, исполнение, мониторинг и управление, завершение* [5].

Состав и взаимосвязи каждого из пяти этапов жизненного цикла проекта представлен на рис. 1.3.

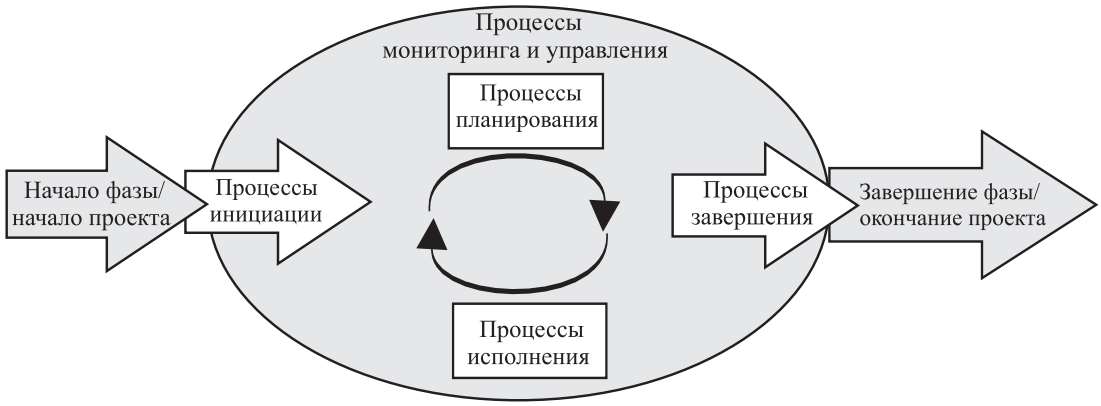


Рис. 1.3 – Группы процессов управления проектами

В рамках *процессов инициации* определяются бизнес-потребности, ради удовлетворения которых разрабатывается проект, включая причины, по которым данный проект является лучшей альтернативой для удовлетворения потребностей, внутренние и внешние заинтересованные стороны в результатах проекта, цели и содержание проекта, формируется команда исполнителей, фиксируются финансовые ресурсы, разрабатываются 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 действий (работ проекта), необходимых и достаточных

для достижения целей проекта. В процессах планирования определяются состав и содержание работ проекта и зависимости между ними; оценивается трудоемкость каждой работы, типы и количество необходимых трудовых ресурсов, разрабатываются календарный план проекта, планы распределения ресурсов, бюджетный план, планы управления рисками.

В составе группы *процессов исполнения* рассматриваются вопросы организации работ по проекту: распределение обязанностей и ответственности; координации работы исполнителей и использования ресурсов; определения механизмов морального и материального стимулирования работы 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 использования результатов проекта; проведение анализа эффективности внедрения; внесение необходимых изменений в историю успеха компании-разработчика.

Каждая из девяти областей знаний делится на соответствующие подразделы.

1. *Интеграция управления проектом*: создание плана проекта, исполнение плана проекта, контроль изменений в проекте.
2. *Управление объемами работ*: 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, управление изменениями объема работ.
3. *Управление сроками проекта*: определение взаимосвязей работ, 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 отчетности, подбор и получение необходимых для проекта трудовых ресурсов, распределение персонала по работам проекта, постоянное повышение квалификации, индивидуальной и командной производительности труда.
7. *Управление коммуникациями*: определение потребностей участников проекта в информации и планировании информационных потоков, регулярное и своевременное обеспечение участников проекта необходимой информа-

цией, сбор и распространение отчетности о текущем состоянии проекта, достигнутом прогрессе и ожидаемых результатах, создание и утверждение отчетов, необходимых для формального завершения проекта/фазы.

8. *Управление рисками проекта*: планирование управления рисками, идентификация рисков, качественный анализ рисков, количественный анализ рисков, планирование реагирования на риски, мониторинг и контроль процессов управления рисками.
9. *Управление поставками*: определение продуктов и услуг, которые необходимо привлечь извне для выполнения проекта, документирование требований к продуктам и услугам от внешних поставщиков, получение предложений, выбор поставщиков, регулирование отношений с поставщиками, подтверждение о выполнении условий контрактов, разрешение споров.

Распределение работ по этапам жизненного цикла, рекомендованное стандартом, приведено в табл. 1.1.

Таблица 1.1 – Распределение процессов управления проектом по этапам

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

продолжение на следующей странице

Таблица 1.1 — Продолжение

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

Контрольные вопросы по главе 1

1. Дайте понятие товара, перечислите и прокомментируйте определения ПП как рыночного товара.
2. Раскройте особенности и свойства программного продукта как товара на рынке.
3. Перечислите свойства программного продукта как объекта интеллектуальной собственности.
4. Приведите возможные определения программного проекта, его цели, результаты, ограничения.
5. Раскройте смысл правила «железного треугольника» при управлении программными проектами.
6. Приведите понятие жизненного цикла ПП и модели жизненного цикла ПП.
7. Перечислите и прокомментируйте требования к формулировке целей программного проекта.
8. Перечислите и прокомментируйте содержание процессов стандарта PMBOK.
9. Перечислите и прокомментируйте содержание этапов управления проектами стандарта PMBOK.

Глава 2

СТАНДАРТИЗАЦИЯ ОСНОВНЫХ ПРОЦЕССОВ ЖИЗНЕННОГО ЦИКЛА СОЗДАНИЯ ПРОГРАММНЫХ ПРОДУКТОВ

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

2.1.1 Этап «Определение требований»

Руководство к Своду Знаний по Программной Инженерии содержит описание десяти областей знаний, которые условно можно разбить на две группы:

1. Знания, конкретизирующие жизненный цикл разработки программных продуктов:

- *определение требований;*
- *проектирование, конструирование;*
- *тестирование, эксплуатация (поддержка)*.

2. Знания, раскрывающие содержание методов и инструментария разработки программных продуктов:

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

Следует отметить, что данное руководство содержит описание работ, которые необходимо выполнять при создании ПП, и не касается вопросов применения тех или иных языков программирования, архитектурных решений или каких-либо технологий проектирования и разработки ПП.

Остановимся в дальнейшем более подробно только на описании содержания областей знаний, конкретизирующих основные *этапы жизненного цикла разработки программных продуктов* (рис. 2.1) [9].

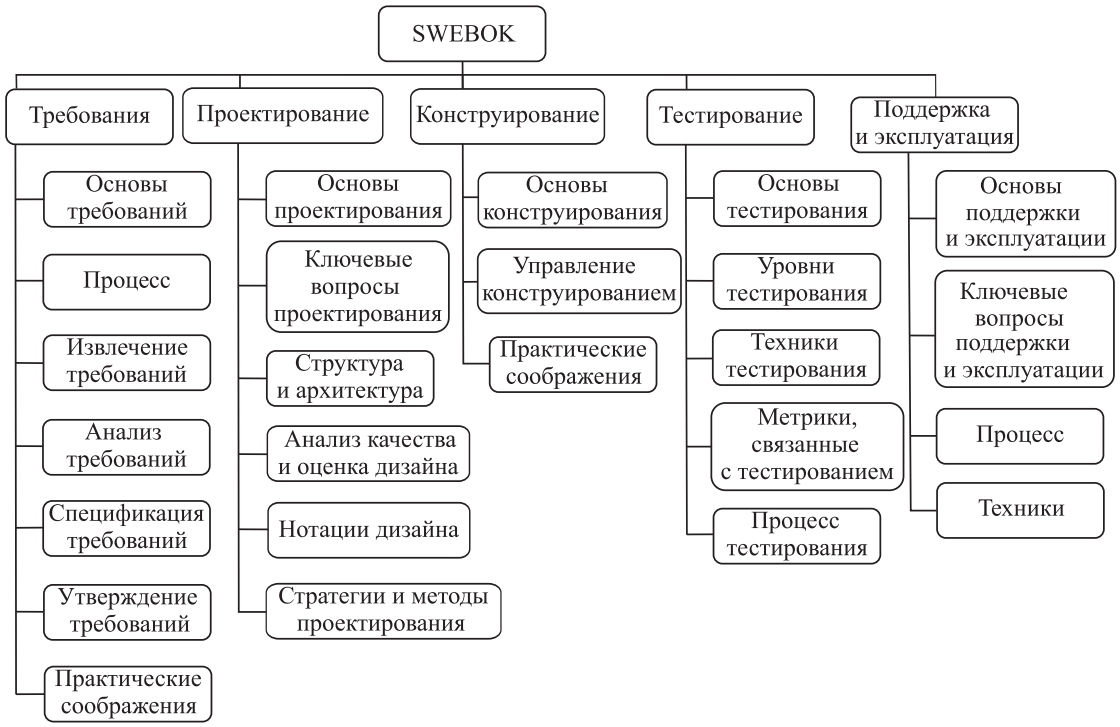


Рис. 2.1 – Содержание этапов жизненного цикла программных продуктов

Определение требований. Под требованиями в данном документе понимаются потребности людей (заказчиков, пользователей, разработчиков), заинтересованных в создании ПП, которые должны быть надлежащим образом представлены и оформлены в виде технического задания или иного документа.

Множество работ по определению и описанию требований условно разбито на группы (рис. 2.2).

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

Функциональные требования описывают перечень бизнес-процессов (функциональных задач) программного продукта, методы преобразования входных данных в результаты.

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

Системные требования описывают требования к программному продукту, состоящему из взаимосвязанных программных и аппаратных подсистем и разных приложений.

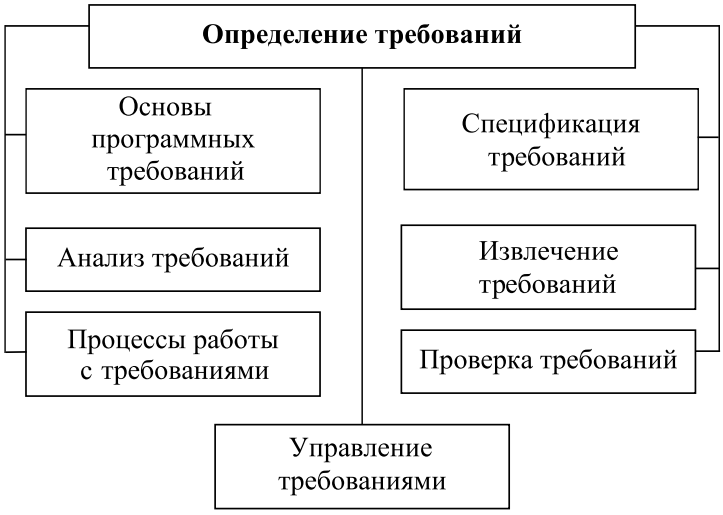


Рис. 2.2 – Область знаний этапа «Определение требований»

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

Участниками процесса работы с требованиями являются:

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

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

Один из ключевых принципов программной инженерии заключается в обеспечении взаимодействия между пользователями и инженерами. Прежде всего, специалисты «по требованиям» — аналитики определяют способы коммуникаций и вза-

имопонимания между заказчиками и исполнителями, которые необходимы для решения задач проекта.

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

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

- процессы изучения потребностей и целей пользователей;
- классификацию требований и их преобразование к требованиям программного обеспечения;
- общесистемное аппаратно-программное обеспечение;
- установление и разрешение конфликтов между требованиями;
- определение их приоритетов;
- принципы взаимодействия со средой функционирования.

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

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

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

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

цессам программной инженерии. Управление изменениями не может быть хаотическим, поэтому отношение к управлению требованиями — как к постоянно действующему бизнес-процессу. Восприятие изменений и возможность их своевременной обработки — вопрос способности проектной команды работать в постоянно меняющихся условиях. Так или иначе, понимание меняющейся природы требований — один из факторов адекватного реагирования на сами изменения, а следовательно, и возможности успешного завершения проекта.

2.1.2 Этап «Проектирование ПП»

Процесс «Проектирование ПП» предназначен для описания на основе анализа требований, архитектуры (программного дизайна) ПП в виде иерархической совокупности компонентов и интерфейсов между ними (рис. 2.3).

Рис. 2.3 – Область знаний этапа «Проектирование ПП»

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

- Архитектурный дизайн — описание многоуровневой структуры компонентов системы.
- Детализированная архитектура — описание поведения характеристик отдельных компонент.

К ***ключевым вопросам проектирования ПП*** относятся декомпозиция архитектуры на функциональные компоненты для независимого и параллельного их выполнения, принципы и механизмы их взаимодействия между собой, обеспечения качества и живучести ПП в целом. Такая декомпозиция на модули сложного ПП производится с целью получения более мелких и относительно независимых программных компонентов, каждый из которых несет различную функциональность.

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

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

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

Анализ качества и оценка программного дизайна. Данный раздел включает мероприятия по анализу сформулированных в требованиях атрибутов оценки качества различных аспектов ПП. Все множество атрибутов предлагается условно разбить на три группы:

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

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

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

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

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

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

взаимодействие и подчеркнута структурная организация объектов, посылающих и принимающих сообщения; Sequence — диаграмма поведения, на которой показано взаимодействие и подчеркнута временная последовательность событий, и др.

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

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

В объектно-ориентированном проектировании методы базируется на концепции объектов, ключевую роль играют наследование, полиморфизм и инкапсуляция, а также абстрактные структуры данных и отображение объектов.

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

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

2.1.3 Этап «Конструирование ПП»

Конструирование программного обеспечения в описания области знаний SWEBOK заключается в создании исходного кода программного продукта посредством комбинации кодирования, верификации (проверки) и отладки программных модулей (рис. 2.4).

Рис. 2.4 – Область знаний этапа «Конструирование ПП»

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

Основы конструирования программного обеспечения определяются следующими положениями:

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

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

Большинство ПП изменяются с течением времени, так как программный продукт не является изолированным от внешнего окружения. Более того, ПП является частью изменяющейся среды и должен меняться вместе с ней, а иногда, и быть источником изменений самой среды.

«Конструирование для проверки» предполагает, что построение ПП должно вестись таким образом, чтобы сам ПП помогал вести поиск причин сбоев, будучи прозрачным для применения различных методов проверки как на стадии независимого тестирования (например, инженерами-тестировщиками), так и в процессе эксплуатации, когда особенно важна возможность быстрого обнаружения и исправления возникающих ошибок.

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

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

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

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

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

К ***методам и инструментам конструирования ПП*** отнесены языки программирования и конструирования, а также программные методы и инструментальные системы (компиляторы, СУБД, генераторы отчетов, системы управления версиями, конфигурацией, тестированием и др.). К формальным средствам описания процесса конструирования ПП, взаимосвязей между человеком и компьютером и с учетом среды окружения отнесены языки конструирования. Основу данного раздела составляет задача понижения сложности конструирования программного продукта, которая обеспечивается применением наиболее подходящих *стилей конструирования*.

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

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

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

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

2.1.4 Этап «Тестирование ПП»

Тестирование ПП — это процесс проверки работоспособности, основанный на использовании конечного набора тестов, сформированных на базе требований к ПП, и сравнении полученных результатов с целевыми показателями качества, заложенными в проекте (рис. 2.5).

В разделе «*Основы тестирования*» приводятся терминология, методы и инструменты подготовки и проведения процесса тестирования, а также показатели оценки данных статистического анализа процесса тестирования. В соответствии с принятой терминологией при тестировании выявляются следующие недостатки: отказы и дефекты, сбои, ошибки. Важно чётко разделять *причину* нарушения работы программ, обычно описываемую терминами *недостаток* или *дефект*, и на-

блюдаемый нежелательный эффект, вызываемый этими причинами, — *сбой*. Термин «*ошибка*», в зависимости от контекста, может описывать и как причину сбоя, и сам сбой. Тестирование позволяет обнаружить дефекты, приводящие к сбоям.

Рис. 2.5 – Область знаний этапа «Тестирование ПП»

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

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

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

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

- *функциональное тестирование*, которое заключается в проверке соответствия выполнения функциональных возможностей ПП;
- *регрессионное тестирование* — тестирование ПП или его компонентов после внесения в них изменений;
- *тестирование эффективности функционирования ПП* — проверка в соответствии со спецификациями требований производительности, пропуск-

ной способности, максимального объема данных, системных ограничений и т. д.;

- *нагрузочное (стресс) тестирование* — проверка поведения ПП при максимально допустимой нагрузке;
- *внутреннее и внешнее тестирование* — альфа (без плана тестирования) и бета (с планом тестирования) тестирование соответственно;
- *тестирование конфигурации* — проверка структуры и идентификации системы на различных наборах данных, а также проверка работы системы в различных конфигурациях;
- *приемочное тестирование* — проверка в соответствии с заранее подготовленной программой и методикой испытаний поведения системы на этапе приемки-сдачи ее заказчику.

Техники тестирования условно разбиты в документе на следующие группы:

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

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

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

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

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

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

Измерение результатов тестирования касается оценки качества получаемого продукта. *Оценка программ в процессе тестирования* базируется на размере программ (например, в терминах количества строк кода или функциональных точек) или их структуре (например, с точки зрения оценки ее сложности в тех или иных

архитектурных терминах). Структурные измерения могут также включать частоту обращений одних модулей программы к другим.

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

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

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

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

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

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

В состав этого процесса должны входить:

1) планирование работ по тестированию (составление планов, тестов, наборов данных) и измерению показателей качества ПП;
2) генерация необходимых тестовых сценариев, соответствующих среде выполнения ПП;
3) проведение тестирования с учетом следующих положений:
   - все работы и результаты процесса тестирования должны обязательно фиксироваться;
   - форма журналирования работ и их результатов должна быть такой, чтобы соответствующее содержание было понятно, однозначно интерпретируемо и повторяемо другими лицами;

- тестирование должно проводиться в соответствии с заданными и документированными процедурами;
- тестирование должно производиться над однозначно идентифицируемой версией и конфигурацией ПП;
- должен осуществляться сбор данных об отказах, ошибках и других непредвиденных ситуациях при выполнении программного продукта;
- должны составляться отчёты по результатам тестирования и оценки характеристик ПП.

2.1.5 Этап «Сопровождение ПП»

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

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

Процесс сопровождения выполняется как перед вводом ПП в эксплуатацию, так и после этого и состоит из планирования деятельности по сопровождению ПП, организации перехода к его полнофункциональному использованию, обучения пользователей и их ежедневной поддержки при работе с текущей версией продукта. Если новый программный продукт должен заменить старый программный продукт, важно обеспечить плавный переход со старого ПП на новый максимально естественно для пользователей.

Область знаний «Сопровождение ПП» состоит из следующих описаний разделов (рис. 2.6).

Рис. 2.6 – Область знаний этапа «Сопровождение ПП»

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

Сопровождение ПП определяется стандартами как:

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

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

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

- Корректирующее сопровождение: «реактивная» модификация программного продукта, выполняемая уже после передачи в эксплуатацию, для устранения сбоев.
- Адаптирующее сопровождение: модификация программного продукта на этапе эксплуатации для обеспечения продолжения его использования с заданной эффективностью (с точки зрения удовлетворения потребностей пользователей) в изменившемся или находящемся в процессе изменения окружении, в первую очередь, подразумевается изменение бизнес-окружения, порождающее новые требования к ПП.
- Совершенствующее сопровождение: модификация программного продукта на этапе эксплуатации для повышения характеристик производительности и удобства сопровождения.
- Профилактическое сопровождение: модификация программного продукта на этапе эксплуатации для идентификации и предотвращения скрытых дефектов до того, когда они приведут к реальным сбоям.

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

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

К группе *технических вопросов* по сопровождению ПП относятся:

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

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

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

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

- Анализируемость: оценка ресурсов, необходимых для диагностики недостатков или причин сбоев, а также для идентификации тех фрагментов ПП, которые должны быть модифицированы.
- Изменяемость: оценка ресурсов, необходимых для проведения заданных модификаций.
- Стабильность: оценка режимов непредусмотренного поведения ПП, включая ситуации, обнаруженные в процессе тестирования.
- Тестируемость: оценка трудозатрат персонала сопровождения и пользователей по тестированию модифицированного ПП.

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

В этом случае возможны четыре варианта развития событий при модификации ПП:

1) пользователям приходится смириться с отсутствием некоторых функциональных возможностей ПП или их несоответствием новым реалиям и выполнять их либо вручную, либо при помощи подручных инструментов, например Word и Excel. Соответственно возрастает трудоемкость обработки, вероятность ошибок (человеческий фактор), требуется обучение и коррекция мотивации персонала, а главное — теряется драгоценное время;
2) в обход ограничений ПП (врожденных или приобретенных в ходе эволюции бизнеса) создаются дополнительные компоненты («заплатки»). Здесь к риску допуска новых ошибок добавляются издержки на исследование «обходных путей», их программирование и отладку;

3) если первые два варианта решения неприемлемы, то тогда разработчик начинает реализацию изменений на уровне исходных текстов, что требует ресурсов, а главное — времени на завершение полного цикла разработки, тестирование и разворачивание новой версии ПП;
4) наиболее пессимистичное развитие ситуации, когда критичные для заказчика требования не только не покрываются ПП и всеми его штатными средствами настройки, но и разработчик в силу различных причин не берется реализовать их путем доработок или переработок программного кода и предлагает заново осуществлять выбор новой платформы и разрабатывать новое решение. Это может привести просто к катастрофе бизнеса, сокращению доли рынка, реальных доходов от продаж, снижению конкурентных преимуществ.

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

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

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

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

Рефакторинг *— трансформация ПП, в процессе которого ПП реорганизуется (не переписываясь) с целью улучшения структуры, без изменения ее функционала.*

Рефакторинг ориентирован на улучшение структурных характеристик и качественных показателей объектно-ориентированных программ без изменения их поведения. Этот процесс реализуется путем изменения отдельных операций над текстами, интерфейсами, средой программирования и выполнения ПП, а также настройки или внесения изменений в инструментальные средства поддержки ПП.

2.2 Государственные стандарты РФ серии ГОСТ Р

В настоящее время в РФ действует ряд государственных стандартов, регламентирующих процессы жизненного цикла создания ПП, разработанные на основе международного стандарта *ISO/IEC 12207 — Information Technology — Software Life Cycle Processes — Процессы жизненного цикла программных средств*:

- *ГОСТ Р ИСО/МЭК 12207–99* «Процессы жизненного цикла программных средств». Стандарт применяется при разработке, приобретении, поставке, эксплуатации и сопровождении программных продуктов и программных компонентов.
- *ГОСТ Р ИСО/МЭК ТО 15271–2002* «Руководство по применению ГОСТ Р ИСО/МЭК 12207». В стандарте основное внимание уделено особенностям, подлежащим учету при прикладном применении ГОСТ Р ИСО/МЭК 12207 в условиях реальных проектов создания программных средств.
- *ГОСТ Р ИСО/МЭК ТО 16326–2002* «Руководство по применению ГОСТ Р ИСО/МЭК 12207 при управлении проектом». Настоящий стандарт предназначен для лиц, отвечающих за управление реализацией основных процессов по ГОСТ Р ИСО/МЭК 12207: заказа, поставки, разработки, эксплуатации и сопровождения.
- *ГОСТ Р ИСО/МЭК 14764–2002* «Сопровождение программных средств». Область применения настоящего стандарта охватывает сопровождение различных программных средств, устанавливает основную структуру, в пределах которой могут быть выполнены, оценены и практически реализованы общие и конкретные планы сопровождения применительно к области действия и объему заданных программных продуктов.
- *ГОСТ Р ИСО/МЭК 15288–2005* «Системная инженерия. Процессы жизненного цикла систем». Настоящий стандарт определяет основы для описания жизненного цикла программных систем, устанавливает соответствующую терминологию.

