ГЛАВА 1. ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММЫХ СРЕДСТВ

Развитие информационных технологий во второй половине XX в. привело к увеличению сложности задач, возлагаемых на программные средства (ПС). Однако отсутствие адекватных методов не позволяло создавать ПС с необходимыми характеристиками. Период конца 60-х – начала 70-х гг. XX в. вошел в историю как время кризиса в области разработки ПС (software crisis). Кризис выражался в том, что сроки выполнения проектов постоянно срывались, стоимость разработки ПС значительно превышала запланированный бюджет, при этом качество ПС оставалось на низком уровне. Некоторые проекты вообще не доводились до завершения.

По данным Standish Group уже к середине 90-х г. кризис принял «хронические» формы и выражался в следующих цифрах:

– США тратит ежегодно более $200 млрд на более чем 170 тыс. проектов разработки ПС в сфере информационных технологий;

– 31,1 % из них закрываются, так и не завершившись;

– 52,7 % проектов завершаются с превышением первоначальных оценок бюджета, сроков и с ограниченной функциональностью;

– потери от недополученного эффекта внедрения ПС измеряются триллионами.

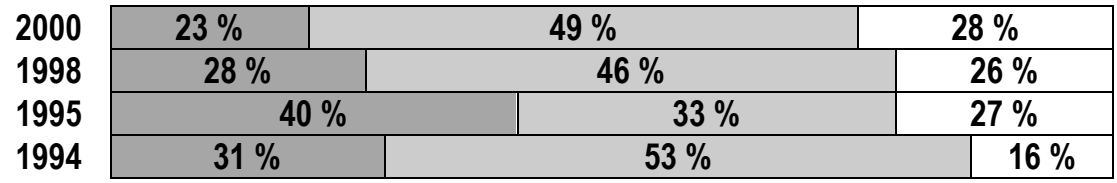


Рис. 1.1. Результаты анализа проектов:

– проваленные проекты; – проблемные проекты; – успешные проекты

Компания Standish Group проанализировала итоги выполнения порядка 30 тыс. проектов, связанных с разработкой ПС, выполненных в американских компаниях в 1994 – 2000 гг. На рис. 1.1 представлены результаты анализа, показывающие соотношение между успешными, проблемными и проваленными проектами. Успешными проектами считались проекты, завершенные вовремя, в рамках бюджета и с выполнением всех запланированных требований. В проблемных проектах зафиксировано: нарушение сроков, перерасход

бюджета и частичное выполнение требований. Проваленные проекты не были доведены до конца из-за значительного превышения сроков разработки, перерасхода бюджета и плохого качества.

Организации-разработчики ПС, осознав возможность остаться без заказов, направили усилия на поиск эффективных методов для решения возникших проблем.

В октябре 1968 г. состоялась конференция подразделения НАТО по науке и технике (г. Гармиш, Германия), на которой присутствовало 50 профессиональных разработчиков ПС из 11 стран. Рассматривались проблемы проектирования, разработки, распространения и поддержки ПС. На этой конференции впервые прозвучал термин software engineering – программная инженерия – как дисциплина, которую необходимо создавать и которой нужно руководствоваться в решении перечисленных проблем. В 1975 г. в Вашингтоне была проведена первая международная конференция, посвященная программной инженерии.

Почти два десятилетия обещаний поднять производительность и качество работ за счет новых методов и средств разработки ПС ушло на осознание того, что причина «хронического кризиса разработки ПС» состоит не столько в отсутствии совершенных методологий, сколько в неспособности организаций управлять технологическим процессом разработки ПС. Для усовершенствования принципов и методов разработки ПС обратились к опыту промышленного проектирования и производства, где был накоплен опыт успешной разработки не менее сложных проектов.

Понятие программной инженерии (software engineering) имеет два важных аспекта (по аналогии с понятием технологии) [3, 22]:

1) *программная инженерия* – это совокупность инженерных методов применяемых на протяжении жизненного цикла ПС;

2) *программная инженерия* – это инженерная дисциплина, охватывающая все аспекты жизненного цикла ПС от начальной стадии разработки требований до завершения использования ПС.

Программная инженерия – это совокупность инженерных методов. Инженеры решают конкретные практические задачи. Если соответствующие методы и подходы существуют, они адаптируют их к решению конкретной проблемы, стремясь достичь максимальной эффективности и качества ПС. Если подходящего метода не существует (проблема неразрешима в рамках существующих теорий) инженер методом проб и ошибок подбирает приемлемое решение для данной конкретной ситуации, применяет его и несет ответственность

за результат. Набор таких инженерных методов, теоретически пока не обоснованных, но получивших неоднократное подтверждение на практике, имеет большое практическое значение. В программной инженерии они получили название лучших практик (*best practices*).

Инженеры ищут решение поставленной перед ними задачи с учетом условий заключенных контрактов. Контракты устанавливают временные, финансовые и организационные (оборудование, люди и др.) рамки на работу инженеров.

Программная инженерия охватывает все аспекты жизненного цикла ПС. Программная инженерия занимается не только техническими аспектами производства ПС (такими инженерными процессами как проектирование, кодирование и т.д.), ее задачам являются:

– управление проектами (планирование, распределение, контроль ресурсов и др.);

– разработка теоретических основ и методов для обеспечения жизненного цикла ПС.

Программная инженерия является быстро изменяющейся областью деятельности. Она прошла несколько этапов развития, в процессе которых сформировались фундаментальные принципы и методы разработки ПС.

1.1. Понятие жизненного цикла ПС

Понятие жизненного цикла ПС занимает центральное место в программной инженерии. Оно образует базу для естественной систематизации методов, ресурсов и результатов на разных этапах разработки, использования и сопровождения ПС.

Понятие жизненного цикла не является специфическим для программной инженерии.

Термином *жизненный цикл* принято отражать совокупность процессов и этапов развития организмов живой природы, технических систем, продуктов производства от момента зарождения или появления потребности их создания и использования до прекращения функционирования или применения. Это соответствует всеобщему закону развития любых изделий, событий или процессов между их началом и концом, которые определяют цикл их создания, существования и применения [24].

После октябрьской конференции 1968 г. в Лондоне состоялась встреча 22-х руководителей проектов по разработке ПС, на которой анализировались те же проблемы проектирования, разработки, рас-

пространения и поддержки ПС. На этой встрече была предложена концепция жизненного цикла ПС (SLC – Software Lifetime Cycle) как последовательности шагов-стадий, которые необходимо выполнить в процессе создания и эксплуатации ПС.

В 1970 г. У.У. Ройс (W.W. Royce) выделил несколько этапов в типичном жизненного цикле ПС и было высказано предположение, что контроль выполнения этапов приведет к повышению качества программных средств и сокращению стоимости их разработки.

С развитием промышленного производства ПС сформировалось понятие жизненного цикла ПС.

Жизненный цикл ПС (software life cycle) – период, который начинается с момента принятия решения о необходимости создания программного средства и заканчивается в момент его полного изъятия из эксплуатации [IEEE Std 610.12 – 1990. IEEE Standard Glossary of Software Engineering Terminology].

Понятие жизненного цикла ПС было заимствовано из промышленной инженерии. В промышленной инженерии *жизненный цикл промышленного изделия* – это последовательность этапов: проектирование, изготовление образца, организация производства, серийное производство, эксплуатация, ремонт, вывод из эксплуатации; состоящих из технологических процессов, действий и операций.

Организация промышленного производства с позиции жизненного цикла позволяет рассматривать все его этапы во взаимосвязи, что ведет к сокращению сроков, стоимости и трудозатрат. В частности, еще недавно наши экономисты выражали беспокойство по поводу того, что зарубежный потребитель сравнительно дешевым советским тракторам предпочитает канадские, цена которых в несколько раз выше. Оказалось, что полная стоимость последних с учетом затрат всего «жизненного цикла существования машин» (включая их техническое обслуживание и ремонт) получается в конечном счете в несколько раз меньше. Не случайно вопрос технологичности с точки зрения не только изготовления, но и последующей эксплуатации имеет в технике первостепенное значение [28].

Механически перенести методики решения инженерных задач промышленного производства на область программирования не удается, т.к. жизненный цикл ПС имеет ряд характерных особенностей по сравнению с техническими объектами.

Основное отличие заключается в том, что в начале жизненного цикла ПС не удается четко определить неизменный набор требований к ПС. Это происходит по разным причинам: пользователи, как пра-

вило, не способны четко сформулировать все, что им необходимо; иногда использование ПС изменяет условия его эксплуатации на столько, что приходится заново пересматривать требования к нему во время разработки и др. Изменчивость требований приводит к тому, что последовательное выполнение этапов жизненного цикла ПС становится не возможным, т.к. постоянно приходится возвращаться на предыдущие этапы для исправления недочетов и уточнений.

Следующие отличие заключается в том, что этап серийного производства программного продукта, в отличие от технических объектов, не требует значительных материальных затрат, т.к. состоит только в его тиражировании (или копировании программного продукта на носители информации).

Еще одно отличие заключается в том, что физического износа программных средств не происходит. Программные средства могли бы существовать вечно, однако из-за морального устаревания они все же заканчивают свой жизненный цикл. Считают, что ПС морально устарело, если оно перестало удовлетворять актуальным требованиям, и его дальнейшая модификация не целесообразна.

По особенностям и свойствам жизненного цикла программные средства делят на два крупных класса – малые и большие [24].

Первый класс (малые ПС) составляют относительно небольшие ПС, создаваемые одиночками или небольшими коллективами (3 – 5) специалистов, которые [24]:

1) создаются преимущественно для получения конкретных результатов автоматизации научных исследований или для анализа относительно простых процессов самими разработчиками ПС;

2) не предназначены для массового тиражирования и распространения как программного продукта на рынке, их оценивают качественно и интуитивно преимущественно как «художественные произведения»;

3) не имеют конкретного независимого заказчика осуществляющего финансирование и определяющего требования к ПС;

4) не ограничены стоимостью, сроками создания, трудоемкостью, требованиями качества и документирования;

5) не подлежат независимому тестированию, гарантированию качества и сертификации.

В случае несложных ПС нет необходимости в регламентировании их жизненного цикла, в длительном применении и сопровождении множества версий, в формализации и применении профилей стандартов и сертификации качества. Их разработчики не применяют

регламентирующих, нормативных документов, вследствие чего жизненный цикл таких ПС имеет не предсказуемый характер по структуре, содержанию, качеству и стоимости основных процессов «творчества» [24].

