БАШКИРСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ
им. М. Акмуллы

М. А. Абдрафиков, В. Е. Гвоздев,
Р. Ф. Маликов, А. Р. Исхаков

Управление программными проектами: теория и практика

Учебное пособие

Уфа 2015

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВПО «БАШКИРСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ
им. М. АКМУЛЛЫ»

М.А.Абдрафиков, В.Е.Гвоздев,
Р.Ф.Маликов, А.Р.Исхаков

Управление программными проектами: теория и практика

Учебное пособие

Уфа 2015

УДК 005:004
ББК 32.973-02
А 67

Печатается по решению учебно-методического совета
Башкирского государственного педагогического университета
им. М.Акмуллы

Абдрафиков М.А. и др. Управление программными проектами: теория и практика: учеб. пособие [Текст] / М.А. Абдрафиков, В.Е. Гвоздев, Р.Ф. Маликов, А.Р. Исхаков. – Уфа: Изд-во БГПУ, 2015. – 128 с.

В данном учебном пособии рассматриваются основные понятия управления проектами, приводится перечень требований, предъявляемых к руководителю проекта сведения, о процессе планирования проекта, о метриках процессов, согласно серии стандартов ESA PSS-05-XX. Приведены методы планирования проекта, оценки размера и стоимости программного проекта, оценки затрат, отображения хода работ, а также методы и описания инструментов, поддержки управления, включая планирование проекта, решения проектных задач и методы прогнозирования экономических характеристик производства программных продуктов. Для проведения практики приведены методы оценивания сложности, продолжительности действия проекта, анализа сложных информационных систем, анализа тенденций и трендов изменения состояния сложных программных систем

Предназначено для бакалавров и магистров, обучающихся по укрупненному направлению подготовки 090000 «Информатика и вычислительная техника», а также аспирантов, инженеров, научных работников, специализирующимся в области проектирования программных и информационных систем.

Рецензенты: В.И.Васильев, д-р тех. наук, профессор УГАТУ(Уфа);
В.Н.Задорожный, д-р техн. наук, профессор ОмГТУ(Омск).

ISBN 978-5-87978-902-7

СОДЕРЖАНИЕ

Введение

Современные специалисты в области информатики, информационных систем и технологий в рамках университетской подготовки получают подготовку по информатике в широком его смысле. Эта подготовка предусматривает обучение основам программирования на языках Pascal, Fortran и др., сред программирования Delphi и группы Visual Studio, современным языкам программирования (C++, JAVA), языкам объектно-ориентированного, дискретно-событийного программирования, UML–моделированию, параллельному программированию и др.

Опыт показывает, что при подготовке инженеров по проектированию и разработке информационных систем необходимы знания не только по инженерии проектирования, но и по управлению проектами, оценки характеристик проекта, качества, конфигурации и соответствии их различным стандартам.

В связи с этим предметом обучения современных студентов, будущих разработчиков программного обеспечения, менеджеров программных проектов, тестировщиков, верификаторов, контролеров качества и др. должны стать не только теоретические и прикладные методы проектирования, но и инженерные методы управления коллективом, планирования и оценивания качества выполняемых работ, укладывания в заданные сроки и оценки стоимости проекта.

Необходимость программных разработок в различных областях знаний, в производстве и повседневной жизни возрастает. В связи с постоянно увеличивающими объемами таких разработок требуется готовить кадровый потенциал, способный решать проблемы создания новых программных продуктов на инженерной профессиональной основе, используя накопленный запас знаний в области проектирования и управления проектами.

Эти знания позволят сформировать компетентность инженера по управлению программными проектами.

К ним мы относим:

– знания методов управления проектами, включая методы планирования проектов;

–знания инструментов проектирования, планирования и оперативного управления;

– умения оценки проектных решений, характера тенденций и экономических характеристик;

– владение навыками алгоритмизации и кодирования программного обеспечения;

– владение методиками анализа и планирования проектов;

– владение методиками оценки характеристик программного проекта.

В первой главе учебного пособия рассматриваются основные понятия управления проектами, приводится перечень требований, предъявляемых к руководителю проекта, сведения о процессе планирования проекта, о метриках процессов согласно стандартам ESA PSS-05-XX.

Во второй главе рассмотрены методы планирования проекта, оценки размера и стоимости программного проекта, оценки затрат, отображения хода работ.

В третьей главе приведены методы и описания инструментов, поддержки управления, включая планирование проекта, решения проектных задач.

Четвертая глава посвящена прогнозированию экономических характеристик производства программных продуктов.

В пятой главе приведены описания практических работ по методам оценивания сложности, продолжительности действия проекта, анализа сложных информационных систем, анализа тенденций и трендов изменения состояния сложных программных систем.

При подготовке учебного пособия использовались материалы, представленные как в отечественных, так и в открытых зарубежных источниках.

Областью применимости описываемых в пособии подходов являются разработки, трудоемкость которых от двух до двадцати человеко-лет.

В конце каждой главы студентам предлагаются контрольные вопросы и или задания по изложенной теме. В конце пособия – рекомендуемая литература, список сокращений и глоссарий.

Данное пособие предназначено для студентов, аспирантов, преподавателей, инженеров и научных работников.

ГЛАВА 1. ОСНОВНЫЕ ПОНЯТИЯ В УПРАВЛЕНИИ ПРОЕКТАМИ

Управление программным проектом есть процесс планирования, организации, обеспечения персоналом, отслеживания состояния, контроля и лидерства в программном проекте.

Каждый программный проект должен иметь управляющего (менеджера), который возглавляет команду исполнителей и осуществляет взаимодействие с инициатором проекта; поставщиками; высшим руководством. Основными задачами менеджера являются:

- разработка плана управления проектом;
- выделение основных ролей проекта и распределение ролей среди персонала;
- информирование персонала о тех задачах, которые он должен решать в рамках проекта в соответствии с его ролью;
- руководство проектом, когда необходимо брать на себя ответственность за принятие ключевых решений и стимулировать персонал к результативной и эффективной работе;
- контроль за ходом выполнения проекта;
- представление информации инициатору проекта и высшему руководству о ходе реализации проекта.

На рис.1.1 представлена архитектура системы оперативного управления проектом [*i=SA-PSS*-05-08].

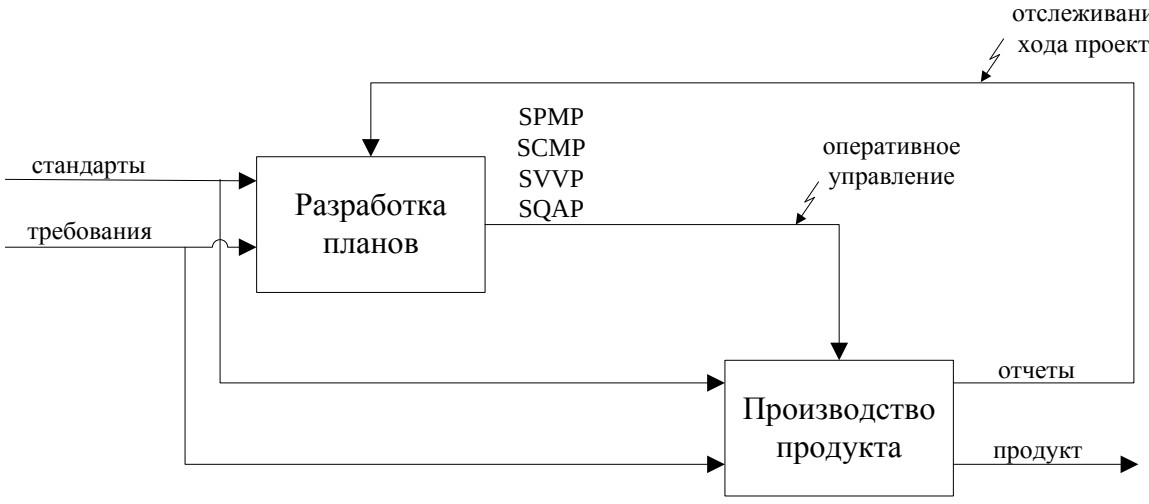


Рис.1.1. Архитектура системы оперативного управления проектом

Здесь:

SPMP – Software Project Management Plan (План управления программным проектом);

SCMP – Software Configuration Management Plan (План управления конфигурацией);

SVVP – Software Verification & Validation Plan (План верификации и валидации программного продукта);

SQAP – Software Quality Assurance Plan (План обеспечения качества программных продуктов).

Входами этой системы, создающими основу как для планирования проекта, так и для производства продукта, являются стандарты реализации разных стадий проекта и требования пользователей. Планы составляют основу управления проектом; управления конфигурацией; верификации и валидации; обеспечения качества. Планы являются составной частью системы оперативного управления производством продукта. Отчеты о ходе проекта; ведомости; о выполнении пакетов работ; по вопросам обеспечения качества; о результатах тестирования обеспечивают обратную связь с процессом планирования. Планы могут корректироваться по результатам представления отчетов.

Руководитель проекта является движущей силой в системе управления проектом.

1.1. Роль и зона ответственности руководителя проекта

При назначении руководителю проекта должны быть определены границы полномочий:

- в цели деятельности;
- мере ответственности;
- границы, в которых руководитель уполномочен самостоятельно принимать решения.

Целью любого руководителя проекта является поставка продукта требуемого качества в оговоренные сроки, причем затраты на получение продукта не должны выходить за границы бюджета проекта. И хотя перечень обязанностей руководителя проекта может зависеть от особенностей устройства организации – исполнителя; от особенностей проекта, тем не менее в любом случае обязанностями руководителя проекта являются планирование и прогнозирование. Дополнительными зонами ответственности руководителя проекта являются:

(1) *межличностные контакты*, что предполагает:

- лидерство в команде разработчиков;
- взаимодействие с инициатором проекта, высшим руководством, поставщиками;
- выступать примером для исполнителей (моряки называют это «быть носовым украшением»);
- представлять команду исполнителей на официальных мероприятиях;

(2) *сбор и распространение информации*, что означает:

- наблюдение и сбор данных о результативности персонала и изменении состояния проекта во времени;
- доведение до сведения команды исполнителей задач проекта;
- информирование инициатора проекта и высшего руководства о ходе проекта;
- выступление в качестве делегата от команды исполнителей;

(3) *принятие решений*, что означает:

- распределение ресурсов в соответствие с планом реализации проекта, а также перераспределение ресурсов, если того требуют обстоятельства (руководитель проекта, в том числе, отвечает за то, чтобы затраты не выходили за бюджет проекта);
- ведение переговоров с инициатором проекта относительно содержания положений контракта; ведение переговоров с руководителем организации относительно ресурсов проекта; ведение переговоров с персоналом относительно поручаемых им задач;
- оперативное разрешение проблем, негативно влияющих на реализацию опорного плана проекта. Причинами нарушения плана являются, например, отказы оборудования, либо проблемы с персоналом.

Руководитель проекта должен выявлять людей, с которыми ему предстоит общаться как внутри собственной организации, так и вне неё. К таковым относятся субъекты, играющие следующие роли:

- инициаторов проекта;
- поставщиков;
- соисполнителей;
- основного подрядчика;
- разработчиков других систем.

При выделении внешних интерфейсов руководителю проекта следует:

- убедиться в том, что в контрактных документах команды исполнителей и всех внешних групп, с которыми контактируют исполнители, совпадают формулировки всех пунктов всех документов;
- привлекать к коммуникациям между командой исполнителей и внешними группами как можно меньше людей;
- убедиться в том, что никто из исполнителей, вовлеченных в проект, не контактирует более чем с семью группами («правило семи»).

1.2. Планирование проекта

Независимо от размеров проекта хорошее планирование является необходимой предпосылкой его успешной реализации. Процесс планирования проекта представлен на рис.1.2

Пятью важнейшими сферами планирования являются:

- описание продуктов;
- описание видов деятельности (активностей);
- оценка объемов ресурсов и временных затрат;
- формирование сетевых графиков работ;
- построение расписания и оценка стоимости работ.

Процесс, структура которого представлена на рис.1.2, может применяться как к проекту в целом, так и к отдельной фазе проекта.

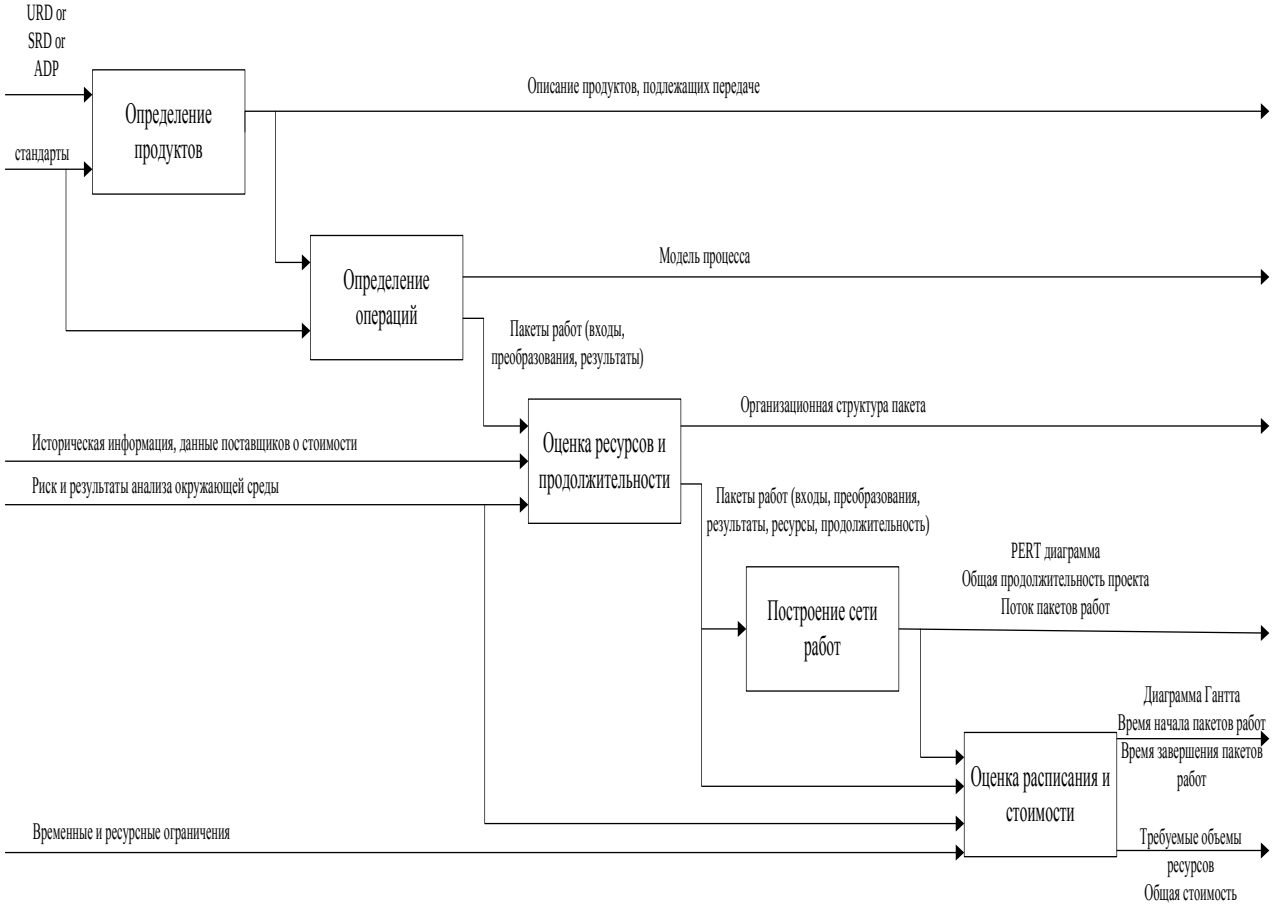


Рис. 1.2. Структура процесса планирования

Составление плана носит итерационный характер. В принципе, каждый из видов работ может быть связан с любым другим, ему предшествующим видом работ, петлей обратной связи. Иначе говоря, результаты, полученные при выполнении последующих видов работ, могут потребовать пересмотра решений, принятых при выполнении предыдущих видов работ. На приведенном рисунке

эти петли обратных связей не показаны с тем, чтобы не усложнять рисунок. Итерационный характер разработки плана служит основой его рациональности.

Исходными данными для планирования программного проекта являются:

- спецификации требований пользователей; либо спецификации системных требований; либо результаты проектирования архитектуры (в зависимости от того, для какой стадии программного проекта выполняется планирование);
- стандарты реализации продуктов и процедур;
- исторические данные, служащие основой оценивания необходимых объемов ресурсов и предполагаемой длительности выполнения работ;
- данные о стоимостях поставок;
- предполагаемые риски;
- факторы внешней среды (например, такой, как новые технологии) реализации работ в рамках программного проекта;
- временные ограничения (например, даты поставки программных продуктов);
- ресурсные ограничения (например, наличие персонала).

Первыми результатами, получаемыми при разработке плана управления проектом, являются:

- описания продуктов, подлежащих передаче;
- модель процесса разработки в составе модели жизненного цикла; предполагаемые к использованию методы проектирования и реализации; предполагаемые к использованию инструментальные средства;
- WBS (Work Breakdown Structure), т.е. иерархически организованная совокупность взаимосвязанных пакетов работ;
- организационная структура проекта, в которой определены роли и подчиненность участников проекта;
- сетевые графики, определяющие последовательность выполнения пакетов работ, общее время выполнения проекта, временные характеристики каждого из пакетов работ;
- график проекта, в котором определены начальное и конечное время выполнения каждого из пакетов работ;
- перечень ресурсов, необходимых для выполнения проекта;
- оценка общей стоимости проекта.