Остановимся более подробно на описании и применении стандарта *ГОСТ Р ИСО/МЭК 12207–99 «Информационная технология. Процессы жизненного цикла программных средств»*. В основу стандарта положены следующие базовые понятия [10]: система, модель жизненного цикла, процесс.

Система *(system) — комплекс, состоящий из процессов, технических и программных средств, устройств и персонала, обладающий возможностью удовлетворять установленным потребностям или целям.*

Модель жизненного цикла *(life cycle model) — структура, состоящая из процессов, работ и задач, включающих в себя разработку, эксплуатацию и сопровождение программного продукта, охватывающая жизнь системы от установления требований к ней до прекращения ее использования.*

Процесс *(process) — набор взаимосвязанных работ, которые преобразуют исходные данные в выходные результаты.*

Все работы, которые могут выполняться в жизненном цикле программных средств, распределены по пяти основным, восьми вспомогательным и четырем организационным процессам (рис. 2.7).

Каждый *процесс* жизненного цикла разделен на набор *работ*; каждая работа разделена на набор *задач*. Процесс, работа или задача по мере необходимости инициируются и выполняются другим процессом, причем нет заранее определенных последовательностей.

Основные процессы жизненного цикла включают следующие процессы: заказ, поставку, разработку, эксплуатацию, сопровождение.

Процесс заказа состоит из следующих работ, выполняемых заказчиком: подготовки заявки на подряд, подготовки и корректировки договора, надзора за поставщиком, приемки и закрытия договора. Процесс начинается с определения потребностей заказчика в программном продукте или программной услуге. Далее следуют подготовка и выпуск заявки на подряд, выбор поставщика и управление процессом заказа вплоть до завершения приемки программного продукта или программной услуги.

Процесс подготовки начинается с описания потребности приобретения готового ПП. Далее заказчик формулирует функциональные, коммерческие, организационные и потребительские свойства ПП, а также требования к безопасности, защите и другие критические требования наряду с требованиями к проектированию, тестированию и соответствующим стандартам и процедурам.

Процесс поставки включает работы, выполняемые поставщиком: подготовку ответа, подготовку договора, планирование, выполнение и контроль, проверку и оценку, поставку и закрытие договора. Процесс может быть начат с решения о подготовке предложения в ответ на заявку на подряд, присланную заказчиком, или с подписания договора и вступления с заказчиком в договорные отношения по поставке программного продукта или программной услуги. Процесс продолжается определением процедур и ресурсов, необходимых для управления и обеспечения проекта, включая разработку и выполнение проектных планов по поставке ПП или программной услуги заказчику.

Процесс разработки включает следующие работы, выполняемые разработчиком: подготовку процесса; анализ требований к системе; проектирование системной архитектуры; анализ требований к программным средствам; проектирование программной архитектуры; техническое проектирование программных средств;

программирование и тестирование программных средств; сборку программных средств; квалификационные испытания программных средств; сборку системы; квалификационные испытания системы; ввод в действие программных средств; обеспечение приемки программных средств.

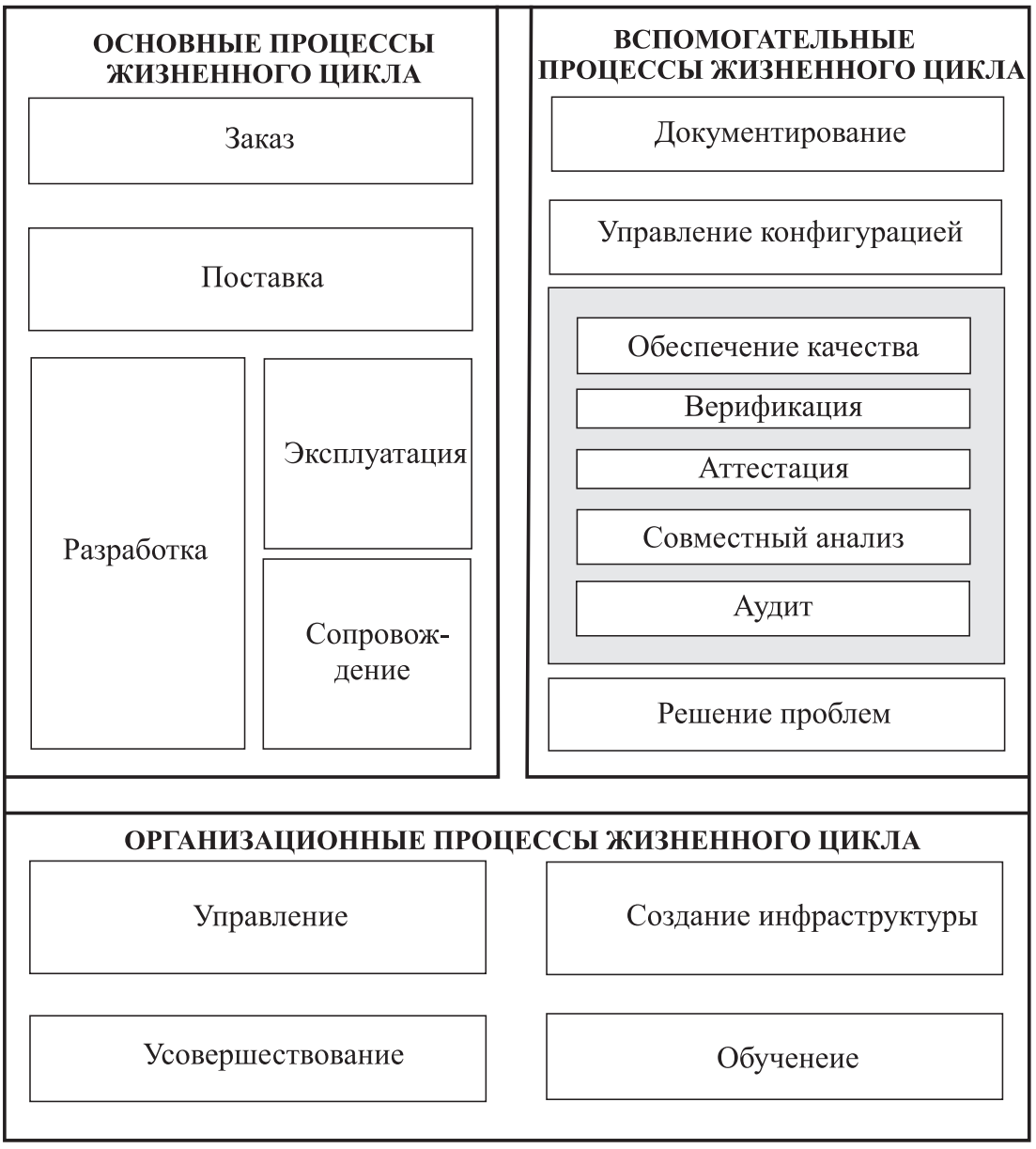


Рис. 2.7 – Структура стандарта

Процесс эксплуатации состоит из следующих работ, выполняемых группой внедрения заказчика: подготовки процесса; эксплуатационных испытаний; эксплуатации системы; поддержки пользователя. Процесс охватывает эксплуатацию программного продукта и поддержку пользователей в процессе эксплуатации.

Процесс сопровождения состоит из следующих работ, выполняемых группой сопровождения разработчика (поставщика): подготовки процесса; анализа проблем и изменений; внесения изменений; проверки и приемки при сопровождении; переноса; снятия с эксплуатации. Данный процесс реализуется при изменениях (моди-

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

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

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

Процесс управления конфигурацией является процессом применения административных и технических процедур на всем протяжении жизненного цикла программных средств (продуктов, комплексов, компонентов): обозначения, определения и установления состояния программных объектов; управления изменениями и выпуском объектов; описания и информирования о состоянии объектов и заявок на внесение изменений; обеспечения полноты, совместимости и правильности объектов; управления хранением, обращением и поставкой объектов. Данный процесс состоит из следующих работ: подготовки процесса; определения конфигурации; контроля конфигурации; учета состояний конфигурации; оценки конфигурации; управления выпуском и поставкой.

Процесс обеспечения качества является процессом обеспечения гарантий по соответствию программных продуктов и процессов в жизненном цикле проекта установленным требованиям и утвержденным планам. С точки зрения беспристрастности обеспечение качества должно быть организационно и полномочно независимым от субъектов, непосредственно связанных с разработкой программного продукта или выполнением процесса в проекте. Обеспечение качества может субъективно (внутренне или внешне) зависеть от того, демонстрируются ли доказательства качества продукта или процесса под управлением поставщика или заказчика. При обеспечении качества могут использоваться результаты других вспомогательных процессов, таких как верификация, аттестация, совместные анализы, аудит и решение проблем. Данный процесс состоит из следующих работ: подготовки процесса; обеспечения качества продукта в соответствии с условиями договора; обеспечения качества процесса (соответствия условиям договора); обеспечения систем качества (соответствия ГОСТ Р ИСО 9001).

Процесс верификации является процессом определения степени соответствия функционирующих программных продуктов требованиям или условиям, сформулированным в договорных документах. Для оценки эффективности затрат и выполняемых работ верификация должна как можно раньше реализовываться в соответствующих процессах, таких как поставка, разработка, эксплуатация или сопро-

вождение. Данный процесс может включать анализ, проверку и испытание (тестирование).

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

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

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

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

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

Организационные процессы жизненного цикла включают следующие процессы: управление, создание инфраструктуры, усовершенствование и обучение.

Процесс управления состоит из общих работ, которые могут быть выполнены как разработчиком, так и заказчиком, управляющими соответствующими процессами. Администратор отвечает за управление проектом, работами и задачами

соответствующего процесса или процессов, таких как заказ, поставка, разработка, эксплуатация, сопровождение или вспомогательные процессы. Данный процесс состоит из следующих работ: подготовки и определения области управления; планирования; выполнения и контроля; проверки и оценки; завершения.

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

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

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

Настоящий стандарт содержит полный набор описания процессов жизненного цикла ПП для некоторого типового проекта с максимально возможным составом процессов, работ и задач, которые используются в следующих случаях: *при приобретении и поставке программного средства* или отдельно поставляемого ПП; при оказании *услуги* по эксплуатации и сопровождению ПП, обучению пользователей; *при разработке* как тиражного так и заказного ПП. Поэтому при решении практических задач стандарт необходимо адаптировать к конкретному проекту.

2.3 Серия стандартов «Единая система программной документации (ЕСПД): ГОСТ 19.102–77 ЕСПД «Стадии разработки»

ГОСТ 19.102–77 ЕСПД «Стадии разработки» устанавливает следующие стадии разработки программных продуктов и программной документации независимо от их назначения и области применения (табл. 2.1).

Таблица 2.1 – Стадии разработки, этапы и содержание работ

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
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

продолжение на следующей странице

Таблица 2.1 — Продолжение

Стадии разработки	Этапы работ	Содержание работ
III. ТЕХНИЧЕСКИЙ ПРОЕКТ	Разработка технического проекта	Уточнение структуры входных и выходных данных. Разработка алгоритма решения задачи. Определение формы представления входных и выходных данных. Определение семантики и синтаксиса языка. Разработка структуры программы. Окончательное определение конфигурации технических средств
	Утверждение технического проекта	Подготовка плана мероприятий по разработке и внедрению программ. Разработка пояснительной записки. Согласование и утверждение технического проекта
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

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.

Контрольные вопросы по главе 2

1. Какие выгоды от соблюдения стандартов имеют заказчики и разработчики?
2. Раскройте содержание этапа «Определение требований».
3. Раскройте содержание этапа «Проектирование ПП».
4. Раскройте содержание этапа «Конструирование ПП».
5. Раскройте содержание этапа «Тестирование ПП».
6. Раскройте содержание этапа «Сопровождение ПП».
7. Проведите сравнительный анализ стадий разработки (создания) по стандарту 19-й серии и ГОСТ Р ИСО/МЭК 12207–99.
8. Раскройте содержание основных процессов стандарта ГОСТ Р ИСО/МЭК 12207–99 «Процессы жизненного цикла программных средств».
9. Перечислите организационные и вспомогательные процессы стандарта ГОСТ Р ИСО/МЭК 12207–99.
10. Перечислите и прокомментируйте работы стандарта 19-й серии — стадия «Техническое задание».
11. Перечислите и прокомментируйте работы стандарта 19-й серии — стадия «Техническое проектирование».
12. Перечислите и прокомментируйте работы стандарта 19-й серии — стадия «Рабочее проектирование».

Глава 3

МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ПРОДУКТА

3.1 Каскадная модель

Деятельность по созданию программного продукта описывается в виде *модели жизненного цикла* как последовательности процессов, работ и задач, обеспечивающих разработку, эксплуатацию и сопровождение программного продукта, отражающей эволюцию изменения продукта, начиная от формулировки требований к ней до прекращения ее использования. Такая последовательность может быть или не быть линейной, поскольку фазы могут следовать друг за другом, повторяться или происходить одновременно (рис. 3.1).

В литературе приводятся описания следующих моделей жизненного цикла разработки ПП: каскадной, V-образной, прототипирования, быстрой разработки приложений, инкрементной, спиральной [11, 12].

Одной из первых, применяемых на практике моделей была *каскадная модель*, или «*водопад*», в которой каждая работа выполняется один раз и только в определенной последовательности. При этом делается допущение, что каждая работа будет выполнена настолько тщательно, что после ее завершения и перехода к следующей работе возвращения к предыдущей не потребуется (рис. 3.2).

Отличительным свойством каскадной модели можно назвать то, 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 случае если: требования к ПП максимально четко определены, понятны и не изменяются; разрабатывается новая версия уже существующего продукта и вносимые изменения четко определены и управляемы; осуществляется разработка ПП типовых бизнес-процессов, содержание которых закреплено законодательно.

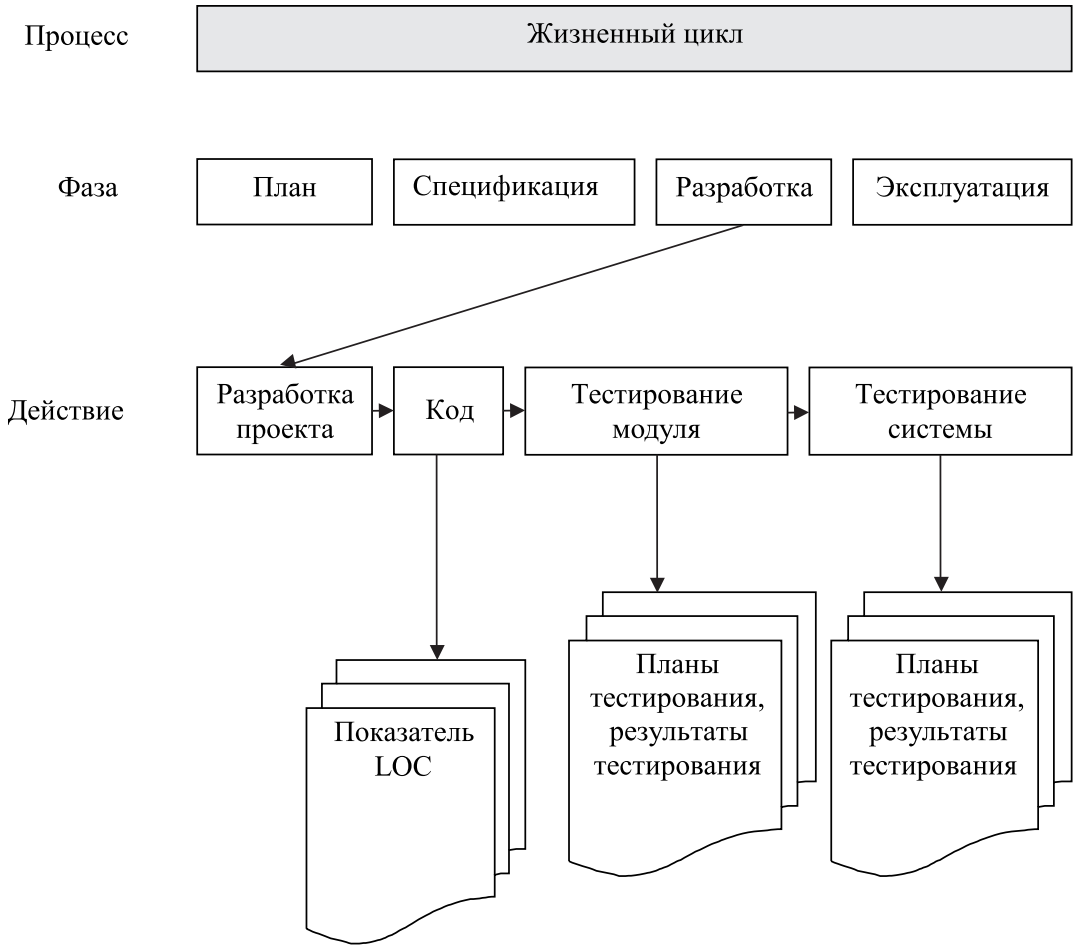


Рис. 3.1 – Обобщенная схема процесса

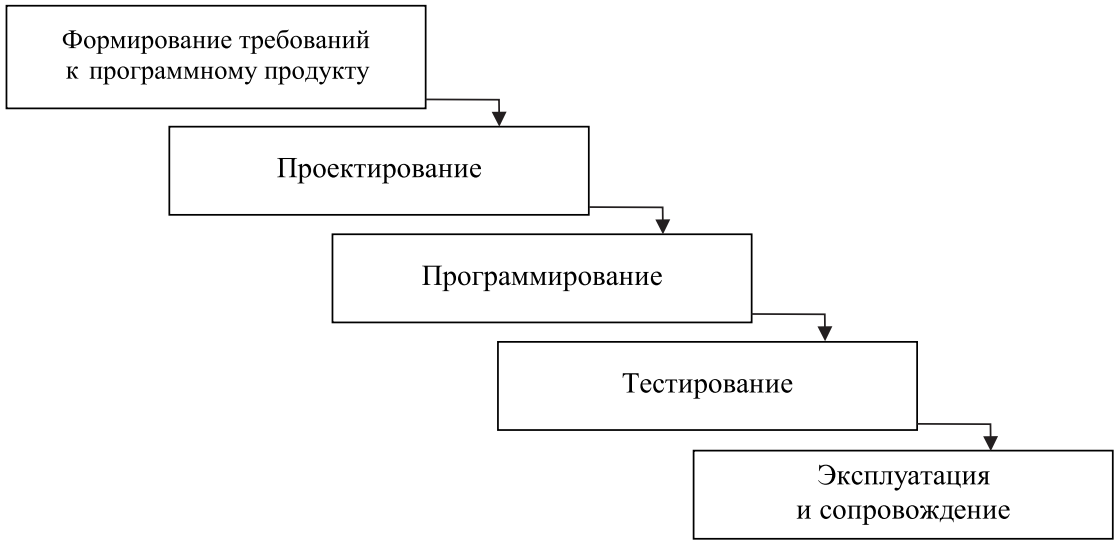


Рис. 3.2 – ЖЦ каскадной модели

Основные *положительные* моменты применения каскадной модели заключаются в следующем: модель проста и понятна заказчикам; каждая последующая

фаза начинается лишь тогда, когда полностью завершено выполнение предыдущей фазы; на каждом этапе 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 больших проектов, в которых задействовано несколько больших команд разработки ПП.

Недостатки каскадной модели особо остро проявляются в случае, когда трудно (или невозможно) четко сформулировать требования или требования могут меняться в процессе создания продукта. Кроме того, любая попытка вернуться на одну или две фазы назад, чтобы исправить какую-либо ошибку, приведет к значительному увеличению затрат и нарушению сроков разработки; интеграция программных компонентов, на которой обычно выявляется большая часть ошибок, выполняется в конце разработки, что сильно увеличивает стоимость устранения ошибок; большое запаздывание с получением и оценкой качества разрабатываемого ПП. В этом случае разработка ПП имеет циклический характер, когда результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних стадиях. Эти изменения связаны, как правило, с ошибками разработчиков, допущенными на ранних стадиях разработки и выявленными на стадии тестирования, изменениями требований в процессе разработки, связанными либо с неготовностью заказчиков правильно сформулировать требования, либо с изменениями требований, вызванными изменениями бизнес-процессов предметной области.

Таким образом, постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате *реальный процесс разработки* принимает другой вид (рис. 3.3).

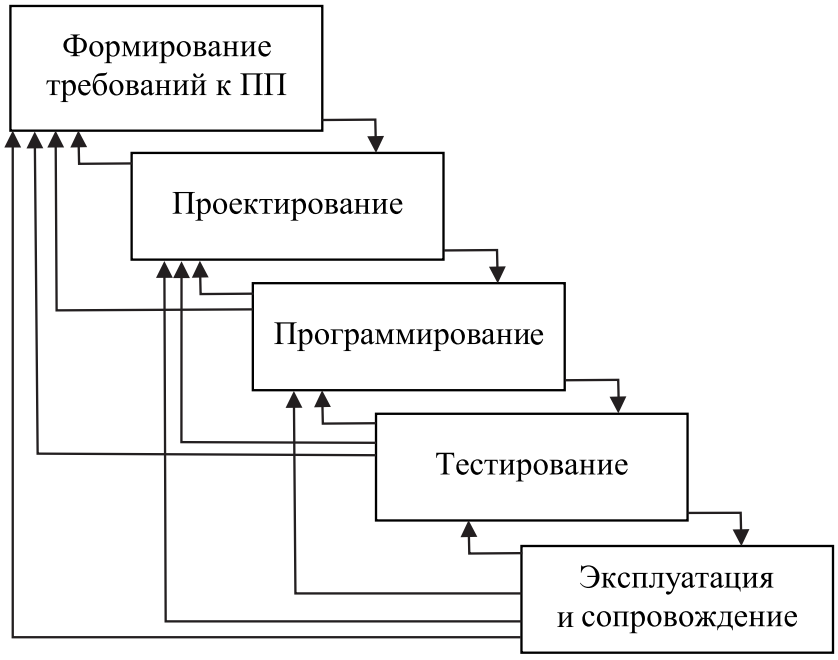


Рис. 3.3 – Реальный процесс разработки на базе каскадной модели

Модифицированная версия каскадной модели является в значительной степени менее жесткой, чем ее первоначальная форма. Здесь включаются итерации между фазами, параллельные фазы и менеджмент изменений. Обратные стрелки предполагают возможность существования итераций между действиями в рамках фаз. Несмотря на то, что модифицированная каскадная модель является значительно более гибкой, чем классическая модель, она все же не является наилучшим выбором для выполнения проектов по ускоренной разработке.