Второй класс (большие ПС) составляют крупномасштабные комплексы ПС для сложных систем управления и обработки информации, оформляемые в виде программных продуктов с гарантированным качеством. Они отличаются следующими особенностями и свойствами их жизненного цикла [24]:

1) большой объем, высокая трудоемкость и стоимость создания таких комплексов ПС определяют необходимость тщательного анализа экономической эффективности всего их жизненного цикла и (при необходимости) конкурентоспособности на рынке;

2) от заказчика, финансирующего программный проект, разработчикам необходимо получать конкретные требования к функциям и характеристикам ПС;

3) для организации и координации деятельности персонала при создании или совершенствовании крупных программных продуктов необходимы квалифицированные менеджеры проектов;

4) в проектах сложных ПС с множеством различных, функциональных компонентов, участвуют специалисты разной квалификации и специализации, от которых требуется высокая ответственность за качество результатов их деятельности;

5) от разработчиков проектов требуются гарантии высокого качества, надежности функционирования и безопасности применения поставляемых программных продуктов;

6) необходимо применять индустриальные, регламентированные стандартами процессы, этапы и документы, а также методы, комплексы средств автоматизации и технологии обеспечения жизненного цикла комплексов ПС.

Крупномасштабные комплексы ПС являются компонентами систем, определяющими их функциональные свойства, увеличивающими сложность и создающими предпосылки для последующих изменений их жизненного цикла. Реализация процессов жизненного цикла таких ПС зависит от многих факторов: от персонала, технических, организационных и договорных требований и сложности проекта. Организованное, контролируемое и методичное отслеживание динамики изменений в жизненном цикле сложных ПС, их согласованная разработка при строгом учете и контроле каждого изменения, является основой эффективного, поступательного развития каждой крупной

системы [24].

1.2. Жизненный цикл ПС с позиции теории деятельности

При определении базовых понятий в области производства ПС используется работа И.Н. Скопина [28]. В качестве основы для формирования этих понятий И.Н. Скопин применяет теорию деятельности, положения которой он адаптирует к предметной области управления программными проектами.

Процесс развития программного проекта рассматривается как одна большая производственная функция, выполнение которой приводит проект от замысла к готовому программному продукту и далее, от поставки программного продукта потребителям до завершения его эксплуатации. Производственная функция по реализации проекта разбивается на составляющие, для каждой функции определены субъекты-исполнители этой функции.

Проект рассматривается как динамическая система взаимосвязанных целенаправленных деятельностей субъектов-исполнителей, выполняющих определенные функции. Целенаправленность означает, что для каждой функции определена задача, для решения которой она выполняется. Динамичность означает изменение системы во времени. С точки зрения теории деятельности для каждой работы этой системы определен набор элементов (табл. 1.1).

Таблица 1.1

Элементы деятельности

№ п/п	Элемент деятельности и его описание
1	*Субъект-исполнитель* – исполнитель деятельности, обладающий определенными качествами и возможностями, позволяющими заниматься ее выполнением. Субъекты, исполнители программных проектов, могут быть индивидуумами либо группами индивидуумов. Кратко: *субъект – это тот, кто в состоянии выполнять данную деятельность*
2	*Цель* – назначение деятельности, явно выделяющее направление ее развития. Цель деятельности всегда соотносится с другими составляющими системы деятельностей (место данной деятельности в системе, как и в каком качестве используются результаты

Продолжение табл. 1.1

№ п/п	Элемент деятельности и его описание
2	деятельности в других проектных и внешних деятельностях). Кратко: *цель – это то, для чего данная деятельность выполняется*
3	*Ресурсы и материалы* – то, что используется или перерабатывается при выполнении деятельности. Ресурсы могут быть многократно используемыми, возобновляемыми или расходуемыми. К ресурсам, связанным с аспектом расходования, можно отнести: электроэнергию, оборудование и др. В ряде аспектов планирования время можно рассматривать как расходуемый и ограниченный ресурс. Термин «материалы» чаще используется, когда аспект расходования не играет существенной роли или отсутствует вовсе (например, сведения о предметной области, спецификации и требования к ПС используются, но не расходуются). Кратко: *ресурсы и материалы – это то, из чего в данной деятельности продуцируются результаты*
4	*Средства и инструменты.* Обычная деятельность предполагает применение вспомогательных средств, которые поддерживают ее выполнение либо являются необходимыми для получения осмысленных результатов. *Средства* – общее понятие, отражающее возможность применения объекта в различных деятельностях. *Инструменты*, как правило, специализируются для конкретных видов деятельности. Однако и микроскопом можно забивать гвозди. Данный пример показывает условность разграничения между средствами и инструментами, хотя это и удобно с определенной точки зрения. Средства и инструменты могут быть внешними по отношению к системе деятельностей проекта или могут разрабатываться в рамках проекта. В последнем случае нужно различать ситуации, когда разработка ведется с расчетом на независимое использование вне проекта и когда внешнее применение создаваемых средств и инструментов не предполагается. Кратко: *средства и инструменты – это то, с помощью чего в данной деятельности продуцируются результаты*

Окончание табл. 1.1

№ п/п	Элемент деятельности и его описание
5	*Методы* – это совокупность следующих элементов: 1) приемов для деятельности данного типа, которые целесообразно применять при ее выполнении; 2) предписаний, соглашений и рекомендаций, принимаемых для проекта, организации и т.д.; 3) регламентов, запрещающих или ограничивающих возможность применения средств, инструментов и других методов, а также задающих ограничение доступа к общим ресурсам. Методы разных методологий могут значительно отличаться. Кратко: *методы – это то, что указывает на способ выполнения данной деятельности*
6	*Результат*. При выполнении деятельности субъект-исполнитель создает различные продукты, обусловленные и не обусловленные целями. Совокупность всех продуктов, полученных в ходе выполнения деятельности, называется ее *результатом*. Продукты можно рассматривать как материализованные плоды деятельности, которые появляются в ходе ее выполнения. Результат может содержать продукты, полезные с точки зрения целей проекта и целей деятельности, и бесполезные продукты, не способствующие продвижению проекта к его глобальным целям. Полезность или бесполезность продукта определяется проектными соглашениями. Общий критерий полезности – включение продукта в другие деятельности проекта в качестве атрибута. Кратко: *результат – это то, что фактически продуцируется в данной деятельности*
7	*Окружение*. Любая деятельность является определенной частью некоторой общей системы деятельностей, охватывающей группу субъектов-исполнителей. Деятельности, субъекты которых не попадают в выделенную группу, являются *окружением* данной системы. Окружение связано с выделенной системой следующими способами: 1) из окружения поставляются элементы деятельностей системы; 2) деятельностям окружения передаются результаты деятельностей системы; 3) система в целом или ее отдельные деятельности являются элементами деятельностей окружения

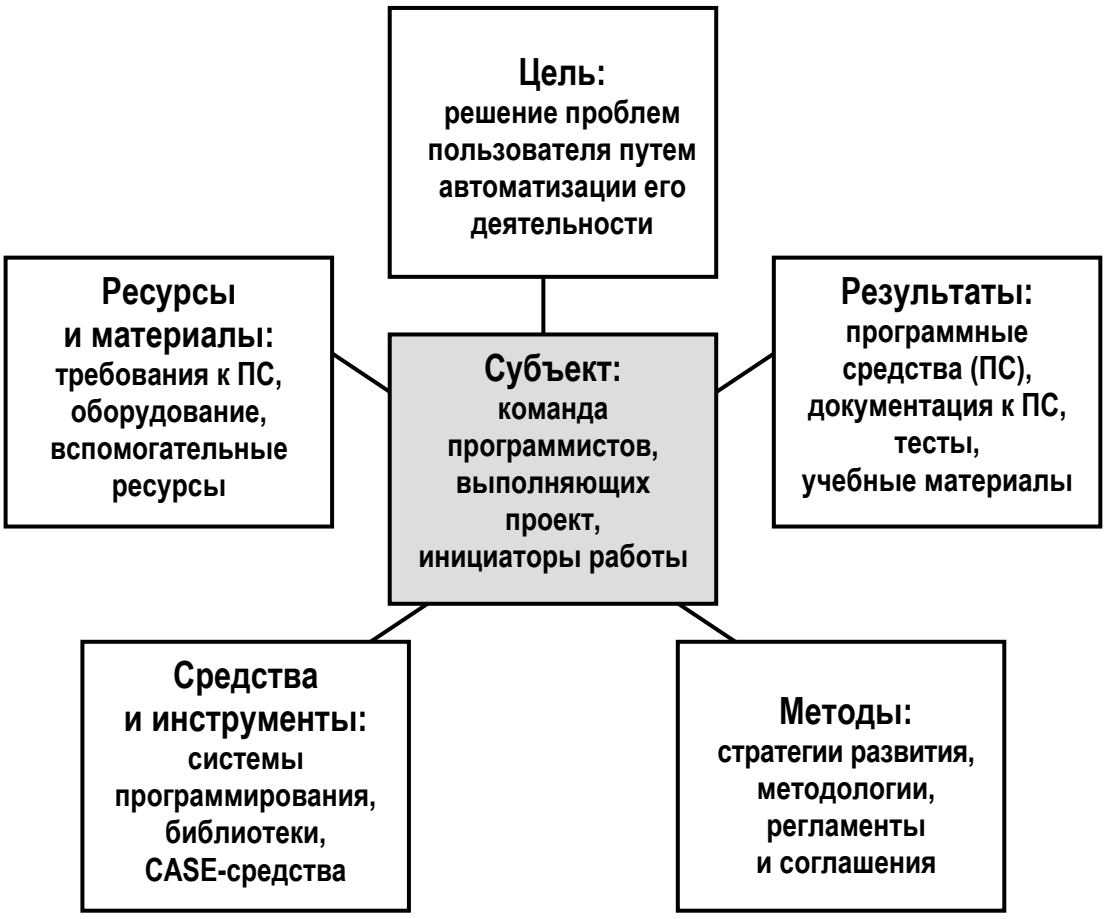


Рис. 1.2. Деятельность выполнения программного проекта

На абстрактном уровне деятельность выполнения любого программного проекта можно изобразить в виде схемы, представленной на рис. 1.2. Соединение блоков и выделение блока-субъекта означает активность этого блока и пассивность всех остальных, т.е. невозможность выполнения деятельности без воздействия субъекта.

1.3. Понятие модели жизненного цикла ПС

Жизненный цикл ПС следует рассматривать как основу деятельности менеджера программного проекта, с ним связываются цели проекта (окончательные и промежуточные), распределение и контроль расходования ресурсов, а также другие аспекты управления развитием проекта. Для успешной реализации проекта объект проектирования должен быть адекватно описан, т.е. должны быть построены полные и непротиворечивые модели объекта. Существуют различные подходы к

моделированию жизненного цикла ПС, отражающие те или иные аспекты разработки ПС и связанной с ней деятельности [28].