1.2.1. Определение продуктов и описание способов действия

Определение продуктов. В таблице 1.1 приведены перечни продуктов, соответствующие каждой стадии проекта согласно стандарту ESA PSS-05-0. Стандарты определяют оглавление (структуру) каждого из документов. Содержание же каждого из разделов документа зависит от особенностей реализуемого проекта.

Таблица 1.1

Стадия	Входные данные (продукты)	Выходные данные (продукты)
Проектирование системных требований (SR)	Спецификация требований пользователей (URD)	Спецификация системных требований (SRD)
Проектирование архитектуры (AD)	Спецификация системных требований (SRD)	Архитектура программной системы (ADD)
Детальное проектирование (DD)	Архитектура программной системы (ADD)	Детальная структура компонент программной системы (DDD), коды, руководство пользователя (SUM)
Передачи программного продукта	Детальная структура компонент программной системы (DDD), коды, руководство пользователя (SUM)	Акт приемки-передачи программного продукта (STD)

Следует обратить внимание на то, чтобы в максимально возможной степени не накладывались необоснованные ограничения на проектные решения при определении продуктов. Следует также обратить внимание на физические характеристики продукта (например, бумажный документ, либо компьютерный файл); язык написания документа; язык программирования.

Описание видов работ (активностей). После описания продуктов следующим шагом является построение модели процесса и выделение пакетов работ.

Построение модели процесса. Модель процесса создания программного продукта должна давать ответы на следующие вопросы:

- виды деятельности, связанные с созданием программного продукта;
- входы и результаты каждого вида деятельности;
- роли, соотнесенные каждому виду деятельности.

Ключевыми решениями при построении модели являются:

- выбор модели жизненного цикла, т.е. шаблонов стадий жизненного цикла;
- изменения, вносимые в типовые модели процессов, соотнесенных с определенными стадиями модели жизненного цикла;

•выбор методов и инструментов для поддержки деятельности, связанной с реализацией определенных стадий модели жизненного цикла.

(А). Выбор модели жизненного цикла

Ниже приводятся рекомендации по выбору моделей, наиболее часто используемых при реализации проектов, трудоемкость которых – от двух до двадцати человеко-лет. Предпосылками к выбору модели жизненного цикла *«водопад»* являются:

- наличие исчерпывающей совокупности высококачественных, стабильных требований пользователей;
- короткое время выполнения проекта (не более двух лет);

Предпосылками к выбору *инкрементальной* модели жизненного цикла являются:

- требование поэтапной передачи программного продукта с учетом приоритетов требований пользователей;
- потребность в высокой эффективности интеграции программного продукта с другими частями системы (например, потребность подключения в первую очередь телекоммуникационной и телеметрических подсистем обусловлена необходимостью начала тестирования системы);
- требование в кратчайшие сроки подтвердить правильность выбранного способа реализации программного продукта.

Предпосылками к выбору *эволюционной* модели жизненного цикла являются:

- необходимость проведения экспериментов с программным продуктом для уточнения существующих и ранее не сформулированных, но фактически важных, с точки зрения пользователей, требований;
- в том случае, если способ реализации программного продукта в обозримом будущем может измениться вследствие появления новой технологии;
- ожидание появления новых требований пользователя, причем содержание этих требований не определено;
- значительно более сложная реализация некоторых требований по сравнению с другими, при невозможности изменения сроков поставки программного продукта заказчику.

(Б). Адаптация типовых моделей процессов, соответствующих разным стадиям проекта. Типовые модели процессов, соответствующие стадиям формирования системных требований, проектирования архитектуры и детального проектирования, представлены на рис.1.3 – 1.5.

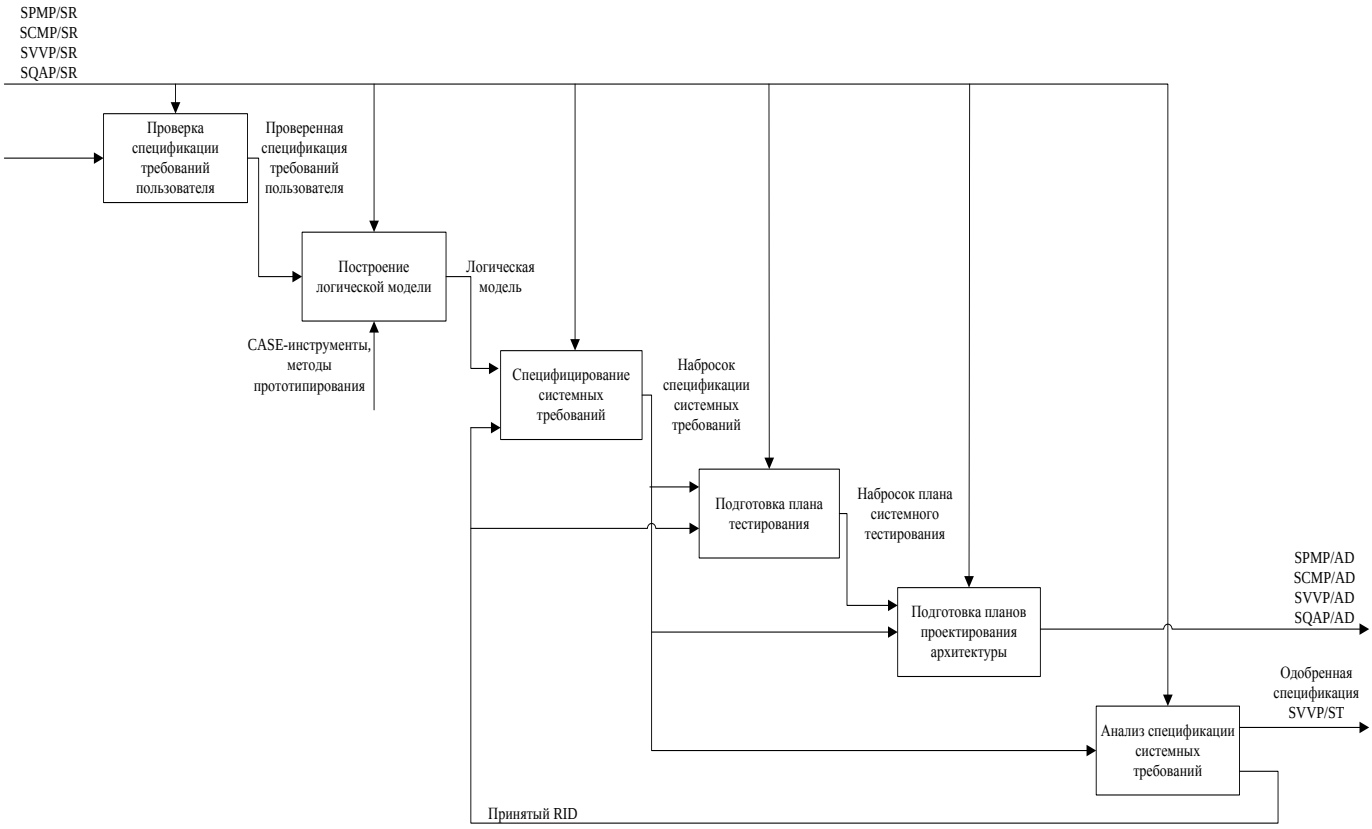


Рис.1.3.SR фаза процесса моделирования

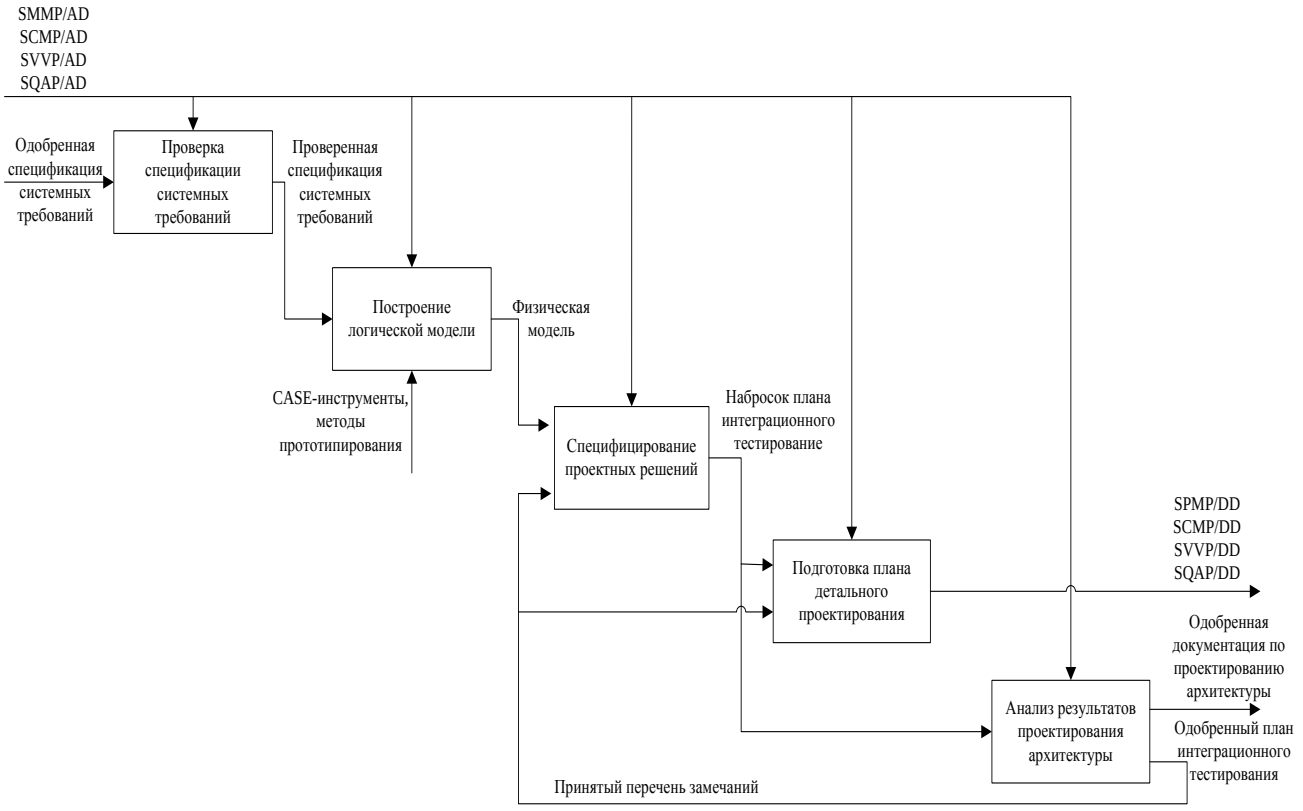


Рис.1.4.AD фаза процесса моделирования

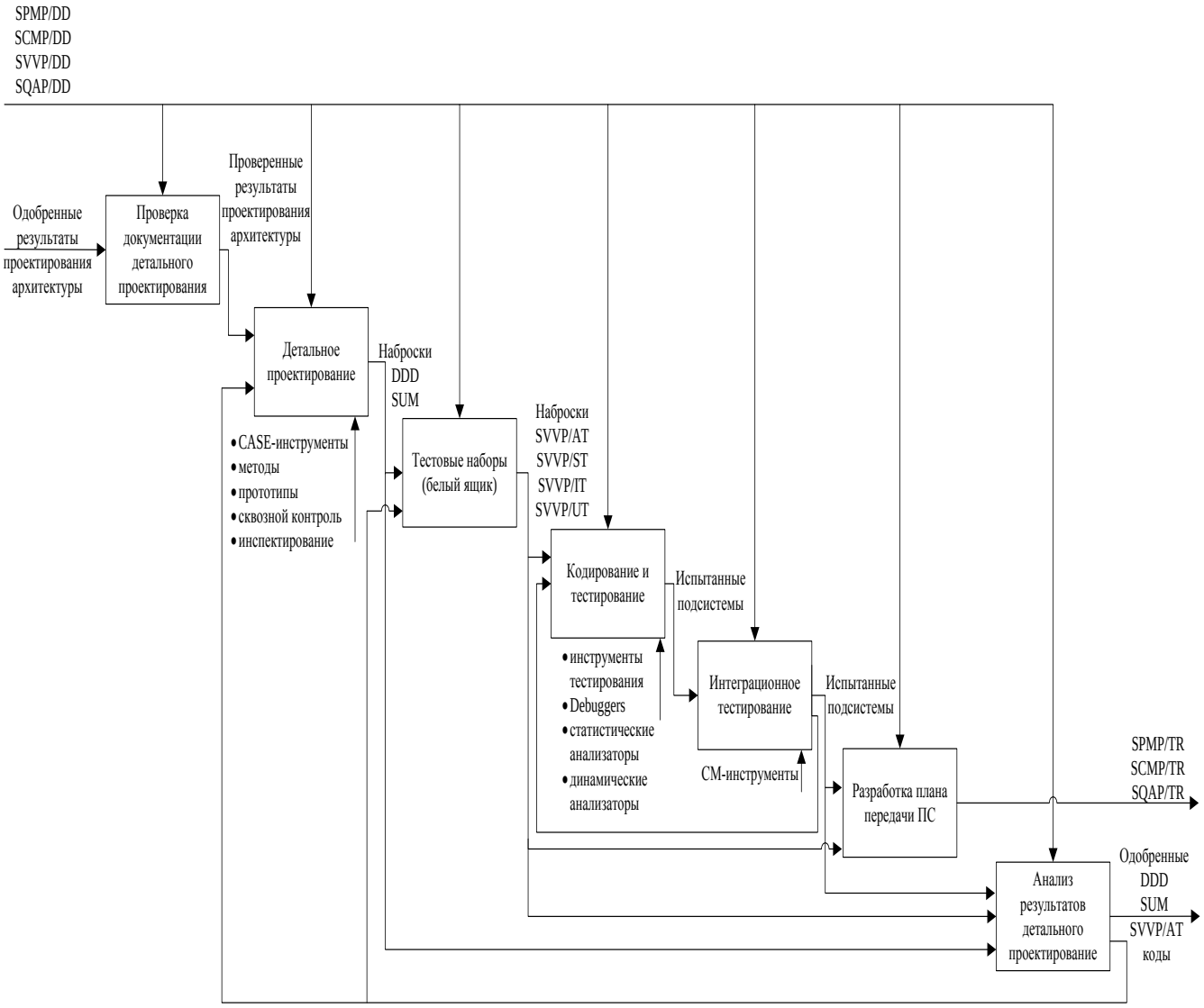


Рис.1.5. DD фаза процесса моделирования

Руководители проектов должны «настроить» эти модели процессов в соответствии с потребностями проектов. Например, специфицирование тестов программных компонентов может быть отложено до появления кодов, так как после завершения стадии детального проектирования может оказаться недостаточно информации для генерации тестов методом белого ящика.

Представленные модели требуют дальнейшей декомпозиции активностей, выделенных в моделях. Детальность описания процессов должна быть такой, чтобы:

- руководитель проекта в любой момент времени мог оценить фактическое состояние любой активности;
- каждый участник проекта знал, что он должен делать в любой момент времени.

(В). Выбор методов и инструментов

Руководитель проекта, учитывая мнение разработчиков, должен выбрать инструмент для реализации каждой из активностей, представленной на моделях.

Выделение пакетов операций (работ). Активностям, выделенным в моделях, ставится в соответствие пакет операций, решающих определенные задачи. Соотношение между выделенными операциями и задачами может быть «один к одному» (например, в случае конструирования логической модели), либо «один ко многим» (например, тестирование кодов «методом белого ящика» может потребовать реализации многих пакетов операций). Разложение сложной активности в совокупность простых операций требует хорошего понимания содержания проекта и логических взаимосвязей, лежащих в основе процесса.

В качестве критериев, используемых при выделении пакетов операций, могут выступать следующие:

- когерентность (англ.-coherence): каждая из задач, реализуемых в пакете операций, должна обеспечивать достижение единой цели, т.е. операции являются взаимозависимыми;
- степень связанности (англ.-coupling): операции в пакете должны быть минимально связанными, что позволит разработчикам в минимальной степени зависеть друг от друга;
- непрерывность (англ. – continuity): операция должна заполнять все отведенное для неё время, что позволит максимизировать эффективность деятельности;
- стоимость (англ. – cost): время выполнения работы исполнителей должно лежать в диапазоне от одной недели до одного месяца.

Уровень детализации описания работ определяется тем, с какой точностью необходимо выполнить оценку ресурсных и временных затрат на следующей стадии планирования.

Замечание. При докладах руководству не следует чрезмерно детализировать описание работ.

Пакеты операций должны быть иерархически организованы, так, чтобы взаимосвязанные работы оказывались в одной группе. Иерархическая структура операций (работ) именуется «Work Breakdown Structure – WBS». Ниже приведен пример WBS для проекта среднего размера. В этом примере каждому пакету работ ставится в соответствие четырехзначный идентификатор, что соответствует наличию четырех иерархических уровней в WBS (рис.1.6).