Попытки оптимизации каскадной модели привели к возникновению других типов моделей разработки ПП. Прототипирование программ позволяет обеспечить полное понимание требований, в то время как инкрементные и спиральные модели позволяют повторно возвращаться к фазам, соотнесенным с классической каскадной моделью, прежде чем полученный продукт будет признан окончательным.

3.2 V-образная модель

V-образная модель была создана с целью помочь работающей над проектом команде в планировании с обеспечением дальнейшей возможности тестирования системы. В этой модели особое значение придается действиям, направленным на верификацию и аттестацию продукта. Она демонстрирует, что тестирование продукта обсуждается, проектируется и планируется на ранних этапах жизненного цикла разработки. План испытания приемки заказчиком разрабатывается на этапе планирования, а компоновочного испытания системы — на фазах анализа, разработки проекта и т. д. Этот процесс разработки планов испытания обозначен пунктирной линией между прямоугольниками V-образной модели (рис. 3.4).

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

Модель включает в себя следующие фазы:

- *составление требований к проекту и планирование* — определяются системные требования и выполняется планирование работ;
- *анализ требований к продукту и их спецификация* — составляется полная спецификация требований к программному продукту;
- *проектирование системной архитектуры* (высокоуровневое проектирование) — определяется структура программного продукта, взаимосвязи между основными его компонентами и реализуемые ими функции;
- *детальное* (техническое) *проектирование* — определяется алгоритм работы каждого компонента;
- *разработка программного кода (кодирование)* — выполняется преобразование алгоритмов в готовый ПП;
- *модульное тестирование* — выполняется проверка каждого компонента или модуля программного продукта;
- *интеграционное тестирование* — осуществляются интеграция модулей программного продукта и его тестирование;

- *системное тестирование* — выполняется проверка функционирования программного продукта после установки его в соответствии со спецификацией нефункциональных требований на программно-аппаратную платформу заказчика;
- *эксплуатация и сопровождение* — запуск программного продукта в производство. На этой фазе в программный продукт могут вноситься поправки и может выполняться его модернизация.

Рис. 3.4 – ЖЦ V-образной модели

Основные *преимущества* V-образной модели заключаются в следующем:

- большая роль придается верификации и аттестации программного продукта, начиная с ранних стадий его разработки, все действия планируются;
- предполагаются аттестация и верификация не только самого программного продукта, 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 программных продуктов, главным требованием для которых является высокая надежность, когда доступными являются информация о методах решения функциональных задач и технологии их реализации, а персонал владеет необходимыми умениями и опытом в работе с данной технологией. Подобно своей предшественнице, каскадной модели, V-образная модель лучше всего срабатывает тогда, когда вся информация о требованиях доступна заранее.

3.3 Модель прототипирования

Основной идеей *модели прототипирования* является максимальное вовлечение пользователя в процесс разработки для 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, не обязательно представляющая собой все свойства системы, благодаря которой 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».

Как правило, под ***прототипом*** *понимается действующий программный компонент, реализующий отдельные функции и внешние интерфейсы разрабатываемого программного продукта.*

Использование модели прототипирования позволяет уже на этапе 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. 3.5.

Начало жизненного цикла разработки помещено в центре эллипса. Жизненный цикл разработки программного продукта начинается с совместной разработки конечными пользователями и разработчиками плана проекта, затем выполняется быстрый анализ предметной области, формулируются предварительные требования, проектируются база данных, пользовательский интерфейс и функционал бу-

дущего прототипа программного продукта. В результате этой работы создается также *документ*, содержащий частичную спецификацию требований к программному продукту, который в дальнейшем является основой для итерационного цикла *быстрого прототипирования*.

Рис. 3.5 – ЖЦ 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, он утверждается, 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, и для выполнения нефункциональных требований.

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

Модель протипирования обладает целым рядом *преимуществ*:

- 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 прототип позволяет гибко выполнять проектирование и разработку, включая несколько итераций на всех фазах жизненного цикла разработки;
- уменьшается число 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 заключаются в следующем:

- 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, что когда прототип превращается в конечный продукт, 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 итераций, которые будут необходимы;
- при выборе инструментальных средств прототипирования (операционные системы, технологии проектирования, языки программирования, алгоритмы решения функциональных задач) разработчики могут остановить свой выбор на неэффективных решениях, только чтобы продемонстрировать свои способности.

Модель прототипирования рекомендуется применять в случаях когда:

- выполняется новая, не имеющая аналогов разработка;
- заказчик неохотно соглашается на фиксированный набор требований, требования к программному продукту заранее неизвестны и могут уточняться в процессе разработки;
- разработчики не уверены, какие решения относительно пользовательского интерфейса и функционала следует выбрать, какую оптимальную архитектуру или алгоритмы следует применять.

3.4 Модель быстрой разработки приложений — RAD

В модели RAD пользователь задействован не только при определении требований, но и на всех остальных фазах жизненного цикла разработки ПП: проектиро-

вании, кодировании, тестировании, внедрении. 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 является короткое время перехода от определения требований до создания полной системы. Разработка каждого интегрированного продукта ограничивается четко определенным периодом времени, который, как правило, составляет 60 дней и называется временным блоком. В состав каждого временного блока входят 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, быстрого и качественного анализа промежуточных результатов, выделение необходимых ресурсов.

В RAD-модели конечный пользователь играет *решающую роль* (рис. 3.6). В тесном взаимодействии с разработчиками он участвует в формировании требований и апробации их на работающих прототипах. Таким образом, в начале жизненного цикла на конечного пользователя выпадает большая часть работы, но в результате этого создаваемая система формируется более быстро.

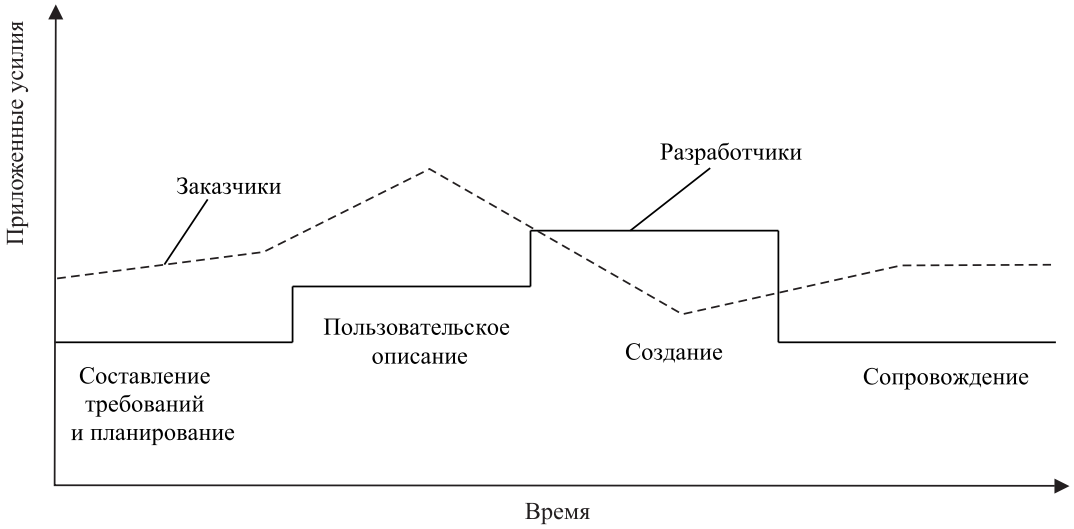


Рис. 3.6 – Модель быстрой разработки приложений

Модель включает в себя следующие фазы:

- *составление требований и планирование* — сбор требований осуществляется с использованием так называемого метода совместного планирования требований (планирование работ по созданию программного продукта и составление требований к программному продукту выполняются одновременно), который заключается в структурном анализе и обсуждении решаемых задач (будущего функционала);
- *пользовательское описание* — проектирование программного продукта, 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 программного продукта, а также поставка его за определенное время заказчику;
- *сопровождение* — проведение пользователем приемочных испытаний, установка программного продукта и обучение пользователей.

Модель обладает следующими *достоинствами*:

- использование современных инструментальных средств позволяет сократить время ЖЦ разработки;
- постоянное присутствие заказчика сводит до минимума риск неудовлетворения продуктом и гарантирует соответствие системы коммерческим потребностям и надёжность программного продукта в эксплуатации;
- основное внимание переносится с разработки документации на создание кода, причем при этом справедлив принцип «получаете то, что видите»;
- повторно используются компоненты уже существующих программ.

В то же время ей присущи и *недостатки*:

- если заказчики не могут постоянно участвовать в процессе разработки, то это может негативно сказаться на качестве программного продукта;
- для работы нужны высококвалифицированные кадры как разработчиков, так и пользователей, умеющих пользоваться современными инструментальными средствами;
- использование модели может оказаться неудачным в случае отсутствия 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.

Рассмотренную RAD-модель можно применять при разработке программных продуктов, которые хорошо поддаются моделированию, когда требования к программным продуктам хорошо известны, а заказчик может принять непосредственное участие на всех этапах ЖЦ процесса разработки.

3.5 Инкрементная модель жизненного цикла разработки

Инкрементирование представляет собой процесс поэтапной реализации ПП путем постепенного расширения его функциональных возможностей, при этом

необходим заранее сформированный полный набор требований. На ранних этапах жизненного цикла (разработка и анализ 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 требований первостепенной важности, играющих основную роль для успешного выполнения проекта и снижающих степень риска. Последующие итерации направлены на улучшение функциональных возможностей программного продукта (рис. 3.7).

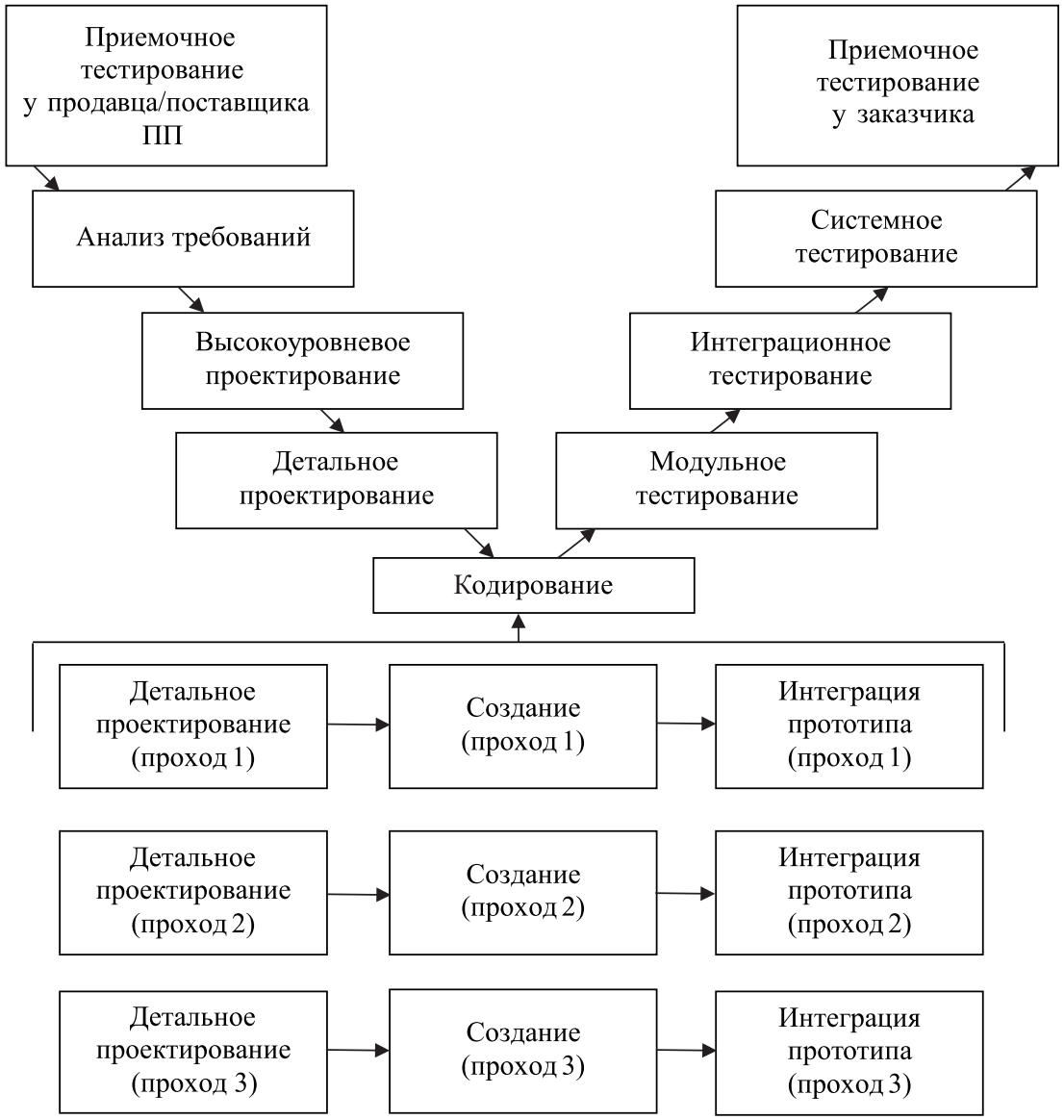


Рис. 3.7 – Инкрементная модель

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.

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 более ранних этапах разработки;
- риск распределяется на несколько меньших 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 раннее 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:

- для выделения инкрементов определение требований полной функциональной системы должно осуществляться в начале жизненного цикла;
- поскольку создание некоторых модулей будет завершено 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 необходимы хорошее планирование и проектирование: руководство должно заботиться о распределении работы, а технический персонал должен соблюдать субординацию в отношениях между сотрудниками;
- может возникнуть тенденция к оттягиванию решений трудных проблем на будущее с целью продемонстрировать руководству успех, достигнутый на ранних этапах разработки.

Инкрементные модели целесообразно использовать, если для этого имеются следующие предпосылки:

- если большинство требований и функционал можно сформулировать заранее, но их появление ожидается через определенный период времени;
- если рыночное окно слишком «узкое» и существует потребность быстро поставить на рынок продукт, имеющий функциональные базовые свойства;
- для проектов, на выполнение которых предусмотрен большой период времени разработки, как правило, более одного года;
- при разработке программ, связанных с низкой или средней степенью риска;
- при выполнении проекта с применением новой технологии, что позволяет пользователю адаптироваться к системе путем выполнения более мелких инкрементных шагов, без резкого перехода к применению основного нового продукта.

3.6 Спиральная модель

Спиральная модель была предложена как альтернатива каскадной модели, учитывающая повторяющийся характер разработки ПП, при этом предусмотрено также использование метода прототипирования или быстрой разработки приложений. Модель отображает базовую концепцию разработки, которая заключается в том, что каждый цикл представляет собой набор операций, которому соответствует такое же количество стадий, как и в модели каскадного процесса, начиная с общей формулировки требований и заканчивая кодированием каждой отдельной программы (рис. 3.8).

Основные принципы спиральной модели можно сформулировать следующим образом:

- Разработка вариантов продукта, соответствующих различным вариантам требований с возможностью вернуться к более ранним вариантам.
- Создание прототипов ПП как средства общения с заказчиком для уточнения и выявления требований.
- Планирование следующих 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.

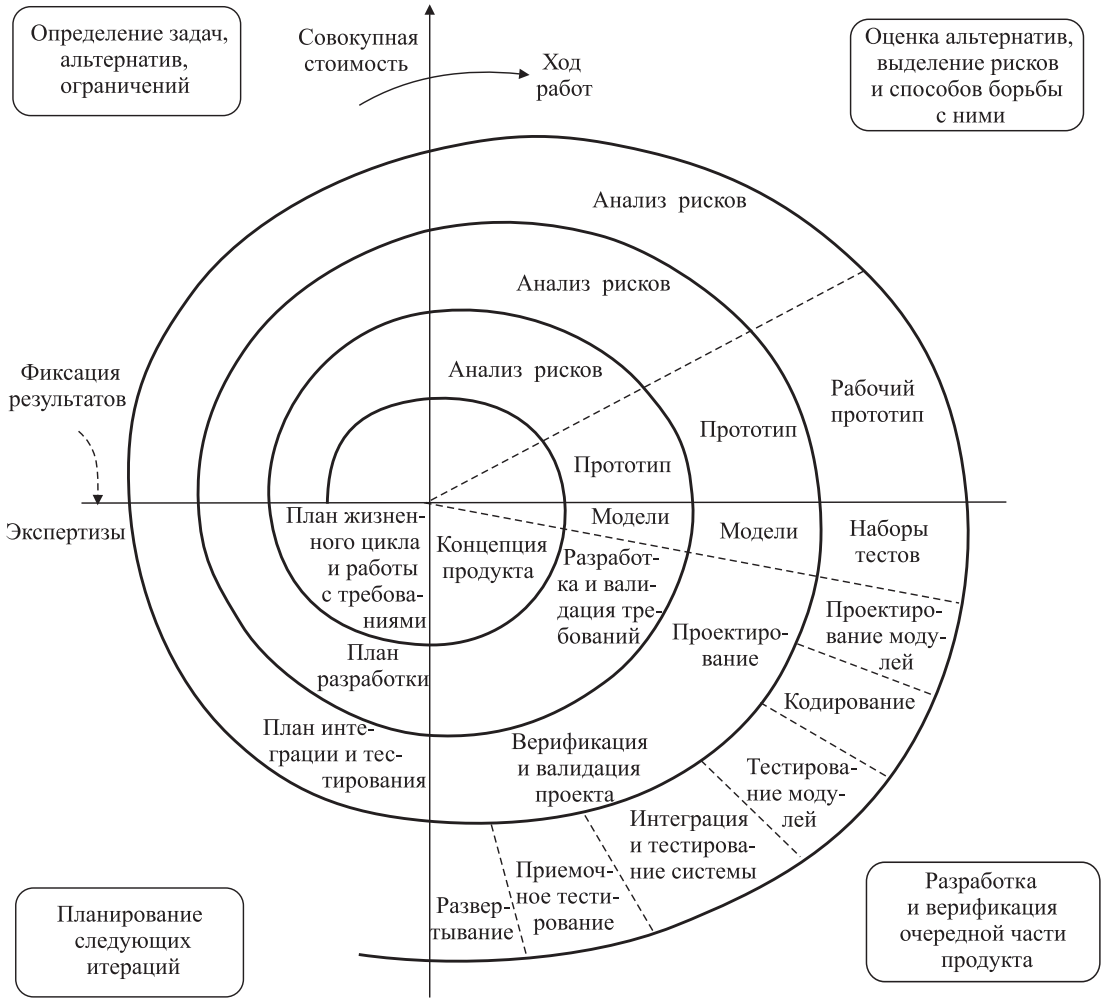


Рис. 3.8 – ЖЦ спиральной модели

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 разработке 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 с высокой частотой и на ранних этапах модели, что обеспечивает создание нужного продукта высокого качества;
- усовершенствование административного управления над процессом обеспечения качества, правильностью выполнения процесса разработки, затратами, соблюдением графика и кадровым обеспечением, что достигается путем выполнения обзора в конце каждой итерации;
- при использовании спиральной модели 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 чрезмерным;
- при выполнении действий на этапе вне процесса разработки возникает необходимость в переназначении разработчиков.

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

Спиральную модель целесообразно применять, когда:

- пользователи не уверены в своих потребностях или когда требования слишком сложны и могут меняться в процессе выполнения проекта и для анализа и оценки требований необходимо прототипирование;
- достижение успеха не гарантировано и необходима оценка рисков продолжения проекта;
- проект является сложным, дорогостоящим и обоснование его финансирования возможно только в процессе его выполнения;
- ПП реализуется с применением новых средств разработки, что связано с риском их освоения и своевременного достижения ожидаемого результата;
- в силу ограниченности ресурсов проект возможно выполнять только по частям.

3.7 Методика выбора модели жизненного цикла разработки ПП

Все приведенные выше жизненные циклы разработки ПП представляют собой логически взаимосвязанную совокупность процессов и работ, описывающих действия разработчиков и пользователей по создания программных продуктов, начиная с определения требований и заканчивая приемкой — сдачей готового продукта. Каждая модель имеет присущие ей характеристики, определяющие ее применение для определенных типов проектов.

В [11] в качестве таких характеристик моделей ЖЦ ПП предлагается использовать:

- особенности выявления и анализа требований к программному продукту;
- требования к составу и квалификации команды разработчиков программного проекта;
- степень участия коллектива пользователей при реализации программного проекта;
- характеристики сложности проекта.

В таблицах 3.1–3.4 приведены значения параметров приведенных выше характеристик для каждой из шести моделей ЖЦ программного продукта.