Модель жизненного цикла ПС (software life cycle model) – структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении жизненного ПС. Модель жизненного цикла ПС зависит от специфики ПС и условий, в которых оно создается и функционирует [3].

Одна из целей моделирования заключается в такой поддержке жизненного цикла ПС, которая в конечном итоге приводит к повышению производительности труда и качества результатов. Это накладывает определенные требования на модели жизненного цикла ПС.

В зависимости от целей моделирования выделяют два вида моделей жизненного цикла ПС: иллюстративные и инструментальные [28]. Такое разделение не является строгим, в одной модели могут сочетаться свойства (или часть свойств) как иллюстративной, так и инструментальной модели.

Иллюстративная модель иллюстрирует (отображает) жизненный цикл ПС, указывая, на какие его аспекты необходимо обратить внимание. Если цель моделирования ограничивается выделением и иллюстрацией некоторого значимого аспекта жизненного цикла ПС, то достаточно приближенных (неточных) иллюстративных моделей. Однако для повышения качества процесса управления, требуются более точные инструментальные модели.

Инструментальная модель служит основой для организации проектных работ, предоставляя различные средства систематизации работ в соответствии с производственным процессом. Инструментальные модели дают возможность оперировать своими элементами, и через это – влиять на ход моделируемого процесса, в данном случае – процесса выполнения проекта. Приведем основные свойства инструментальной модели.

1. Атрибутивность – с элементами модели связаны определенные атрибуты, необходимые для управления проектом. Эти атрибуты можно задавать или извлекать, т.е. размещать информацию о проекте в некотором хранилище или получать ее из хранилища.

2. Расширяемость – элементы модели допускают пополнение, в результате чего модель становится более детализированной, точнее отражающей реальный процесс. Для модели жизненного цикла расширяемость означает возможность добавления в нее элементов, указывающих на составляющие процесса разработки, т.е. этапы, процессы, работы и др.

3. Масштабируемость – возможность увидеть модель с разной степенью детализации, от охвата всего процесса и до конкретной работы.

4. Интегрированность с инструментами поддержки. Это качество не самой модели, а CASE-средств, совместно с которыми она используется.

Мера, в которой модели обладают этими свойствами, может служить основой для сравнения их инструментальных возможностей.

Инструментальная модель должна быть реальной поддержкой деятельности менеджера

1) давая полную картину процесса разработки проекта (историю развития, текущее состояние и варианты продвижения вперед);

2) предоставляя средства декомпозиции жизненного цикла ПС на этапы, процессы и работы с возможностью их уточнения;

3) предоставляя средства для организации планирования процесса разработки включая:

– установление сроков выполнения работ;
– выделение и отслеживание ресурсной обеспеченности работ;
– распараллеливание производственных операций;
– распределение производственных операций между исполнителями.

Реализация всех свойств инструментальной модели может привести к потере наглядности, поэтому для ее представления используют комплекс средств: различные календарные планы, графики сетевого планирования, сетевые диаграммы и др.

Последовательность и время выполнения, содержание и взаимосвязь процессов жизненного цикла ПС могут значительно меняться от проекта к проекту. По сути, жизненный цикл каждого ПС уникален. Поэтому современные концепции разработки ПС, включая различные методологии, методики, стратегии развития проекта, регламенты и соглашения, требуют адаптации к условиям выполнения конкретного проекта.

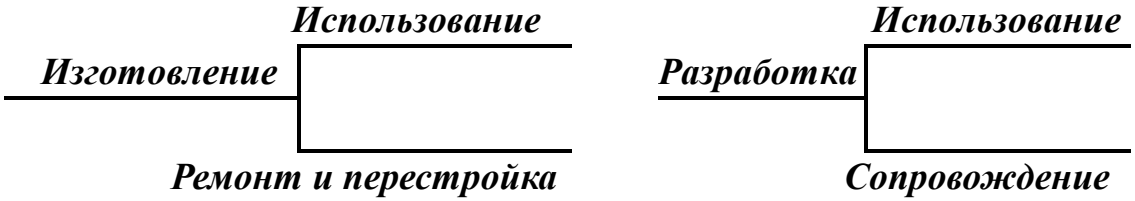
Строгой классификации моделей жизненного цикла ПС не существует, тем не менее, выделяют несколько моделей, наиболее ярко отражающих те или иные аспекты жизненного цикла ПС, к ним относятся: обобщенная, каскадная, спиральная, инкрементная и др. Большинство этих моделей относится к иллюстративному виду, и лишь не многие имеют свойства инструментальных моделей.

Рассмотрим наиболее известные модели жизненного цикла ПС.

1.4. Обобщенная модель жизненного цикла ПС

Одним из поводов обращения к понятию жизненного цикла ПС является потребность в систематизации работ в соответствии с производственным процессом.

В общем виде жизненный цикл любого изделия состоит из трех фаз: изготовление, использование, ремонт и перестройка (рис. 1.3, а). Нужно отметить, что чем больше затрат вложено в изделие, тем больше внимания уделяется фазе «Ремонт и перестройка». Для простого изделия, например, зубочистки данная стадия практически отсутствует, в то время как на ремонт и поддержание большого здания затрачиваются значительные усилия.



а) б)

Рис. 1.3. Взаимосвязь фаз жизненного цикла:
а) изделия; б) программного средства

Программное средство имеет те же фазы жизненного цикла, но, учитывая специфику создания ПС, их называют иначе: разработка, использование, сопровождение (рис. 1.3, б). В отличие от других изделий определяющей стадией для ПС является сопровождение. Статистика показывает, что на разработку ПС затрачивается 50 % времени и 38 % трудозатрат, остальные же затраты приходятся на стадию сопровождения. Это происходит по двум причинам. Во-первых, хорошее ПС требует больших затрат, поэтому при его создании сразу планируется возможность дальнейшего изменения. Во-вторых, ПС проще модифицировать, т.к. не происходит их физического износа.

Модель представленная на рис. 1.3, б называется *обобщенной моделью жизненного цикла ПС*. Рассмотрим подробнее ее фазы: использование, разработку и сопровождение.

1. Фаза «Использование». Несмотря на то, что фаза использования следует за фазой разработки (рис. 1.3, б), именно она является определяющей, т.к. ее характеристики влияют на другие фазы жизненного цикла ПС.

Перечислим основные характеристики фазы использования:

1) тип использования ПС: диалоговый (например, служащий трансагенства) или автономный (например, бухгалтер фирмы). В случае диалогового режима особое внимание приходится обращать на интерфейсную часть;

2) периодичность использования ПС (например, крайне важно учесть эффективность использования памяти при разработке ПС, работающего постоянно, и не так важно – для ПС, работающего раз в год);

3) последствия отказов – при разных типах использования, иногда отказ приемлем, иногда – нет;

4) количество и уровень пользователей (например, пятьсот пользователей, использующих ПС по-разному, влияют на его разработку совершенно иначе, чем пятьсот «одинаковых пользователей»);

5) тип ПС (прикладное, системное и т.п.);

6) масштабность (например, за один день возможно написать бухгалтерскую задачу, в то время как ПС по управлению спутником пишется годы сотнями людей);

7) сложность бывает техническая и логическая (много логических переходов);

8) ясность – существуют приложения, которые трудно запрограммировать, например, качество нефти определяется на вкус;

9) является ПС продуктом или индивидуальным проектом. Индивидуальный проект отличается от продукта двумя аспектами: определением требований и широтой функций. *Определяя требования* к продукту нужно не просто удовлетворить требования большого числа пользователей, но и выиграть соревнование с другими системами, при этом тесных связей с пользователем нет. Для индивидуального проекта такая связь существует, в этом случае важно удовлетворить требования конкретного заказчика. Проект отличается от продукта также *широтой функций,* например, для бухгалтерских задач новую операционную систему создавать не нужно, однако для космического проекта это, скорее всего, потребуется.

2. *Фаза «Разработка».* На этой фазе могут использоваться различные методы разработки ПС и стратегии развития программного проекта.

3. *Фаза «Сопровождение».* Сопровождение – это процесс модификации ПС с целью исправления выявленных ошибок и изменения его функциональных возможностей. Существует два типа сопровождения – в узком и в широком смысле. В узком смысле сопровождение ПС – это поддержка его в актуальном состоянии в условиях из-

меняющейся среды функционирования, а также обеспечение его работоспособности. В широком смысле сопровождение ПС включает в себя следующие виды деятельности: планирование и координация разработки ПС; исправление ошибок; модификация функций и добавление новых; разработка средств диагностики; разработка средств обучения; организация рекламы; тиражирование программного средства и документации к нему; поставка ПС пользователям, оказание услуг по внедрению и эксплуатации; обучение пользователей работе с ПС.

Реальные процессы жизненного цикла ПС не соответствуют обобщенной модели. Обобщенную модель можно использовать только для определения основных первичных понятий и систематизации процессов жизненного цикла ПС.

1.5. Каскадная модель жизненного цикла ПС

Долгое время ПС пытались разрабатывать, следуя каскадной модели 1970 – 1985 гг. (лавинообразная модель, модель «водопада» (waterfall)). Она определяет основные этапы жизненного цикла ПС, придерживаясь принципа их последовательного выполнения. В соответствии с этой моделью необходимо сначала проанализировать требования к будущей системе, спроектировать, создать и протестировать систему, а затем установить ее у пользователей. Выделяют следующие этапы разработки ПС с использованием данного подхода (рис. 1.4):

1) анализ;
2) проектирование;
3) реализация;
4) отладка и тестирование;
5) внедрение;
6) сопровождение.

Согласно каскадной модели переход с одного этапа на следующий должен осуществляться только после того, как будет полностью завершена работа на текущем этапе (рис. 1.4). Каждый этап заканчивается получением некоторых результатов, которые служат в качестве исходных данных для следующего этапа. При завершении каждого этапа должен выпускаться полный комплект документации, достаточный для того, чтобы разработка могла быть продолжена другой командой разработчиков.

Рассмотрим содержание основных этапов каскадной модели жизненного цикла ПС.