1000 Управление программным проектом
1100 Стадия UR
1110 Подготовка SPMP/SR
1200 Стадия SR
1210 Подготовка отчета SPM
1220 Подготовка SPMP/AD
1300 Стадия AD
1310 Подготовка отчета SPM
1320 Подготовка SPMP/DD
1400 Стадия DD
1410 Подготовка отчета SPM
1420 Подготовка SPMP/TR
1430 Уточнение SPMP/DD
1500 Стадия TR
1510 Подготовка отчета SPM
2000 Производство программного продукта
2100 Стадия UR
2110 Проектирование требований
2200 Стадия SR
2110 Построение логической модели
2220 Прототипирование
2230 Подготовка проекта SPD
2240 Подготовка окончательной редакции SPD
2300 Стадия AD
2400 Стадия DD
2410 Производство компонента К1
2411 К1 документирование (DDD)
2412 Кодирование К1
2413 Тестирование К1
2420 Производство компонента К2
2421 К2 документирование (DDD)
2422 Кодирование К2
2423 Тестирование К2
2430 Производство компонента К3
2431 К3 документирование (DDD)
2432 Кодирование К3
2433 Тестирование К3
2440 Производство компонента К4
2441 К4 документирование (DDD)
2442 Кодирование К4
2443 Тестирование К4
2450 Производство компонента К5
2451 К5 документирование (DDD)
2452 Кодирование К5
2453 Тестирование К5
2460 Интеграция и тестирование
2461 Стадия интеграции 1
2462 Стадия интеграции 2
2463 Стадия интеграции 3
2464 Стадия интеграции 4
2470 Разработка руководства пользователя (SUM)
2500 Подготовка отчета об инсталляции
3000 Управление конфигурацией программного продукта
3100 Стадия UR
3110 Подготовка SCMR/SR
3200 Стадия SR
3210 Сопровождение библиотеки
3220 Определение статуса
3230 Разработка SCMP/AD
3300 Стадия AD
3310 Сопровождение библиотеки
3320 Определение статуса
3330 Разработка SCMP/DD
3400 Стадия DD
3410 Сопровождение библиотеки
3420 Разработка SCMP/TR
3500 Стадия TR
3510 Сопровождение библиотеки
3520 Определение статуса
4000 Верификация и валидация
4100 Стадия UR
4110 Разработка SVVP/SR
4120 Разработка SVVP/AT
4130 Анализ результатов выполнения UR
4200 Стадия SR
4210 Сквозной контроль
4220 Трассировка
4230 Анализ результатов стадии SR
4240 Разработка плана SVVP/ST
4300 Стадия AD
4310 Сквозной контроль
4320 Трассировка
4330 Анализ результатов стадии AD
4340 Разработка SVVP/IT
4400 Стадия DD
4410 Анализ компонентов
4411 Анализ компонента К1
4412 Анализ компонента К2
4413 Анализ компонента К3
4414 Анализ компонента К4
4415 Анализ компонента К5
4420 Разработка SVVP/UT
4430 Разработка SVVP/IT
4440 Разработка SVVP/ST
4450 Разработка SVVP/AT
4500 Стадия TR
4510 Выполнение тестирования приемлемости
4520 Предварительный анализ приемлемости
5000 Обеспечение качества программного продукта
5100 Стадия UR
5110 Разработка SQAP/SR
5200 Стадия SR
5210 Отчеты о SQA
5220 разработка SQAP/AD
5300 Стадия AD
5310 Отчеты о SQA
5320 разработка SQAP/DD
5400 Стадия DD
5310 Отчеты о SQA
5320 разработка SQAP/TR
5500 Стадия TR
5510 Отчеты SQA

Рис 1.6. Пример WBS

Количество пакетов работ обычно возрастает с увеличением масштаба проекта. Малые проекты (трудозатраты – не более двух человеко-лет) могут иметь на нижнем уровне иерархии не более десяти пакетов работ. Большие проекты (трудозатраты – более двадцати человеко-лет) могут иметь более сотни пакетов. Малым проектам может быть поставлена в соответствие одно, либо двухуровневая иерархическая схема работ. Проектам среднего размера – двух, либо четырехуровневая схема. Большим проектам – четырех, либо пятиуровневая иерархическая модель. В средних и больших проектах для обозначения работ могут использоваться как символьные, так и цифровые идентификаторы.

Новые пакеты работ могут выделяться при возникновении новых задач в ходе реализации проекта. Такие задачи не должны объединяться с ранее выделенными, но не родственными с ними задачами. В некоторых проектах представляется разумным создать внеплановые пакеты работ, специально для объединения разнородных задач, которые неизбежно появляются в любом проекте. Вместе с тем, такие «внеплановые» работы должны отвлекать на себя лишь нескольких процентов ресурсов проекта. Новые пакеты, предназначенные для решения однородных задач, выделяют в том случае, если для решения задач требуется привлечение значительных ресурсов. Пакетам работ могут ставиться в соответствие стандарты, руководства и инструкции, в которых приводятся рекомендации по выполнению работ. Ниже в таблице 1.2 приведена типовая форма описания пакета работ.

Таблица 1.2

Проект: Стадия:	Ссылочные данные для W.P.
(-) Наименование W.P.: (-) Исполнитель: (-) Основная компонента: (-) Начало создания: Планируемая дата (-) Завершение создания: Планируемая дата (-) Руководитель W.P.	Бланк 1 из 1 Источник, на который осуществляется ссылка: Дата создания источника, на который осуществляется ссылка

Такая форма должна заполняться для каждого пакета работ. Следует избегать дублирования информации.

1.2.2. Оценка объемов ресурсов и продолжительности выполнения работ

Следующим шагом планирования является:

- оценивание объемов трудовых ресурсов;
- оценивание трудоемкости;
- оценивание стоимости внешних приобретений;

- оценивание продолжительности работ.

Оценивание объемов трудовых ресурсов. Оценивание объемов трудовых ресурсов начинается с выделения ролей исполнителей. Ниже приводятся примеры ролей:

- руководитель проекта;
- лидер команды;
- программист;
- проектировщик тестов;
- разработчик библиотек;
- специалист по обеспечению качества.

Руководитель проекта также должен определять взаимоотношения между ролями, а также эффективно координировать выполнение работ. При формировании организационной структуры следует стремиться к реализации следующих правил:

- следует убедиться, что каждый член команды подчиняется одному и только одному руководителю (принцип единоначалия);
- следует убедиться, что у каждого руководителя находится в подчинении не более семи подчиненных («правило семи»).

Структуру команды проекта следует изобразить в виде органограммы (например, как на рис. 1.7). Пунктирная линия между отделом качества организации и специалистом, отвечающим за качество проекта, нарушает принцип единоначалия. Это обусловлено необходимостью обеспечения высшего руководства независимой и объективной информацией о качестве и безопасности проекта. Высшее руководство описывает состояние проекта понятиями, соответствующими зонам ответственности руководителя проекта. Руководитель проекта описывает состояние проекта характеристиками пакетов работ.

Оценивание трудоемкости. Руководитель проекта с помощью членов своей команды и, возможно, внешних экспертов, должен выполнить детальное исследование пакетов работ и оценить трудоемкость их реализации (например, в виде человеко-часов; -дней; -месяцев).

Там, где это возможно, оценки должны опираться на исторические данные. При этом качество оценок зависит от качества исходных данных. Должна быть обеспечена сопоставимость данных с учетом таких характеристик, как квалификация персонала; риски проекта; степень новизны методов и технологий, используемых при выполнении работ. Формальные приемы, такие как COCOMO (Boehm) и FPA(Function Point Analysis) не могут использоваться в качестве инструмента на этой стадии планирования. Эти методы могут ис-

пользоваться в качестве инструмента оценки стоимости после того, как план сформирован.

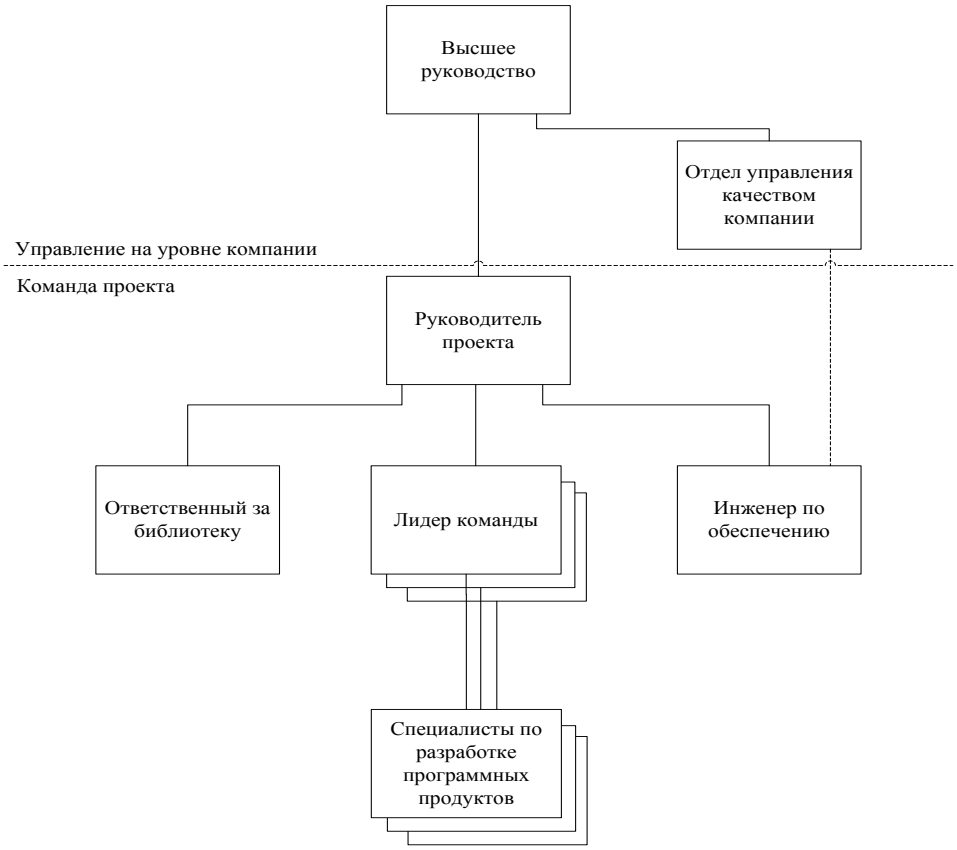


Рис.1.7. Пример органограммы для проекта среднего масштаба

Примерное распределение трудозатрат по разным стадиям программного проекта представлено на рис. 1.8.

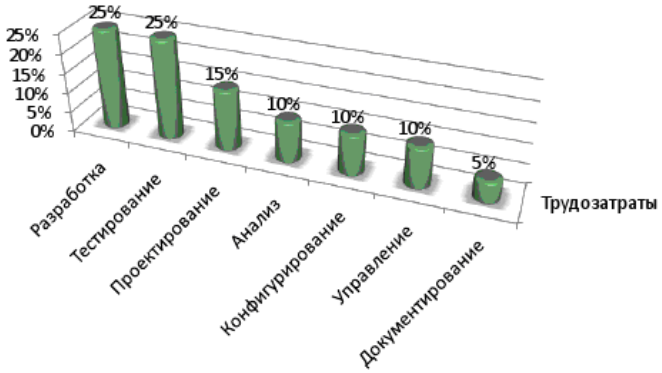


Рис. 1.8. Примерное распределение трудозатрат по основным производственным процессам при разработке ПО

Оценивание стоимости внешних приобретений. Внешние приобретения, как правило, включают в себя:

- коммерческие продукты, используемые в составе конечного продукта;

- коммерческие продукты, которые используются для получения конечного продукта, но не входят в него составной частью (например, инструментальные средства);
- материалы (т.е. расходные материалы, не входящие в накладные расходы);
- внутреннее оборудование (например, компьютеры и инструменты для тестирования);
- внешние услуги (например, размножение бумажных документов);
- транспортные и командировочные расходы;
- упаковка и погрузка;
- страхование.

Оценка продолжительности работ. Предполагаемая продолжительность выполнения работ может быть вычислена на основе оценки трудозатрат и исторических данных о продуктивности, либо исходя из каких-либо иных практических соображений (например, продолжительность такого вида деятельности, как совещание, посвященное анализу полученных результатов, может быть определена в один день).

Знание продолжительности выполнения каждого пакета работ необходимо для того, чтобы построить сетевой график работ и на его основе вычислить общую продолжительность реализации проекта на следующей стадии планирования. При оценивании продуктивности следует рассчитывать среднюю, а не максимальную продуктивность.

Замечание. В литературных источниках [6], [27] [14] отмечается, что выполнение иных, кроме непосредственного создания программного продукта, функций может занимать 25% времени персонала. В [6] указывается цифра в 50%. Это существенно снижает реальную производительность по сравнению с пиковой производительностью. Более того, индивидуальная продуктивность (такая, например, как часто упоминаемый диапазон продуктивности в 10 – 20 строк кода в день) соответствует реальной средней производительности.

Построение сетевого графика работ. При представлении последовательности выполнения работ, связанных с реализацией проекта, пакеты работ играют роль узлов, соединяемых стрелками. Последовательность стрелок представляет собою путь, либо часть пути в проекте. В сети работ не допускается наличия замкнутых путей (контуров). Главной целью построения сетевого графика работ является получение реалистичного пути работ, в результате выполнения которых будут получены все ожидаемые результаты.

Двумя важными побочными результатами построения сетевого графика работ являются:

- выделение критического пути;
- замедление темпов события, т.е.время, на которое может быть отложено выполнение отдельной работы без угрозы нарушения сроков представления результатов всего проекта.

Критический путь – это наиболее длинный путь в графе. Время выполнения критического пути – это ожидаемое время завершения проекта. Замедление темпов события есть разница между наиболее ранним и наиболее поздним временем, когда может начаться выполнение пакета работ. Иными словами, это то время, на которое может быть задержано начало пакета работ без последствий для общего времени выполнения проекта. Работы, лежащие на критическом пути, имеют нулевое время замедления темпов работ.

В общем случае, в состав сетевого графика включают только те работы, выполнение которых требует в качестве исходных данных результаты выполнения других работ; либо те работы, результаты выполнения которых создают исходные данные для других работ. Внешние работы, которые связаны с начальными и конечными узлами графа, не принимаются в рассмотрение. За счет этого упрощается процедура анализа, и это не влияет на значение длины критического пути. Далее: графы операций, который ставится в соответствие сложному проекту, следует декомпозировать на подграфы (например, по правилу: каждой стадии проекта – свой подграф). Подобный модульный подход упрощает процедуру анализа проекта, его оценивание и управление.

1.2.3. Разработка расписания и оценка общей стоимости работ

Последней стадией планирования является разработка расписания; определение требований к ресурсам; и, наконец, оценка общей стоимости проекта. Граф работ накладывает ограничения на расписание, но не формирует его. Руководитель проекта должен разработать расписание, что означает определение времени начала и завершения выполнения каждого из пакетов работ так, чтобы:

- соблюдались ограничения на время выполнения проекта и выделяемые ресурсы;
- минимизировались общие затраты;
- минимизировалась раздробленность ресурсов;
- уменьшался любой риск, который может препятствовать полноценному выполнению проекта в оговоренные сроки.

Время начала и завершения выполнения пакетов работ зависят от сроков реализации проекта (например, от даты поставки продукта) и ресурсных ограничений (например, доступности персонала и оборудования). Время начала и завершения выполнения пакетов работ должны быть определены в первую очередь, так как это снижает количество альтернативных вариантов составления расписаний. Если общее время реализации проекта не позволяет выполнить в срок поставку программного продукта, руководителю проекта следует вернуться к стадии планирования работ; перераспределить пакеты работ; переоценить ресурсы и продолжительность операций и затем модифицировать графы работ.

Минимальная общая стоимость проекта складывается из стоимости всех пакетов работ. Однако рабочая стоимость проекта должна рассчитываться из общего времени, затрачиваемого каждым из исполнителей проекта. При таком подходе к оцениванию стоимости учитываются не только непосредственное время выполнения пакета работ, но и время ожидания результатов соисполнителей. Задача руководителя проекта состоит в разработке такого расписания, при котором стоимость реализации проекта наименьшая.

В составленном расписании работы, которым ставится в соответствие высокий риск, должны начинаться как можно раньше с тем, чтобы замедление темпов события выступало в качестве «аварийного запаса» на случай возникновения каких-либо непредвиденных событий.

Однообразие ресурсов (англ. – resource smoothing – «сглаживание ресурсов»), необходимых для выполнения работ, позволяет уменьшить риск несвоевременного выполнения пакетов работ. Руководителю проекта нужно стараться составить такое расписание работ, чтобы работы, использующие схожие ресурсы, шли одна за другой так, чтобы не было перерывов во времени использования ресурсов. Это обусловлено тем, что:

- из-за перерывов в работе искажается представление исполнителей о реальном положение дел. Кроме того, может потребоваться повторное освоение забытых технологий;
- оборудование оказывается закрепленным за проектом даже в то время, когда оно не используется.

1.3. Руководство и управление рисками проекта

Одной из главных функций руководителя проекта является определение направления деятельности и практическое руководство командой проекта.

Помимо умения решать административные вопросы руководитель проекта должен быть компетентен в следующем:

- методы и инструменты;
- стандарты проектирования и кодирования;
- построение логических моделей;
- проектирование требований к программному продукту;
- построение физических моделей;
- проектирование архитектуры;
- конфигурационный менеджмент;
- верификации и валидации программных продуктов;
- обеспечение качества.

В средних и больших проектах руководитель часто делегирует право управления большинством обыденных технологических вопросов ответственному исполнителю (лидеру команды исполнителей).

Риск является неотъемлемым признаком любого проекта. Опорной точкой управления рисками является не бегство от опасностей, а снижение возможности их негативного влияния на успешную реализацию проекта. Инструментами управления рисками являются:

- постоянное отслеживание угроз успешному завершению проекта;
- разработка планов проекта, реализация которых минимизирует возможность негативного воздействия реализованной угрозы;
- разработка планов действий на случай реализации угрозы;
- реализация «запасных» планов действий в случае возникновения нештатной ситуации.

Управление рисками никогда не должно исходить из предпосылки, что «все будет хорошо». Напротив, руководитель проекта всегда должен спрашивать себя «что скорее всего пойдет не так как нужно?». В планах проекта должно присутствовать описание рисков, а также должно быть указано, как риски учитываются при разработке плана.

Типичными факторами риска для программных проектов являются:

- фактор мастерства;
- факторы, обусловленные планированием;
- технологические факторы;
- внешние факторы.

1.3.1. Фактор мастерства

К нему относятся:

- опыт и квалификация руководителя проекта;
- опыт и квалификация персонала;

- уровень зрелости поставщиков.