Таблица 3.1 – Характеристики модели ЖЦ в зависимости от особенностей процесса выявления требований к ПП

Требования	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
Являются ли требования легко определимыми и/или хорошо известными?	Да	Да	Нет	Нет	Да	Нет
Могут ли требования заранее определяться в цикле?	Да	Да	Нет	Нет	Да	Да
Часто ли будут изменяться требования?	Нет	Нет	Да	Да	Нет	Нет
Нужно ли демонстрировать требования с целью определения?	Нет	Нет	Да	Да	Да	Нет
Требуются ли для демонстрации возможностей ПП проверка концепции?	Нет	Нет	Да	Да	Да	Нет
Будут ли требования отражать сложность системы?	Нет	Нет	Да	Да	Нет	Да
Обладает ли требование функциональными свойствами на раннем этапе?	Нет	Нет	Да	Да	Да	Да

Таблица 3.2 – Характеристики модели ЖЦ в зависимости от квалификации команды разработчиков

Команда разработчиков проекта	**Каскадная**	**V-образная**	**Прототипирование**	**Спиральная**	**RAD**	**Инкрементная**
Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	Нет	Нет	Да	Да	Нет	Нет
Является ли технология предметной области проекта новой для большинства разработчиков?	Да	Да	Нет	Да	Нет	Да
Являются ли инструменты, используемые проектом, новыми для большинства разработчиков?	Да	Да	Нет	Да	Нет	Нет
Изменяются ли роли участников проекта во время жизненного цикла?	Нет	Нет	Да	Да	Нет	Да
Могут ли разработчики проекта пройти обучение?	Нет	Да	Нет	Нет	Да	Да
Является ли структура ПП более значимой для разработчиков, чем гибкость?	Да	Да	Нет	Нет	Нет	Да
Будет ли менеджер проекта строго отслеживать прогресс команды?	Да	Да	Нет	Да	Нет	Да

продолжение на следующей странице

Таблица 3.2 — Продолжение

Команда разработчиков проекта	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
Важна ли легкость распределение ресурсов?	Да	Да	Нет	Нет	Да	Да
Приемлет ли команда равноправные обзоры и инспекции, менеджмент/обзоры заказчика, а также стадии?	Да	Да	Да	Да	Нет	Да

Таблица 3.3 – Характеристики модели ЖЦ в зависимости от участия в проекте коллектива пользователей

Коллектив пользователей	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
Будет ли присутствие пользователей ограничено в жизненном цикле?	Да	Да	Нет	Да	Нет	Да
Будут ли пользователи знакомы с определением системы?	Нет	Нет	Да	Да	Нет	Да
Будут ли пользователи ознакомлены с проблемами предметной области?	Нет	Нет	Да	Нет	Да	Да
Будут ли пользователи вовлечены во все фазы жизненного цикла?	Нет	Нет	Да	Нет	Да	Нет

продолжение на следующей странице

Таблица 3.3 — Продолжение

Коллектив пользователей	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
Будет ли заказчик отслеживать ход выполнения проекта?	Нет	Нет	Да	Да	Нет	Нет

Таблица 3.4 – Характеристики модели ЖЦ в зависимости от сложности проекта

Тип проекта и риски	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
Будет ли проект идентифицировать новое направление продукта для организации?	Нет	Нет	Да	Да	Нет	Да
Будет ли проект иметь тип системной интеграции?	Нет	Да	Да	Да	Да	Да
Будет ли проект являться расширением существующей системы?	Нет	Да	Нет	Нет	Да	Да
Будет ли финансирование проекта стабильным на всем протяжении жизненного цикла?	Да	Да	Да	Нет	Да	Нет
Ожидается ли длительная эксплуатация продукта в организации?	Да	Да	Нет	Да	Нет	Да
Должна ли быть высокая степень надежности?	Нет	Да	Нет	Да	Нет	Да

продолжение на следующей странице

Таблица 3.4 — Продолжение

Тип проекта и риски	Каскадная	V-образная	Прототипирование	Спиральная	RAD	Инкрементная
Будет ли система изменяться, возможно, с применением непредвиденных методов, на этапе сопровождения?	Нет	Нет	Да	Да	Нет	Да
Является ли график ограниченным?	Нет	Нет	Да	Да	Да	Да
Являются ли «прозрачными» интерфейсные модули?	Да	Да	Нет	Нет	Нет	Да
Доступны ли повторно используемые компоненты?	Нет	Нет	Да	Да	Да	Нет
Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)?	Нет	Нет	Да	Да	Нет	Нет

Постановку задачи выбора наиболее эффективной модели ЖЦ, в зависимости от особенностей конкретного проекта можно представить в следующем виде.

Пусть известно множество моделей жизненного цикла ПП — $R = \{1, \ldots, r, \ldots, 6\}$. Каждая модель $r \in R$ описывается четырьмя группами характеристик $I = \{1, \ldots, j, \ldots, 4\}$, где $j = \{1, \ldots, j, \ldots, n\}$ — множество показателей входящих в состав каждой группы. Тогда $X_r = \{x_{ij}^r\}$, $i = \overline{1, 4}$, $j = \overline{1, n}$ — множество нормативных показателей r-ой модели жизненного цикла ПП. Каждый из показателей x_{ij}^r описывается в качественной шкале оценивания (да, нет) и характеризуется наличием либо отсутствием j-ого свойства модели в i-ой группе.

Пусть $X_p = \{x_{ij}^p\}$ — множество показателей, характеризующих особенности программного продукта. Мера сходства (близости) показателей жизненного цикла ПП с нормативными показателями каждой из моделей жизненного цикла определяется по расстоянию Хемминга $\rho\left(x_{ij}^r, x_{ij}^p\right) = \left|x_{ij}^r - x_{ij}^p\right|$, которое определяет количество совпадение (либо несовпадение) показателей ПП с показателями нормативных моделей ЖЦ. Тогда задача выбора модели жизненного цикла для реализации про-

граммного продукта состоит в выборе той модели, в которой совпадения показателей модели и проекта будет максимальным: $r_p = \min_{r \in R} \sum_{r=1}^{4} \sum_{j=1}^{n} |x_{ij}^r - x_{ij}^p|$.

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. Выбор и адаптация модели жизненного цикла разработки проекта оказывает влияние на содержание плана разработки продукта, в соответствии с задачами и целями конкретного проекта.

Контрольные вопросы по главе 3

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

Глава 4

КОМАНДООБРАЗОВАНИЕ

4.1 Организация командной работы над проектом

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

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

Инвестор *— физическое или юридическое лицо (группа лиц), предоставляющие в любой форме финансовые ресурсы для проекта.*

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

Куратор проекта *— представитель исполнителя, уполномоченный принимать решение о выделении ресурсов и внесении необходимых изменений в проекте.*

Руководитель проекта *— проект-менеджер, физическое лицо, которому заказчик и инвестор делегируют полномочия по руководству работами при осуществлении проекта.*

Соисполнители проекта *— физические или юридические лица, выполняющие на договорной основе отдельные виды работ по проекту.*

Команда проекта *— группа специалистов, формируемая в зависимости от потребностей, условий проектирования и организационной структуры выполнения проекта.*

Сложные программные проекты не выполняются индивидуально, они разрабатываются коллективом разработчиков, состоящим из множества разноплановых специалистов. В этом случае объективно требуется *распределение работ* и *ответственности* между членами команды проекта. При этом каждый из членов команды в зависимости от своей квалификации выполняет вполне конкретные *функциональные роли*.

В соответствии с методологией Microsoft Solutions Framework в команде проекта рекомендуется выделять семь функциональных ролевых групп [13].

Обязанности *в функциональных ролевых группах* распределены следующим образом:

1) *группа управления программой* — 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 [6] 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.

Группа проектирования и разработки ПП обеспечивает проектирование базы данных системы, компонентов и подсистем в соответствии с общей архитектурой, разработку архитектурно значимых модулей и интерфейса пользователя, проектирование, реализацию и отладку отдельных модулей системы.

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

Участники *группы обеспечения реализации проекта*, как правило, не входят в команду проекта. Они выполняют работы в рамках своей профессиональной деятельности. К этой группе можно отнести следующие проектные роли: *разработчик документации; переводчик; дизайнер графического интерфейса; разработчик учебных курсов; специалист по маркетингу и продажам; специалист по инструментальным средствам*.

Нет однозначных рекомендаций и по выделению функциональных (должностных) обязанностей участников команды внутри групп. В соответствии с методологией Microsoft Solutions Framework в команде проекта выделяются шесть видов специализации сотрудников: *менеджер проекта*, *архитектор*, *бизнес-аналитик*, *разработчик*, *тестер*, *менеджер по работе с заказчиками*. В [14] 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 проектирования архитектуры высокого уровня и контроля ее выполнения; *разработчик* — специалист, ответственный за создание качественного программного кода; *тестировщик* — сотрудник, отвечающий за удовлетворение функциональных и нефункциональных требований к ПП; *инженер по качеству* в обязанности которого входят: управление качеством конечного продукта, управление качеством процесса разработки, управление качеством организации работ по проекту; *технический писатель* — разработчик технической документации на программный продукт; *технолог разработки* — специалист, отвечающий за поддержку работоспособности принятой модели ЖЦ ПП и управление версиями.

Функциональные ролевые группы составляют основу для проектирования *организационной структуры управления программными проектами*. Образно организационную структуру системы управления можно сравнить с оркестром. Оркестранты играют на разных инструментах разные партии, но для достижения общего звучания область свободы каждого музыканта должна быть ограничена. Дирижер, согласуясь с партитурой, объединяет всех музыкантов для исполнения произведения [15].

Существует несколько взаимодополняющих определений организационной структуры управления:

- форма распределения задач и полномочий по принятию решений между лицами или структурными подразделениями;
- количество и состав звеньев и ступеней управления, их соподчиненность и взаимосвязь;
- пространственно-временное расположение элементов структуры и взаимосвязей для достижения целей проекта.

Анализ приведенных определений показывает, что при проектировании организационной структуры требуется решить следующие принципиальные вопросы:

- определить виды специализации подразделений и отдельных сотрудников;
- определить необходимое и достаточное количество подразделений, подчиняющихся одному руководителю, и сотрудников в каждом подразделении;
- определить целесообразное количество уровней управления и количество 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 основным видам конечных продуктов либо услуг (проект А, проект В, проект С).

Первый вид специализации порождает *функциональную* структуру управления, второй — *проектную (линейную)* структуру управления. Комбинацией описанных выше классических структур является *матричная* структура управления, в которой совмещаются функциональное и линейное управление при достижении баланса между функциональными и линейными руководителями проекта.

Подробное описание каждой из структур их достоинства и недостатки приведены в [11, 15].

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

1) постоянная корректировка структуры управления в соответствии с целями и задачами организации, обусловленная динамическим характером процесса распределения функций по службам структуры;
2) закрепление всех функций за соответствующими службами, при этом для больших фирм данная задача должна быть обязательно доведена до логического конца;
3) закрепление каждой функции за одной службой во избежание дублирования, влекущего рост накладных расходов на содержание аппарата и возникновение просчетов в управлении;
4) обеспечение завершенности процесса проектирования организационных структур. Проектирование структуры не должно заканчиваться процедурой распределения функций по структурам. Необходима разработка регламентных информационных технологий управления (принятия решений), ориентированных на достижение конкретных целевых результатов.

Распределение функциональных обязанностей сотрудников в команде производит руководитель проекта с учетом профессиональных качеств программиста и его типа личности. Эти обязанности излагаются в должностных инструкциях.

Одна из первых попыток классификации людей по *типу личности*, сделанных еще в трудах Гиппократа, основана на разделении по темпераментам. *Темперамент* (от лат. *temperamentum* — надлежащее соотношение частей) — характеристика индивида со стороны динамических особенностей его психической деятельности, т. е. темпа, ритма, интенсивности отдельных психических процессов и состояний. Темперамент определяется наследственностью и устойчиво проявляется в течение всей жизни человека. Описанные выше характеристики группируются в четыре типа темперамента [7]:

- *холерик*, имеющий самые скоростные темпоритмы, много и быстро говорящий, без промедления отвечающий собеседнику, часто перебивающий: когда собеседник только начал о чем-то говорить, холерик уже все понял и имеет готовый ответ;
- *флегматик* — спокойный, миролюбивый и сдержанный человек, никогда не перебивает собеседника, умеет внимательно выслушать и кивает в знак согласия, у него мягкие и неторопливые движения, негромкий голос;
- *сангвиник* — деловитый, выносливый и работоспособный человек, с развитым самоконтролем, нередко трудоголик, стремится к карьерному росту и высоким заработкам;
- *меланхолик* — чувствительный, обидчивый и очень ранимый человек, легко расстраивается даже при мелких неудачах, любит жаловаться на судьбу, искренне верит, что самая «тяжелая доля» и самые «тяжкие испытания» из всех возможных выпали именно ему.

В [16] в качестве примера отмечается, что наиболее продуктивными специалистами являются *флегматики*, из них получаются как грамотные и настойчивые в реализации программисты, живущие в реальном мире и обладающие конкретностью, точностью и практичностью, стремящиеся к специализации, так и успешные архитекторы — руководители проектов, рассматривающие широкий спектр возможностей, абстрагирующиеся от технических деталей, склонные обобщать и теоретизировать.

Описания качеств, характерных для каждого из типов личности сотрудника-флегматика (специалиста-флегматика, программиста-флегматика, архитектора-флегматика), приведены в табл. 4.1.

Таблица 4.1 – Характеристика командно-функциональных ролей сотрудников-флегматиков

Специалист-флегматик	**Программист-флегматик**	**Архитектор-флегматик**
Любят тишину, позволяющую сосредоточиться на работе	Любят решать проблемы, используя прошлый опыт и стандартные подходы	Любят решать новые, сложные проблемы
Способны работать долгое время над одним проектом, не отрываясь	Любят применять уже освоенные навыки и знания	Больше любят осваивать новые навыки и знания, чем их применять
Интересуются фактами и идеями, лежащими в основе работы	Склонны не доверять своей интуиции и игнорировать ее	Склонны следовать своей интуиции, удачно или неудачно
Сначала думают, а потом (может быть) действуют	Редко ошибаются в фактах	Могут допускать ошибки в фактах
продолжение на следующей странице		

Таблица 4.1 — Продолжение

Специалист-флегматик	Программист-флегматик	Архитектор-флегматик
Воспринимают телефонные звонки как помехи работе	Любят работу с практическим уклоном	Любят работу с новаторским уклоном
Генерируют идеи путем уединенных размышлений	Говоря о работе, сначала сообщают конкретные детали	Говоря о работе, сначала представляют ее общий обзор
Любят работать в уединении	Предпочитают работать установившимся способом, оттачивая свое мастерство	Любят изменения в своей работе, порой весьма радикальные
Испытывают затруднения, вспоминая имена и лица людей	Обычно действуют методично, продвигаясь шаг за шагом к цели	Действуют по настроению в соответствии с приливами сил
Склонны предпочитать письменное общение устному	Обращают внимание на специфику каждой ситуации	Обращают внимание на новые возможности и интересные проблемы
Предпочитают учиться на основе чтения литературы, а не обсуждений	Принимают существующую ситуацию как данность, с которой надо работать	Склонны усложнять решаемую задачу
Лучше всего работают, когда могут планировать работу и придерживаться планов		Задаются вопросом о причинах сложившейся ситуации
Любят приводить в порядок и завершать свои дела		
Могут не замечать новые возникающие дела, которые необходимо сделать		
Испытывают удовлетворение, приняв решение по поводу ситуации		
Быстро принимают окончательное решение		

продолжение на следующей странице

Таблица 4.1 — Продолжение

Специалист-флегматик	Программист-флегматик	Архитектор-флегматик
Стремятся к упорядоченной структуре и планированию времени		
Используют записи при планировании деятельности		
Приступая к новой работе, стремятся выявить только существенные моменты		
Не любят отрываться от работы ради более срочного дела		
Планируют работу так, что каждый шаг выполняется вовремя		

Команда, которая начинает проект, не остается неизменной, она проходит определенные стадии формирования и, как правило, количественно растет по мере развития проекта. Поэтому процесс должен постоянно адаптироваться к этим изменениям. Главный принцип: не люди должны строиться под выбранную модель процесса, а модель процесса должна выбираться под возможности конкретной команды, чтобы обеспечить наивысшую эффективность ее работы.

4.2 Роль руководителя в команде

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 [7]:

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*. Технологии разработки ПП весьма быстро морально устаревают, а качественно новые версии инструменталь-

ных средств появляются иногда с периодичностью раз в полгода. В связи с этим необходимо постоянное повышение квалификации сотрудников;

3) *высокая самооценка программиста*. Узкая специализация и высокая профессиональная квалификация в конкретной области часто вызывают завышенную самооценку своих возможностей у сотрудников. В связи с этим у руководителя возникают две проблемы:

   - руководитель должен сам точно представлять реальные возможности своих сотрудников, в противном случае неприятные неожиданности неизбежны;
   - сотруднику с высокой самооценкой трудно что-либо приказать, его необходимо убедить, что бывает непросто, в силу того что сам руководитель вряд ли может быть авторитетом в той области, в которой сотрудник является узким специалистом;

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 (из 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 невозможно управлять. 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 [16] стратегий:

1) стратегию «Директивное управление», при которой руководитель указывает, направляет, устанавливает. Характеризуется жестким назначением работ, строгим контролем сроков и результатов;
2) стратегию «Объяснения», при которой лидер объясняет и проясняет свои решения, убеждает. Характеризуется сочетанием директивного и коллективного управления;
3) стратегию «Участие», при которой лидер участвует, поощряет, сотрудничает, проявляет преданность. Характеризуется преобладанием коллективного принятия решений, обменом идеями, поддержкой инициативы подчиненных;
4) стратегию «Делегирование», при которой лидер делегирует 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 всего 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.

4.3 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 литературе описываются две модели управления: «бюрократическая модель» и «модель участия» [15].

Бюрократическая 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 необходимо 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, которая автоматически приводит 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* этой модели является четкая организация труда, разграничение ролей обязанностей и ответственности, хорошо организованная система контроля. Модель хорошо сочетается с каскадной моделью жизненного цикла и применима в тех же случаях, что и каскадная модель.

Недостатки модели связаны с тем, что административная система стремится к самосохранению (стабильности в работе), игнорируется активная роль сотрудников при выполнении проекта, отсутствует мотивация к творческому труду. Модель плохо восприимчива к изменению ситуации — новые типы проектов, применение новых технологий, оперативная реакция на изменения требований пользователей, поведения рынка и т. д.

Модель участия ориентирована на развитие и использование творческих способностей сотрудников к выполнению 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.

Основной принцип использования модели можно сформулировать так: «Работаем спокойно, работаем вместе» [12]:

- 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.

4.4 Основные положения мотивации 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 [7, 16]:

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 в будущем физиологических потребностей (пенсия);
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 как личности.

При этом уровни 1-й и 2-й отнесены к низшим потребностям, а уровни 3, 4, 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 понятия побуждения и вознаграждения.

Побуждение *— это порождаемое потребностью ощущение недостатка в чем-либо, имеющее определенную направленность.*

Вознаграждение (поощрение) *— это все, что человек считает ценным для себя.*

Все 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 и вознаграждения).

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 т. д.;
- 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.

Какие именно потребности надо удовлетворять, чтобы мотивировать сотрудников, зависит от личностных качеств конкретного человека: *особенностей характера, профессионального опыта, внешних обстоятельств, жизненной ситуации*.

Так, при организации работы программиста-флегматика, принимая во внимание особенности его темперамента, необходимо [16]:

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 план-графиков 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. 4.2 [16].

Таблица 4.2 – Зависимость мотивации участника команды от опыта

Потребности	**Предпочтения сотрудников в зависимости от профессионального уровня, %**		
	Начинающий	**Опытный**	**Мастер**
Материальные (зарплата, условия труда, социальный пакет)	50	20	—
Безопасность (стабильность компании, востребованность)	—	20	—
Принадлежность к команде (возможность учиться у более опытных коллег, опыт участия в успешном проекте, признание в коллективе)	40	20	10
Самоуважение (профессиональный рост, карьерный рост, возможность «выделиться» за счет индивидуальных качеств, позволяющих выполнять работу лучше других, повышение в должности, самостоятельность и ответственность в работе)	10	30	40
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 [16]:

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 надо находить достойные задачи, которые его заинтересует, поручать ему системную интеграцию проекта, реализацию архитектурно значимых компонентов («скелета» системы).

Основные рекомендации по разработке подходов к мотивации сотрудников сводятся к следующему [7, 16]:

- 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 мотиватором только в случае устойчивой связи между ее размером и результатами труда. При этом оптимально разбиение зарплаты на три составляющих: часть, определяемая должностью (постоянна и равна у всех сотрудников с равными должностями); часть, связан-

ная 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, нужно использование нематериальных вознаграждений и льгот.

В [17] 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 [7, 16]:

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.

Контрольные вопросы по главе 4

1. Перечислите и опишите роли участников проекта.
2. Перечислите и 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.

Глава 5

ИНИЦИАЦИЯ ПРОГРАММНОГО ПРОЕКТА

Любая IT-компания при создании ПП 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 из бизнес-моделей имеет свои особенности. При использовании заказной модели имеется риск разработать «под заказ» программный продукт, работающий с ошибками, непригодный для сопровождения и модификации. Кроме того, возможен риск «затянуть» проект или попасть в слишком опасную зависимость от постоянно меняющихся требований заказчика и т. д.

Использование продуктовой модели предполагает наличие востребованного на рынке продукта и обеспечение его тиражирования. С точки зрения оценки бизнеса компании-разработчика продуктовая модель более перспективна, в силу того что сама компания является непосредственным производителем новых проектов. При этом малыми ресурсами могут быть созданы инновационные продукты, имеющие большой экономический и коммерческий потенциал. Однако несмотря на привлекательность продуктовой модели, ее практическая реализация связана с решением трудной и важной задачи *выбора и оценки привлекательности идеи будущего программного продукта*. Это связано с тем, что ошибки, допущенные на этой стадии, существенно 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*. Недостаточное внимание именно этой фазе неизбежно приводит к существенным проблемам разработки и дальнейшей коммерциализации продукта.

В процессе инициации проекта [8]:

- создается творческое ядро команды по разработке будущего программного продукта;

- формируется и оценивается ряд привлекательных идей программного проекта;
- на основе наиболее привлекательных идей разрабатываются концепции реализации программного проекта, (определяется, в чем состоит целесообразность и выгодность проекта для компании, выбирается продуктово-рыночное направление продвижения ПП, определяются источники привлечения инвестиций и т. д.);
- проводится отбор наиболее перспективной концепции разработки будущего программного продукта.

Процедуру оценки перспективности идеи программного проекта можно описать в виде следующих этапов.

5.1 Подготовительный этап

На этом этапе один из инициаторов проекта должен сформировать команду для организации «мозгового штурма», приняв на себя на период проведения «мозгового штурма» 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. Эти ограничения и допущения должны задавать пограничные условия генерации идей.

5.2 Этап генерации идей

На этом этапе происходит непосредственная генерация идей по разработке программного продукта. 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 процедуру группового обсуждения конкретной проблемы с целью выработки альтернативных вариантов ее разрешения [18]. При использовании метода необходимо соблюдать следующие принципы:

1. Сознательное генерирование как можно большего количества вариантов. 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 ПП ведущих фирм, и далее генерировать варианты решений внутри этого интервала. Не рекомендуется оценивать рынок, конкурентоспособность, возможные проблем продвижения, а также думать о технологиях реализации идеи.

2. При организации работ на этапе генерации альтернатив необходимо помнить о существовании факторов, как тормозящих работу, так и способствующих ей. К их числу можно отнести психологическую несовместимость экспертов, различную инертность мышления, эмоциональные и физическое состояние и т. д. Совокупность этих факторов должна позволять эксперту не только самому продуктивно генерировать варианты, но и создавать условия для успешной деятельности других участников.
3. При значительном количестве альтернатив и ограниченном времени на принятие решения рекомендуется проводить предварительное «грубое отсеивание», не сравнивая альтернативы количественно, а лишь проверяя их на присутствие некоторых качеств, желательно для любой приемлемой альтернативы.
4. При генерации идей запрещена всякая критика, не только явная словесная, но и скрытая — в виде скептических улыбок, мимики, жестов и т. д. В ходе штурма между участниками должны быть установлены свободные и доброжелательные отношения. Надо, чтобы идея, выдвинутая одним участником штурма, подхватывалась и развивалась другим.