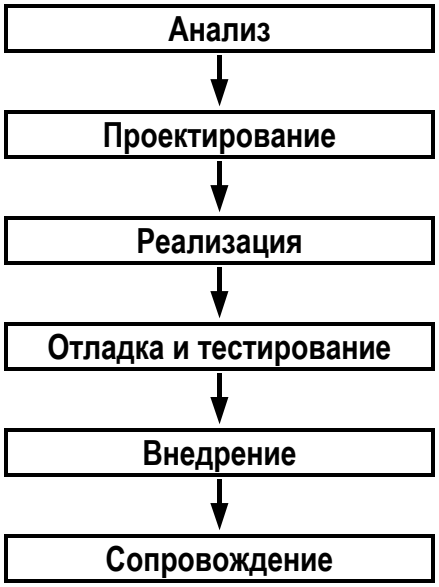


Рис. 1.4. Каскадная модель жизненного цикла ПС

Целью этапа *«Анализ»* является описание задачи, которое должно быть полным, последовательным, доступным для чтения и обзора различными заинтересованными сторонами, а также позволяющим производить сравнение с реальными условиями. В ходе этого этапа уточняются требования, приведенные в задании на проектирование, и разрабатываются спецификации на ПС. Итогом выполнения этого этапа являются эксплуатационные и функциональные спецификации, которые содержат конкретное описание ПС. *Эксплуатационные спецификации* должны содержать сведения о быстродействии ПС, затратах памяти, требуемых технических средствах, надежности и т.п. *Функциональные спецификации* определяют функции, которые должно выполнять ПС. Спецификации должны быть полными, точными и не противоречивыми.

Целью этапа *«Проектирование»* является разбиение исходной задачи на подзадачи меньшей сложности. На этом этапе выполняется следующая деятельность: выбор структуры информации; формирование структуры ПС и разработка алгоритмов, которые задаются спецификацией; определение состава модулей с разделением их на иерархические уровни; фиксация межмодульных интерфейсов.

Результатом проектирования является разбиение системы на отдельные модули, дальнейшая декомпозиция которых нецелесообразна.

Следующий этап – *«Реализация».*

Целью этапа *«Отладка и тестирование»* является выявление ошибок, проверка работоспособности ПС и его соответствия специ-

фикации. В ходе этого этапа решаются следующие задачи: подготовка данных для отладки; планирование отладки; разработка тестов; испытание ПС. Результатом данного этапа является протестированное и отлаженное ПС.

«Внедрение» – это процесс передачи ПС заказчику.

Этап *«Сопровождение»* был рассмотрен в пункте «Обобщенная модель жизненного цикла ПС».

Каскадный подход хорошо зарекомендовал себя при построении однородных систем, в которых каждое приложение представляет собой единое целое и для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования.

Положительные стороны применения каскадного подхода заключаются в следующем:

1) на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;

2) выполняемые в логичной последовательности этапы работ позволяют планировать сроки их завершения и соответствующие затраты.

Каскадный подход был официальной методологией, применявшейся во многих проектах. Однако можно с уверенностью сказать, что в чистом виде он не использовался ни разу.

Рассмотрим трудности, которые возникают, если пытаться строго следовать каскадной модели.

В начале проекта перед разработчиками стоит практически не осуществимая задача полностью определить все требования к ПС. Несмотря на то, что функции к ПС, тщательно и всесторонне анализируются и обсуждаются с пользователями, реально на стадии анализа удается собрать не более 80 % требований к ПС.

На этапе проектирования определяют архитектуру ПС. Возникающие на этом этапе проблемы обсуждаются с пользователями, это приводит к уточнению требований и появлению новых. Таким образом, осуществляется возврат на этап анализа.

После нескольких уточнений на этапе проектирования наступает этап написания программного кода. На этом этапе может обнаружиться, что некоторые из принятых ранее решений невозможно осуществить. Приходится возвращаться к проектированию и пересматривать эти решения.

После завершения кодирования наступает этап тестирования. Здесь может выясниться, что требования не были достаточно детализированы или их реализация некорректна. Нужно возвращаться назад на этап анализа и пересматривать эти требования.

Наконец ПС готово и поставлено пользователям. Поскольку прошло уже много времени и условия, вероятно, уже изменились, пользователи воспринимают его без большого энтузиазма, отвечая примерно так: «Да, это то, что я просил, но не то, что я хочу!»

Реальный процесс создания ПС никогда полностью не укладывался в жесткую схему каскадной модели. В процессе создания ПС постоянно возникает потребность в уточнении или пересмотре ранее принятых решений, т.е. осуществляется возврат на предыдущие этапы. В результате, реальный процесс создания ПС принимает вид, представленный на рис. 1.5.

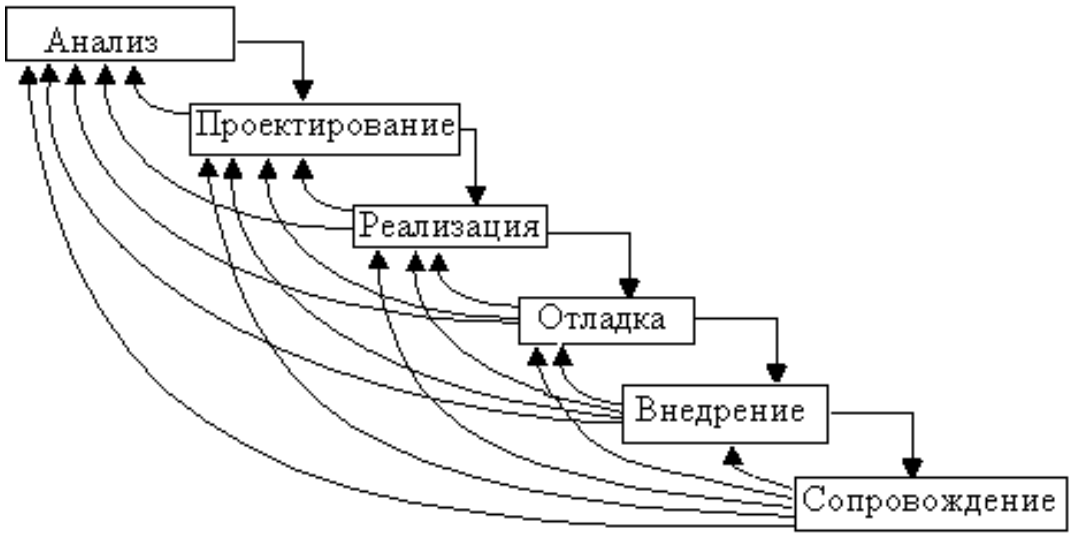


Рис. 1.5. Реальный процесс разработки ПС

Основным недостатком каскадного подхода является существенное запаздывание с получением результатов. Пользователи могут внести существенные замечания только после того, как система будет полностью завершена. В случае неточного изложения требований пользователи получают ПС, не удовлетворяющее их потребностям, приходится переделывать значительную часть ПС.

Существуют несколько модификаций классической каскадной модели жизненного цикла ПС. Одна из них – модель с промежуточным контролем, в которой каждый этап завершается проверкой полученных результатов. Это обеспечивает большую надежность по сравнению с классической каскадной моделью, однако продолжительность периода разработки увеличивается. Еще одна модификация каскадной модели рассмотрена в ГОСТ ИСО/МЭК ТО 15271.

1.6. Модель жизненного цикла ПС по Гантеру

Модель жизненного цикла ПС должна служить основой организации взаимоотношений между разработчиками. Это приводит к необходимости определения в модели [28]:

1) *контрольных точек*, задающих организационно-временные рамки проекта;

2) *организационно-технических (производственных) функций*, которые выполняются при развитии проекта.

Данный подход реализован в модели жизненного цикла ПС Гантера в виде матрицы «фазы – функции» [4]. Модель Гантера имеет два измерения:

1) фазовое, отражающее этапы выполнения проекта и сопутствующие им события;

2) функциональное, показывающее, какие производственные функции выполняются в ходе развития проекта, и какова их интенсивность на каждом из этапов.

1. Фазовое измерение. В модели Гантера жизненный цикл ПС распадается на перекрывающие друг друга фазы (этапы) (табл. 1.2).

Таблица 1.2

Описание этапов жизненного цикла ПС в модели Гантера

№ п/п	Описание этапа
1	*Этап исследования* начинается, когда необходимость разработки признана руководством проекта (контрольная точка 0). Здесь же для проекта обосновываются необходимые ресурсы (контрольная точка 1) и формулируются требования к разрабатываемому ПС (контрольная точка 2)
2	*Анализ осуществимости* начинается на этапе исследования, когда определены исполнители проекта (контрольная точка 1), и завершается утверждением требований (контрольная точка 3). Цель этапа – определить возможность конструирования ПС с технической точки зрения (ресурсы, квалификации и т.п.), будет ли ПС удобно для практического использования; решение вопросов экономической и коммерческой эффективности
3	*Конструирование* начинается обычно на этапе анализа осуществимости, как только документально зафиксированы предварительные цели проекта (контрольная точка 2), и заканчивается утверждением проектных решений в виде официальной спецификации на разработку (контрольная точка 5)

Окончание табл. 1.2

№ п/п	Описание этапа
4	*Программирование* начинается на этапе конструирования, когда становятся доступными основные спецификации на отдельные компоненты ПС (контрольная точка 4), но не ранее утверждения соглашения о требованиях (контрольная точка 3). Цель этапа – реализация компонентов с последующей сборкой ПС. Он завершается, когда разработчики заканчивают документирование, отладку, компоновку и передают ПС службе, выполняющей независимую оценку результатов работы (независимые испытания начались – контрольная точка 7)
5	*Оценка* является буферной зоной между началом испытаний и практическим использованием ПС. Этап начинается, как только проведены внутренние (силами разработчиков) испытания ПС (контрольная точка 6) и заканчивается, когда подтверждается готовность ПС к эксплуатации (контрольная точка 9)
6	*Использование* начинается ближе к концу этапа оценки, когда готовность ПС к эксплуатации проверена и может организовываться передача ПС на распространение (контрольная точка 8). Этап продолжается, пока ПС находится в действии и интенсивно эксплуатируется. Он связан с внедрением, обучением, настройкой и сопровождением, возможно, с модернизацией ПС. Этап заканчивается, когда разработчики прекращают систематическую деятельность по сопровождению и поддержке данного программного ПС (контрольная точка 10)

На рис. 1.6 представлено фазовое измерение модели Гантера. Жирной чертой (с разрывом и стрелкой, обозначающей временное направление) изображен процесс разработки. Контрольные точки и наименования событий указаны под этой чертой. Они пронумерованы. Развитие проекта в модели привязывается к этим контрольным точкам и событиям.

2. Функциональное измерение. На протяжении фаз жизненного цикла ПС разработчики выполняют определенные организационные производственные функции, которые группируются в классы родственных функций. С точки зрения моделирования жизненного цикла ПС важно следующее:

– классы родственных функций можно считать выполняемыми в течение всего хода развития проекта;

– содержание (цели) функции на различных этапах претерпевает изменение;

– интенсивность функции меняется как при переходе от этапа к этапу, так и в рамках работ одного этапа.