Опыт и квалификация руководителя проекта. Руководитель проекта является членом команды исполнителей, чья деятельность оказывает решающее влияние на результаты проекта. Неопытный руководитель проекта представляет собою значительный риск, и инициаторам проекта, осуществляющим назначение на должность руководителя, следует обеспечивать баланс между сложностью проекта и фактическим опытом руководителя проекта. Занятость руководителя проекта на неполный рабочий день также представляет собою риск, так как снижается возможность быстро решать связанные с проектом проблемы.

Руководство проектами – такая же профессия, как и все остальные. Неподготовленные руководители проекта являются источником риска, так как они не знают, в чем состоит руководство. Лицо, перемещаемое с должности исполнителя на должность руководителя проекта, должно пройти подготовку, необходимую для выполнения этой роли. Руководство проектом должно быть зоной ответственности одного человека. Коллективное руководство недопустимо. Единоначалие является гарантией единства слова и дела.

Опыт и квалификация персонала. Недостаточная квалификация, изобретательность и опытность персонала являются источниками риска для проекта. Руководитель проекта должен парировать этот риск посредством:

- оценивания пригодности персонала для вовлечения его в проект;
- подбора сотрудникам задач, соответствующих их опыту, квалификации и изобретательности;
- привлечения к реализации проекта персонала, ранее подтвердившего свою квалификацию, изобретательность и опытность с тем, чтобы использовать их компетентность при реализации нового проекта.

Зрелость поставщиков. Опытность поставщиков и подтверждающие это официальные документы являются ключевым фактором в оценке риска проекта. Индикаторами зрелости являются:

- успешная реализация в прошлом аналогичных систем;
- использование стандартов программной инженерии как основы деятельности исполнителя;
- наличие сертификата ISO 9000
- наличие программы постоянного совершенствования процесса производства программных продуктов.

Опыт реализации аналогичных систем является важным показателем квалификации разработчиков программных продуктов. Недостаток опыта являет-

ся причиной низких оценок, возможности ошибок, увеличения стоимости продукта (так как исправление ошибок требует дополнительных затрат). Стандарты могут существовать в организации, однако недостаточное внимания к их реальному использованию является причиной получения от них малой отдачи. Руководители проектов должны быть уверены в том, что содержание стандартов правильно понимается; стандарты признаются персоналом, и они действительно используются в практической деятельности. Возможно, для того, чтобы добиться этого, руководителю проекта потребуется содействие персонала, отвечающего за подтверждение качества.

Наличие программы постоянного совершенствования производства является важным признаком реальной зрелости организации. Возглавлять такую программу должна группа наиболее квалифицированных специалистов в области программной инженерии. Основной задачей этой группы является сбор данных о текущем процессе, их анализ и принятие решений по улучшению процесса.

1.3.2. Факторы планирования

Факторами планирования, которые могут явиться источниками риска, являются:

- точность оценок;
- недостаточное для выполнения проекта время;
- слишком большое время, выделяемое для выполнения отдельной работы в составе проекта;
- точечные отказы;
- местонахождение персонала;
- непродуманное распределение зон ответственности;
- изменение профиля персонала.

Точность оценок. Для успешной реализации проекта необходима точная оценка требуемых ресурсов. Недооценка необходимых ресурсов имеет гибельные последствия в случае, если в последующем получить дополнительные ресурсы не представляется возможным. Последствием переоценки необходимых ресурсов является их потеря вместо того, чтобы им быть использованными там, где это необходимо.

К операциям, с трудом поддающимся оценке, относятся:

- тестирование;
- интеграция, в особенности с внешними системами;
- затраты, связанные с передачей продукта конечному пользователю;

•анализ результатов, включая исправление ранее полученных результатов.

Некоторые ресурсы обязательно должны быть зарезервированы на случай возникновения непредвиденных ситуаций, обусловленных внезапно проявившимися ошибками. Объем таких ресурсов должен соотноситься с рисками проекта. Делать подобные оценки особенно сложно в случае реализации проектов, аналогов которых ранее у разработчиков не было.

Недостаточное для выполнения проекта время. Последствие недостаточного времени, выделяемого для выполнения работ, является необходимость параллельного выполнения значительного числа работ, следствием чего является вынужденное увеличение числа исполнителей. Чем меньше выделяемое время, тем больше опасность получить неуправляемый проект.

Руководители проектов не должны соглашаться на нереально короткие сроки выполнения проекта. Руководителям проектов следует избегать искусственного сокращения сроков реализации проекта с тем, чтобы поставить программный продукт ранее требуемого времени. Им следует использовать все выделенное время с тем, чтобы рационально использовать трудовые ресурсы.

Сталкиваясь с проблемой нарастающего отставания от плана и в условиях быстро приближающегося срока сдачи продукта, следует помнить закон Брукса (Brooks Law): включение дополнительных трудовых ресурсов в отстающих программных проектах еще больше увеличивает отставание [14] и закон Б. Боэма (рис.1.8).

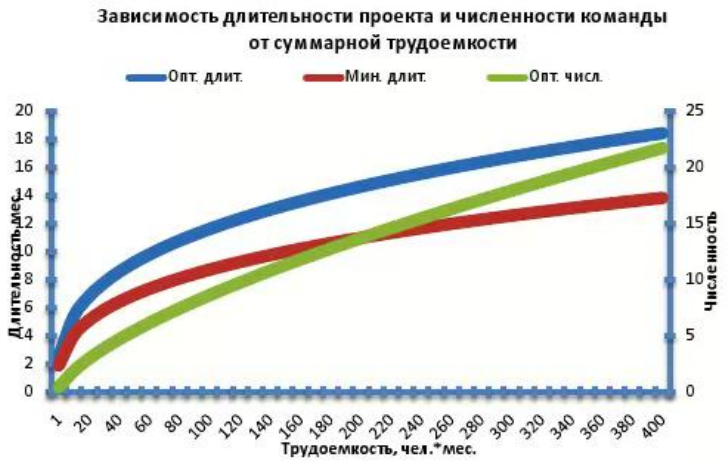


Рис.1.8. Закон Б.Боэма

Слишком большое время, выделяемое для выполнения отдельной работы в составе проекта

Рисками, обусловленными слишком большим временем, выделяемым для выполнения проекта, являются:

- возможность изменения требований пользователей;
- изменение состава исполнителей;
- необходимость перехода на новые технологии разработки.

Слишком большое время выполнения проекта является последствием слишком раннего начала проекта. Руководителю проекта следует определить оптимальное время для начала проекта путем тщательного планирования. В случае реализации больших проектов рекомендуется ориентироваться на использование инкрементальной, либо эволюционной модели жизненного цикла. Использование любой из упомянутых моделей преследует одну и ту же цель: передать пользователю программный продукт, реализующий полезные с его точки зрения функции, пока не произошло изменения требований. Если этого не удается сделать, то жизнеспособность проекта оказывается под вопросом, т.к. программный продукт на момент своего появления может оказаться невостребованным из-за того, что он уже морально устарел.

Честолюбивые цели в совокупности с ограниченным бюджетом часто являются причиной возникновения длительных проектов. По причине того, что стоимость единицы рабочего времени управленцев является фиксированной, слишком большое время реализации проекта приводит к тому, что увеличивается часть средств (в долях от стоимости конечного продукта), обусловленная оплатой труда управленцев. Более того, при длительном времени реализации проекта изменения в технической и экономической сферах могут привести к тому, что цели проекта перестанут быть актуальными, что может привести к закрытию проекта до получения какого-либо результата.

Точечные отказы (англ.- single-point failures) возникают в проектах в случае, когда жизненно важные для реализации проекта ресурсы прекращаются, причем когда-либо получить недостающие ресурсы уже не представляется возможным. Руководителям проекта следует анализировать каждую операцию со следующих позиций:

- надежность источников;
- насколько возможно использование резервных источников.

Руководители должны разработать план использования запасных вариантов источников сырья, если в этом возникнет потребность.

Местоположение персонала. Территориальный разброс местонахождения персонала, задействованного в проекте, может явиться причиной плохого информационного взаимодействия между сотрудниками. Руководителю проекта следует располагать персонал настолько близко друг к другу, насколько это

возможно. Это способствует улучшению информационного взаимодействия персонала, а также гибкому распределению поручений.

Непродуманное распределение зон ответственности является одной из главных угроз проекта. Жизненно важная для проекта работа может оказаться невыполненной только потому, что её никому не поручили. Задания могут повторно выдаваться без получения какого-либо результата, т.к. за их выполнение никто не несет ответственности. В проекте должны быть четко определены роли, а также задачи, относящиеся к каждой из ролей. При описании пакетов работ должна быть четко определена ответственность за выполнение каждой из работ.

Изменение профиля персонала. Быстрый рост численности персонала может создать проблемы, потому что новым участникам нужно время для ознакомления с проектом. После начала проекта основным источником знаний о нем становится персонал, участвующий в проекте. Ограниченность «обучающего» персонала ограничивает возможности быстрого развития проекта.

Быстрое уменьшение численности персонала также может создать проблемы, т.к. уход специалистов до завершения проекта может сильно замедлить темп решения возникающих проблем. В случае, если специалист покидает проект, должно быть определено время, в течение которого он передает свои знания остающемуся в проекте персоналу.

Численность персонала изменяется в течение жизненного цикла проекта: от малой численности на стадии проектирования требований до пикового значения на стадии тестирования и испытания компонент, с последующим уменьшением вновь до малого числа к началу стадии передачи продукта пользователю. Руководителям проекта следует избегать резких изменений в профиле трудовых ресурсов и создавать условия к тому, чтобы можно было выдержать прилив персонала без ущерба для хода проекта.

1.3.3. Технологические факторы

В качестве источников риска могут выступить следующие технологические факторы:

- технические новшества;
- зрелость и пригодность методов разработки;
- зрелость и эффективность инструментальных средств;
- качество приобретенных программных продуктов.

Технические новшества неизбежно связаны с риском. Руководители проекта должны оценить степень технической новизны каждой части системы. Каждый новый компонент может вызвать проблемы, так как:

- недостаточно убедительных доказательств их пригодности;
- сложно точно оценить объем усилий, необходимых для их создания.

Руководителю проекта следует оценить затраты и выгоду от нового компонента. Одним из способов снижения риска, обусловленного техническими новшествами, является прототипирование. Затраты на создание прототипа существенно меньше, чем разработка всей системы, так что пригодность демонстрируется до того, как в разработку будут осуществляться основные вложения. Еще одним преимуществом от использования прототипа является возможность повысить точность оценивания предполагаемых затрат.

Зрелость и практическая значимость методов. Новые методы, как правило, незрелые, и для их улучшения необходим опыт практического использования. Кроме того, новые методы недостаточно поддержаны инструментальными средствами. Выбор непригодного средства оборачивается выполнением ненужной работы и, возможно, в получение в результате реализации проекта невостребованного программного продукта. Поэтому, прежде чем использовать новый метод, необходимо проанализировать, для решения каких проблем он может быть использован. Руководителю проекта необходимо оценить сильные (например, высокую практическую значимость) и слабые (например, недостаточная зрелость) стороны новых методов, прежде чем принять решение о целесообразности их использования. Следует в обязательном порядке изучить, имело ли место применение метода в аналогичных приложениях.

Подтверждением практической значимости метода являются результаты верификации и валидации программного продукта.

Зрелость и эффективность инструментальных средств. Доказано, что наличие инструментальных средств может оказаться не преимуществом, а недостатком. Примерами этому могут служить следующие:

- персонал не был предварительно обучен их использованию;
- они не соответствуют подходам к реализации проекта;
- в ходе реализации проекта произошла смена инструментальных средств;
- с использованием инструментальных средств связаны бо́льшие накладные расходы, нежели при использовании «ручных» технологий.

Целью использования инструментальных средств является повышение производительности и качества. Руководитель проекта должен тщательно

взвесить затраты и выгоды, связанные с использованием в проекте инструментальных средств.

Качество приобретаемых программных продуктов. Решение использовать при реализации проекта проверенный программный продукт с хорошо известными характеристиками связано с малым риском. Однако ориентация на новый коммерческий программный продукт, либо использование хорошо известного программного продукта в новых условиях, может быть сопряжено с б*о*льшими рисками, так как:

- оно в реальности может работать не так, как было заявлено, либо как ожидалось;
- их использование может не обеспечить требований по качеству, надежности, сопровождаемости и безопасности, предъявляемых к проекту.

Новые коммерческие программные продукты должны как можно раньше пройти всестороннюю оценку с тем, чтобы избежать неприятных сюрпризов в дальнейшем. Иными, помимо технических характеристик, аспектами изучения коммерческих программных продуктов являются:

- размеры поставщика, его возможности и опыт;
- способность поставщика обеспечить поддержку программного продукта;
- планы будущего развития программного продукта.

1.3.4. Внешние факторы

Источниками угроз на разработку проекта, могут оказаться:

- качество и стабильность требований пользователей;
- определенность и стабильность внешних интерфейсов;
- качество и пригодность внешних систем.

Качество и стабильность требований пользователей. Наличие высококачественного документа «Спецификация требований пользователей» является необходимым условием успешной реализации проекта. Содержание этого документа является эталоном, относительно которого будет оцениваться успех проекта. Руководителю проекта следует выполнить анализ качества спецификации требований пользователей прежде чем начинать проектирование программного продукта и реализующего его проекта.

Определенность и стабильность внешних интерфейсов. Внешние интерфейсы могут стать источником риска в случае нестабильности их описания; недостаточного описания; либо в случае, когда они являются нестандартными.

Интерфейсы с внешними системами должны быть представлены в документах, именуемых Interface Control Documents – ICDs. Разработка таких документов должна быть выполнена как можно раньше, т.к. внешние интерфейсы являются одним из ограничений проекта. Руководитель проекта должен предусмотреть адекватные ресурсы для обеспечения взаимодействия создаваемого программного продукта с внешними системами.

Негативные последствия от запаздывания доступа к внешней системе могут быть смягчены посредством:

- временного добавления в программный продукт возможности «шунтирования» обращений к внешней системе;
- разработки программного продукта, симулирующего внешнюю систему.

При принятии окончательного решения о подходах к парированию угроз, обусловленных поздним предоставлением доступа к внешней системе, следует сопоставить стоимость разработки симулятора и возможные потери, связанные с поздней поставкой системы.

1.4. Измерение процессов проекта и продукта

Руководителю проекта следует в максимально возможной степени использовать количественные показатели для измерения продукта, а также процессов проекта. Это важно с точки зрения эффективного оперативного управления проектом; с точки зрения планирования будущих проектов; совершенствования производственных процессов в организации.

Формирование системы метрик предполагает реализацию следующей последовательности шагов:

1. Исследовать внешнюю среду проекта и по результатам проведенного анализа выделить приоритетные цели (например, финансовые; сроки разработки; надежность и т.д.).

2. Преобразовать приоритетные цели в совокупность подцелей таким образом, чтобы каждой подцели были поставлены в соответствие с количественным показателем (например, затраты в человеко-месяцах). Процедура построения дерева целей и выделения измеримых целей описана, например, в работе В.И. Николаева и В.М.Брука [26].

3. Выполнить сбор исходных данных и оценить соответствие характеристик проекта целям проекта.

4. Внести изменения в план реализации проекта с тем, чтобы ликвидировать отклонение от целей проекта.

5. Внести улучшения в производственный процесс. Для этого группа, отвечающая за непрерывное улучшение процесса, на основе анализа измерительных данных вносит изменения в стандарты и процедуры, связанные с реализацией процессов.

Исследование внешней среды проекта и выделение приоритетных целей. Целью исследования внешней среды является понимание того, что нужно получить в результате реализации проекта (определение приоритетных целей), а также оценка реалистичности достижения этого.

Анализ целей и описание метрик. Общими требованиями к целям, вытекающим из приоритетных целей, являются:

- не превышать плановых затрат на каждую операцию;
- не превышать планового времени выполнения каждой операции;
- получить подтверждение комплектности продукта;
- получить подтверждение надежности продукта.

Метрика – это количественная мера того, в какой степени система, компонент или процесс обладают заданными свойствами. Руководитель проекта должен поставить в соответствие каждой из целей как минимум одну метрику. Для обеспечения сопоставимости метрик, соответствующих разным проектам, они должны конструироваться таким образом, чтобы в наибольшей степени соответствовать действующим организационным и промышленным стандартам.

Метрики процесса. Возможными метриками для оценки затрат, связанных с реализацией проекта, являются:

- количество используемых ресурсов;
- количество потерянных ресурсов.

Возможными метриками для измерения времени реализации проекта являются:

- фактическая продолжительность операции (в днях, неделях, месяцах);
- сдвиг в операциях (фактическое начало – плановое начало).

Возможными метриками для измерения хода проекта являются:

- число выполненных пакетов работ;
- число решенных задач (по отношению к плановому числу).

Метрики продукта. Количество произведенной продукции может измеряться:

- числом написанных строк кода (не считая комментариев);
- числом закодированных модулей и протестированных программных единиц;

- числом внедренных функций;
- количеством страниц написанной документации.

Сравнение фактических величин с их плановыми значениями служит основой оценки уровня завершенности разработки.

Возможными метриками для оценки надежности программного продукта являются:

- покрытие модулей тестами;
- цикломатическая сложность модулей;
- интеграционная сложность программ;
- число сообщений о критических, с точки зрения создаваемого программного продукта, проблемах;
- число сообщений о некритических, с точки зрения создаваемого программного продукта, проблемах;
- число изменений, вносимых в продукт после первого издания или представления.

1.5. Сбор метрических данных и характеристик производительности

Руководителю проекта следует убедиться в том, что собираемые метрические данные отражают реальное состояние проекта. Например, прежде чем подписать заключение о завершение пакета работ, следует проанализировать реальные затраты. Подсчет сообщений о проблемах, возникающих в ходе реализации проекта, должен быть частью процесса «Оценка состояния конфигурации». Расчет цикломатической сложности может быть типовой процедурой, предшествующей инспекции кода, либо проектирования теста для программной единицы.