Отдельным направлением поиска перспективных идей является поиск в Интернете информации о недовольстве пользователями функционала ПП ведущих компаний. В этом случае необходимо подумать о разработке компонентов, позволяющих разрешить эти проблемы.

Процессом решения задачи управляет руководитель проекта, который обеспечивает соблюдение правил классического «мозгового штурма».

Результатом стадии генерации является множество идей, сформулированных в вид нескольких предложений. При описании идеи необходимо отразить:

- необходимость или потребность будущего программного продукта для общества;
- цели и основные результаты проекта;
- тип рынка и потенциальных потребителей, наличие конкурентов.

5.3 Этап обсуждения и оценки привлекательных идей

Процедура обсуждения и выбора привлекательных идей может быть проведена в форме дискуссии — открытом коллективном обсуждении сгенерированных идей. Основной задачей дискуссии является всесторонний анализ положительных и отрицательных моментов каждой идеи: актуальности и осуществимости, коммерческой привлекательности, возможности реализации, продуктов конкурентов, наработок и опыта команды и др. Поскольку на этой стадии важно понять, действительно ли существует проблема, которую команда разработчиков собирается решать при помощи своего продукта, обсуждение и оценка идеи должны быть проведены с представителями потенциальной целевой аудитории в виде опроса или собеседования.

Отбор наиболее привлекательной идеи можно производить как по итогам коллективного обсуждения идей, 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 идей по степени их перспективности. Наиболее привлекательной идее присваивается ранг, равный единице, второй — ранг, равный двум, и т. д. Интегральная оценка привлекательных идеи определяется суммированием рангов экспертов.

Результатом стадии отбора идей является несколько привлекательных идей, жизнеспособность которых обоснована экспертами. Каждая идея, ориентированная на определенные сегменты потребителей, должна быть далее воплощена в конкретный вариант концепции программного проекта. Если в ходе дискуссии окажется, что 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.

5.4 Разработка концепций программного проекта

В [5] отмечается, что концепция проекта разрабатывается, прежде всего, на основе анализа потребностей бизнеса и предназначена для подтверждения и согласования всеми участниками проекта единого видения целей, задач и результатов проекта. Концепция должна быть изложена в понятной форме, в ней, прежде всего, необходимо отразить вопросы, которые помогут глубже понять содержание идеи и проверить ее на практичность и эффективность реализации.

Содержание концепции программного проекта должно раскрывать следующие вопросы:

Бизнес-обоснование потребности или необходимости разработки ПП. В [19] необходимость или потребность для общества будущего программного продукта предлагается рассматривать в четырех аспектах: *коммерческая привлекательность* (появление новой потребности рынка в программных продуктах или услугах, изменение потребительского спроса на ПП); *изменение законодательства*, регулирующего состав и содержание бизнес-процессов в конкретных предметных областях; *научно-технический прогресс* в области развития программно-аппаратных средств информатизации (например, проект по разработке программного продукта для портативных устройств); *потребность отраслей социальной сферы* в информационно-коммуникационных услугах, финансирование которых берет на себя государство. После обоснования необходимости следует раскрыть следующие вопросы [17]:

- *Какие у потенциального заказчика существуют проблемы, 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?*

Цели, ограничения, допущения и основные результаты программного проекта. *Цели* программного проекта следует определять в виде желаемого результата достигаемого командой проекта при его успешной реализации. Формулировки целей должны быть: конкретными, измеримыми, согласованными, реальными, ограниченными по срокам. При описании желаемого результата программного продукта следует делать акцент на преимуществах, которые этот продукт несет потенциальным покупателям [17]:

- *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?*

В качестве одного из результатов должно быть краткое описание архитектуры будущего программного продукта как совокупности программных модулей (компонентов), с перечислением их функционала.

Основные сегменты рынка и потенциальные пользователи. Разработка данного раздела концепции должна начинаться с описания типа рынка с выделением его целевых сегментов и выявления множества потенциальных потребителей, которым может быть интересен продукт, особенности продвижения ПП в выбранных сегментах рынка и способы оказания услуг, отвечающих запросам потребителей. При выборе рынка отдавайте предпочтение горизонтальным рынкам.

Кроме этого, предлагается привести ответы на следующие вопросы [17]:

- *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?*

Экономика программного проекта. По одному из методов, описанных в [15], рассчитываются трудозатраты (человеко/месяцев) на разработку программных продуктов, бюджет проекта, рыночная цена продажи одной лицензии, срок окупаемости проекта.

Потенциал исполнителей. Кратко описывается история успеха команды проекта, наличие сертификатов и других документов, подтверждающих потенциал, определяется, сколько и каких исполнителей понадобится для реализации проекта, потребуется ли передача отдельных работ на аутсорсинг.

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 должна ответить на вопрос: есть ли у команды реальная возможность реализовать проект в приемлемые

сроки функционалом, востребованным потенциальным пользователем и в пределах имеющихся финансовых ресурсов?

5.5 Этап выбора перспективной концепции будущего ПП

5.5.1 Оценка перспективности концепции на основе мнения экспертов

В литературе описывается несколько подходов к выбору и обоснованию наиболее приоритетных для компании концепций проектов. Так, в [19] для решения этой задачи предлагается рассматривать, прежде всего, коммерческую необходимость проекта, оценивая ее с использованием методов финансового анализа. В качестве критериев отбора концепций проекта рекомендуется рассматривать следующие показатели: период окупаемости, поток денежных средств, анализ затрат и результатов, внутреннюю норму прибыли. Описание этих показателей и методик расчета приведено в [3]. Очевидно, что использование этого подхода для оценки перспективности концепции на ранней стадии программного проекта связано с низкой достоверностью численных значений показателей.

Другая группа методов оценки концепций программных проектов основана на использовании метода экспертных оценок. Основная идея метода состоит в том, чтобы использовать интеллект людей, их способность находить решения в слабоформализованных задачах при большом количестве качественной информации, наличии противоречивых целей, критериев и ограничений.

В общем виде постановка задачи оценки перспективности концепции программного проекта может быть представлена в следующем виде [15]. Пусть разработано множество концепций будущего ПП $I = \{1, 2, \ldots, i, \ldots, m\}$. Каждую из концепций предлагается оценивать по множеству критериев $H = \{1, 2, \ldots, h, \ldots, n\}$. Важность каждого из критериев для оценки перспективности концепции может быть задана в виде вектора относительной значимости критериев $Q = \{q_1, q_2, \ldots, q_h, \ldots, q_n\}$. Для оценки перспективности концепции создается экспертная группа, в которой множество экспертов описывается выражением $S = \{1, 2, \ldots, s, \ldots, d\}$. Относительная значимость (важность) каждого эксперта задается в виде вектора $K = \{k_1, k_2, \ldots, k_s, \ldots, k_d\}$. Требуется определить численные значения перспективности каждой из концепций и упорядочить их по убыванию величин по выражению (5.1):

$$x_i = \sum_{h=1}^{n} \sum_{s=1}^{d} q_h k_s x_{is}^h, \; i = \overline{1, m}. \tag{5.1}$$

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 [5] 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 методических рекомендациях по разработке научно-технических программ для оценки проектов рекомендуется использовать следующий набор критериев: *практическая востребованность* — проект должен иметь направленность на реальные и первоочередные проблемы пользователей; *обозримость* — проект должен быть реализован в приемлемые сроки; *эффективность* — проект должен быть направлен на получение максимального социального либо экономического эффекта при разумных затратах на его реализацию; *коммерческая привлекательность* — проект должен обладать привлекательностью для рыночного тиражирования; *потенциал исполнителей* — наличие у команды проекта, достаточной численности и опыта создания и внедрения подобных проектов; *реализуемость* — содержание проекта не противоречит действующему законодательству, требуемые объемы финансирования соответствуют возможностям команды проекта; *научно-технический уровень* — проект должен содержать новые, претендующие на получение патента технические решения, не имеющие аналогов.

Субъективный характер восприятия экспертами перспективности концепции программного продукта приводит к расхождению их мнений по этому вопросу. В связи с этим возникает необходимость количественной оценки степени согласованности экспертов. Для этих целей используется *дисперсионный коэффициент конкордации*, значение которого равно единице, если все оценки экспертов одинаковы, и нулю, если они различны. Если степень согласованности экспертов, недостаточна, экспертиза может быть повторена. При этом в открытой либо закрытой дискуссии должны участвовать эксперты, имеющие крайние точки зрения.

5.5.2 Гибридная модель оценки перспективности концепции

В [20] описана гибридная модель оценки перспективности концепции рыночного программного продукта, основу которой составляет сеть функциональных зависимостей. В формализованном виде модель сети функциональных зависимостей можно представить в виде ориентированного графа $G = (X, U)$, где $X = x_i$ — множество вершин графа представлено набором показателей, описывающих различные свойства объекта моделирования, $U = \{u_{ij}\}$ — множество направленных дуг графа, показывающих взаимосвязь между показателями.

В истоках графа находятся первичные показатели, характеризующие объект моделирования и значение которых не зависит от других показателей. В стоке сети расположен результирующий (целевой) показатель. Значения остальных показателей зависят от первичных и вычисляются через совокупность функциональных зависимостей.

Структура графа определяется используемой системой классификации показателей объекта моделирования. В данной работе авторами для четкой фокусировки продукта под требования рынка предлагается использовать три группы показателей: *продукта, рынка и проекта* (рис. 5.1).

На базе предложенной классификации построена гибридная модель оценки перспективности концепции рыночного программного продукта (рис. 5.2). Функциональная сеть состоит из 54 параметров, расположенных в шести слоях направленного графа (Приложение А) [20].

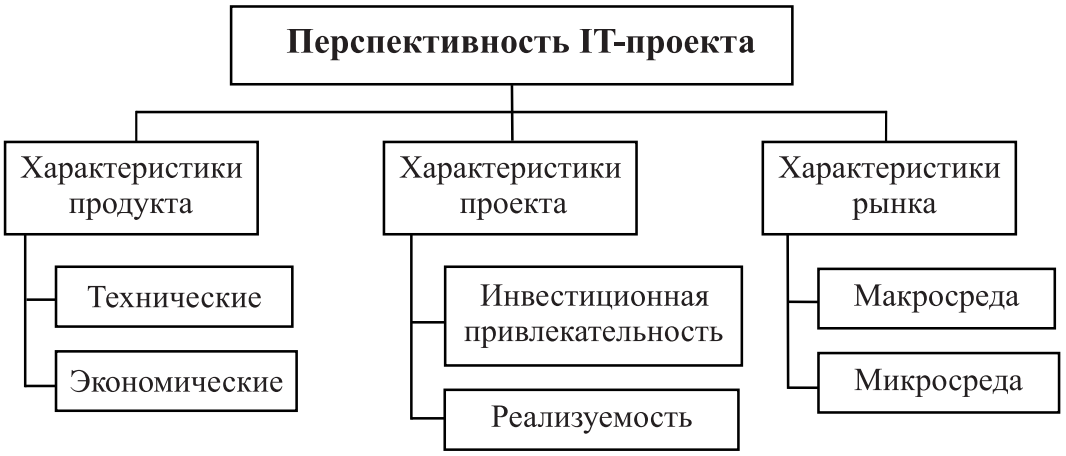


Рис. 5.1 – Классификация показателей оценки перспективности IT-проекта

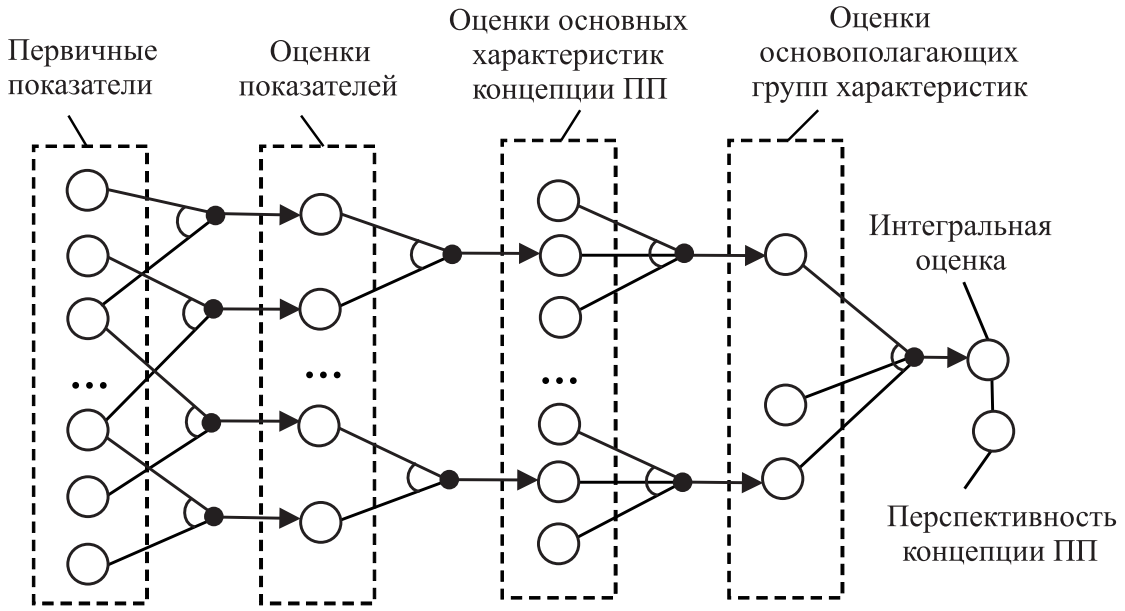


Рис. 5.2 – Общая структура модели оценки перспективности концепции рыночного ПП

В первом слое расположены первичные показатели, влияющие на перспективность концепции рыночного программного продукта, например: x_1 — затраты на разработку программного продукта; x_2 — средняя стоимость часа разработчика; x_4 — стоимость приобретаемого программного обеспечения; … x_7 — необходимость дополнительного привлечения сотрудников; x_8 — необходимость привлечения соисполнителей; … x_{35} — наличие государственной поддержки отрасли; … x_{37} — уровень развития у потенциальных пользователей информационно-коммуникационных технологий.

Второй слой сети составляют показатели, отражающие затраты на разработку, готовность команды к реализации проекта, уровень конкуренции на рынке и т. д. Для их вычисления используются либо математические функции x_{38} — затраты на разработку ($x_{38} = x_1 \cdot x_2 + x_3 + x_4 + x_5$), либо продукционные правила, x_{40} — уровень конкуренции (если x_{11} = «высокое» и x_{12} = «средняя» и x_{13} = «высокая», то

x_{40} = «высокий»; если x_{11} = «высокое» и x_{12} = «низкая» и x_{13} = «низкая», то x_{40}= «низкий»).

Третий слой сети составляют оценки основных показателей концепции программного проекта: x_{44} — вероятность успешной реализации, x_{45} — инвестиционная привлекательность и т. д. Значения данных показателей формируются с помощью правил-продукций.

В четвертом слое сети производится оценка перспективности по трем основополагающим свойствам концепции проекта: x_{50} — перспективность проекта, x_{51} — привлекательность рынка, x_{52} — перспективность программного продукта.

Интегральная оценка перспективности концепции программного продукта (показатель x_{53}) вычисляется по формуле: $x_{53} = a_1x_{50} + a_2x_{51} + a_3x_{52}$, где a_1, a_2, a_3 — весовые коэффициенты показателей четвертого слоя, задаваемые командой экспертов.

Качественная интегральная оценка перспективности концепции программного продукта определяется с помощью следующих продукционных правил:

- ЕСЛИ $0.8 < x_{53} < 1$, ТО x_{54} = «очень высокая».
- ЕСЛИ $0.6 \leqslant x_{53} < 0.8$, ТО x_{54} = «высокая».
- ЕСЛИ $0.4 \leqslant x_{53} < 0.6$, ТО x_{54} = «средняя».
- ЕСЛИ $0.2 \leqslant x_{53} < 0.4$, ТО x_{54} = «ниже среднего».
- ЕСЛИ $0 \leqslant x_{53} \leqslant 0.2$, ТО x_{54} = «низкая».

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

Контрольные вопросы по главе 5

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 гибридной модели оценки перспективности концепции программного проекта.

Глава 6

ПЛАНИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОЕКТА

6.1 Основное содержание этапов планирования и реализации программного проекта

Особенности программного продукта как результата интеллектуальной деятельности предъявляют определенные требования к этапам планирования и реализации программных проектов. С точки зрения классической теории управления, состав и структуру бизнес-процессов этих этапов можно представить в виде следующей схемы (рис. 6.1).

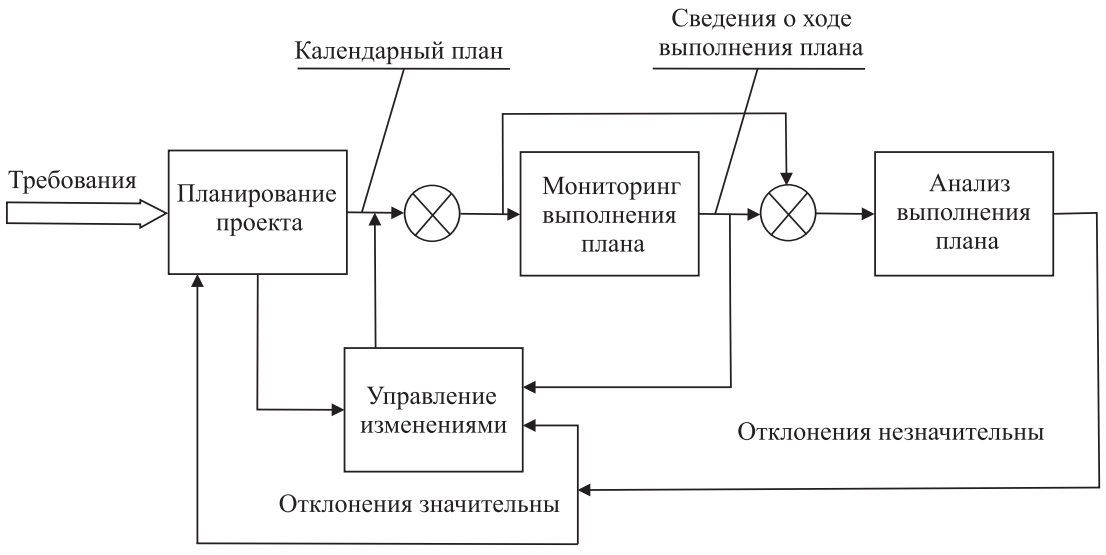


Рис. 6.1 – Схема управления проектом

Планирование проекта включает в себя следующий стандартный набор действий [8]:

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

В процессе календарного планирования программного проекта необходимо распределить взаимосвязанное множество работ по исполнителям, определить длительность, сроки начала и окончания выполнения работ. Круг вопросов, связанных с решением этой задачи, рассматривается в рамках прикладной науки — теории расписаний [21].

Вероятностный характер длительности выполнения отдельных работ увеличивает (усиливает) важность функции контроля и анализа плановых заданий и при необходимости корректировки первоначального варианта календарного плана. *Контроль и анализ выполнения работ* ориентированы на сравнение плановых и фактических показателей и определение отклонений от сформированных плановых заданий, при этом контролировать необходимо все четыре составляющие эффективности программного проекта: функционал, сроки, бюджет и качество. Мониторинг состояния этих характеристик проекта можно организовать путем:

- проведения отчетных совещаний по вопросам состояния выполнения проекта;
- контроля и анализа отклонений плановых и фактических характеристик проекта от графика выполнения, состояния бюджета, показателей качества.

В литературе принято выделять следующие виды контроля: *традиционный контроль* (контроль по фактическим отклонениям на конкретную дату) и *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: «Что произойдет, если тенденция по реализации проекта будет сохраняться?».

Если размеры отклонений достигли некоторой критической величины, необходима выработка управленческих решений по их устранению. При этом возможны два варианта изменений первоначального плана проекта: корректировка плановых заданий за счет привлечения дополнительных ресурсов; разработка нового варианта календарного плана.

Периодичность процессов контроля, анализа и корректировки плана программного проекта объективно требуют организации процесса *управления изменениями*. Для программных проектов управление изменениями можно организовать путем ведения хранилища данных — «библиотеки рабочего варианта ПП». Очевидно, что первыми элементами хранилища должны быть техническое задание и календар-

ный план проекта. В связи с возможными нарушениями целостности данных при многочисленных синхронных изменениях доступ к хранилищу должен быть строго ограничен.

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* [11]. Стандарт отвечает на вопрос: «Какими признаками должна обладать профессиональная организация по разработке ПП?». В соответствии со стандартом *модель технологической зрелости* — это описание стадий эволюции, которые проходят организации-разработчики по мере того, как они (организации) определяют, реализуют, измеряют, контролируют и совершенствуют процессы разработки программных проектов (создания программных продуктов).

Разработчики стандарта определили пять уровней технологической зрелости, по которым заказчики могут оценивать потенциальных претендентов на получение контракта на разработку ПП, компании разработчики совершенствовать процессы управления программными проектами. Каждый из уровней технологической зрелости соответствует определенному этапу развития компании по управлению и непрерывному усовершенствованию процесса разработки ПП.

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.

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.

Определенный. Базовые, вспомогательные и организационные процессы задокументированы, стандартизованы и интегрированы в унифицированную для данной компании технологию управления программными проектами, которая и используется при разработке ПП.

Управляемый. Управление всеми процессами по разработке ПП осуществляется по количественным оценкам. Детальные и объективные показатели о качестве исполнения процессов разработки ПП и характеристики самого продукта в соответствии с регламентом собираются и накапливаются.

Оптимизируемый. Совершенствование процессов разработки ПП осуществляется непрерывно как на основе количественного анализа эффективности процессов, так и использовании инновационных методов и технологий.

6.2 Содержательные модели структурной декомпозиции проекта

Структурная декомпозиции работ проекта (Work Breakdown Structure, WBS) является ключевым элементом в процессах преодоления *сложности программных проектов* и представляет собой описание перечня необходимых действий команды

проекта для его успешного завершения. В этот перечень должны быть включены производственные, управленческие и административные действия, обеспечивающие разработку ПП и управление программным проектом [8].

Сложные проекты выполняются коллективами разработчиков, что вызывает необходимость наличия определенных методик декомпозиции проекта на отдельные работы, выполняемые командой проекта, распределения работ между членами коллектива. Основной операцией декомпозиции является разбиение целого на части.