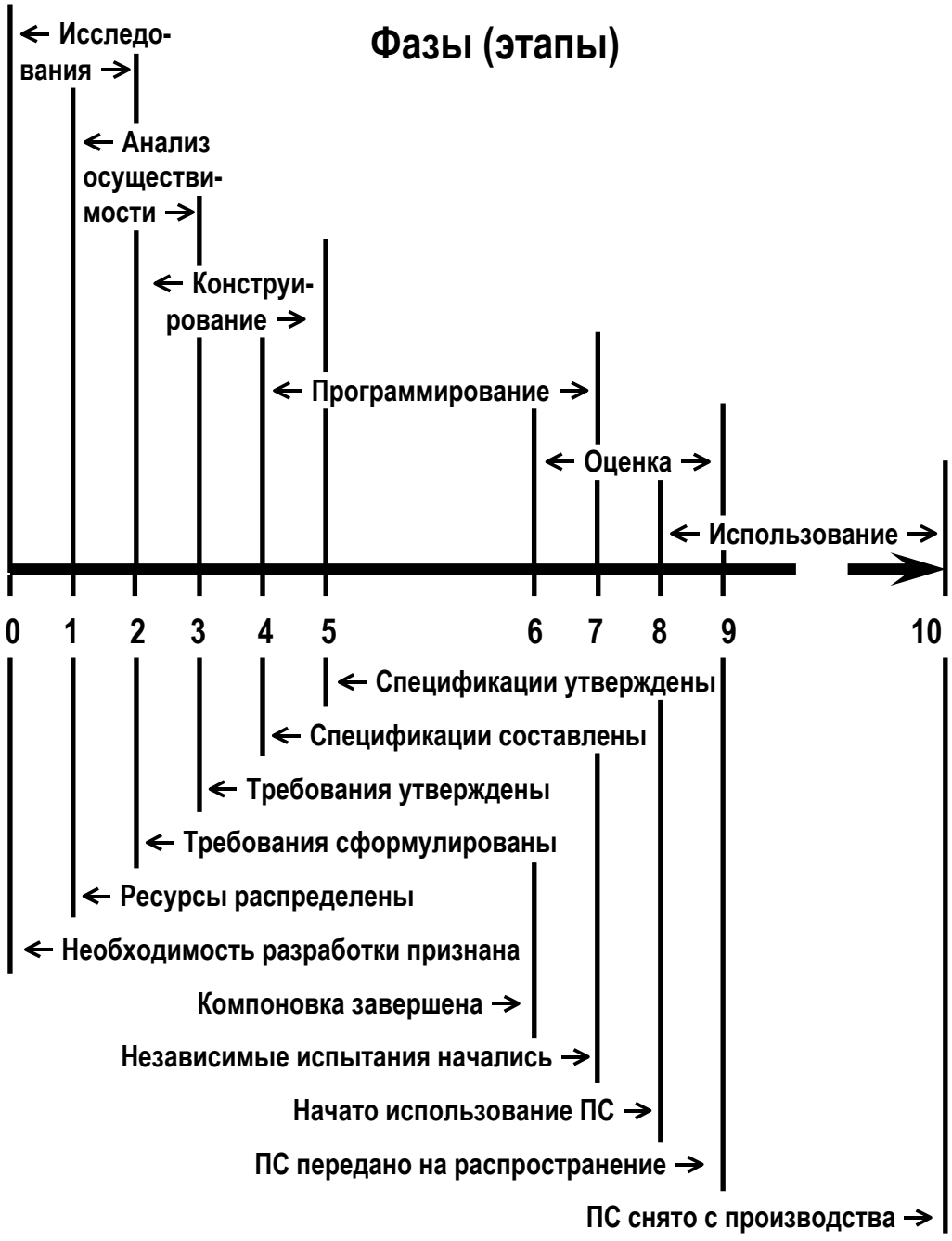


Рис. 1.6. Фазовое измерение модели «фазы – функции» по Гантеру

Состав функций и их интенсивность зависят от специфики проекта и выполняющей его команды. В качестве интенсивности можно использовать, например трудозатраты, важность работ, потребность в ресурсах.

В модели Гантера одна функция может выполняться на нескольких этапах. Схема привязки функций к этапам должна сохраняться для всех видов проектов и коллективов. Исходя из этих предпосылок, Гантер строит второе – функциональное измерение своей модели. Набор функций (классов функций) приведен в табл. 1.3.

Таблица 1.3

Описание функций жизненного цикла ПС в модели Гантера

№ п/п	Описание функции
1	*Планирование* – функция, которая должна выполняться с самого начала и до конца развития любого проекта. Содержание планов соответствует контрольным точкам
2	*Разработка* – функция, которая пронизывает весь проект. Содержание соответствует разработке рабочего продукта этапа
3	*Обслуживание* – функция, обеспечивающая максимально комфортную обстановку выполнения некоторой деятельности. Обслуживаемые виды деятельности меняются при переходе от одного этапа к другому, и могут распространяться на несколько этапов
4	*Выпуск документации* – документация в проекте включает в себя не только техническое описание системы. Она рассматривается как полноправный вид рабочих продуктов, сопровождающий другие рабочие продукты: программы, диаграммы моделей системы и прочее. Первый рабочий продукт в проекте – утвержденные требования. И именно он первым оформляется как документ. Поэтому до утверждения требований говорить о выпуске документации как о выполняемой производственной функции не приходится. Вместе с тем начиная с этого момента интенсивность деятельности по выпуску документации растет и достигает своего максимума на этапе программирования перед этапом оценки. Далее интенсивность падает и незначительно растет к концу этапа проверки, когда приходится исправлять обнаруженные ошибки в документах, предоставляемых пользователям. И здесь видно, что содержание функции меняется даже в пределах одного этапа: это не одно и то же – создавать техническое описание и готовить руководство по эксплуатации

Окончание табл. 1.3

№ п/п	Описание функции
5	*Испытания* – испытывать приходится все рабочие продукты проекта. О вариантности содержания этой функции можно судить по целям испытательных работ: подтверждения, обзоры, верификация, тестирование, аттестация и переаттестация. Поскольку испытывать можно лишь то, что готово для испытания, активизация функции начинается тогда, когда первичные требования к проекту сформулированы и нуждаются в проверке
6	*Поддержка использования рабочих продуктов* – функция, выполнение которой необходимо в связи с передачей продукта в эксплуатацию. Содержание ее связано с обучением и созданием необходимой для использования продуктов инфраструктуры. Пользователями результатов анализа сначала являются разработчики архитектуры, а затем другие разработчики. Пользователи результатов конструирования – программисты, а затем другие заинтересованные лица. По этой схеме организуется эксплуатация всех рабочих продуктов, в т.ч. и поставляемых программных продуктов. Всякий раз требуется свой содержательный уровень поддержки
7	*Сопровождение* – отличается от поддержки тем, что оно организуется для внешнего использования продуктов. Оно предполагает организацию поддержки и на этой основе выстраивает мероприятия, нацеленные на активную обратную связь с пользователями, чтобы можно было реагировать на их отклики, в т.ч. на изменение требований к продукту. Другой мотивировкой выделения сопровождения как особой функции является то, что для этих видов работ чаще всего приходится выделять специальные кадровые ресурсы, в частности со своей ролевой структурой, которая не сводится к структуре проектной группы

Состав производственных функций и их интенсивность могут меняться от проекта к проекту в зависимости от его особенностей. К примеру, если исходная квалификация коллектива не очень высока, в список функций может быть добавлено обучение персонала.

Функциональное измерение модели – функции, выполняемые при реализации проекта, накладываются на фазовое измерение, это дает изображение матрицы «фаз – функций» (рис. 1.7, на рисунке интенсивность выполняемых функций отражается густотой закраски клеток матрицы).

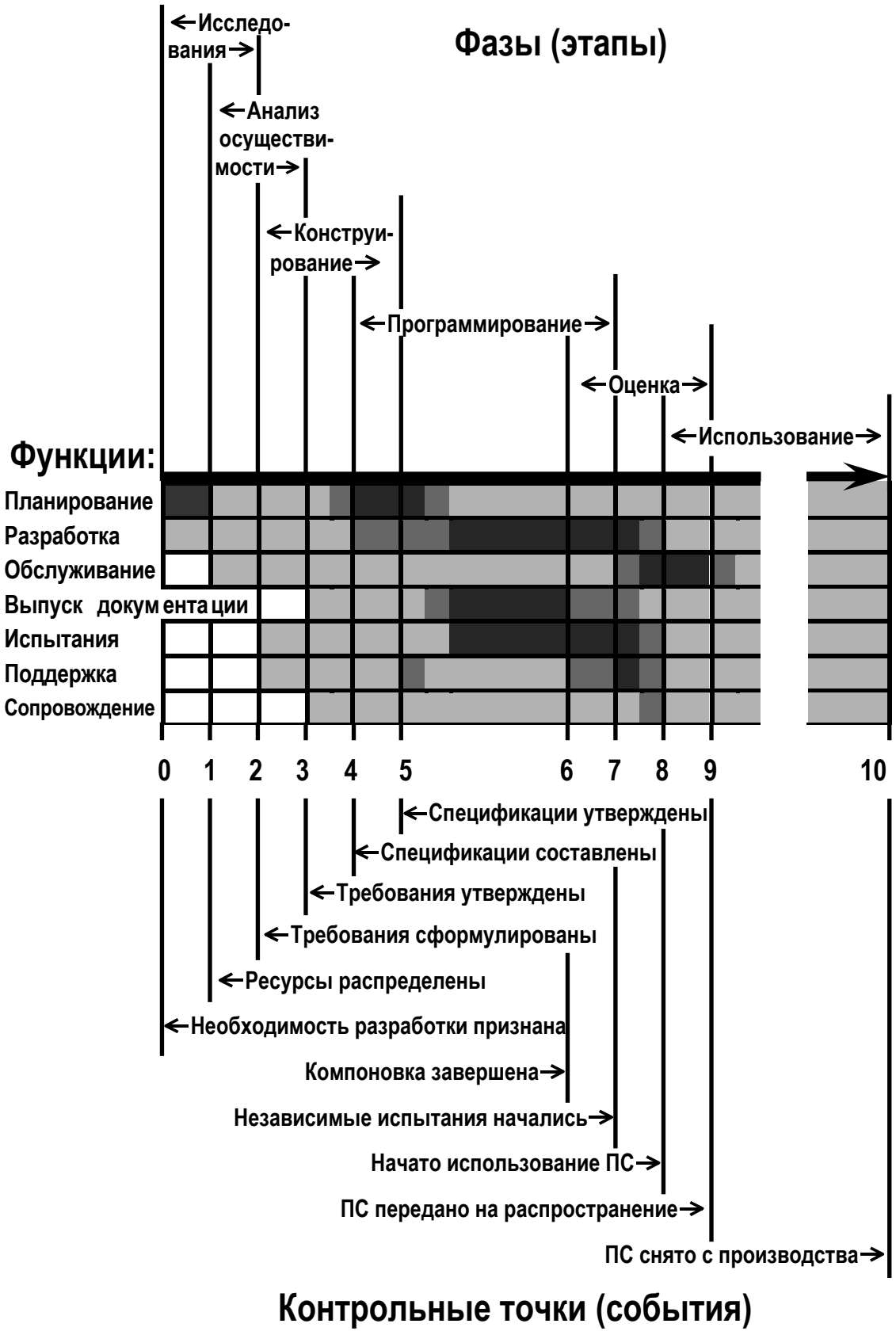


Рис. 1.7. Матрица «фазы – функции» модели Гантера

Модель Гантера учитывает соотношение производственных функций (технологических операций) и фаз жизненного цикла (этапов), этим она отличается от других моделей. В то же время задача отражения итеративности в модели Гантера в явном виде не предусматривается. Хотя само по себе перекрытие смежных фаз проекта и выпуск соответствующей событиям документации есть путь к минимизации возвратов к выполненным этапам, более содержательные средства описания итераций в модель не закладываются.