Улучшение производительности. Метрические данные должны составлять базу для подготовки отчета по управлению проектом, планирования будущих проектов, выполнения исследований, направленных на совершенствование процессов производства программных продуктов. Руководителю проекта следует использовать метрические данные для улучшения процессов и продуктов. Например:

- сопоставление прогнозных и фактических затрат позволяет быстро оценить недостатки исходных оценок и помочь руководству перепланировать проект с тем, чтобы улучшить его исполнение;
- анализ закономерностей в местоположении выявленных проблем на этапе интеграции и системного тестирования может способствовать вынесению обоснованного заключения о том, целесообразно ли дальнейшее проведение

работ по улучшению качества и надежности, либо продукт может быть передан пользователю.

Отчетность о проекте. Точная и своевременная отчетность необходима для оперативного управления проектом. Отчеты о проделанной работе служат основой подготовки руководителем проекта сводного доклада о ходе проекта инициаторам проекта и вышестоящему руководству. Инициаторы проекта и вышестоящее руководство должны анализировать отчеты и проводить регулярные встречи с руководителем проекта для критического анализа состояния проекта.

Руководителю проекта следует проводить частые дискуссии с разработчиками с тем, чтобы те имели объективное представление о текущем состоянии проекта. Разработчики должны предоставлять руководителю проекта официальные отчеты о готовности пакетов работ и таблицы учета времени, затраченного на выполнение работ.

*Отчеты о ходе проекта***.** Руководитель проекта должен регулярно (например, раз в месяц) представлять отчеты о ходе работ, в которых описано:

- техническое состояние;
- состояние ресурсов;
- соответствие расписанию;
- проблемы;
- финансовое состояние.

*Отчеты о готовности пакетов работ***.** По мере готовности пакета работ, ответственный за реализацию пакета член команды исполнителей (называемый «руководителем пакета работ») должен известить об этом руководителя проекта посредством отчета о завершении реализации пакета работ. Руководители проектов принимают предоставленный пакет и оформляют передачу документом, именуемым «сертификат выполнения пакета работ». Одним из способов документирования этого процесса – добавить поля о сдаче и приемки результатов выполнения пакетов работ в типовую форму описания пакетов работ.

*Учет времени***.** В организациях следует создать систему учета времени с тем, чтобы руководители проекта могли отслеживать временные затраты персонала, вязанные с реализацией проекта. В системе учета времени должны содержаться сведения по дням и часам о том, чем занимался каждый сотрудник за отчетный период. Эта информация необходима для подготовки руководителем проекта текущих отчетов о состоянии проекта. Хорошая система учета времени позволяет регистрировать усилия, затрачиваемые на выполнение от-

дельных работ в пакете, а не затраты на проект в целом. Наличие данных такой гранулярности позволяет упростить сбор данных о затратах, понесенных в ходе реализации проекта, и обеспечить создание баз данных, позволяющих получать более точные оценки в будущем.

Контрольные вопросы:

1. Сформулируйте основные задачи менеджера проекта?
2. В чем роль и зона ответственности руководителя программного проекта?
3. В чем состоит структура процесса планирования?
4. Каковы стадии проекта по стандарту и основные продукты по входным и выходным данным?
5. Какие основные виды пакетов работ можно выделить ?
6. Опишите AA и DD фазы процесса моделирования.
7. Как оцениваются объемы трудовых ресурсов и как они распределяются?
8. Как внешние приобретения возникают при производстве ПП и как можно оценить продолжительность работ?
9. Каковы критерии составления расписания работ?
10. В чем заключаются факторы риска, связанные с опытом и квалификацией команды?
11. Какие риски могут появиться при планировании программного продукта?
12. Какие технологические факторы риска могут возникнуть при производстве ПП?
13. Опишите внешние факторы источников риска.
14. Каковы метрики программных продуктов и как их определить?

ГЛАВА 2. МЕТОДЫ И ПЛАНЫ УПРАВЛЕНИЯ ПРОГРАММНЫМИ ПРОЕКТАМИ

2.1. Методы планирования проекта

Различают методы планирования проекта для поддержки следующих операций:

- моделирование процессов;
- оценивание ресурсов и продолжительности;
- построение сетевого графика операций;
- построение расписания.

Моделирование процессов. Целью моделирования процесса производства программного продукта является определение ролей; их зон ответственности; операций, а также указание входов и выходов операций. Часто такую модель называют «потоком работ» (англ.-workflow). В качестве изобразительных средств в таких моделях обычно используют текст и диаграммы. Примерами могут служить SADT-диаграммы, сети Петри; предикатная логика. Моделирование процессов, предполагает построение как статические, так и динамические модели.

Оценивание ресурсов и продолжительности. Выше упоминались методы оценивания, основанные на экспертных оценках. В их основе лежат оценки времени, затрачиваемого каждым из исполнителей в ходе реализации проекта. Альтернативными методами оценивания являются:

- использование исторических данных;
- COCOMO [3,4, 18];
- анализ функциональных точек [1,2, 19];
- анализ затрат по операциям [21-23];
- метод Делфи [25];
- оценка затрат на интеграционное и системное тестирование [20,28];
- оценка затрат на документирование [18].

Некоторые из этих методов основаны на использовании формул. При использовании формальных соотношений следует помнить положение, сформулированное А.Альбрехтом, автором метода функциональных точек: «Следует проводить различие между двумя типами оценок затрат, связанных с выполнением работ: первая оценка связана с анализом задач; вторая – с построением формулы. Первая оценка основана на анализе задач, обеспечивая, таким образом, команду проекта планом работ. Формулы рекомендуется использовать для того, чтобы выполнить валидацию результатов анализа задач, а также для

того, чтобы осуществить прогноз на базе первичных оценок». Барри Боэм дает схожие по смыслу советы, которые слишком часто игнорируются теми, кто использует его метод.

Использование исторических данных. Существо метода оценивания стоимости проекта на основе исторических данных заключается в сравнении текущего проекта с предыдущими проектами. Условиями применения этого метода являются:

- наличие точных данных о стоимости предыдущих проектов;
- схожесть предыдущих проектов с текущим проектом.

Если все проекты достаточно схожи, то в качестве оценки стоимости может быть принята средняя величина стоимости, определяемая по совокупности схожих проектов. В общем случае незначительное различие между проектами приводит к тому, что имеет место интервальная оценка вместо точечной. Оценка может быть выполнена после того, как будут выделены схожие проекты или пакеты работ.

Использование подхода, основанного на анализе исторических данных, предпочтительно в организациях, в которых выполняется серия однотипных проектов, в особенности, если в организациях налажена система сбора и хранения данных о затратах, связанных с реализацией каждого пакета работ. В этом случае экспериментальным данным нет равноценной замены [20].

Ограничениями этого подхода являются:

- невозможность учета в оценках влияния, оказываемого применением новых методов и инструментов;
- невозможность выделить в оценке влияния неэффективных операций, приведших к потерям.

Чтобы обойти это ограничение, необходимо, чтобы все специфические особенности проекта, которые влияют на величину стоимости, необходимо документировать, это позволит руководителям проектов соответствующим образом настраивать свои оценки. Одна из целей подготовки PHD (Project History Document) является передача последующим руководителям проектов данных о стоимости разработок, а также предоставить информацию о причинах, приведших к таким затратам.

Метод оценки стоимости программного продукта (COCOMO). Constructive Cost Model (COCOMO) есть формализованный метод оценки общей стоимости разрабатываемого программного продукта. Исходными данными для COCOMO является оценка числа строк исходного кода. Это является слабым местом метода, так как:

- приемлемая по точности оценка числа строк кода может быть получена лишь в конце стадии проектирования архитектуры, что с точки зрения реализации проекта является достаточно поздней оценкой;
- понятие «строка кода» может по-разному толковаться в зависимости от языка программирования и принятыми в конкретном проекте соглашениями;
- понятие «строка кода» неприменимо к некоторым современным технологиям программирования, например, к визуальному программированию.

Выделяют базовый, промежуточный и детальный варианты COCOMO. *Базовый* вариант использует простые формулы для оценки затрат на реализацию проекта (в человеко-месяцах), а также общую продолжительность реализации проекта в зависимости от ожидаемого количества строк кода. *Промежуточный* вариант уточняет базовый за счет дополнения его пятнадцатью параметрами, характеризующими усилия. Грубые погрешности оценок могут быть обусловлены неверным выбором значения упомянутых параметров. В работах [8-9] приводятся критерии, позволяющие повысить обоснованность выбора параметров. *Детальный* вариант основан на использовании независимой последовательности параметров для каждой стадии проекта.

Анализ функциональных точек — стандартный метод измерения размера программного продукта с точки зрения пользователей системы. Метод разработан Аланом Альбрехтом (Alan Albrecht) в середине 70-х гг. и впервые опубликован в 1979 г.[1]. Function Point Analysis (FPA) позволяет оценивать затраты на реализацию проекта на основе предполагаемой функциональности [6]. FPA рекомендуется использовать в конце фазы проектирования требований к программному продукту. Этот метод достаточно хорошо совмещается с методами структурного анализа, позволяя оценить общие затраты на реализацию программной системы.

Оценивание стоимости разработки выполняется на основе подсчета входов, выходов системы, а также хранилищ данных. На основе этих результатов в итоге получается показатель «Function Point» (FP). Значения FP в дальнейшем переводятся в число строк кода (с учетом предполагаемого языка реализации программного продукта). Полученное значение числа строк кода затем, используя COCOMO, переводится в оценку предполагаемых затрат и расписание работ.

Метод предназначен для оценки на основе логической модели объема программного продукта количеством функционала, востребованного заказчиком и поставляемого разработчиком. Несомненным достоинством метода является то, что измерения не зависят от технологической платформы, на кото-

рой будет разрабатываться продукт, и он обеспечивает единообразный подход к оценке всех проектов в компании.

При анализе методом функциональных точек надо выполнить следующую последовательность шагов (Рис.2.1):

1. Определение типа оценки.
2. Определение области оценки и границ продукта.
3. Подсчет функциональных точек, связанных с данными.
4. Подсчет функциональных точек, связанных с транзакциями.
5. Определение суммарного количества не выровненных функциональных точек (UFP).
6. Определение значения фактора выравнивания (FAV).
7. Расчет количества выровненных функциональных точек (AFP).

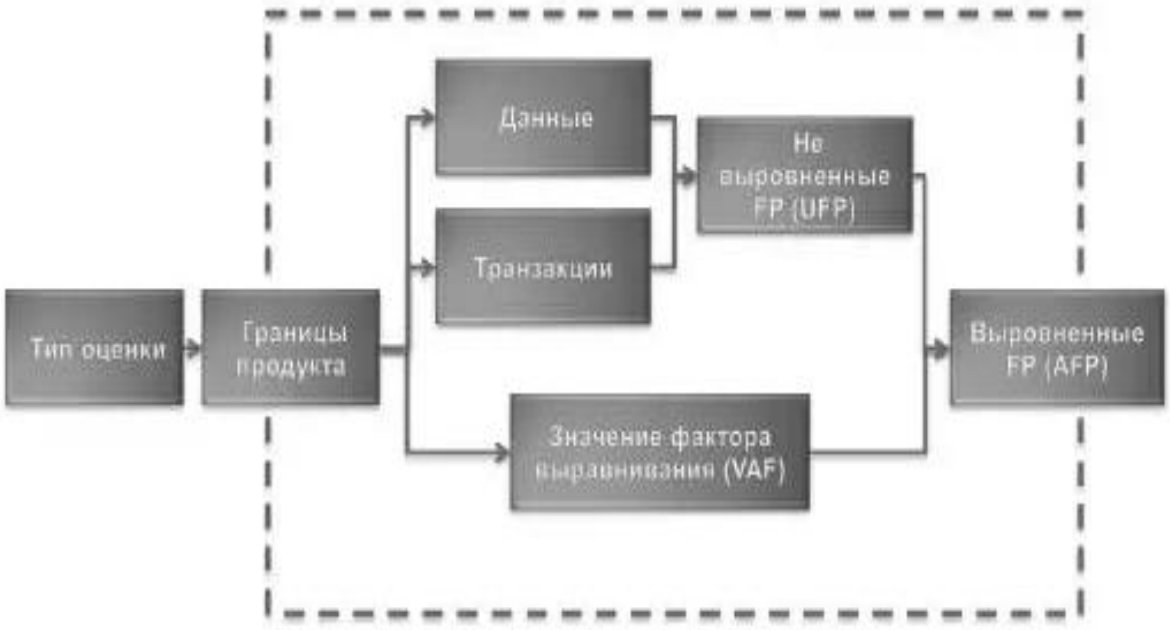


Рис.2.1. Процедура анализа по методу функциональных точек

Анализ затрат по операциям. Анализ затрат по операциям предполагает изучение затрат на реализацию каждой из операций по данным о ранее реализованных проектах. Результаты анализа позволяют:

- оценить долю затрат на каждую операцию в общих затратах на реализацию проекта;
- подтвердить обоснованность удельной доли расходов на каждую из операций;

- подтвердить обоснованность оценок относительно доли предполагаемых затрат на операцию в будущих проектах на основе данных о реализованных операциях.

Стадии проекта можно рассматривать как операции высокого уровня. Известна технология оценивания затрат на разных стадиях проекта, именуемая «phase percent method». Как и при использовании всех методов, ориентированных на обработку исторических данных, необходимо обеспечивать сопоставимость сырых данных по таким признакам, как квалификация персонала; риски проекта; использование новых методов и технологий; использование более эффективных способов реализации работы. Например, использование новых методов анализа и проектирования и CASE-инструментов требует увеличения доли затрат на стадии формирования спецификации системных требований и проектирования архитектуры.

Метод Делфи (англ.- Delphi) обычно используется при оценивании стоимости проекта. Допущением этого метода является то, что эксперты, независимо друг от друга, стремятся дать «хорошую» оценку. Реализация метода требует наличия нескольких экспертов, способных дать оценку стоимости, а также наличие координатора процесса. Основными шагами при реализации метода являются следующие:

- координатор представляет эксперту для ознакомления спецификацию, а также знакомит с формой, в соответствие с которой он должен дать оценку;
- каждый из экспертов анонимно заполняет предложенную форму, в которой дает объяснение данной им оценке (эксперт может задавать вопросы координатору, но не может обсуждать проблему с другими экспертами);
- если консенсус не достигнут, координатор готовит сводный обзор всех оценок, распространяет его среди экспертов, после чего процесс повторяется, начиная с первого шага.

В свободном обзоре должны приводиться только результаты без объяснения того, что лежит в их основе. Если же после первого прохождения цикла получаются схожие оценки, проводится совещание, посвященное обсуждению независимых оценок с тем, чтобы прийти к общей согласованной оценке.

Оценка затрат на интеграционное и системное тестирование. Затраты на интеграционное и системное тестирование в основном зависят от следующих факторов:

- количества и критичности дефектов, выявленных в программном продукте после тестирования программных единиц;

- количества и критичности дефектов, допускаемых в поставляемом продукте;
- количества интеграционных и системных тестов, а также их продолжительности;
- среднего времени исправления дефекта.

Усилия, связанные с тестированием интеграции и системным тестированием, могут быть оценены на основе модели тестирования интеграции; предположений относительно числа дефектов, выявленных при тестировании программных модулей, а также на основе оценки средних затрат на исправление одного дефекта.

Затраты усилий на исправление дефектов могут меняться в очень широких пределах. Результаты практической деятельности позволяют утверждать, что устойчивой оценкой затрат на исправление одной ошибки в коде является 0,5 человеко-дня. В [ESA PSS-05-08] отмечается, что 20-50% дефектов выявляются на стадии производства программного продукта. От пяти до 15% дефектов на 1000 строк кода приходится на начало тестирования интеграции. От одного до четырех дефектов по 1000 строк кода остается после завершения системного тестирования.

Оценка затрат на документирование. Существует множество определений программного продукта. В том числе можно утверждать, что программный продукт состоит из документации и кодов. Документация является критически важным результатом создания программного продукта, а не «накладными расходами». При оценивании объема работ, связанного с реализацией программного продукта, должны учитываться не только количество кодов, которые необходимо разработать, но также и объем документации.

Исследования, проведенные Б.Боэм [8, 9], показали, что затраты на одну страницу документации лежат в диапазоне от двух до четырех часов. Б.Боэм также установил, что соотношение между объемом кода и числом страниц документации лежит в диапазоне 10 – 150 страниц текста на 1000 строк кода, причем значение медианы составляет порядка 50. Средняя результативность составляет 20 строк кода в день, включая затраты на подготовку документации. Из этих цифр следует, что примерно половину времени программные инженеры тратят на подготовку документации (действительно, 50 страниц документации на 1000 строк кода означает одну страницу документации на 20 сток кода; на подготовку одной страницы нужно четыре часа, т.е. половина рабочего дня).

2.2. Отображение хода проекта

К методам отображения хода проекта относятся:

- таблица (графики) выполнения работ;
- диаграмма выполнения проекта;
- диаграмма достижения вех.

Таблица (графики) выполнения работ. Таблица (графики) выполнения работ используются для демонстрации того, сколько ресурсов было израсходовано для выполнения каждого пакета работ и сколько ресурсов еще необходимо потратить. Для каждого из пакетов, входящих в состав WBS, должны быть сделаны предварительные оценки объемов необходимых ресурсов. Эти оценки должны быть представлены в Плане Управления Программным Проектом (англ. – Software Project Managing Plan -SPMP). В конце каждого отчетного периода необходимо выполнить сбор следующих параметров.

1. Предварительная оценка потребных ресурсов.
2. Фактическое потребление ресурсов за отчетный период.
3. Потребление ресурсов на конец отчетного периода нарастающим итогом (кумулятивные затраты).
4. Оценка дополнительных ресурсов, необходимых для завершения реализации пакета работ.