В существующих стандартах нет четких рекомендаций по способам выделения элементов следующего уровня при декомпозиции. Переход от чисто эмпирического умозрительного подхода деления целого на части к формализованному возможен при использовании формальных моделей декомпозиции. Для дальнейшего изложения материала введем следующие определения [18]:

- *декомпозиция* — процедура формального разбиения проекта на составляющие его элементы (разбиение целого на части);
- *модель декомпозиции* — набор формальных элементов, обеспечивающих однозначное разбиение целого на части.

Для однозначного определения множества элементов декомпозиции предполагается использовать три вида моделей:

1) *модель состава*, предназначенную для определения формального набора элементов проекта в целом либо его отдельных частей;
2) *модель жизненного цикла*, обеспечивающую выделение строго упорядоченной совокупности элементов, описывающих эволюционное преобразование проекта от момента его инициации до момента завершения;
3) *модель структуры*, описывающую формальное содержание работы либо задачи проекта.

Формирование состава элементов в перечисленных моделях декомпозиции целесообразно производить, основываясь на отечественных и международных стандартах на разработку программных систем. На рис. 6.2 с использованием «модели структуры» представлен вариант *структурной декомпозиции работ программного проекта* в виде элементов архитектурного дизайна ПП, где:

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

Глубина детализации зависит от размера и сложности программного продукта, культуры управления, стандартов жизненного цикла, специфики и масштабов проекта и т. д. С одной стороны, проект должен быть рассмотрен максимально всесторонне и полно. По мере увеличения глубины декомпозиции расширяются возможности планирования, управления и контроля работ. С другой — полученные результаты должны быть доступны для понимания и анализа. Чрезмерная декомпозиция может привести к непродуктивной управленческой трудоемкости, неэффек-

тивному использованию ресурсов и снижению эффективности реализации проекта. В этом случае *понятие полноты* работ проекта вступает в противоречие с *понятием их элементарности* [17]. Поэтому декомпозиция целого на части должна производиться до получения работ, понятных исполнителю и которые могут быть достаточно адекватно оценены по срокам исполнения и требуемым ресурсам. Корректность процедуры декомпозиции требует проверки того, что низкоуровневые элементы это именно те элементы, которые необходимы и достаточны для получения соответствующих результатов более высокого уровня.

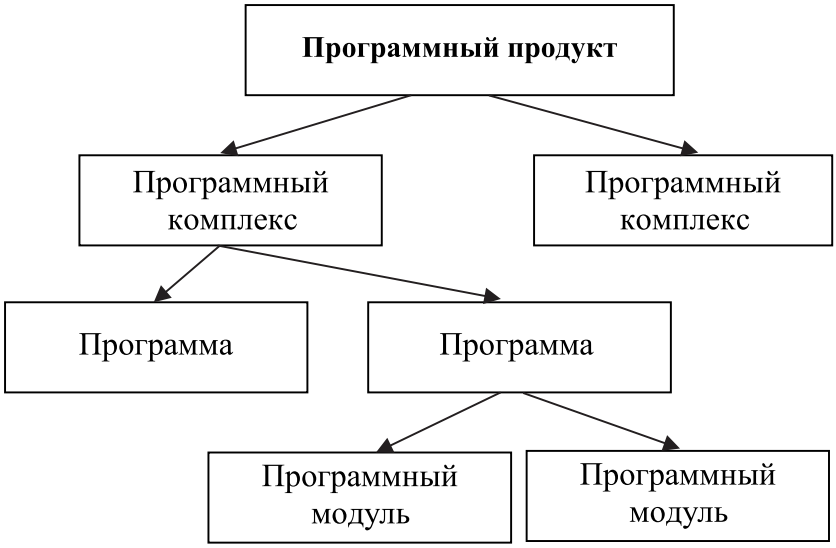


Рис. 6.2 – Структура программного продукта

Состав и последовательность разработки каждого из элементов архитектурного дизайна ПП зависит от используемых отечественных и международных стандартов, регламентирующих процессы ЖЦ ПП. Так, в соответствии с ГОСТ Р ИСО/МЭК 12207–99 «Информационная технология. Процессы жизненного цикла программных средств» [10] последовательность разработки ПП можно представить в виде процессов, работ и задач. Множество *процессов* соответствует этапам жизненного цикла ПП. Процессы декомпозируются далее на совокупность взаимосвязанных *работ* как деятельности по преобразованию объектов планирования из исходных (входных) данных в выходные результаты. Если после очередного этапа декомпозиции работа не удовлетворяет требованию элементарности, ее следует разложить далее на совокупность задач. В этом случае работа определяет достижение промежуточного результата, а *задача* является конкретным действием достижения этого результата и рассматривается, как элементарная единица планирования, поддающаяся достоверной оценке и управлению. В дальнейшем будем понимать термины «работа» и «задача» как слова синонимы, если по ним можно адекватно определить трудоемкость, назначить исполнителей и определить потребности в ресурсах.

Вариант распределения работ по элементам архитектурного дизайна программного продукта при использовании *каскадной модели* ЖЦ представлен в табл. 6.1.

Таблица 6.1 – Дорожная карта распределения работ по объектам планирования

Работы	**Объекты**					
	Программный продукт	**Программный комплекс**		**Программа**		**Программный модуль**
		1	2	1	2	1, 2
1. Разработка функциональных требований к ПП	+					
2. Разработка системных требований к ПП	+					
3. Разработка технического задания на ПП						
4. Проектирование архитектурного дизайна как многоуровневой структуры ПП	+					
5. Проектирование архитектуры программных комплексов, программ		+	+	+	+	
6. Программирование программных модулей						++
7. Разработка технической документации на модуль						++
8. Модульное тестирование						++
9. Сборка программы				+	+	
10. Разработка пользовательской документации на программу				+	+	
11. Тестирование программ				+	+	
12. Интеграция программ в программный комплекс		+	+			
13. Разработка эксплуатационной документации на программный комплекс		+	+			
14. Тестирование программного комплекса		+	+			
15. Сборка программного продукта	+		+			
16. Разработка эксплуатационной документации на ПП	+		+			

продолжение на следующей странице

Таблица 6.1 — Продолжение

Работы	Объекты					
	Программный продукт	Программный комплекс		Программа		Программный модуль
		1	2	1	2	1, 2
17. Организация приемки-сдачи ПП	+		+			
18. Ввод в эксплуатацию ПП	+		+			

Структурная декомпозиция работ может производиться с различной степенью детализации, например, на ранней стадии проекта в качестве единицы планирования можно рассматривать работу «Разработка программного модуля 1», а далее, при необходимости, ее можно детализировать на три отдельных задания (элементарные работы): программирование, разработка документации, тестирование.

Структура зависимостей между работами и задачами проекта может быть представлена как в виде *иерархического вложенного списка задач*, так и в виде «*дерева задач*» [11]. В иерархической структуре WBS множество задач и последовательность их выполнения определяется полным перебором элементов архитектурного дизайна ПП и используемого набора работ (задач) над их преобразованием (рис. 6.3). Вложение определяет уровни иерархий архитектуры ПП. Подобный тип описания структуры WBS рекомендуется для сложных программных проектов с большим количеством задач. Такие списки целесообразно создавать с помощью электронных таблиц, что позволяет руководителю проекта формировать различные сортировки и произвольные выборки заданий.

Описание структуры зависимостей между работами и задачами в виде «дерева задач» представлено на рис. 6.4. Такое описание структуры позволяет визуально представить сложность программного продукта и проводить изменение архитектурного дизайна (добавлять либо удалять его элементы). Однако размеры графического представления дерева велики, и эти процедуры достаточно трудоемки.

Определение состава и последовательности работ проекта является основой для разработки календарных планов реализации проекта, распределения и закрепления трудовых ресурсов, определения контрольных сроков выполнения отдельных этапов проекта.

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. 6.3 – Иерархический список заданий программного проекта

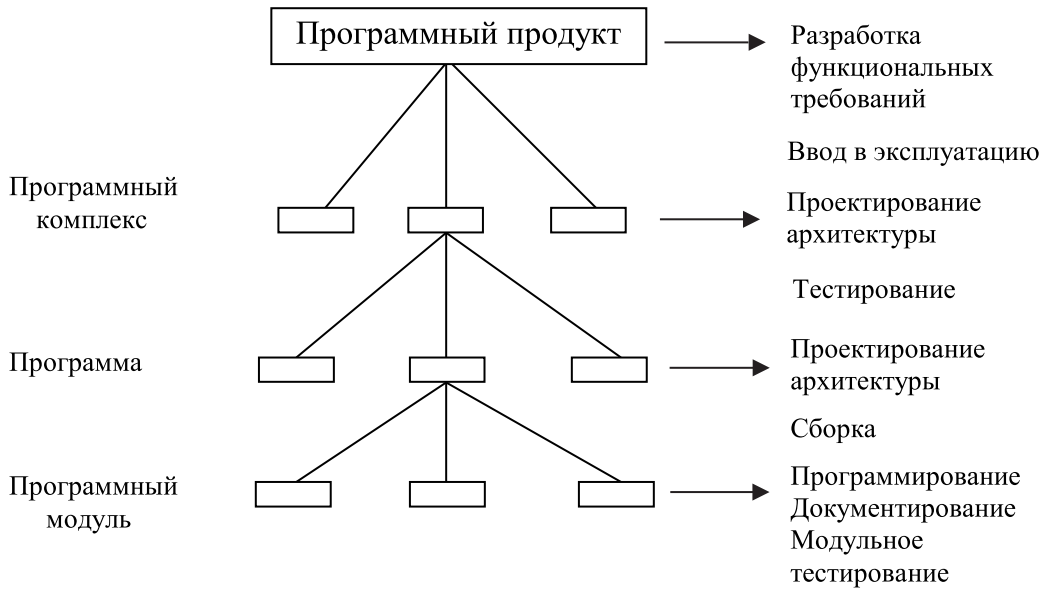


Рис. 6.4 – Древовидная структура зависимостей между работами и задачами

6.3 Математические модели планирования программных проектов

6.3.1 Содержательная и математическая модели формирования календарного плана программного проекта

Сформированное множество работ и задач предоставляет собой основу для оценки, планирования, исполнения, мониторинга и контроля выполнения проекта. Каждая задача из множества $I = \{1, 2, \ldots, i, \ldots, n\}$ формулируется как элементарная работа по преобразованию конкретного объекта планирования. Для выполнения работ потребуется один или несколько типов специалистов (архитекторов, бизнес-аналитиков, разработчиков, тестеров, технических писателей и т. д.) $Z = \{1, 2, \ldots, z, \ldots, m\}$. Каждый специалист в заданный момент времени может выполнять только одну работу, при этом для выполнения одной работы может потребоваться нескольких разнопрофильных специалистов. Закрепление специалистов по работам может быть оформлено в виде матрицы распределения полномочий (ответственности).

Первичными оценочными характеристиками работ являются:

- $T = \{t(i, z)\}$ — трудозатраты (в человеко-месяцах) на выполнение работы;
- $d(i, z) = f(i, k(z))$ — длительность работы, устанавливающая количество периодов времени (в часах, днях, неделях, месяцах), необходимых для ее выполнения. Определяется, как функции от количества привлекаемых специалистов;
- $s(i, z)$ — стоимость выполнения работы конкретным специалистом.

Существующие методики определения трудозатрат достаточно подробно изложены в [11, 22]. Однако на ранних этапах разработки ПП целесообразно использовать метод PERT-анализа (Project Evaluation and Review Technique) [11]. Его суть заключается в том, что для каждой работы проекта указывают три оценки трудоемкости — оптимистическую o, пессимистическую p и реалистическую 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 условии наличия множества проблем. Исходная предпосылка заключается в том, что если что-то плохое может произойти, оно обязательно произойдет.

Ожидаемая оценка трудозатрат — нечто среднее между оптимистической и пессимистической оценками. Это наиболее вероятное оценивание трудоемкости работ.

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 других случаях оценку трудозатрат можно определять как *среднее арифметическое* от оптимистической, пессимистической и ожидаемой оценок. Длительность выполнения работы обратно пропорциональна коли-

честву привлекаемых специалистов: $d(i, z) = d(i, z)/k(z)$. Например: «Разработка программного модуля 1» была оценена экспертами в 48 человеко-дней, для выполнения работы привлекаются 4 специалиста, тогда длительность работы составит 12 календарных дней.

Учитывая вероятностный характер выполнения работы, в модель структурной декомпозиции можно вводить буферные работы типа «Непредвиденные обстоятельства проекта» с соответствующей оценкой их длительности и типами привлекаемых специалистов. Примерами таких работ могут являются работы <*Разработка программного модуля 1*> и <*Разработка программного модуля 2*>.

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

По аналогии с [21] представим графически процесс реализации программного проекта в виде точек на плоскости. Каждая работа как точка на плоскости может быть описана в виде $<i, z, d(i, z), t^{н}(i, z), t^{к}(i, z)>$.

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

Технологическая зависимость точки определяется условием: <*последующая работа начинается после полного 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) (два задания) связаны временной зависимостью 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 использования ими 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, описанные 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 заданий привлекаются четыре типа специалистов $Z = \{z_1, z_2, z_3, z_4\}$. Последовательность выполнения заданий специалистами представлена в табл. 6.2. Каждая работа состоит из определенного количества заданий.

Таблица 6.2 – Последовательность выполнения работ

Работы	**Количество заданий**		
	1	**2**	**3**
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 задания каждой из работ 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. 6.5).

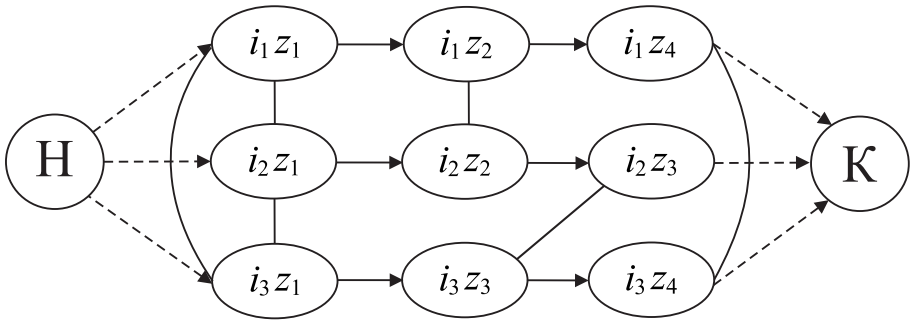


Рис. 6.5 – Смешанный граф

6.3.2 Алгоритм формирования календарного плана программного проекта

Многовариантная процедура 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 расчета методом критического пути времен начала и окончания всех заданий графа G'.

В основу метода критического пути положены расчеты нескольких дат выполнения каждой работы, включая:

- раннюю дату начала;
- раннюю дату окончания;
- позднюю дату начала;
- позднюю дату окончания.

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

Поздней датой начала выполнения работы называется самое позднее время его начала, при котором сохраняется общая длина пути и выполняются условия взаимосвязей между работами. Соответственно *поздняя дата окончания* работы также отличается от ранней на величину длительности задания.

Расчет критического пути состоит из следующих шагов [23].

На первом шаге для каждой работы вычисляются ранние даты начала и окончания ее выполнения. Вычисления начинаются с начальной работы и продолжаются до тех пор, пока не будет определено время, завершающее последнюю работу проекта.

На втором шаге вычисления начинаются с завершающей работы проекта и далее для каждой работы определяются поздние даты начала и окончания ее выполнения.

На третьем шаге, с использованием результатов вычислений на предыдущих шагах определяется совокупность взаимосвязанных работ, у которых раннее и позднее времена начала и окончания совпадают. Эти работы и образуют *критический путь* программного проекта. Длина критического пути (продолжительность реализации программного проекта) определяется путем суммирования длительностей всех работ, лежащих на этом пути.

На четвертом шаге необходимо вычислить резервы времени для некритических работ проекта. *Резерв времени* работы как разность между поздним и ранним временем начала определяет максимально допустимое время, на которое можно отложить момент окончания выполнения работы (сдвинуть время ее начала), при этом длина критического пути должна остаться неизменной. Очевидно, что все работы, входящие в критический путь, имеют нулевой резерв времени.

Резерв времени является показателем гибкости планирования сроков некритических работ в ориентированном графе. Рассчитывают обычно полный и свободный резервы времени. Полный резерв времени представляет собой максимальную продолжительность задержки выполнения работы, не вызывающую задержки в осуществлении всего проекта. Свободный резерв времени работы является показателем, характеризующим период времени, на который работа может быть отложена, не влияя на раннее начало выполнения последующих работ, и вычисляется как разность между минимальным временем начала всех работ, следующих за данной, и ранним сроком окончания данной работы.

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

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».

Поиск оптимального критического пути основан на замене по определенному правилу в графе $G = (X, \overline{U}, U)$ ребер на направленные дуги. В теории расписаний такие правила получили название *функции предпочтения, или приоритетов*. Ниже приводятся несколько классических правил предпочтения, используемых при решении задач теории расписаний [21].

Правило «кратчайшей операции» — SIO — первой выбирается работа с наименьшей длительностью выполнения.

Правило «*первым пришел — первый обслуживается*» — *FIFO* — первой выбирается работа с наименьшим резервом времени.

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

Очевидно, что поочередное использование каждого из этих правил позволяет получить различные варианты критического пути на графе G. Задача заключается в нахождении такого 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.

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 диаграммы Ганта, где слева перечисляются наименования работ и исполнители, а справа горизонтальные линии, длина которых, в соответствии с принятой шкалой временных параметров планирования соответствует длительности работ;
- в виде сетевых диаграмм (моделей), изображенных в виде совокупности узлов (работ) и дуг (взаимосвязей между ними) с указанием всех параметров работ: время начала и окончания, величина полного и свободного резерва времени.

6.4 Алгоритм выравнивания ресурсов

Рассчитанный по методу критического пути календарный план проекта может содержать неравномерную нагрузку (перегрузка либо простой) различных типов исполнителей в определенных интервалах планового периода проекта. Очевидно, что это негативно сказывается на ритмичности использования трудовых ресурсов. Эту проблему можно частично решить путем перепланирования работ, имеющих ненулевые резервы времени. Процедура перераспределения ресурсов может быть основана:

- на сдвиге некритических работ вперед либо назад — изменении дат начала и окончания в пределах резерва времени;
- разбиении некритических работ на части и сдвиге отдельных частей назад либо вперед;
- увеличении либо уменьшении количества исполнителей привлекаемых для выполнения работ;
- увеличении либо уменьшении длительности выполнения работ;
- сдвиге критических работ в «буферные зоны».

Ниже приводится описание одного из возможных алгоритмов выравнивания ресурсов, основанного на методе критической цепи [13].

В основу метода положено предположение, 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.

Представим графически календарный план реализации программного проекта в виде линейной диаграммы Ганта, где слева направо равномерно отсчитываются периоды времени (недели, месяцы), сверху вниз перечисляются работы, причём каждая работа представляется отрезком, начало и конец которого размещаются в нужном периоде (рис. 6.6).

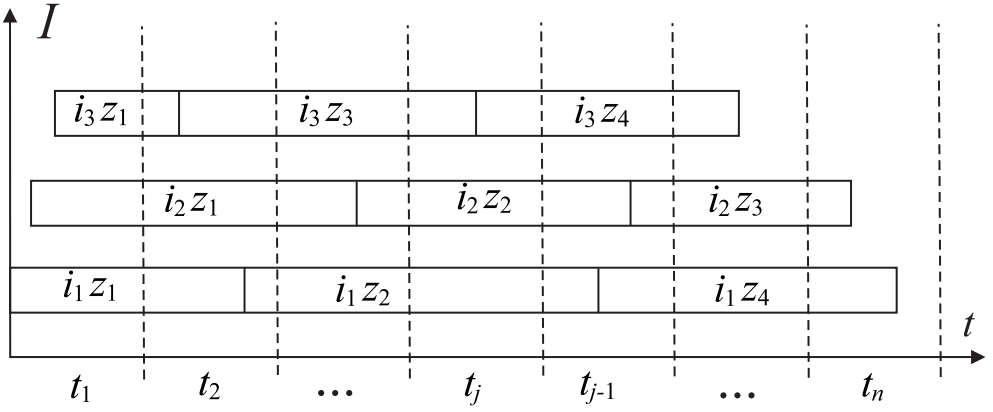


Рис. 6.6 – Линейная диаграмма Ганта

Каждая работа, выполняемая одним или несколькими типами специалистов, характеризуется продолжительностью выполнения и плановой трудоемкостью, интенсивность потребления в каждом интервале времени равномерна, прерывание процесса выполнения работы недопустимо. Объемы трудовых ресурсов в каждом интервале времени ограничены.

Сущность алгоритма выравнивания ресурсов при заданном времени выполнения работ описана в [17, 23]. Интервал планирования $[0, T_k]$ 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:

- если работа может быть выполнена разными специалистами, то предпочтение отдается исполнителю с наименьшей на данный момент времени загрузкой;
- проверяется, выполняются ли ограничения на имеющиеся ресурсы в данном интервале планирования.

Если ограничение по ресурсам не выполняется, работа двигается влево на один интервал времени. 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 мало, 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.

6.5 Рекомендации по управлению ЖЦ программных проектов

Описанные ниже рекомендации предложены и использованы Дж. Маккартни при разработке приложений в среде MS + 4.0 для продукта Microsoft Solution Framework, созданного фирмой Microsoft. Рекомендации объединены в три раздела правил: «Выпустить», «Лучший проект», «Выпустить точно в срок» [24].