1.7. Спиральная модель жизненного цикла ПС

Любая человеческая деятельность, которая требует творчества и инноваций, идет путем проб и ошибок, итеративно развивающегося процесса. Разработка ПС – сложный процесс, и его аккуратное поэтапное выполнение не всегда возможно.

В сложных ПС итеративность возникает регулярно на любом этапе жизненного цикла ПС из-за ошибок допущенных на предыдущих этапах и изменений внешних требований к ПС. Если игнорировать необходимость возврата, то система будет содержать дефекты проектирования, некоторые требования будут не учтены, возможны и более серьезные последствия.

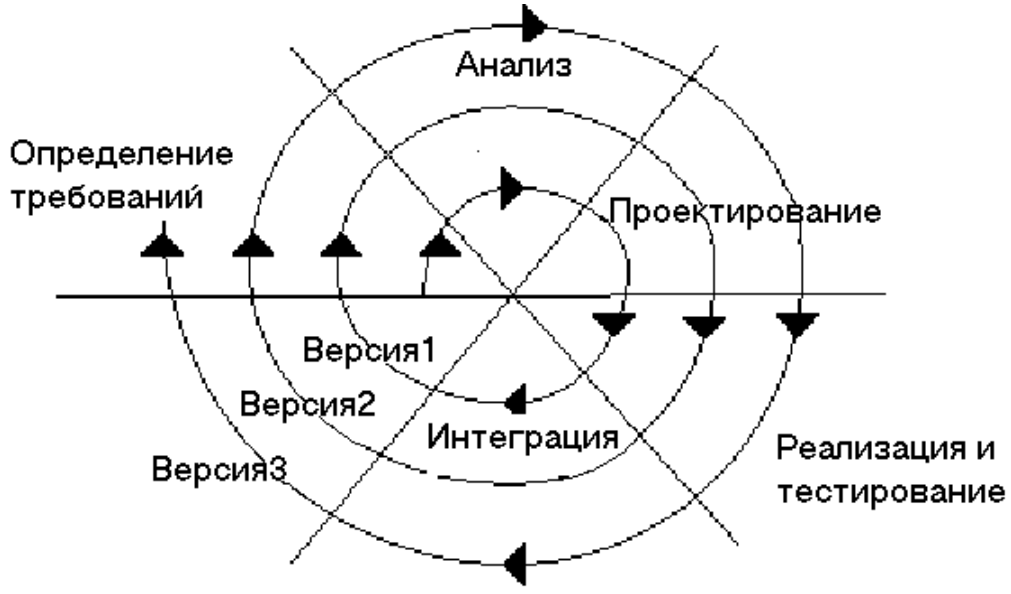


Рис. 1.8. Спиральная модель жизненного цикла ПС

Для преодоления проблем, характерных каскадной модели, бы-

ла выработана спиральная модель жизненного цикла ПС (1986 – 1990 гг.), поддерживающая итеративный процесс разработки (рис. 1.8). Ее принципиальная особенность заключается в том, что ПС создается не сразу, как в каскадной модели, а по частям и возврат на предыдущие этапы планируется заранее. Анализ, проектирование, реализация, отладка и установка системы выполняются по несколько раз. В рамках такой концепции проект можно рассматривать как последовательность небольших «водопадиков».

Спираль, раскручивающаяся от центра, послужила основой для многочисленных вариаций на тему отражения в модели жизненного цикла итеративного развития проекта. По-видимому, первым предложил ее Боэм в работе [14]. Один из вариантов спиральной модели Боэма приведен на рис. 1.9.

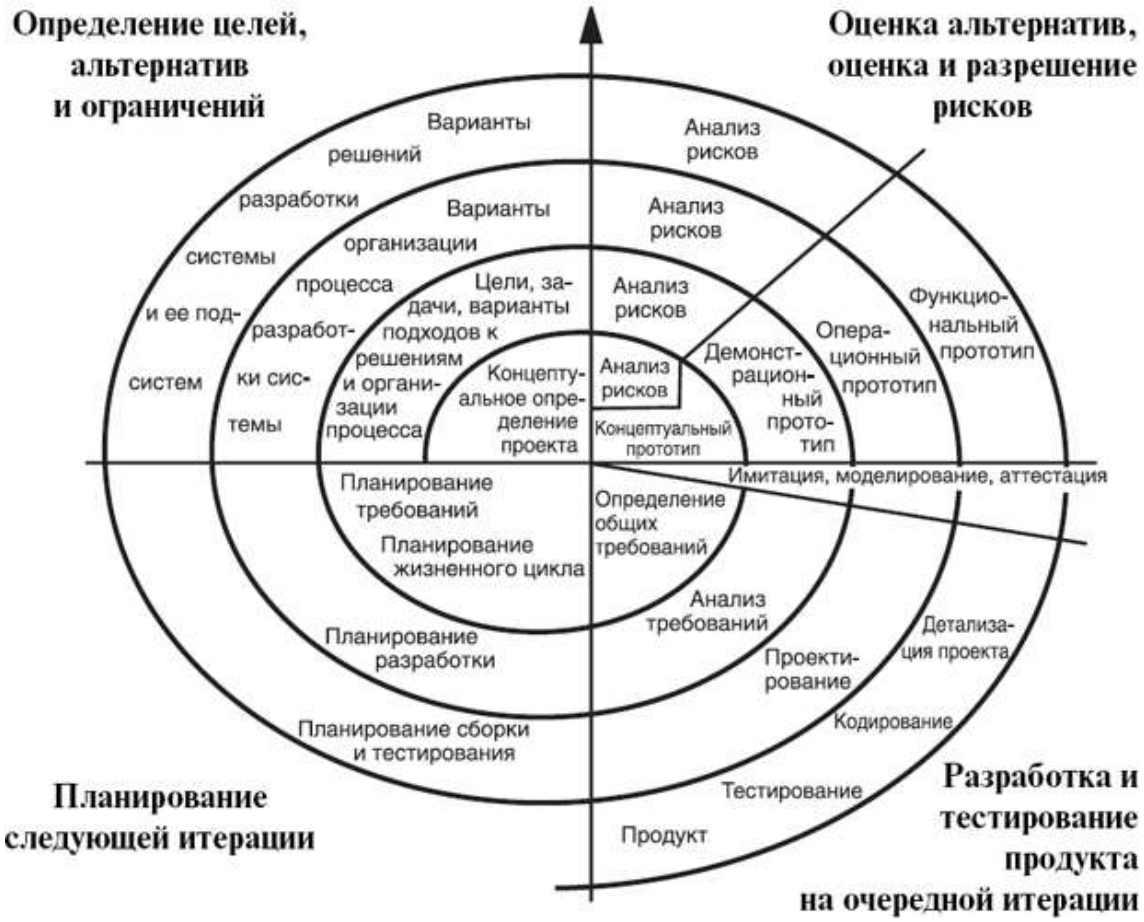


Рис. 1.9. Спиральная модель жизненного цикла ПС по Боэму

Спиральная модель обладает следующими характеристиками.

1. Итеративная разработка большое значение придает началь-

ным этапам: анализ и проектирование. Невозможно сразу выявить все требования к системе и создать идеальную архитектуру. Нужно учитывать появление новых требований и изменение архитектуры в процессе разработки, планируя несколько итераций.

Итеративный процесс предполагает, что требования к системе и созданная архитектура вновь и вновь подвергаются анализу и проектированию. При этом в каждом цикле анализ-проектирование-эволюция принятые решения развиваются, приближаясь к требованиям конечного пользователя (часто даже не высказанным), оставаясь при этом простыми, надежными и открытыми для дальнейшего изменения. Таким образом, осуществляется адаптация к изменяющимся требованиям и условиям развития проекта.

2. Реализуемость технических решений проверяется путем создания прототипов системы. Каждый виток спирали соответствует созданию версии ПС, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Детали проекта постепенно углубляются и последовательно конкретизируются, в результате чего выбирается обоснованный вариант, который доводится до реализации. Главная же задача – как можно быстрее показать пользователям очередную версию системы, тем самым, активизируя процесс уточнения и дополнения требований.

3. Разработка итерациями отражает объективно существующий спиральный цикл создания систем. На каждом этапе не требуется полного завершения работ. Это позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем; недостающая работа выполняется на следующей итерации.

4. Основная проблема спирального цикла – определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом. План составляется на основе статистических данных, полученных в предыдущих проектах и личного опыта разработчиков.

Итеративный процесс разработки ПС (спиральная модель жизненного цикла ПС) в большей степени, чем последовательный (каскадная модель жизненного цикла ПС), гарантирует, что созданное ПС оправдает ожидания заказчика.

1.8. Модель жизненного цикла ПС RUP

Сегодня 51 % программных разработок применяют CASE-

средства, поддерживающие RUP – Rational Unified Processing [6]. RUP претендует на роль универсальной основы любых программных разработок. Поэтому ее авторы предлагают общую модель жизненного цикла, которая пригодна для всех развивающихся проектов.