Суммирование третьего и четвертого показателей позволяет получить оценку на следующий месяц, т.е. содержание первого пункта приведенного списка. Оценки дополнительных ресурсов, необходимых для завершения выполнения пакета работ, должны переопределяться в конце каждого отчетного периода, ***но не рассчитываться путем вычитания из кумулятивных затрат на предыдущий отчетный период***. Затраты на реализацию пакетов работ должны быть представлены в «таблице выполнения проекта», пример которой приведен на рис. 2.2. В этой таблице в колонках один и два приведены идентификаторы и имена работ. Плановые оценки затрат, отнесенные к пакетам работ, представлены в столбце три. В последних колонках представлены (для каждого месяца) значения предыдущих оценок (PrEst), затраты за отчетный период (ExpP); затраты нарастающим итогом (Cum) и оценки дополнительных ресурсов (ToGo). В последней колонке представлены текущие оценки затрат, связанные с выполнением пакета работ. Из приведенной таблицы видно, что затраты, связанные с реализацией пакетов работ 2210 и 2220, изначально были занижены. Недостатки ресурсов в 10 человеко-дней для пакета 2210 были выявлены и скорректированы в январе. Оценки для пакета 2220

были частично (на пять человеко-дней) скорректированы в январе. Недостаток ресурсов вновь увеличился на пять человеко-дней в феврале.

WPid	Name	Plan	January		February		March		Total
			PrEst	ToGo	PrEst	ToGo	PrEst	ToGo	
			ExpP	Cum	Exp	Cum	Exp	Cum	
2210	Logical	20	20	10	30	0	30	0	30
	Model		20	20	10	30	0	30	
2220	Prototype	30	30	15	35	5	40	0	40
			20	20	15	35	5	40	
2230	SRD	20	20	20	20	10	20	0	20
	draft		0	0	10	10	10	20	
2240	SRD	5	5	5	5	5	5	0	5
	final		0	0	0	0	5	5	
2200	SR phase	75	75	50	90	20	95	0	95
	total		40	40	35	75	20	95	

Рис.2.2. Таблица выполнения проекта
Источник: [ESA-PSS-05-08]

Графики хода проекта позволяют представить в удобной форме в обобщенном виде тренды стоимости выполнения работ (рис.2.3.).

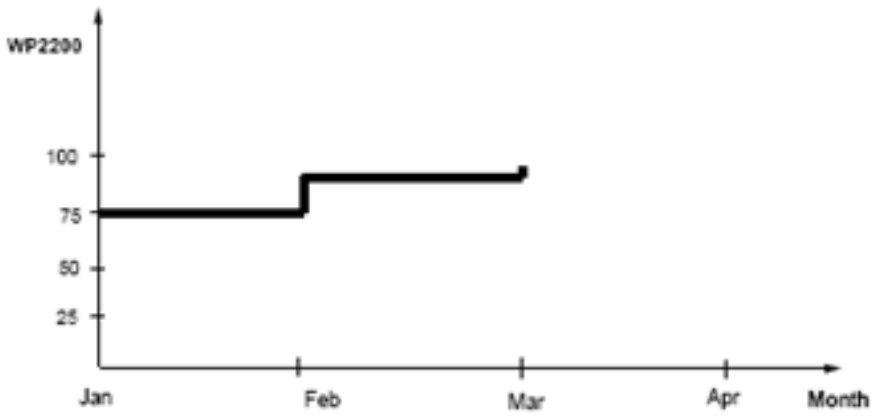


Рис.2.3 График хода проекта, соответствующий данным рис. 2.2
Источник: [ESA-PSS-05-08]

Построение таблиц и графиков, отражающих ход проекта, соответствующих каждому отчетному периоду, позволяют:

- повысить точность получаемых оценок состояния проекта;
- оценить дополнительный объем ресурсов, необходимый для завершения выполнения пакетов работ.

Ежемесячное оценивание того, что уже выполнено и что еще предстоит сделать, важно с точки зрения обеспечения хорошего оперативного управления ресурсами проекта. Раннее выявление тенденций в перерасходе ресурсов позволяет своевременно принять меры, направленные на исправления положения дел.

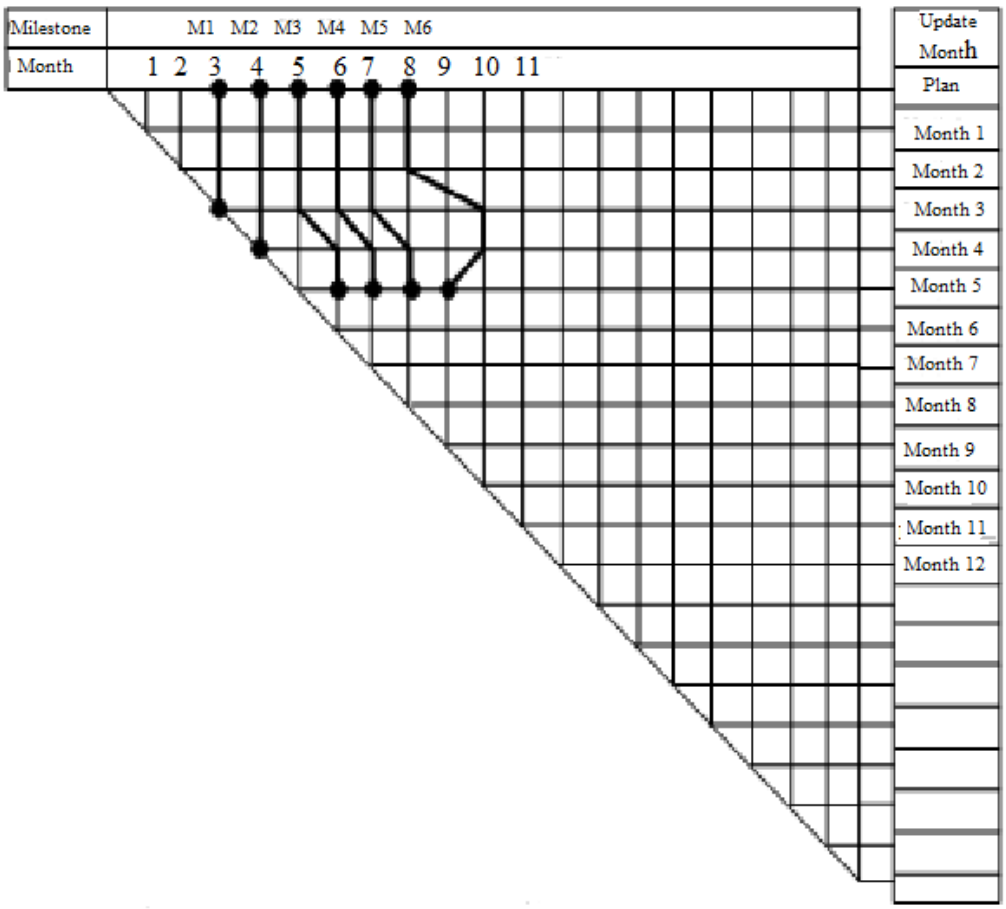
Графики трендов вех используются для формирования отчета о том, когда веха достигнута (либо будет достигнута). Вначале выполняется первичная оценка даты достижения вехи. Эти оценки помещаются в раздел «расписание» Плана управления программным проектом (Software Project Management Plan - SPMP). По завершении каждого отчетного периода собирают следующие данные:

- предварительные даты достижения вех;
- новые оценки дат достижения вех.

Полученные данные следует представить в виде графика, отражающего тренд изменения в датах достижения вех. Пример диаграммы тренда приведен на рис. 2.4.

Вертикальная ось показывает, что отчетным периодом является один месяц. Рисунок демонстрирует состояние дел через пять месяцев после начала проекта. Вехи M_1 и M_2 были достигнуты согласно расписания. Даты достижения вех M_3, M_4, M_5 оказались сдвинутыми относительно плановых дат на месяц, т.к. реализация «integration stage 2» заняла на один месяц больше, чем планировалось. Дата достижения вехи M_6 (конец стадии детального проектирования) изначально была сдвинута на два месяца из-за того, что была объявлена задержка поставки симулятора на два месяца. Обращение с жалобой на поставщика симулятора привела к тому, что дата поставки вновь была сдвинута, но уже на месяц ранее.

В данном примере показано, как продемонстрировать соответствие реального хода проекта его расписанию. Наклонные линии отражают изменения. График трендов является мощным инструментом, способствующим пониманию того, что происходит в проекте. Сдвиг вех в каждом отчетном периоде свидетельствует о существовании серьезных проблем. График трендов вех отражает даты прохождения контрольных точек в рамках текущего одобренного плана. Ранее разработанные, но уже отмененные планы в рассмотрение не принимаются. Отмененные планы следует пересматривать при обновлении Плана Управления Программным Проектом.



	key to milestones and dates	Plan date	Plan month
M1	DDD complete	17/03	3
M2	Integration stage 1 end	28/04	4
M3	Integration stage 2 end	26/05	5
M4	Integration stage 3 end	23/06	6
M5	Integration stage 4 end	21/07	7
M6	DD phase end	28/08	8

Рис. 2.4. Пример диаграммы тренда для DD фазы проекта
Источник: [ESA-PSS-05-08]

2.3. Состав плана управления программным проектом

Все операции, входящие в состав программного проекта, должны быть документально представлены в Плане Управления Программным Проектом (SPMP). Этот документ служит основой контроля хода программного проекта. План содержит четыре раздела, каждый их которых соотнесен с определенной стадией программного проекта. Эти разделы называются:

- план управления программным проектом на стадии проектирования системных требований;

- план управления программным проектом на стадии проектирования архитектуры;
- план управления проектом на стадии детального проектирования;
- план управления проектом на стадии передачи в эксплуатацию.

Каждый раздел Плана Управления Программным Проектом (SPMP) должен:

- содержать описание проектной организации;
- содержать описание руководящего (административного) процесса;
- содержать эскизное (в общих чертах) описание технического подхода, в особенности методов, инструментов, технологий;
- содержать описание WBS;
- содержать оценки затрат, связанных с реализацией проекта;
- содержать описание сети работ с указанием предполагаемого времени начала и завершения работ;
- содержать оценки рисков проекта.

Стиль. План управления проектом должен:

- быть ясным;
- быть полным;
- быть кратким;
- допускать внесение изменений в содержание;
- иметь согласованные между собою части.

2.4. Вспомогательная информация

Разделы плана, посвященные разработке системных требований, проектирования архитектуры; детального проектирования и передачи продукта в эксплуатацию реализуются в разное время. Должен обеспечиваться независимый контроль конфигурации каждого раздела. Каждый из разделов должен содержать следующую вспомогательную информацию:

a – введение;

в – содержание;

c – информацию о статусе документа;

d – информацию о изменениях, внесенных в документ с момента последнего издания.

Типовая структура раздела «содержание» следующая.

Введение

1. *Общая информация о проекте*

- цели;

- перечень продуктов, поставляемых заказчику в результате реализации проекта;
- модель жизненного цикла;
- главные операции;
- контрольные точки (вехи);
- требуемые ресурсы;
- расписание;
- бюджет.

Приводится укрупненный план реализации проекта. В следующих разделах может содержаться уточненный план работ.

2. Продукты, поставляемые в результате реализации проекта

В этом разделе приводится перечень всех документов, планов и версий программного продукта, поставляемых в ходе реализации проекта. В этот список должны быть включены такие объекты, как прототипы, демонстрационные материалы, инструментальные средства.

3. Ссылочные материалы

В этом разделе приводится полный перечень всех цитируемых при разработке плана документв, а также документов, на которые осуществляются ссылки. В описании документов следует указывать его наименование, автора, дату выпуска. Каждый документ должен обозначаться как цитируемый, либо ссылочный. Следует указывать номера отчетов, номера журналов и издательство в случае использования таких документов.

4. *Определения и сокращения*

В этом разделе приводятся определения всех терминов, список и расшифровки всех сокращений, используемых в плане. Допускается вместо этого указывать ссылки на документы, в которых приводятся определения и сокращения.

Пример описания разделов проекта

1. Организация проекта

1.1. Модель процесса

В этом разделе приводится описание операций, выполняемых на некоторой стадии проекта, а также входов и выходов операций. Описание должно содержать перечисление основных функций проекта, которые необходимо учитывать при определении общей продолжительности проекта. К числу таких функций относятся: управление проектом; управление конфигурацией; верификация и валидация; подтверждение качества, а также основные функ-

ции, которые необходимо выполнить для достижения целей проекта. Описание процесса может быть текстовым, либо графическим.

1.2. Организационная структура

В этом разделе приводится описание внутренней структуры управления проектом. Инструментом графического представления связей отчетов, контроля и коммуникаций являются органограммы. Типовыми ролями в структуре системы управления являются:

- руководитель проекта;
- лидер команды;
- программисты;
- библиотекарь программных компонентов;
- инженер по обеспечению качества программного продукта.

1.3. Организационные границы и интерфейсы

В этом разделе должны быть определены взаимоотношения между командой проекта и группами внешних правообладателей при реализации определенной фазы проекта. К внешним правообладателям относятся:

- вышестоящая организация;
- клиенты;
- пользователи;
- соисполнители;
- поставщики;
- организации, выполняющие независимую верификацию и валидацию программного продукта;
- независимая организация, осуществляющая оценку качества программного продукта.

Процедуры контроля каждого внешнего интерфейса и зоны ответственности при контроле должны быть документально оформлены. Например:

- имя документа, обеспечивающего контроль интерфейсов Interface Control Document – ICD);
- ответственные за обеспечение согласованного содержания ICD;
- сотрудник, ответственный за обеспечение доступа к ICD.

2. Зоны ответственности

В этом разделе должны быть описаны роли, перечисленные в организационной структуре, и их зоны ответственности. Должно быть дано краткое описание роли и список вопросов, за решение которых отвечает роль.

3. Процессы управления

3.1 *Цели и приоритеты управления*

В этом разделе должны быть определены цели управления и соответствующие им приоритеты на выделенной стадии проекта. Также должны быть приведены результаты обсуждений, в ходе которых был найден баланс целей.

3.2 *Допущения, взаимосвязи и ограничения*

В этом разделе для каждой стадии проекта должны быть определены:

- допущения, положенные в основу разработки плана;
- внешние события, способные повлиять на ход проекта;
- ограничения проекта.

Технические вопросы упоминаются только в том случае, если они существенны с точки зрения реализации плана. Часто достаточно трудно явно определить допущения, взаимосвязи и ограничения. Иногда разумно не пытаться выделять классы ограничений, а просто составить их перечень. Например:

- ограничения на размер бюджета;
- ограничения расписаний;
- ограничения на местоположение участников проекта (например, некий участник должен находиться в определенном помещении);
- программное и аппаратное обеспечение, которое должно быть закуплено в ходе выполнения проекта;
- возможности симуляторов и других устройств, используемых в ходе испытаний;
- возможности внешних систем, с которыми должна взаимодействовать создаваемая программная система.

3.3. *Управление риском*

В этом разделе должны быть определены риски проекта, а также определены действия, которые следует предпринять на определенной стадии проекта в случае реализации опасностей. Для описания рисков может использоваться таблица следующей структуры (пример заимствован из [ESA-PSS-05-08]).

Risk	Description	Prob.	Action	Decision Date	Impact
1	New user requirements	High	Change to evolutionary development	1/Jun/1997	High
2	Installation of air conditioning	Medium	Relocate staff while work is done	1/April/1998	Medium

Рис. 2.5

3.4. *Механизмы мониторинга и контроля*

В этом разделе должны быть определены все форматы документов, используемых при оперативном управлении проектом. Либо должны быть даны ссылки на источники, в которых приводятся описания этих документов. В этом разделе также должны быть представлены следующие сведения:

- периодичность проведения встреч инициаторов проекта с руководителем проекта для обсуждения хода реализации проекта;
- периодичность представления отчетов о ходе проекта;
- об изменениях, вносимых в типовую структуру отчета о ходе проекта (см. следующую главу);
- общая политика проведения анализа и аудита (детальная информация по этим вопросам должна быть представлена в Плане Валидации и Верификации Программного Проекта).

3.5. *План управления персоналом*

В этом разделе плана должны быть определены имена, роли, уровни квалификации персонала, привлекаемого к реализации проекта на определенной его стадии. Информация о загрузке персонала должна приводиться за каждый отчетный период. Также ежемесячно должна приводиться информация об общей численности персонала, задействованного в реализации проекта.

4. Технологический процесс

4.1. *Методы, инструменты и технологии*

В этом разделе должно быть приведено описание методов, инструментов и технологий, используемых при реализации определенной стадии проекта.

4.2. *Документация на программные продукты*

В этом разделе приводится план разработки документации на определенной стадии проекта, либо дается ссылка на документ, содержащий описание плана. Для каждого из создаваемых документов в плане разработки документации должно быть определено:

- имя документа;
- анализ требований;
- утвержденные требования.

Некоторые документы относятся к категории выходных результатов. Другие являются внутренними документами, такими как технические записи, либо планы (например, План Управления Конфигурацией Программного Продукта). Все документы должны быть пронумерованы. План разработки документации может содержать описание форматов и структуры документов, либо содержать ссылку на документы, где приводится соответствующая информация.

4.3 ***Функции, поддерживающие проект***

В этом разделе содержатся общие сведения о планах разработки следующих поддерживающих функций:

- управления конфигурацией программного продукта;
- верификации и валидации программного продукта;
- обеспечения качества программного продукта.

В этом разделе также должны даваться ссылки на План Управления Конфигурацией программного продукта; План Верификации и Валидации Программного Продукта; План Управления Качеством Программного Продукта.