Раздел 1. «ВЫПУСТИТЬ»

1. *Вы не знаете того, чего вы не знаете*. В любом проекте всегда существуют неопределенности, но человеку свойственно отвергать свое незнание, подменяя истинное знание предположениями. Неизвестно, что является ошибкой, поэтому старайтесь быть критичными к себе. Найдите для своей команды мотивацию к тому, чтобы она постаралась обнаружить и понять максимум «белых пятен». Не пытайтесь обмануть себя предположениями.
2. *Старайтесь иметь четкое представление о состоянии проекта*. В проекте должны участвовать люди, способные объективно оценить готовность системы. Разработчики всегда более оптимистичны, чем тестировщики. Поощряйте плохие новости, и тогда снизится риск неожиданного провала в самом конце проекта. Самый лучший способ — полагаться на объективные данные о состоянии проекта в определенный период времени.
3. *Помните о правиле «железного треугольника»*. Существуют три взаимосвязанных компонента любого проекта: ресурсы (люди и деньги), функциональность и сроки. Изменение одного из них всегда влияет на остальные. Не забывайте, что эти условия можно выполнять параллельно.
4. *Старайтесь быть на виду*. При планировании надо разбивать работы на более мелкие, на выполнение которых требуется один или несколько дней. Тогда о том, что вы начинаете отставать, вы узнаете достаточно рано и сможете предпринять корректирующие шаги. Неделя для проекта разработки — это вечность. Каждый проект, который отстает на месяц, вначале отставал на один день. Узнайте об отставании прежде, чем наступит момент, когда исправить что-то будет уже невозможно. Несомненно, подобный стиль управления весьма трудоемок и периодически вас будут в нем обвинять. Но если вам удается довести до участников проекта понимание того, что его цель — предоставить готовый продукт в определенное время, ваши коллеги примут этот стиль.
5. *Используйте контрольные точки с отсутствием дефектов*. В каждом приложении можно выделить 20% функциональности, которая займет 80% времени; в свою очередь, в оставшихся 80% функциональности можно опять выделить 20% и т. д. Реализуйте сначала полностью первые 20% функций, протестируйте программу, разработайте документацию. И только когда все это будет готово, двигайтесь дальше. Использование контрольных точек позволяет быстро понять, какие части проекта могут стать проблемными, дают возможность сфокусироваться на достижении целей каждой контрольной точки.

6. *Бойтесь разработчиков, сидящих в башнях из слоновой кости*. Разработчики должны уметь работать в команде, демонстрировать свой прогресс, делиться опытом и проверять то, что уже сделано. Избегайте «примадонн», которые считают себя умнее всех. Несомненно, проект лишь выиграет, если в команде появится пара гениев или просто отличных специалистов, но еще лучше будет, если их талант признает и команда проекта. Для определения состава команды проекта может пригодиться правило, принятое в Microsoft: *шесть* разработчиков, *три* инженера по качеству, *один* менеджер, *два* технических писателя.
7. *Плохая дата — не просто плохая дата*. Обычно вы заранее знаете, что опоздаете, и настаиваете на смене даты, но с каждой отсрочкой теряете доверие клиента. Вот хорошее правило: перенос даты должен происходить только в тех случаях, когда точно известны все составляющие и причины задержки. Изменение сроков требует дополнительных ресурсов, поэтому следующая контрольная точка должна быть реалистичной. В противном случае подобный «проект» никогда не закончится.
8. *Сдвинув сроки, не проваливайте их*. Перенос срока — это симптом, который свидетельствует о выявлении «белых пятен». Если сдвиг сроков выполнения проекта стал неожиданностью — значит, используемые методы управления командой проекта надо менять. В то же время задержка имеет и положительный эффект — дает возможность понять причины отставания и перераспределить ресурсы. Поэтому, сдвигая сроки, внимательно анализируйте ошибки и старайтесь их больше не допускать.
9. *Чем проще — тем лучше*. Лучше обещать меньше, но сделать больше, чем пообещать больше и не выполнить. Для заказчика важнее результат, а не обещания. А для успеха проекта стоит выбирать максимально простые, но надежные решения.
10. *Время для проектирования — это время для проектирования*. Часто в погоне за красотой реализации, выбирается концепция проекта, совершенно не укладывающаяся в сроки, всегда выбирайте из альтернативных решений наименее рискованное и оптимальное по времени реализации. Помните, что время — весьма ограниченный ресурс, который дан для того, чтобы достичь успеха, а не удивить.
11. *Если вы не можете что-то собрать, значит, вы не сможете это выпустить*. Продукт должен постоянно собираться (компилироваться вместе со всеми компонентами системы, документацией и программой установки). Это необходимо потому, что с самого начала нужно определить все составляющие будущего продукта и адекватно оценивать степень их готовности. В противном случае вы обязательно что-нибудь забудете, и это станет весьма неприятной неожиданностью, особенно под окончание работ. Пусть на начальном этапе это будут скромные зачатки будущей системы — придет время, и из них вырастет отличный продукт. С другой стороны, промежуточная версия продукта — хороший способ продемонстрировать заказчику факт готовности того или иного фрагмента работы.

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

Раздел 2. «ЛУЧШИЙ ПРОДУКТ»

1. *Заказчик — это ваше все*. Старайтесь в каждой последующей версии учитывать даже весьма странные, нечетко выраженные пожелания клиентов. Часто в этих требованиях скрывается то, что делает продукт уникальным и отличимым от других, добавляет ему маркетинговой привлекательности и заставляет пользователей использовать его долгие годы. Помните, что заказчик лучше вас знает, что ему нужно.
2. *Двигайтесь правильным курсом*. Цели проекта — это основной результат вашей разработки. Коммерческий успех продукта основывается именно на ней, поэтому цели должны быть конкретными, измеримыми, согласованными, реальными, ограниченными по срокам. Старайтесь, как можно раньше сформулировать цели и сохранить веру в них вплоть до конца проекта.
3. *Будьте гибким*. Часто по ходу проекта требования к системе могут изменяться — будьте готовы к этому. Старайтесь постоянно проверять, насколько мнение пользователя соответствует поставленной цели. Используйте для этого промежуточные версии продукта, вовлекая заказчика в процесс работы с системой как можно раньше.
4. *Соблюдайте баланс*. Правильно расставьте акценты при реализации функциональных и нефункциональных требований. Ни в коем случае не увлекайтесь наращиванием какой-либо из функциональностей продукта, это станет причиной дискомфорта пользователей и может привести к серьезным проблемам со сроками реализации продукта.
5. *Развивайте продукт постепенно*. Правильное развитие выглядит так: ранние стадии разработки определяют более поздние, ошибки не должны повторяться, а результат отвечает потребностям конечного пользователя. Плавное развитие вселяет в вас ощущение предсказуемости и стабильности процесса разработки.
6. *Продукт — это иерархия компонентов*. Следуя этому принципу, отдельным компонентам проекта следует уделять внимание пропорционально их важности, что обеспечивает стабильность и сбалансированное развитие. Иерархию очень удобно использовать как скелет для постепенного расширения системы, сначала вы реализуете основу, а затем наполняете ее будущим содержимым; новые компоненты опираются на уже разработанные.
7. *Все должны разделять общее видение продукта*. Все члены команды должны знать, какие цели должны быть достигнуты, как продукт должен выглядеть, какова стратегия его разработки. Если в команде появляются желаю-

щие скорректировать цели, постарайтесь дать им слово и аргументировать свое видение, быть может, оно лишь улучшит будущий продукт. Все противоречия должны быть разрешены, а видение продукта приведено к единству.

Раздел 3. «ВЫПУСТИТЬ ТОЧНО В СРОК»

Ваша главная задача — выпустить продукт:

- команда обязана поставить продукт в срок, а все члены команды должны верить в то, что это возможно;
- каждый член команды должен понимать, что от него для этого требуется, а менеджер должен сделать все от него зависящее, чтобы иметь все шансы сделать то, что требуется;
- каждый член команды должен не просто хотеть, а гореть желанием достичь желаемого результата;
- выпуск продукта в срок должен стать целью каждого члена команды, самым ожидаемым событием для всех!

Контрольные вопросы по главе 6

1. Приведите и прокомментируйте схему взаимосвязи основных фаз ЖЦ программного проекта.
2. Раскройте назначение и содержание международного стандарта CMM — модель зрелости процессов разработки программного обеспечения.
3. Поясните технологию и раскройте содержание и представление структурной декомпозиции работ программного проекта.
4. Приведите содержательную и математическую постановку задачи формирования календарного плана проекта.
5. Раскройте содержание и приведите пример алгоритма определения критического пути.
6. Раскройте содержание и приведите пример алгоритма выравнивания ресурсов.
7. Раскройте содержание правил «Выпустить», «Лучший проект», «Выпустить точно в срок» при реализации программного проекта.

Глава 7

УПРАВЛЕНИЕ РИСКАМИ ПРОГРАММНОГО ПРОЕКТА

7.1 Основные понятия риска и рискообразующих факторов

В литературе под риском проекта понимается событие или условие, которое может произойти либо не произойти в будущем и оказать негативное влияние на цели проекта. Нельзя сказать заранее, произойдет ли это событие, но точно известно, что его возникновение скажется на функционале, качестве, стоимости или сроках выполнения проекта.

В [25] предлагается рассматривать риск с трех точек зрения: риск как возможность угрозы бизнесу; риск как негативное событие, не позволяющее достичь в полной мере цели проекта; риск как неопределенность между возникающими неблагоприятными ситуациями и возможными действиями по их устранению. В данном случае, в дальнейшем, под *риском* будем понимать наступление события, которое может возникнуть в процессе реализации программного проекта и негативно повлиять на степень достижения одной или нескольких характеристик целей проекта: содержания, стоимости, сроков, качества.

Учитывая явную логическую взаимосвязь между целями проекта и возможными рисками можно предположить, что при разработке программного проекта могут возникнуть четыре типа (категории) рисков:

- срыв плановых сроков проекта;
- превышение стоимости (бюджета) проекта;
- критические отклонения по составу и содержанию проекта (невыполнение функциональных требований);
- критическое отклонение по показателям качества проекта (невыполнение нефункциональных требований).

В табл. 7.1 приведены возможные для каждого из этапов ЖЦ ПП варианты формулировок целей и рисков проекта.

Таблица 7.1 – Соотношение целей и рисков программного проекта

Этапы ЖЦ	Цели	Риски
Инициализация	Выбрать продуктово-рыночное направление и разработать концепцию будущего коммерческого ПП	Ошибки в выборе функционала и как следствие невостребованность ПП, нарушение сроков окупаемости проекта
Разработка	Разработать коммерческий ПП с требуемым функционалом при ограничениях на сроки и бюджет проекта	Критические отклонения по срокам и бюджету проекта
Продвижение	Обеспечить в определенном интервале времени заданный уровень 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	Обеспечить процесс 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* [25].

Для стандартизации и унификации терминологии и формулировок рискообразующих факторов, оценки их влияния на конкретные цели и отдельные фазы программного проекта необходима их систематизация и классификация по определенным признакам. В данном случае для классификации факторов риска будем использовать иерархический метод классификации, при котором множество рискообразующих факторов последовательно, в соответствии с выбранными основаниями (признаками) классификации, разбивается на подмножества (рис. 7.1) [26].

Рис. 7.1 – Классификация факторов риска программного проекта

На первом уровне классификатора в качестве основания классификации используется модель жизненного цикла программного проекта: инициализация — разработка — продвижение — внедрение.

На втором уровне для каждого из этапов 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 обуславливается как политикой государства в отношении бизнеса малых IT-компаний, так и различными ситуациями на рынках. 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, 11, 25, 27–29] в табл. 7.2 и 7.3 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.

Таблица 7.2 – Состав внутренних первичных факторов риска программных проектов

Основания классификации	**Первичные 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 существующими у заказчиков информационными системами
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. Технологии 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 исполнителями новых инструментальных средств разработки ПП
4. Технология управления ПП	1. Отсутствие у разработчика эффективной методологии управления ПП. 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.

продолжение на следующей странице

Таблица 7.2 — Продолжение

Основания классификации	Первичные факторы риска
	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

Таблица 7.3 – Состав внешних первичных факторов риска программных проектов

Основания классификации	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
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.

продолжение на следующей странице

Таблица 7.3 — Продолжение

Основания классификации	Первичные факторы риска
	5. Несоответствие нефункциональных характеристик ПП имеющимся у потребителей программно-аппаратным средствам и коммуникациям. 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

7.2 Управления рисками: идентификация

Процесс управления рисками включает следующие логически взаимосвязанные этапы: *идентификацию рисков, анализ рисков, планирование рисков, мониторинг и управление рисками* [8]. Необходимо отметить, что описанные этапы являются логическими шагами и не обязательно должны следовать друг за другом в строгом хронологическом порядке.

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 каждого из факторов риска следует проводить на естественном языке с разъяснением причинно-следственной связи между реально существующей

причиной и потенциально возможным, еще не случившимся событием или ситуацией (рис. 7.2) [1]. *Условие* содержит описание причины, которая может сделать результат 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.

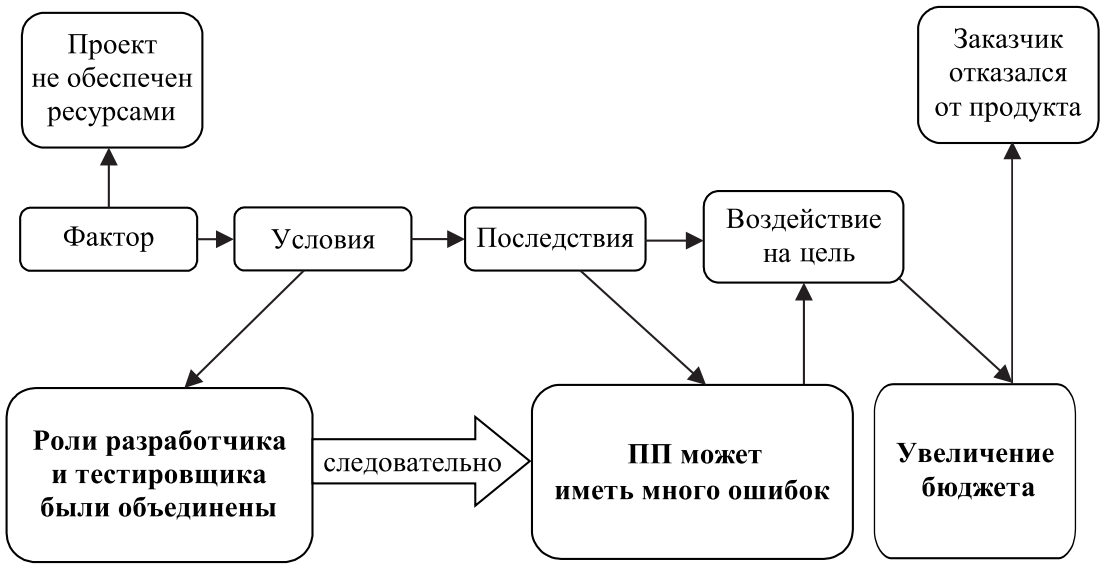


Рис. 7.2 – Основные характеристики риска и их взаимосвязи

Последовательность действий команды проекта по выявлению и описанию рисков может быть определена на основе предложенного 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. 7.4).

Таблица 7.4 – Фрагмент описания схемы рискообразующих факторов

Факторы	**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
продолжение на следующей странице			

Таблица 7.4 — Продолжение

Факторы	Условие	Последствия	Воздействие на цели
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

7.3 Управления рисками: анализ

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 на них;
- вероятность достижения целей проекта.

Термин «*вероятность*» означает меру возможности того, что появление риска, описанное в его формулировке, действительно наступит и определяется, как соотношение количества проектов, когда рискообразующие факторы имели место к общему числу проектов. 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 из целей проекта: стоимости, времени, содержания и качества.

Оценка может производиться на основании результатов опросов или совместных совещаний с экспертами. В число опрашиваемых могут входить члены команды проекта, а также специалисты, имеющие широкие познания в этой области.

Значения показателей вероятности и воздействия могут оцениваться экспертами 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) [11]. 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 следующей формуле:

$$P(x_j) = \frac{p_1(x_j) + 4p_2(x_j) + p_3(x_j)}{6}, \qquad (7.1)$$

где $p_1(x_j)$, $p_2(x_j)$, $p_3(x_j)$ — соответственно оптимистическая, пессимистическая и реалистическая вероятности наступления фактора.

Однако учитывая сформулированные в разделе 1 особенности ПП, оценки рискообразующих факторов не всегда можно описать с помощью числовых значений. В этом случае для этих целей необходимо использовать качественную шкалу с несколькими градациями, например <низкая, средняя, высокая> вероятности. Кроме того, в отсутствие 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, а в качестве инструмента задания значений характеристик использовать один из видов функций принадлежности, например *треугольную функцию* как наиболее простое и часто используемое на практике математическое выражение (рис. 7.3) [30].

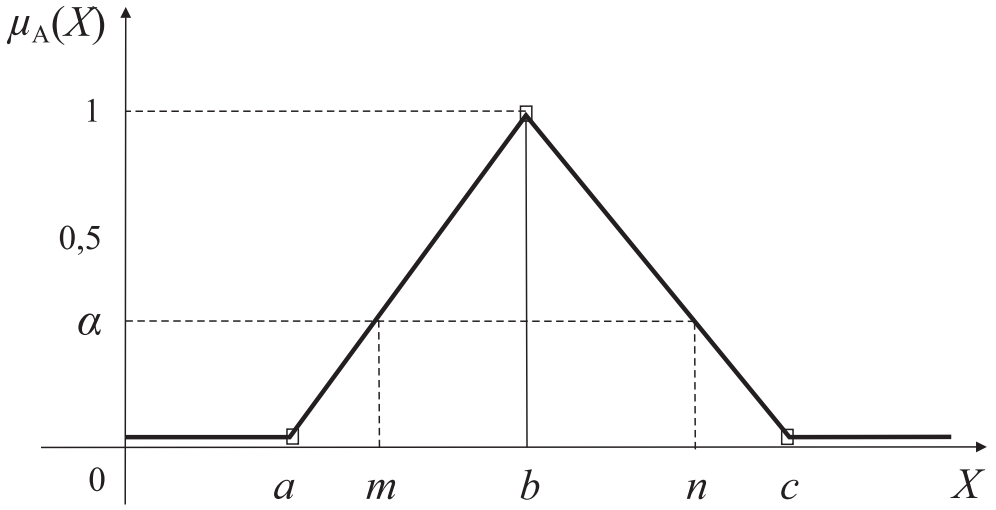


Рис. 7.3 – Вид треугольной функции принадлежности

Математически треугольный вид функции принадлежности можно описать в следующем виде: $P_1 = (m_1, n_1) = \left(a_1 + \alpha \cdot (b_1 - a_1),\ c_1 + \alpha \cdot (b_1 - c_1)\right)$, где a — минимальное, b — модальное и c — максимальное, значения оцениваемых характеристик,

что соответствуют пессимистическому, базовому и оптимистическому вариантам развития сценария; $m = a + \alpha \cdot (b - a)$, а $n = c + \alpha \cdot (b - c)$.

Варианты оценок рискообразующих факторов и их воздействие на цели проекта представлены в табл. 7.5 [25]. Интервалы оценок вероятностей и воздействий в виду присутствия неопределенности могут пересекаться.

Таблица 7.5 – Оценки рискообразующих факторов

	Шкалы оценивания				
Вероятность	Очень низкая	Низкая	Умеренная	Высокая	Очень высокая
Воздействие	Незначительное	Умеренное	Высокое	Критичное	Катастрофическое
Оценка Цель	< 0.15	0.1–0.4	0.2–0.6	0.5–0.9	> 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

По результатам идентификации и анализа описанных выше показателей необходимо выделить множества 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. 7.6) [25]. 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. Для определения степени критичности влияния фактора целесообразно ввести пороговое значение критичности (например, > 0.33).

Таблица 7.6 – Матрица вероятности и воздействий

Воздействие **Вероятность**	< 0.15	0.1–0.4	0.2–0.6	0.5–0.9	> 0.8
> 0.8	0–0.15	0.08–0.4	0.16–0.6	0.4–0.9	0.64–1
0.5–0.9	0–0.135	0.05–0.36	0.1–0.54	0.25–0.81	0.4–0.9
0.2–0.6	0–0.09	0.02–0.24	0.04–0.36	0.1–0.54	0.16–0.6
0.1–0.4	0–0.06	0.01–0.16	0.02–0.24	0.05–0.36	0.08–0.4
< 0.15	0–0.0225	0–0.06	0–0.09	0–0.135	0–0.15

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. 7.7.

Таблица 7.7 – 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

Интегральная оценка критичности и характеристика близости наступления рискообразующего фактора являются основой для их ранжирования. Ранг определяет его порядковый номер в полной совокупности рисков проекта. Чем выше ранг, тем более опасен риск (табл. 7.8).

Таблица 7.8 – 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. 7.9.

Таблица 7.9 – 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: Критично
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 следующим категориям опасности последствий [8]:

- 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.

Влияние *отдельных групп рискообразующих факторов на цели проекта*, определение *вероятности достижения целей проекта* и категории итогового риска можно определить на основе модели функциональных зависимостей. В формализованном виде модель сети функциональных зависимостей можно представить в виде ориентированного графа $G = (X, U)$, где $X = \{x_i\}$ — множество вершин графа представлено набором атрибутов, описывающих различные характеристики объекта моделирования, $U = \{u_{ij}\}$ — множество направленных дуг графа, показывающих взаимосвязь между атрибутами [31].

В истоках графа находятся первичные атрибуты, характеризующие объект моделирования и значение которых не зависит от других атрибутов. В стоке сети расположен результирующий (целевой) атрибут. Значения остальных атрибутов зависят от первичных и вычисляются через совокупность функциональных зависимостей.

С учетом вышеизложенного и предложенной системы классификации первичных рискообразующих факторов структуру модели сети функциональных зависимостей определения количественных оценок риска можно представить в виде следующего ориентированного графа (рис. 7.4).

В первом слое модели описываются первичные рискообразующие факторы, на втором, третьем и четвертом слоях вычисляются промежуточные риски программного проекта, в последнем слое определяется итоговая оценка риска.

Вероятность наступления промежуточного и итогового рисков определяется по двум выражениям:

1) вероятность наступления x_i-го риска есть некоторая функция первичных факторов, равная сумме вероятностей независимых событий, в которых реализуется хотя бы одни из факторов: $P(x_i) = 1 - \prod_{j=1}^{n} \overline{p(x_j)}$;
2) вероятность наступления x_i-го риска, которая представляет собой некоторую функцию, равную произведению вероятностей независимых событий, при условии, что все факторы имеют место: $P(x_i) = \prod_{j=1}^{n} p(x_j)$.