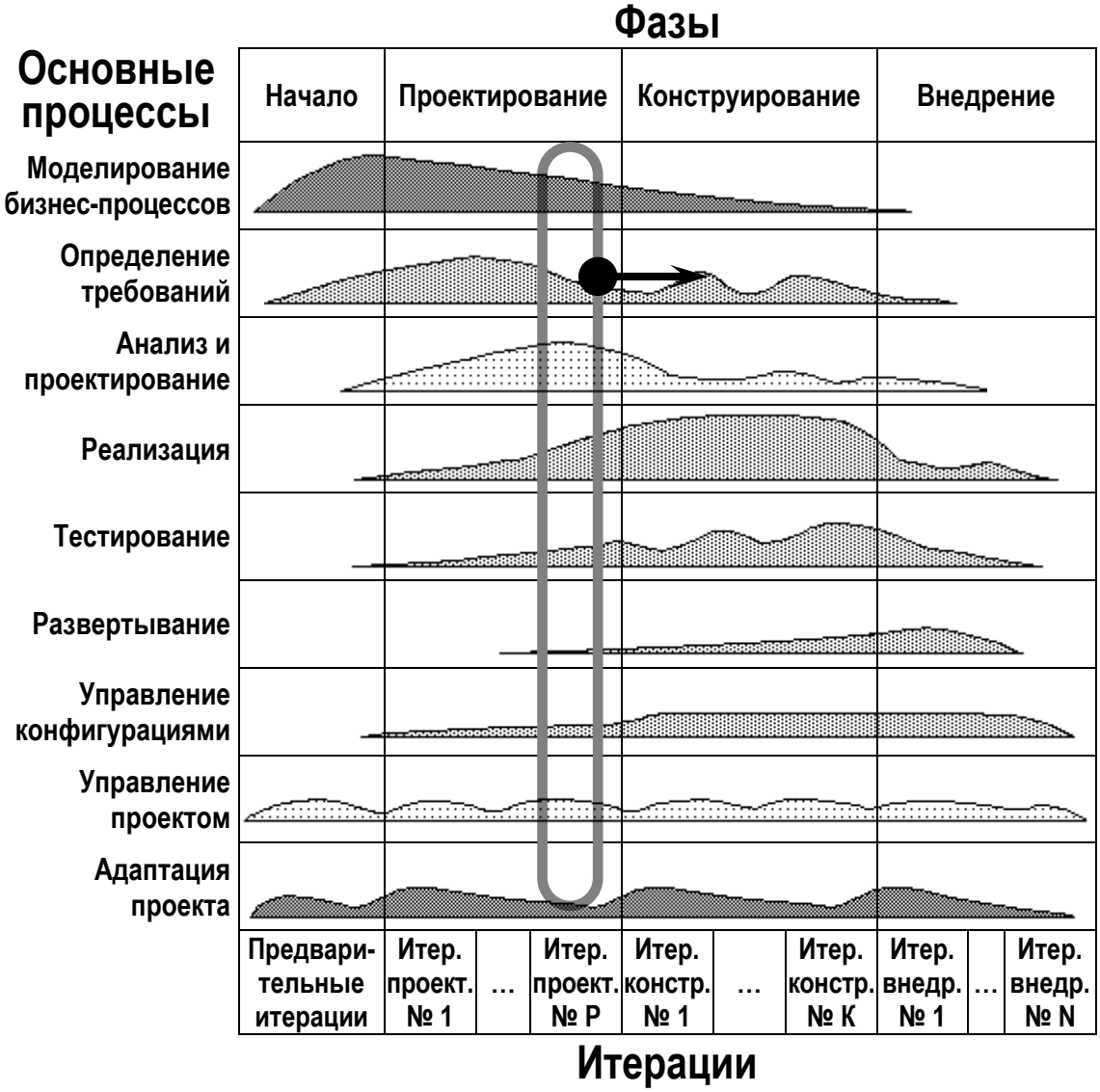


Рис. 1.10. Модель жизненного цикла RUP

Модель RUP задается в виде матрицы интенсивностей функций, выполняемых на этапах (фазах), которые проецируются на итерации (рис. 1.10). Авторы CASE-средств, поддерживающих RUP, неизменно подчеркивают иллюстративный стиль изображения интенсивностей. Для каждой итерации можно указать: в какой фазе она находится в данный момент (серый овал на рис. 1.10), а также какая рассматрива-

ется функция (жирная точка и стрелка, ведущая к очередной фазе). Для каждой функции показана интенсивность ее выполнении в виде фигур разной формы (больший объем фигуры на определенной фазе свидетельствует о большей интенсивности выполнения функции).

1.9. Модель процессов MSF

Предложение Microsoft Solutions Framework (MSF), касающееся жизненного цикла, исходит из идеи механического соединения каскадной модели MSF и самой примитивной спиральной модели. По мнению ее авторов, модель процессов MSF (рис. 1.11) «объединяет в себе лучшие принципы каскадной и спиральной моделей. Она сохраняет преимущества упорядоченности каскадной модели, не теряя при этом гибкости и творческой ориентации спиральной модели, учитывает необходимость постоянного пересмотра, уточнения и оценки проектных требований, стимулирует активное взаимодействие между проектной группой и заказчиком, который оценивает ход и результаты работы на протяжении всего проекта». Недостатки моделей, основанных на раскручивающейся спирали, присущи также MSF: невозможность отслеживания временных соотношений между сроками выполнения работ, трудности дополнения специфичных этапов. К тому же ориентация на всеобщность лишает модель и тех преимуществ, которые демонстрирует, например, модель Боэма, снабженная конкретным механизмом интерпретации [28].

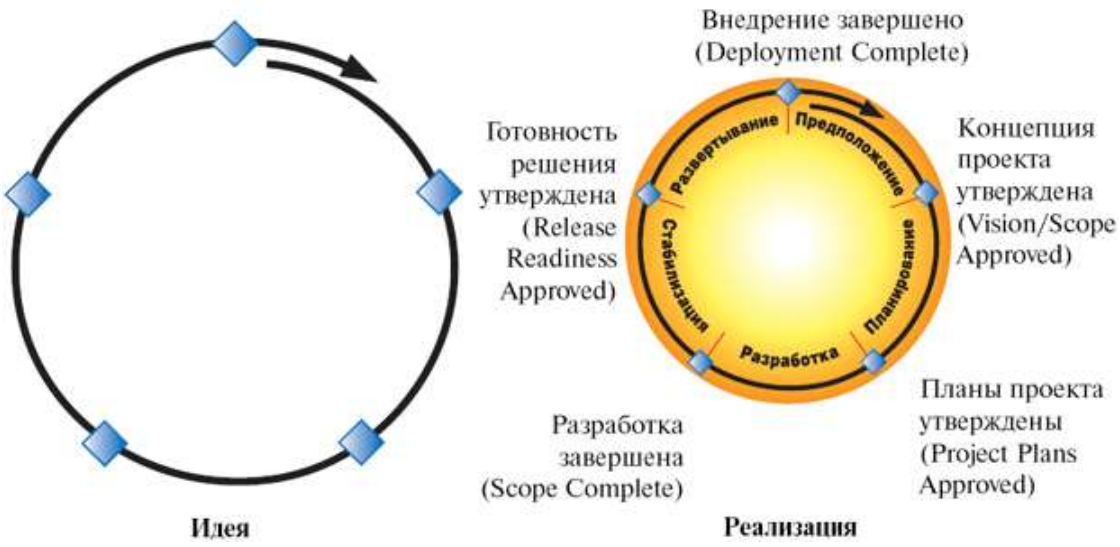


Рис. 1.11. Модель процессов MSF

1.10. Жизненный цикл ПС в методологиях быстрого развития проектов

Как утверждают сторонники быстрого развития, их методологии не нуждаются в том, чтобы четко фиксировать этапы развития разработки программного проекта [1]. Однако они понимают, что само понятие жизненного цикла ПС полезно для представления процесса разработки на концептуальном уровне. Что же касается деятельности менеджера, то в этом подходе в противовес жестким методологиям провозглашаются самодисциплина и сотрудничество вместо дисциплины и подчинения; планирование, контрольные и другие функции носят здесь такой характер, который позволяет менеджеру в большей мере сосредоточиться на руководстве командой, чем на управлении. В результате отслеживание процесса не требует, к примеру, специальных документов о достигнутых результатах и проблемах, для которых нужна специальная поддержка. По этой причине модели жизненного цикла быстрого развития не претендуют на инструментальность, и в таком ключе их рассматривать не имеет смысла. Тем не менее, понятия контрольных точек и контрольных мероприятий, распределения ресурсов и оценки остаются, хотя их содержание становится менее формализованным [28].

Жизненный цикл ПС в любой методологии быстрого развития можно описать следующим образом.

1. Выделена начальная фаза, т.к. приходится выполнять работы, которые не являются характерными для основного процесса.
2. Серия максимально коротких итераций, состоящих из шагов:

– выбор реализуемых требований;
– реализация только отобранных требований;
– передача результата для практического использования;
– короткий период оценки достигнутого (в зависимости от объема работ, этот период можно назвать этапом или контрольным мероприятием).

3. Фаза заключительной оценки разработки проекта.

Реальные быстрые методологии конкретизируют эту схему, дополняют ее теми или иными методиками.

1.11. Модель жизненного цикла ПС в экстремальном программировании

Основная концепция экстремального программирования изло-

жена К. Беком в работе [1], при этом автор не использует средства визуального представления модели жизненного цикла ПС. Схема, приведенная на рис. 1.12, взята из работы И.Н. Скопина [28].

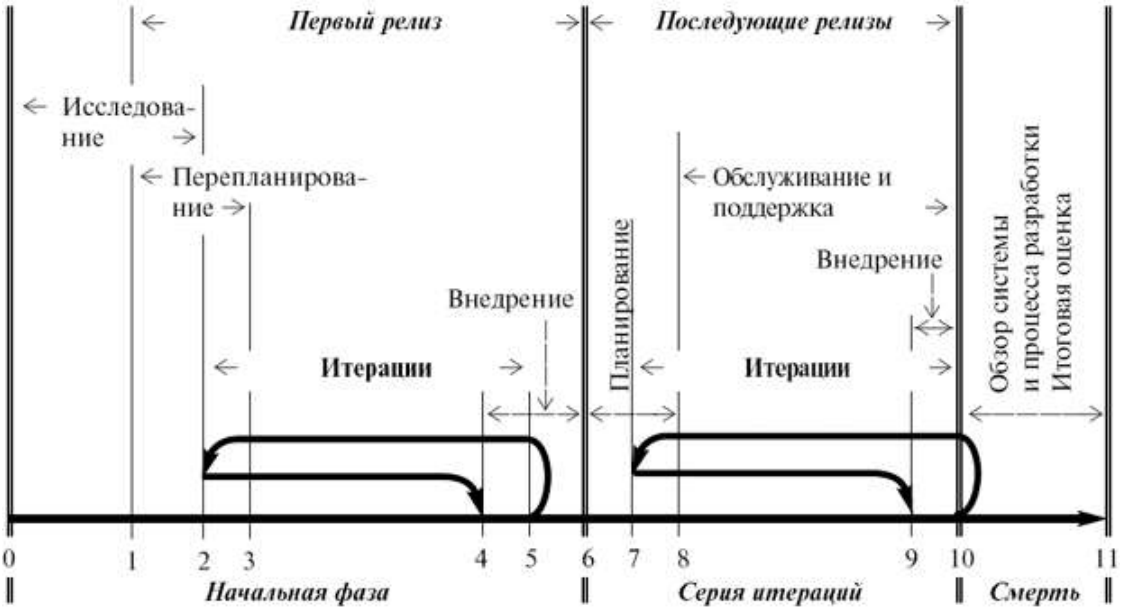


Рис. 1.12. Модель жизненного цикла ПС в экстремальном программировании

Начальная фаза развития проекта в экстремальном программировании состоит из нескольких итераций без выпуска релиза. В этот период нет результатов, которые можно представить пользователю. В качестве задач этой фазы указывается изучение применяемых инструментов, постановка экспериментов с целью определения и последующего построения стартового варианта архитектурного скелета системы, освоение командой методик экстремального программирования.