5. Пакеты работ, расписание и бюджет

5.1. ***Пакеты работ.*** В этом разделе описывается представление операций в виде пакетов работ. Раздел должен начинаться описанием WBS с тем, чтобы дать представление о иерархической взаимосвязи пакетов работ. Каждый блок должен иметь имя и идентификатор. И наоборот, иерархическая модель работ должна содержать перечень имен и идентификаторов пакетов работ. Полное описание пакетов работ может быть приведено в этом разделе, либо вынесено в приложение. Каждое описание пакета работ должно содержать:

- наименование пакета работ;
- ссылочный номер пакета работ;
- организацию, ответственную за разработку пакета;
- основные составляющие операции;
- сведения об управляющем разработкой пакета;
- дату начала работ по реализации пакета;
- дату завершения работ по реализации пакета;
- операции, связанные с реализацией пакета;
- описание предполагаемых выходных результатов.

Пакеты работ должны быть определены для всех операций, включая операции, связанные с управлением проектом; управлением конфигурацией; верификацией и валидацией; подтверждением качества программного продукта.

5.2. ***Взаимосвязи.*** Здесь представлен сетевой график пакетов работ.

5.3. ***Требуемые ресурсы.*** Здесь представлена информация о составе и объему ресурсов, необходимых для реализации основного, вспомогательного и обеспечивающего процессов.

5.4. ***Бюджет и распределение ресурсов.*** Здесь представлена информация о стоимости каждой из работ.

5.5. ***Расписание.*** Здесь представлено расписание работ.

Материалы, не попадающие в предлагаемую структуру, следует помещать в дополнительные приложения. Если материалы, соответствующие какому-либо разделу, в конкретном проекте отсутствуют, в соответствующее место следует поместить фразу «***не используется***» при сохранении нумерации разделов.

Контрольные вопросы:

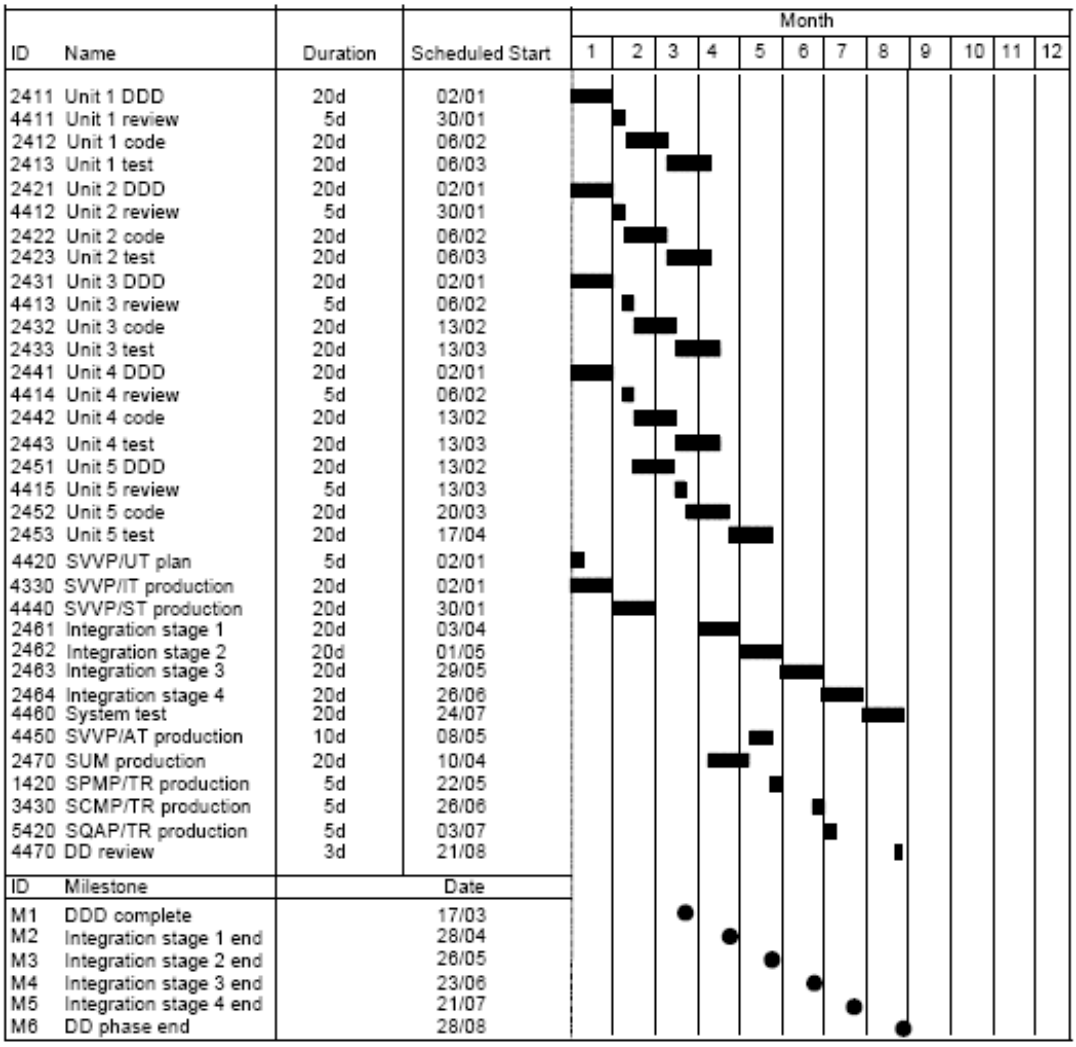
1. Какие технологии используются для моделирования процесса производства программного продукта?
2. Какие методы оценивания ресурсов и их продолжительности используются при реализации проектов и чем они отличаются?
3. Какие методики используются для отображения хода проекта и чем они отличаются?
4. Каковы основные разделы управления программным проектом?
5. Опишите структуру вспомогательной информации для разделов разработки технического задания, проектирования и внедрения программного продукта.

ГЛАВА 3. ИНСТРУМЕНТЫ УПРАВЛЕНИЯ И ПРОЕКТИРОВАНИЯ ПРОГРАММНЫМ ПРОЕКТОМ

3.1. Инструменты планирования программного проекта

а). **Метод сетевого планирования**. Program Evaluation and Review Technique (PERT) – диаграммы являются широко используемым методом проектирования сети операций.

б). **Методы, основанные на расписаниях работ**. Расписание проекта определяет начало пакета работ; продолжительность его выполнения; дату прохождения каждой вехи. Широко используемым инструментом для формирования расписаний являются диаграммы Ганта. На рис.3.1 в качестве примера приведено расписание операций, соответствующих стадии детального проектирования.



ID	Name	Duration	Scheduled Start
2411	Unit 1 DDD	20d	02/01
4411	Unit 1 review	5d	30/01
2412	Unit 1 code	20d	06/02
2413	Unit 1 test	20d	06/03
2421	Unit 2 DDD	20d	02/01
4412	Unit 2 review	5d	30/01
2422	Unit 2 code	20d	06/02
2423	Unit 2 test	20d	06/03
2431	Unit 3 DDD	20d	02/01
4413	Unit 3 review	5d	06/02
2432	Unit 3 code	20d	13/02
2433	Unit 3 test	20d	13/03
2441	Unit 4 DDD	20d	02/01
4414	Unit 4 review	5d	06/02
2442	Unit 4 code	20d	13/02
2443	Unit 4 test	20d	13/03
2451	Unit 5 DDD	20d	13/02
4415	Unit 5 review	5d	13/03
2452	Unit 5 code	20d	20/03
2453	Unit 5 test	20d	17/04
4420	SVVP/UT plan	5d	02/01
4330	SVVP/IT production	20d	02/01
4440	SVVP/ST production	20d	30/01
2461	Integration stage 1	20d	03/04
2462	Integration stage 2	20d	01/05
2463	Integration stage 3	20d	29/05
2464	Integration stage 4	20d	26/06
4460	System test	20d	24/07
4450	SVVP/AT production	10d	08/05
2470	SUM production	20d	10/04
1420	SPMP/TR production	5d	22/05
3430	SCMP/TR production	5d	26/06
5420	SQAP/TR production	5d	03/07
4470	DD review	3d	21/08

ID	Milestone	Date
M1	DDD complete	17/03
M2	Integration stage 1 end	28/04
M3	Integration stage 2 end	26/05
M4	Integration stage 3 end	23/06
M5	Integration stage 4 end	21/07
M6	DD phase end	28/08

Рис.3.1. Диаграмма Ганта
Источник: [ESA-PSS-05-08]

Исходные данные для расписания приведены в разделе 1.2.1.

Основные цели использования инструментов планирования программного проекта

К основным целям относятся:

- выделение пакетов работ;
- определение ресурсов, выделяемых для выполнения каждого из пакетов работ;
- определение фактически доступных ресурсов;
- определение продолжительности выполнения каждого из пакетов работ;
- построение сетевого графика выполнения работ (PERT-диаграммы);
- выделение критического пути;
- построение расписания в виде диаграммы Гантта.

Использование инструментов должно обеспечивать решение следующих задач:

- подсчет общих объемов необходимых ресурсов;
- построение сценариев использования ресурсов;
- разделение проекта на подпроекты;
- выявление конфликтных ситуаций, связанных с использованием ресурсов, либо перерасходованием ресурсов;
- легкость интеграции с текстовыми редакторами.

Общими недостатками инструментов планирования широкого назначения являются:

- фиксированные параметры расписания, которые необходимо сдвигать если деятельность кого-либо из участников отклоняется от плановой;
- невозможность ручного регулирования парциального распределения ресурсов (например, конфликты при распределении ресурсов в случае, когда персонал вынужден распределять свое время между параллельно реализуемыми пакетами работ);
- необходимость внесения изменений в идентификаторы пакетов каждый раз, когда включается новый пакет работ;
- слишком большое число пересекающихся линий в сетевых графиках работ.

Инструменты моделирования процессов предназначены:

- поддерживать составление описания процедур;
- содержать библиотеку «шаблонов процессов», которые могут быть настроены с учетом особенностей конкретного проекта;

- позволять ввести модель процесса в инструмент, поддерживающий процесс.

Известны несколько специализированных инструментов моделирования процессов. Эффективными инструментами моделирования являются DFD и ERD – диаграммы.

Инструменты получения оценок стоимости. Информацию о стоимости проектов следует хранить с тем, чтобы руководитель проекта мог оценить стоимость своего проекта по сравнению со стоимостью схожих проектов. В основе набора инструментов предварительного оценивания стоимости лежит измерение масштаба программного продукта (например, посредством числа строк кода, либо числом функциональных точек). Уточнение этих оценок происходит с учетом, например, таких показателей как требования к надежности, квалификации программистов. Кроме того, инструменты для оценивания стоимости программного продукта позволяют:

- выполнить исследование типа «что-если» применительно к стоимости и затратам времени;
- выполнить оценку разумной продолжительности проекта;
- выполнить разработку интервальных оценок параметров (вместо точечных оценок), увеличивая за счет этого реалистичность оценок;
- получать объяснение того, как были получены оценки;
- производить оценивание при пропущенных данных;
- заменять значения прогнозных оценок фактическими данными по мере их появления, что способствует уточнению оценок по мере продвижения проекта;
- вернуться к более ранним стадиям процесса оценивания стоимости и повторить реализацию процесса с измененными исходными данными.

3.2. Инструменты поддержки оперативного управления процессом

Инструменты поддержки оперативного управления процессом позволяют автоматизировать выполнение операций, необходимых с точки зрения оперативного управления процессом. Использование инструментов возможно лишь при наличии формальной модели процесса, представленной в виде, допускающем компьютерную обработку. Отсутствие комплексной модели процесса реализации программного продукта накладывает ограничение на возможность применения инструментов поддержки процесса.

Инструменты поддержки процесса могут быть интегрированы во внешнюю, по отношению к ним, среду, именуемую «программной инженерией»;

могут координировать использование других инструментов, применяемых при создании программного продукта; поддерживать руководство и оперативное управление разработчиками.

Инструменты поддержки процесса должны предоставлять следующие возможности:

- быстро строить модель процесса;
- подтверждать адекватность модели;
- визуализировать статус модели процесса.

Кроме того, инструменты поддержки процесса должны обеспечивать:

- интерфейсы между моделями процесса и акторами;
- интерфейсы между инструментами планирования проекта, что позволит отслеживать реализацию плана и связанное с этим расходование ресурсов.

Двумя простыми и эффективными инструментами поддержки управления рисками проекта являются:

1) таблица рисков;
2) матрица рисков.

Таблица рисков. Формирование таблицы рисков предполагает реализацию следующей последовательности шагов.

1. Составление перечня рисков проекта.
2. Оценивание вероятности каждого риска.
3. Описание действий по снижению рисков в рамках выбранного подхода, либо при принятии решений по переходу на альтернативный способ реализации проекта.
4. Определить даты принятия решений.
5. Описать степень возможного воздействия на проект в случае реализации источника риска.

Пример таблицы рисков

Таблица 3.1

Риск	Описание	Вероятность	Действия	Дата принятия решения	Воздействия
1	Новые требования пользователя	Высокая	Переход к модели эволюционного развития	1.07.13	Сильное
2	Установка кондиционера	Средняя	Перевод персонала в другие помещения на период выполнения работ	1.04.14	Среднее

Матрица рисков. Формирование матрицы рисков предполагает реализацию следующей последовательности шагов.

1. Составление перечня рисков проекта.
2. Определение вероятности реализации каждого источника риска.
3. Описание степени возможного воздействия.
4. Размещение рисков в соответствие с их вероятностью и степенью воздействия.

Пример матрицы рисков

Таблица 3.2

Вероятность	Высокая			1
	Средняя		2	
	Низкая	3		
		Слабое	Среднее	Сильное
		СТЕПЕНЬ ВОЗДЕЙСТВИЯ		

3.3. Инструментальные средства проектирования

Практически любая современная крупная программная система разрабатывается с применением CASE-технологий по крайней мере на этапах анализа и моделирования, что связано с большой сложностью данной проблематики и со стремлением повысить эффективность работ.

Целью систем анализа и проектирования является определение системных требований и свойств, создание проекта и архитектуры информационной системы, а также детальная «калька» проекта, включающая алгоритмы и определения структур данных.

Системы проектирования баз данных обеспечивают логическое моделирование данных, автоматическое преобразование моделей данных в Третью Нормальную Форму, автоматическую генерацию схем Базы Данных и описаний форматов файлов на уровне программного кода.

В таблице 3.3 приведены некоторые CASE- средства анализа и проектирования программных систем.

Средства проектирования

Таблица 3.3

Система проектирования	Производитель ПО	Нотация DFD	Поддержка	
			Поддержка методологии	Поддержка диаграмм
Системы анализа и проектирования				
BPWin	Logic Works	Гейн-Сарсон	IDEF0/SADT IDEF3/SADT	Функциональной декомпозиции
BPMN	Business Process Management Initiative	Графическая нотация BPMN	Собственная	Бизнес-процессов

Продолжение табл. 3.3

ProKit *Workbench	MDIS	Гейн-Сарсон	Собственная STRADIS	Потоков данных в нотации Гейна-Сарсона, сущность-связь, структурные карты Константайна
CASE. Аналитик	Эйтекс	Гейн-Сарсон	структурного системного анализа Гейн-Сарсона	функциональной декомпозиции, потоков данных, управляющих потоков, структуры данных, сущность-связь
CASE /4/0	MicroTOOL	Йодан (расшир.)	анализа и проектирования систем реального времени Уорда-Меллера	Функциональной декомпозиции, потоков данных, переходов состояний, карты Джексона
Design/IDEF	Meta Software	-	IDEF0, IDEF1 IDEF1X, IDEF/CPN	сущность-связь
Visible Analyst Workbench	Visible Systems	Гейн-Сарсон, Йодан	Информационного моделирования Мартина	Функциональной декомпозиции, сущность-связь, потоков данных в нотации Йодана и Гейна-Сарсона, структурные карты Константайна
Rational Rose	Rational Software	UML, также поддерживается нотация Буча и OMT-2.	Собственная методология "Rational Objectory Process", UML	вариантов использования, взаимодействия объектов, последовательности взаимодействий, переходов состояний, классов, пакетов, компонентов, размещения
UML 2.x	Object Management Group	Собственная	UML	Классов, компонентов, композитной/ составной структуры, кооперации (UML 2.0), объектов, пакетов, профилей (UML 2.2), деятельности, состояний, прецедентов, коммуникации (UML 2.0) / кооперации (UML 4.x), обзора взаимодействия (UML 2.0), последовательности, синхронизации (UML 2.0)
ARIS Toolset	Software AG	Нотация ARIS eEPC	UML	сущность-связь, описания процессов, информационных потоков
Системы проектирования баз данных и файлов				
Designer/2000	Oracle	Гейн-Сарсон	Собственная CASE*Method	сущность-связь, потоков данных, иерархии функций, взаимодействия модулей структурные карты Джексона
ERWin	Logic Works	Гейн-Сарсон	IDEF3/SADT	сущность-связь

Окончание табл. 3.3

EasyCASE	Evergreen CASE Tools	Гейн-Сарсон, Йодан	структурного системного анализа Гейн-Сарсон	сущность-связь, потоков данных, переходов состояний, структурные карты в нотации Константайна
Vantage Team Builer	CAYENNE	Йодан	Структурного анализа и проектирования Чена и Йодана,	сущность-связь в нотации Чена, потоков данных в нотации Йодана, переходов состояний, структурные карты Константайна
Chen Toolkit	Chen & Asssociates	Чена	Чена	сущность-связь, потоков данных, переходов состояний
S-Designor	Sybase/ Powersoft		Информационного моделирования	сущность-связь
SILVERRUN	Computer Systems Advisers	произвольная	DATARUN	потоков данных BPM, сущность-связь ERX и RDM

Эти системы поддерживают те или иные методологии анализа и проектирования информационных систем, которые определяют руководящие указания для оценки и выбора проекта программного продукта, последовательность шагов выполнения работы, правила распределения и назначения операций и методов.