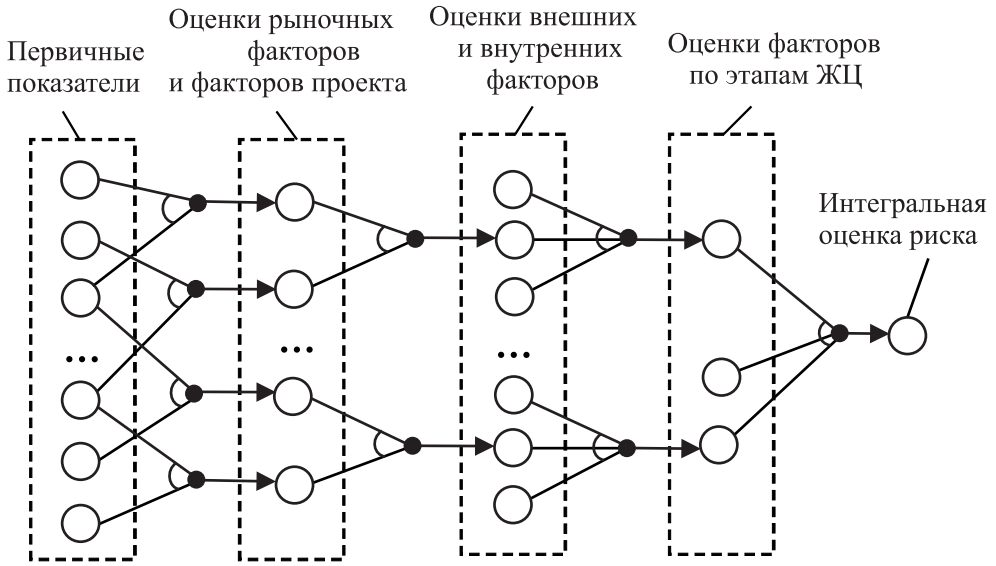


Рис. 7.4 – Структура модели оценки риска программного проекта

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 $x_i \in \Gamma^{-1} x_j$, $j = 1, \ldots, n$.

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 риска программный проект может быть отнесен к следующим категориям: высокорискованный проект — вероятность > 0.8; рискованный проект — вероятность > 0.6; среднерискованный проект — вероятность > 0.4; низкорискованный проект — вероятность > 0.2.

7.4 Управления рисками: планирование мероприятий по реагированию на риски

Процессы планирования реагирования на риски предполагают выбор стратегии по снижению угроз для каждой из целей проекта и разработку планов мероприятий по реализации стратегий. Согласно [8] возможны три вида таких стратегий: уклонение от риска, передача риска, снижение рисков.

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, нежели действия по устранению негативных последствий, предпринимаемые после наступления события.

Одним из возможных математических аппаратов принятия решений по выбору стратегии снижения угроз является математический 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 утвержденный 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 включения 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.

7.5 Управления рисками: мониторинг и управление рисками

Мониторинг рисков — это процесс наблюдения и контроля за ходом исполнения принятых в отношении рисков планов и инициирование изменений в проекте. Другими словами, *мониторинг и управление рисками* — это процесс идентификации, анализа и планирования реагирования на новые риски, отслеживания ранее идентифицированных рисков, а также проверка и исполнение мероприятий реагирования на риски и оценка эффективности этих выполнений [8]. Мониторинг и управление рисками включает в себя 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 предполагает изучение и предоставление в документальном виде результатов оценки эффективности выполнения мероприятий по реагированию на риски, требующие немедленного реагирования, изучение 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 по главе 7

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.

ЗАКЛЮЧЕНИЕ

В настоящее время существует большое количество учебников и учебных пособий по управлению программными проектами, в том числе и в электронном виде. Поэтому с учётом количества академических часов, выделенных для изучения дисциплины, в данном учебном пособии кратко излагаются базовые знания, необходимые и достаточные студенту для понимания проблемы.

Отдельные разделы более подробно рассматриваются в других учебных курсах: метод экспертных оценок, метод сетевого планирования в курсе «Исследование операции»; вопросы определения трудозатрат и стоимости проекта в курсе «Экономика программных проектов»; методы структурной декомпозиции работ в курсах «Теория систем и системный анализ» и «Проектирование и архитектура программных систем». Для самостоятельного изучения вынесены вопросы освоения пакетов прикладных программ по управлению проектами. На данный момент времени рынок ПП по управлению проектами представлен следующими пакетами программ: Microsoft Project, Suretrak Primavera Systems, Spider Project Lite, Primavera Project Planner (P3), Primavera Project Planner Enterprise (P3e), Open Plan Professional, Spider Project Professional, Scitor Project Scheduler, Artemis Project View.

ЛИТЕРАТУРА

[1] Архипенков С. Я. Лекции по управлению программными проектами [Персональный сайт]. — [М., 2009]. — URL: http://www.arkhipenkov.ru/resources/sw_project_management.pdf.

[2] Уэбстер Ф. Основы промышленного маркетинга / Ф. Уэбстер. — М. : Издательский дом Гребенникова, 2005. — 416 с.

[3] Ехлаков Ю. П. Информационные технологии и программные продукты: рынок, экономика, нормативно-правовое регулирование : учеб. пособие / Ю. П. Ехлаков. — Томск : Томский государственный университет систем управления и радиоэлектроники, 2007. — 176 с.

[4] Ехлаков Ю. П. Функциональные модели и организационно-правовые механизмы продвижения прикладных программных продуктов на рынок корпоративных продаж : монография / Ю. П. Ехлаков, А. А. Ефимов. — Томск : Изд-во Томск. гос. ун-та систем управления и радиоэлектроники, 2010. — 172 с.

[5] Кознов Д. В. Введение в программную инженерию [Электронный ресурс] / Д. В. Кознов. — СПб. : Изд-во Санкт-Петербургского ун-та, 2008. — Ч. 1. — URL : http://bookfi.org/book/805204.

[6] Лаврищева Е. М. Методы и средства инженерии программного обеспечения : учебник [Электронный ресурс] / Е. М. Лаврищева, В. А. Петрухин. — М. : Изд-во МФТИ, 2006. — 304 с. — URL: http://bookfi.org/book/809755.

[7] Новиков Ф. А. Управление проектами и разработкой ПО : учеб.-метод. пособие по дисциплине [Электронный ресурс] / Ф. А. Новиков, Э. А. Опалева, Е. О. Степанов. — СПб. : СПбГУ ИТМО, 2008. — 256 с. — URL : http://books.ifmo.ru/file/pdf/430.pdf.

[8] Руководство к своду знаний по управлению проектами (PMBOK). — 4-е изд. — М. : Project Management Institute, 2010. — 496 с.

[9] Руководство к своду знаний по программной инженерии / SWEBOK [Электронный ресурс]. — [2004]. — URL : sorlik.blogspot.com.

[10] ГОСТ Р ИСО/МЭК 12207–99. Информационная технология. Процессы жизненного цикла программных средств [Электронный ресурс]. — URL: http://vsegost.com/Catalog/38/38119.shtml.

[11] Фатрелл Роберт Т. Управление программными проектами. Достижение оптимального качества при минимуме затрат / Роберт Т. Фатрелл, Дональд Ф. Шафер, Линда И. Шафер. — М. : Издательский дом «Вильямс», 2004. — 1136 с.

[12] Карпенко С. Н. Введение в программную инженерию : учеб.-метод. пособие С. Н. Карпенко. — Н. Новгород : Изд-во ННГУ, 2007. — 103 с.

[13] Управление проектами: технология MSF [Электронный ресурс]. — URL: http://www.microsoft.com/rus/msdn/msf/default.mspx.

[14] Салливан Эд. Время — деньги. Создание команды разработчиков программного обеспечения : пер. с англ. / Эд. Салливан. — М. : Русская редакция, 2002. — 364 с.

[15] Ехлаков Ю. П. Организация бизнеса на рынке программных продуктов : учебник / Ю. П. Ехлаков. — Томск : Изд-во Томск. гос. ун-т систем управления и радиоэлектроники, 2012. — 312 с.

[16] Архипенков С. Я. Руководство командой разработчиков программного обеспечения. Прикладные мысли [Персональный сайт]. — [М., 2008]. — URL: http://www.arkhipenkov.ru/resources/sw_project_management.pdf.

[17] Эинк Э. Бизнес для программистов. Как начать свое дело / Э. Эинк. — СПб. : Питер, 2008. — 256 с.

[18] Ехлаков Ю. П. Теоретические основы автоматизированного управления : учебник / Ю. П. Ехлаков. — Томск : Изд-во Томск. гос. ун-та систем управления и радиоэлектроники, 2001. — 337 с.

[19] 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.

[20] Янченко Е. А. Оценка перспективности концепции рыночного программного продукта / Е. А. Янченко, Н. Ю. Хабибулина // Доклады ТУСУРа. — 2013. — 3 (29). — С. 141–151.

[21] Танаев В. С. Введение в теорию расписаний / В. С. Танаев, В. В. Шкурба. — М. : Наука, 1975. — 256 с.

[22] Липаев В. В. Экономика производства программных продуктов / В. В. Липаев. — 2-е изд. — М. : СИНТЕГ, 2011. — 352 с.

[23] Математические основы управления проектами : учеб. пособие / С. А. Баркалов [и др.] ; под ред. В. Н. Буркова — М. : Высшая школа, 2005. — 423 с.

[24] Смольянинов А. Некоторые секреты командной разработки / А. Смольянинов, А. Ложечкин // Открытые системы. — 2005. — №№7, 8. — С. 49–56.

[25] Авдошин С. М. Информатизация бизнеса. Управление рисками / С. М. Авдошин, Е. Ю. Песоцкая. — М. : ДМК Пресс, 2011. — 176 с.

[26] Ехлаков Ю. П. Классификация и описание рискообразующих факторов при создании программных продуктов // Доклады ТУСУРа. — 2013. — 4 (30). — С. 142–147.

[27] Липаев В. В. Анализ и сокращение рисков проектов сложных программных средств / В. В. Липаев. — М. : СИНТЕГ, 2003. — 224 с.

[28] Шапкин А. С. Экономические и финансовые риски / А. С. Шапкин, В. А. Шапкин. — М. : Дашков и К°, 2008. — 543 с.

[29] Фатхутдинов Р. А. Инновационный менеджмент : учебник для вузов / Р. А. Фатхутдинов. — 6-е изд. — СПб. : Питер, 2008. — 448 с.

[30] Леоненков А. Нечеткое моделирование в среде MATLAB и fuzzyTech / А. Леоненков. — СПб. : БХВ-Петербург, 2003. — 736 с.

[31] 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.

Приложение А

ПАРАМЕТРЫ И ФУНКЦИОНАЛЬНЫЕ ЗАВИСИМОСТИ ГИБРИДНОЙ МОДЕЛИ

1 слой — истоки		
Параметр	**Наименование параметра**	**Значение параметра единицы измерения**
x_1	Затраты на разработку ПП	> 0 тыс. руб.
x_2	Средная величина оплаты труда разработчика (в час)	> 0 тыс. руб.
x_3	Стоимость аппаратного оснащения	$\geqslant$ 0 тыс. руб.
x_4	Стоимость ПП	$\geqslant$ 0 тыс. руб.
x_5	Стоимость обучения	$\geqslant$ 0 тыс. руб.
x_6	Структура процесса	Высокий (1); средний (0.5); простая (0)
x_7	Необходимость дополнительного привлечения специалистов в команду проекта	Да — 1, Нет — 0
x_8	Необходимость привлечения соисполнителей	Да — 1, Нет — 0
x_9	Наличие разработанных компонентов и модулей	Да — 1, Нет — 0
x_{10}	Уровень риска проекта	Высокий (1); средний (0.5); низкий (0)
x_{11}	Количество конкурентов	$\geqslant$ 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

продолжение на следующей странице

1 слой — истоки		
Параметр	**Наименование параметра**	**Значение параметра единицы измерения**
x_{15}	Число потенциальных потребителей	> 0
x_{16}	Уровень платежеспособного спроса	Высокий (1); средний (0.5); низкий (0)
x_{17}	Приведенная стоимость ПП	> 0 тыс. руб.
x_{18}	Стоимость годовой эксплуатации	⩾ 0 тыс. руб.
x_{19}	ФОТ IT-подразделения	⩾ 0 тыс. руб.
x_{20}	Стоимость общесистемного ПО	⩾ 0 тыс. руб.
x_{21}	Стоимость аппаратного переоснащения	⩾ 0 тыс. руб.
x_{22}	Сокращение составляющих издержек по реализации бизнес-процессов	> 25% — «Высокая» (1); ⩾ 5%, но ⩽ 25% — «Средняя» (0.5); < 5% — «Низкая» (0)
x_{23}	Сокращение временных потерь	> 25% — «Высокая» (1); ⩾ 5%, но ⩽ 25% — «Средняя» (0.5); < 5% — «Низкая» (0)
x_{24}	Увеличение объемов продаж	> 19% — «Высокая» (1); ⩾ 3%, но ⩽ 10% — «Средняя» (0.5); < 3% — «Низкая» (0)
x_{25}	Функциональные возможности	Высокий (1); средний (0.5); низкий (0)
x_{26}	Надежность	Высокая (1); средняя (0.5); низкая (0)
x_{27}	Практичность	Высокая (1); средняя (0.5); низкая (0)
x_{28}	Эффективность	Высокая (1); средняя (0.5); низкая (0)
x_{29}	Сопровождаемость	Высокая (1); средняя (0.5); низкая (0)
x_{30}	Мобильность	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

продолжение на следующей странице

1 слой — истоки		
Параметр	**Наименование параметра**	**Значение параметра единицы измерения**
x_{34}	Дисконтированный период окупаемости (DPV)	DPV < 12 мес. — «прибыльный» (1); DPV ⩾ 12, но < 36 — «ни прибыльный, ни убыточный» (0.5); DPV > 36 — «убыточный» (0)
x_{35}	Наличие государственной поддержки отрасли	Да — 1, Нет — 0
x_{36}	Наличие входных барьеров	Да — 1, Нет — 0
x_{37}	Уровень развития ИКТ	Высокий (1); средний (0.5); низкий (0)

2 слой			
Параметр	**Наименование параметра**	**Значение параметра**	**Способ определения значений параметра**
x_{38}	Затраты на разработку	тыс. руб.	$x_1 \cdot x_2 + x_3 + x_4 + x_5$
x_{39}	Готовность команды	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 $x_{39} = 1$.

продолжение на следующей странице

2 слой			
Параметр	**Наименование параметра**	**Значение параметра**	**Способ определения значений параметра**
			Если $x_6 = 1$ и $x_7 = 0$ и $x_8 = 0$ и $x_9 = 1$, то $x_{39} = 1$. 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	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 и $x_{40} = 0$. 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 $x_{13} = 0$ 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 $x_{13} = 0$, то $x_{40} = 0$. 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}	Покупательная способность	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 $x_{16} = 1$ или $x_{16} = 0$, то $x_{41} = 1$. Если $x_{14} = 0.5$ и $x_{15} = 0$ или $x_{15} = 0.5$ и $x_{16} = 0$ или $x_{16} = 0$, то $x_{41} = 0$.

продолжение на следующей странице

2 слой			
Параметр	**Наименование параметра**	**Значение параметра**	**Способ определения значений параметра**
			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}	Совокупная стоимость владения	«Высокая» $>$ $>$ 200000 — (1); «Средняя» $\geqslant$ $\geqslant$ 50000, но $\leqslant$ $\leqslant$ 200000 — (0.5); «Низкая» $<$ $<$ 50000 — (0)	$x_{17} + x_{18} + x_{19} + x_{20} + x_{21}$
x_{43}	Степень повышения деятельности организации	«Высокий» $>$ 2 — (1); «Средний» $\geqslant$ 1, но $\leqslant$ — (0.5); «Низкий» $<$ 1 — (0)	$x_{22} + x_{23} + x_{24}$

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 $x_{44} = 1$.
продолжение на следующей странице			

3 слой			
Параметр	**Наименование параметра**	**Значение параметра**	**Способ определения значений параметра**
			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 $x_{39} = 1$ 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 привлекательность	«Высокий» $> 2 — (1)$; «Средний» $\geqslant 1$, но $\leqslant 2 — (0.5)$; «Низкий» $< 1 — (0)$	$x_{31} + x_{32} + x_{33} + x_{34}$
x_{46}	Привлекательность рыночной микросреды	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 $x_{42} = 0$ или $x_{43} = 1$, то $x_{49} = 1$. Если $x_{42} = 1$ и $x_{43} = 0$, то и $x_{49} = 0.5$. 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 $x_{49} = 0$
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 $x_{47} = 0.5$.

продолжение на следующей странице

3 слой			
Параметр	**Наименование параметра**	**Значение параметра**	**Способ определения значений параметра**
			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}	Технический уровень	«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 и $x_{49} = 0.5$. Если $x_{42} = 0.5$ или $x_{42} = 0$ и $x_{43} = 0.5$, то $x_{49} = 0$

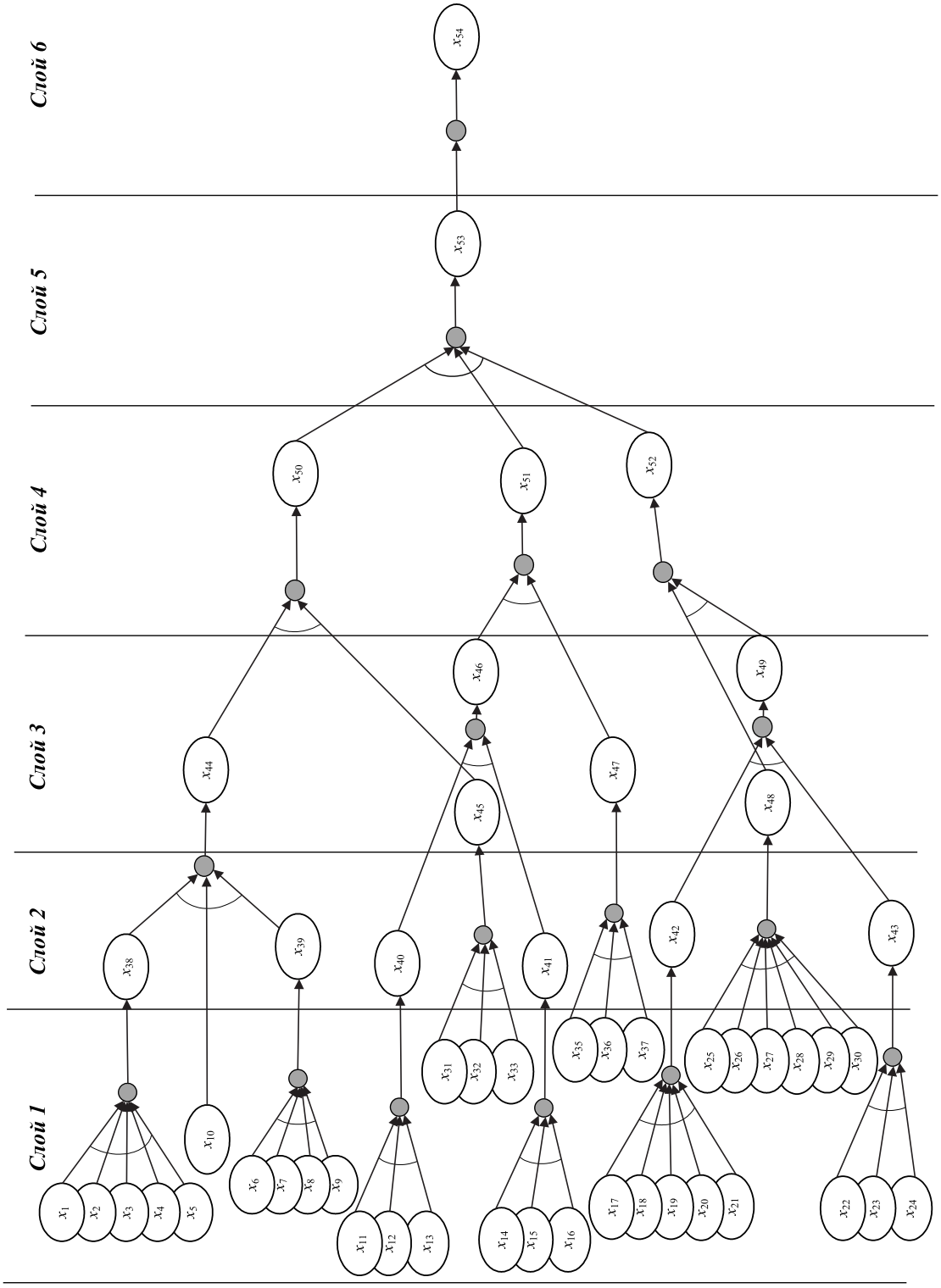
4 слой			
Параметр	**Наименование параметра**	**Значение параметра**	**Способ определения значений параметра**
x_{50}	Перспективность проекта	Высокая (1); средняя (0.5); низкая (0)	Если $x_{44} = 1$ или $x_{44} = 0.5$ и $x_{45} = 1$, то $x_{50} = 1$. Если $x_{44} = 1$ или $x_{44} = 0.5$ и $x_{45} = 0.5$, 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}	Перспективность продукта	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 $x_{50} = 0$.

продолжение на следующей странице

4 слой			
Параметр	**Наименование параметра**	**Значение параметра**	**Способ определения значений параметра**
			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 $x_{44} = 0.5$ и $x_{45} = 0$, то $x_{50} = 0$. Если $x_{44} = 0$ и $x_{45} = 1$, то $x_{50} = 0.5$

5 слой			
Параметр	**Наименование параметра**	**Значение параметра**	**Способ определения значений параметра**
x_{53}	Оценка перспективности концепции	$a — [0; 1]$	$a_1x_{50} + a_2x_{51} + a_3x_{52}$

6 слой			
Параметр	**Наименование параметра**	**Значение параметра**	**Способ определения значений параметра**
x_{54}	Перспективность концепции	«очень высокая»; «высокая»; «средняя»; «ниже среднего»; «низкая»	Если $0.8 \leqslant x_{53} \leqslant 1$, то x_{54} = «очень высокая». Если $0.6 \leqslant x_{53} \leqslant 0.8$, то x_{54} = «высокая». Если $0.4 \leqslant x_{53} \leqslant 0.6$, то x_{54} = «средняя». Если $0.2 \leqslant x_{53} \leqslant 0.4$, то x_{54} = «ниже среднего». Если $0 \leqslant x_{53} \leqslant 0.2$, то x_{54} = «низкая»


Слой 6
Слой 5
Слой 4
Слой 3
Слой 2
Слой 1
x54
x53
x50
x51
x52
x46
x49
x44
x47
x45
x48
x38
x39
x40
x41
x42
x43
x1
x2
x3
x4
x5
x10
x6
x7
x8
x9
x11
x12
x13
x31
x32
x33
x14
x15
x16
x35
x36
x37
x17
x18
x19
x20
x21
x25
x26
x27
x28
x29
x30
x22
x23
x24

Учебное издание

Ехлаков Юрий Поликарпович

УПРАВЛЕНИЕ
ПРОГРАММНЫМИ ПРОЕКТАМИ

Учебное пособие

Корректор Осипова Е. А.
Компьютерная верстка Насынова Н. Е.

Подписано в печать 31.03.14. Формат 60x84/8.
Усл. печ. л. 16,28. Тираж 200 экз. Заказ

Издано в ООО «Эль Контент»
634029, г. Томск, ул. Кузнецова д. 11 оф. 17
Отпечатано в Томском государственном университете
систем управления и радиоэлектроники.
634050, г. Томск, пр. Ленина, 40
Тел. (3822) 533018.