Серия итераций – это стационарный период развития проекта (рис. 1.12), на котором осуществляется постоянное взаимодействие с заказчиком, обеспечивающее нужное и актуальное для пользователей развитие работ. В модели определены контрольные точки, связанные с событиями проекта. Бек выделяет первое внедрение системы в эксплуатацию как особое событие, которое стоит отметить праздничными мероприятиями.

Как только в проекте исчерпывается постоянно пополняемый набор требований (пользовательских историй), исчезает стимул для развития проекта. Бек называет эту ситуацию «смертью» проекта. Со «смертью» проекта использование разработанного ПС не прекращает-

ся. Отсутствие новых требований к ПС свидетельствует о том, что пользовательские нужды обеспечены адекватной поддержкой. Однако возможны и другие причины для «смерти» проекта. Это либо переход пользователей к применению конкурирующей разработки, которая оказалась более подходящей, либо неспособность в разумные сроки реализовать необходимые возможности системы.

Методология экстремального программирования является характерным примером использования стратегии сужения задачи проекта за счет итеративного наращивания возможностей. Можно заметить, что в данном подходе конкретизировано движение требований таким образом, чтобы получить ясные и точные критерии их отбора для реализации с помощью апелляции к заказчику (рис. 1.12). Из этого следуют и границы применимости подхода: когда заказчик плохо представляет, что фактически нужно для поддержки деятельности пользователя, а для аналитического исследования проблем автоматизируемой деятельности возможности нет, экстремальное программирование будет давать сбои. Если воспользоваться метафорой вождения автомобиля, то можно сказать, что этот подход хорошо работает в условиях шоссе, но не на бездорожье [28].

1.12. Адаптивная разработка по Хайсмиту

Хайсмит, автор адаптивной разработки (ASD), в течение многих лет работал с предсказуемыми методологиями. Он занимался их разработкой, внедрял их, учил ими пользоваться и, в конце концов, пришел к выводу, что они глубоко ошибочны в условиях современного бизнеса. В работе [16] он излагает теоретические основы адаптивных разработок, показывая, почему они так важны и к каким последствиям приводит их использование на организационном и руководящем уровне [28].

Основу ASD составляют три нелинейные, перекрывающие друг друга фазы: обдумывание, сотрудничество и обучение, относящиеся к каждому периоду разработки, который завершается выпуском релиза (рис. 1.13). Особое представление о процессе планирования. При обычном планировании отклонения от плана воспринимаются как ошибки, которые нужно исправлять. В адаптивных разработках считается, что отклонения ведут к решениям, которые объективно обусловлены, следовательно, они являются нормой.

Неопределенность в столь непредсказуемой среде преодолевается за счет активного сотрудничества разработчиков. При этом вни-

мание руководства направлено не столько на объяснения, что именно нужно делать, сколько на обеспечение коммуникации, при которой разработчики сами находят ответы на возникающие вопросы. Отсюда следует повышенное внимание к обучению, значение которого в предсказуемых методологиях часто занижается. Хайсмит пишет, что «в адаптивном окружении обучения не избежать всем участникам проекта – и разработчикам, и их заказчикам, поскольку и те и другие в процессе работы должны пересматривать собственные обязательства, а также использовать итоги каждого цикла разработки для того, чтобы подготовиться к следующему» [16].

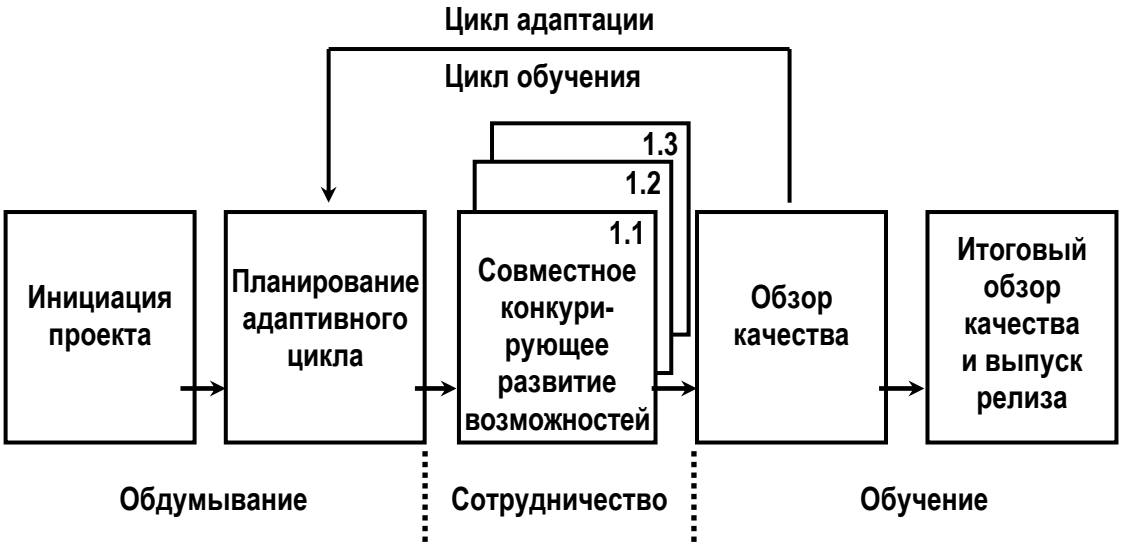


Рис. 1.13. Модель жизненного цикла ПС в концепции адаптивной разработки (ASD)

Автор ASD освещает сложные моменты адаптивных разработок, в частности вопросы обеспечения сотрудничества и обучения во время реализации проекта. И в этом ценность работы Хайсмита, поскольку полученные результаты применимы в самых разных случаях.

ASD не является методологией, это некоторая концепция различных адаптивных разработок. Схема жизненного цикла ПС не является определяющим фактором ASD, могут применяться различные методики и стратегии развития проекта. В этом отношении подход Хайсмита сближается еще с одной методологией быстрого развития, предложенной А. Коуберном. Речь идет о семействе методологий Crystal [15]. Коуберн вводит следующую градацию проектов: по одной оси откладывается количество занятых в проекте людей, по другой – критичность ошибок. Каждая из методологий семейства предназначе-

на для определенной ячейки получившейся сетки. Таким образом, проект, в котором занято 40 человек, и на котором компания может позволить себе потерять некоторую сумму, будет работать по другой методологии, нежели проект для шести разработчиков, от которого зависит существование компании.

Контрольные вопросы

1. Какие два важных аспекта имеет понятие программной инженерии?
2. Дайте определение понятия «жизненный цикл ПС».
3. Укажите основные отличительные особенности жизненного цикла ПС по сравнению с жизненным циклом технических объектов.
4. Каковы существенные особенности разработки «больших» и «малых» ПС?
5. Дайте определения следующих понятий, относящихся к элементам деятельности: субъект-исполнитель, цель деятельности, ресурсы и материалы, средства и инструменты, методы, результат деятельности, окружение системы деятельностей.
6. Дайте определение понятия «модель жизненного цикла ПС». Перечислите известные Вам модели жизненного цикла ПС.
7. Объясните в чем разница между иллюстративными и инструментальными моделями жизненного цикла ПС?
8. Опишите основные свойства и возможности инструментальных моделей жизненного цикла ПС.
9. Каковы принципиальные особенности обобщенной модели жизненного цикла ПС? В каких случаях она применима?
10. Каковы принципиальные особенности каскадной модели жизненного цикла ПС? В чем состоят преимущества и недостатки данной модели?
11. Каковы принципиальные особенности модели жизненного цикла ПС Гантера? В чем состоят преимущества и недостатки данной модели?
12. Каковы принципиальные особенности спиральной модели жизненного цикла ПС? В чем состоят преимущества и недостатки данной модели?
13. Каковы принципиальные особенности модели жизненного цикла ПС RUP?
14. Каковы принципиальные особенности модели процессов MSF?

15. Каковы особенности представления о жизненном цикле ПС в методологиях быстрого развития проектов?

16. Дайте краткую характеристику моделям жизненного цикла ПС экстремального программирования.

17. Каковы принципиальные особенности адаптивной разработки по Хайсмиту?

18. Составьте две сводные таблицы, в одной из них отразите особенности моделей жизненного цикла ПС, которые вам известны, в другой – область применения этих моделей.

Рекомендуемая литература

1. Бек К. Экстремальное программирование / К. Бек. – Спб.: Питер, 2002. – 222 с.

2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: учебник для вузов / А.М. Вендров. – М.: Финансы и статистика, 2000. – 352 с.

3. Гантер Р. Методы управления проектированием программного обеспечения / Р. Гантер. – М.: Мир, 1981. – 392 с.

4. Якобсон А. Унифицированный процесс разработки программного обеспечения / А. Якобсон, Г. Буч, Дж. Рамбо. – СПб.: Питер, 2002. – 496 с.

5. Карпенко С.Н. Введение в программную инженерию. Лекция № 1. Программная инженерия: назначение, основные принципы и понятия [Электронный ресурс] / С.Н. Карпенко. – Электрон. дан. – Нижегородск: ННГУ, 2005. – Режим доступа: http://www.software.unn.ru. – Загл. с экрана.

6. Липаев В.В. Программная инженерия в жизненном цикле программных средств. Лекция из курса «Программная инженерия. Методологические основы» [Электронный ресурс] / В.В. Липаев. – Электрон. дан. – Режим доступа: http://citforum.ru/se/lipaev. – Загл. с экрана.

7. Скопин И.Н. Основы менеджмента программных продуктов [Электронный ресурс] / И.Н. Скопин // Интернет-университет информационных технологий. – Электрон. дан. – 2004. – Режим доступа: http://www.intuit.ru/department/se/msd/1..18. – Загл. с экрана.