Кроме перечисленных систем проектирования, разработаны различные инструментальные CASE–средства: разработки приложений – JAM (JYACC), PowerBuilder (Sybase), New Era (Informix), SQL Windows (Gupta) и др.; планирования и управления проектом – Microsoft Project, SE Companion; тестирования – Quality Works (Segue Software); документирования – SoDA (Rational Software), которые позволяют автоматизировать процессы проектирования, планирования, разработки, тестирования, документирования информационной системы моделирования.

Инструментальные средства, предназначенные для проектирования и функционального моделирования информационных систем, могут быть отнесены к одной из следующих категорий:

- локальные, поддерживающие один-два типа моделей и методов (Design/IDEF, ProCap, S-Designor, “CASE. Аналитик”);
- малые интегрированные средства моделирования, поддерживающие несколько типов моделей и методов (ERwin, BPwin, BPMN) [8, 41, 45] ;
- средние интегрированные средства моделирования, поддерживающие от 4 до 10—15 типов моделей и методов (Rational Rose, Paradigm Plus, Designer/2000) [46];

- крупные интегрированные средства моделирования, поддерживающие более 20 типов моделей и методов (ARIS Toolset).

Контрольные вопросы:

1. Какие методики и инструменты используются для планирования программного проекта?
2. Каковы основные цели использования инструментов планирования программного проекта?
3. Какие возможности предоставляют инструменты поддержки управления проектом?
4. Опишите инструменты управления рисками проектов.
5. Какие инструментальные средства анализа и проектирования используются при разработке программных продуктов и каковы их возможности?

ГЛАВА 4. ПРОГНОЗИРОВАНИЕ ЭКОНОМИЧЕСКИХ ХАРАКТЕРИСТИК ПРОИЗВОДСТВА ПРОГРАММНЫХ ПРОДУКТОВ

Какая бы технология проектирования и реализации программного продукта не использовалась, оценка размера будущего продукта является весьма важной, поскольку она является основой оценивания реальных экономических характеристик производства. Нереальные ожидания участников проекта, основанные на неточных оценках, представляют собой одну из частых причин провала проекта. Зачастую причина провала кроется не в недостаточной производительности команды проекта, а в неточном предвидении уровня этой производительности и размера продукта. Точное прогнозирование экономических характеристик делает возможным создать эффективную систему контроля над ходом проекта.

По мере разработки проекта необходимо пересматривать ранее полученные оценки. Это обусловлено тем, что неопределенность относительно характеристик проекта изменяется по мере реализации проекта.

В настоящем пособии последовательно излагаются три базовых метода (рис. 4.1):

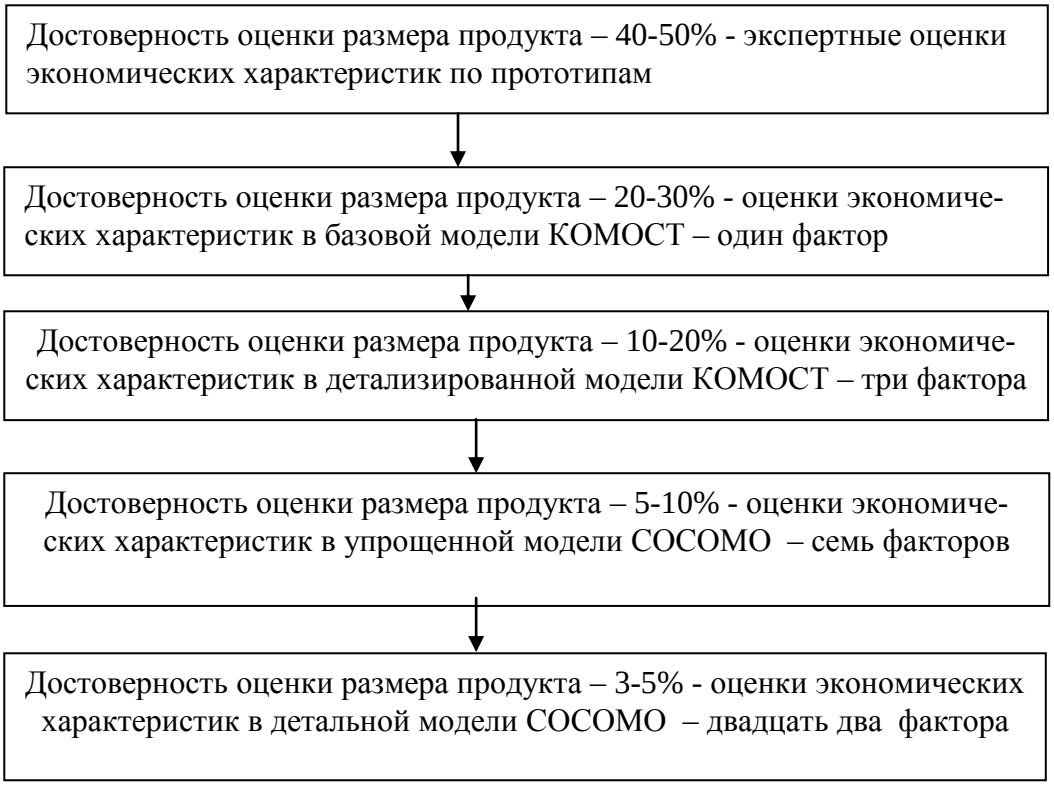


Рис 4.1. Базовые методы оценивания экономических характеристик программных продуктов

- Первичный экономический прогноз при подготовке концепции и технического задания на новый комплекс программ на основе экспертных данных о его размере, средней производительности труда специалистов или стоимости разработки одной строки текста программ – прототипов.
- Прогнозирование основных экономических характеристик при предварительном (эскизном) проектировании на базе расчетных значений трудоемкости и длительности разработки комплекса программ посредством упрощенных моделей с учетом влияния минимума дополнительных факторов.
- Определение возможных экономических характеристик проекта с учетом доступных оценок множества факторов для календарного планирования производства сложного программного продукта с использованием системы прогнозирования COCOMO II.

4.1. Экспертное прогнозирование экономических характеристик производства программных продуктов

В простейшем методе может быть реализован прогноз экономических характеристик производства программного продукта с учетом экспертной оценки минимального числа факторов. Данная методика экспертной оценки экономических характеристик может применяться, когда определены цели, масштаб и общие функции комплекса программ, сформулированных в концепции и первичных требованиях с достоверностью около 30 – 40%. Основная цель такого прогноза – подготовить возможность принять обоснованное решение о допустимости дальнейшего продвижения проекта в область системного анализа, разработки требований и предварительного проектирования.

При первичном экономическом обосновании сложных комплексов программ наибольшее значение имеют три ключевых фактора (таблица 4.1).

- Размер – масштаб подлежащих разработке полностью новых программных компонентов и всего комплекса.
- Размер и относительная доля готовых программных компонентов, которые могут быть заимствованы из предшествовавших продуктов и повторно использованы в новом проекте.
- Относительные затраты ресурсов на создание программного продукта: труда специалистов, времени или бюджета на единицу размера (на строку текста программ) проектируемого комплекса.

Таблица 4.1

Экспертные оценки исходных данных	Средние	Оптимистические	Пессимистические
1. Размер – масштаб комплекса программ (тысячи строк текста с указанием языка программирования)			
2. Относительное число строк в программных компонентах, готовых для повторного использования (%)			
3. Исходная производительность труда при разработке новых программных компонентов (число строк на человеко - месяц)			
4. Исходная стоимость разработки одной строки текста в комплексе программ			

Эти факторы могут быть прогнозируемы квалифицированными экспертами на основе имеющегося у них опыта реализации подобных проектов, а также опубликованных данных. При наличии необходимых данных важно оценить их достоверность и возможную точность – 30 – 40%. Наименее точный из перечисленных факторов определяет достоверность прогнозов экономических характеристик.

При наличии перечисленных исходных данных и положительной оценке целесообразности экспертного прогноза экономики проекта может быть реализован метод, состоящий из следующей последовательности шагов.

- Экспертная оценка размера – масштаба, числа строк (функциональных точек) предполагаемого текста разрабатываемых программ, с учетом размера повторно используемых компонентов и характеристик возможного языка программирования.
- Экспертная оценка возможной средней производительности труда специалистов при разработке комплекса программ и/или стоимости разработки одной строки текста программ продукта.
- Расчет возможной полной трудоемкости и длительности проектирования и производства программного продукта, а также среднего числа специалистов, необходимых для его реализации.
- Обобщение основных экономических характеристик и полной стоимости программного продукта, анализ результатов и экономическое обоснование рентабельности продолжения проектирования и производства комплекса программ.

Особенностью ***экспертного оценивания размера – масштаба комплекса программ*** является то, что прогноз характеристик нового продукта основы-

вается на особенностях реализованных проектов, при реализации которых использовались, например, устаревшие, с современной точки зрения, инструментальные средства проектирования. Экспертный прогноз удельных затрат на строку текста программного продукта обычно относится к полному циклу производства крупных комплексов программ, начиная от разработки концепции и требований до завершения испытаний и передачи продукта заказчику/пользователям. В составе участников проекта учитываются все категории специалистов, обеспечивающих реализацию программного продукта. В [3] со ссылкой на первоисточники утверждается, что несмотря на появление новых методов и инструментальных средств для разработки сложных комплексов программ, средняя производительность при их создании за последние двадцать лет оставалась практически неизменной и составляла около 3000 строк кода на одного разработчика продукта в год (порядка 250 строк на человеко-месяц).

Это отражает то, что уменьшение времени, затрачиваемого на цикл производства крупного продукта, *не может быть достигнуто за счет значительного повышения производительности труда отдельных специалистов*. Причем это практически не зависит от усовершенствований языков программирования, организационных усилий со стороны менеджеров, от наличия или отсутствия отдельных видов инструментария и автоматизации работ, хотя значительную роль играет увеличившаяся доля повторно используемых компонентов.

В литературных источниках приводятся достаточно широкие диапазоны данных о производительности труда. В работе В.В. Липаева [3] со ссылкой на первоисточники указывается: для простых программных продуктов – 8 LOC на человеко-день и 4 LOC на человеко-день для сложных продуктов. Там же приведены широкие диапазоны производительности труда при производстве комплексов программ на ассемблере – 60—500 LOC на человеко-месяц, и 50-300 LOC на человеко-месяц для зыков высокого уровня.

В базовой модели КОМОСТ используются следующие данные:

- для встроенных комплексов программ размером 30 тысяч строк для прогнозов рекомендуется использовать производительность около 140 строк на человеко-месяц, а для крупных программных продуктов размером 500 тысяч строк предлагается значение производительности около 80 строк на человеко-месяц;
- для программ административных систем размером 30 тысяч строк оценка производительности составляет около 220 строк на человеко-месяц, а для комплексов размером 500 тысяч строк – 160 строк на человеко-месяц.

Подчеркнем, что приведенные оценки применимы лишь при экспертной оценке полной трудоемкости производства новых программных продуктов. При применении повторно используемых компонентов обобщенная производительность труда возрастает и зависит от доли таких компонентов в комплексе программ. Для оценки таких ситуаций можно использовать коэффициенты снижения трудоемкости и длительности разработки. Графики этих коэффициентов, приведенные в работе В.В.Липаева [3], представлены ниже.

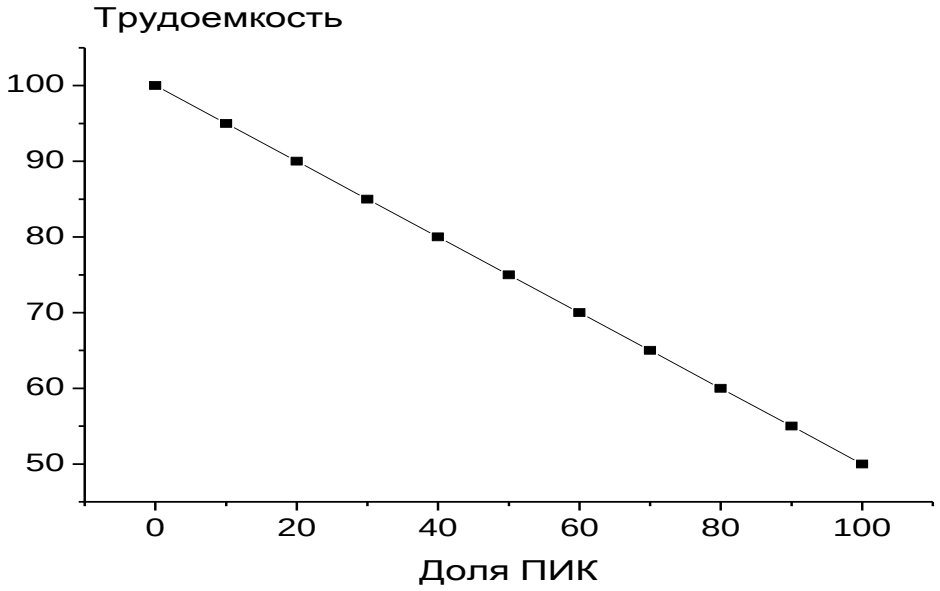


Рис 5.2. Зависимость трудоемкости от доли повторно используемых компонентов (ПИК)

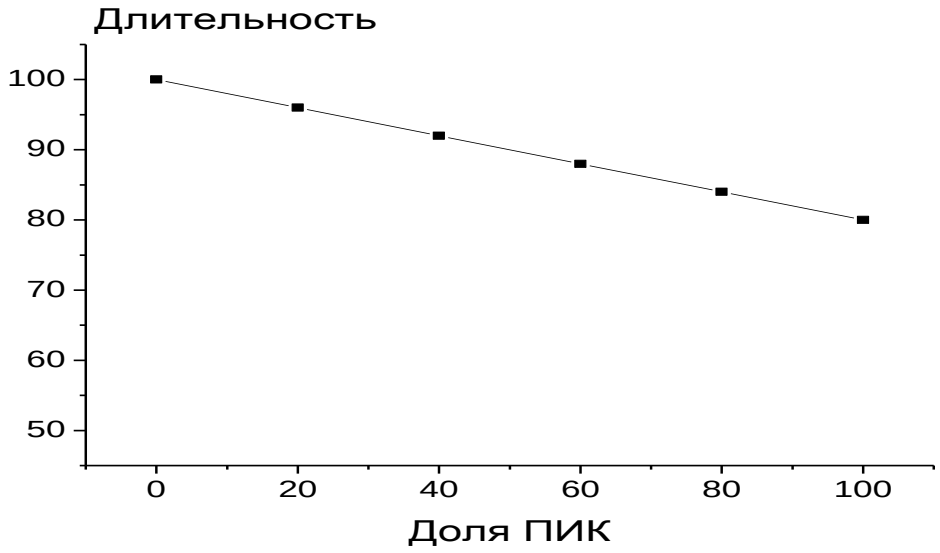


Рис 5.3. Зависимость длительности разработки от доли повторно используемых компонентов (ПИК)

Ориентировочная ***стоимость разработки одной строки теста*** программ реального времени – 100$, административных систем – 20-50$ [2].

Экспертное ***прогнозирование длительности производства*** сложных программных продуктов составляет для Новых проектов программ реального времени размером около 500 тыс. строк – 3,5 года. Для небольших (около 30 тыс. строк) – около года [3].

Экспертная ***оценка необходимого числа специалистов*** определяется делением полной трудоемкости производства программного продукта на длительность его реализации.

Обобщенные экспертные оценки экономических характеристик проекта целесообразно представить в виде таблицы (см. таблицу 4.2)

На основе анализа результатов и оценивания характеристик проекта делается заключение о целесообразности его продолжения.

4.2. Простейшие модели прогнозирования экономических характеристик производства программных продуктов

Методика и сценарии оценки экономических характеристик должны калиброваться по шагам в соответствии с уменьшением неопределенности исходных данных о размере программного проекта.

Таблица 4.2

Результаты прогнозов экономических
характеристик производства программного продукта

Экспертные оценки исходных данных	Средние	Оптимистические	Пессимистические
1. Полная трудоемкость производства Программного продукта – **C** (человеко-месяцы)			
2. Полная длительность производства программного продукта – **T** (месяцы)			
3. Необходимое среднее число Специалистов – **N** (человек)			
4. Средняя производительность Труда **P** специалистов (число строк На человеко-месяц)			

В моделях для первичной оценки экономических характеристик полного цикла производства размер комплекса программ используется в качестве базового доминирующего параметра. Остальные факторы рекомендуется учитывать поправочными коэффициентами при уточнении интегральных показателей. В [2, 4] отмечается, что трудоемкость разработки программного модуля

пропорциональна квадрату размера программы. При разработке комплексов программ большого размера должна возрастать сложность разработки, так как в них усложняются взаимосвязи по информации и управлению, а также становятся более трудоемкими процессы планирования, тестирования и управления проектом. Суммарную трудоемкость разработки сложного комплекса программ можно представить двумя сомножителями. Первый сомножитель (коэффициент **А**) отражает линейное возрастание трудоемкости создания комплексов программ при увеличении их размера.

Экспериментально установлено, что по мере увеличения размера комплекса программ возрастает относительная трудоемкость разработки каждой строки в программе. Это обстоятельство можно учесть вторым поправочным сомножителем, отражающим изменение трудоемкости на разработку каждой строки в программе при увеличении её размера. Для аппроксимации зависимости трудоемкости от размера комплекса программ удобно использовать степенную функцию вида

$$C = A * \Pi^{E} \qquad (4.1)$$

Значения коэффициентов **А** и **Е** приведены в таблице 4.3 [9].

Предполагается, что значение размера комплекса программ задано в тысячах строк ассемблера. Расчет П осуществляется на основе данных таблицы 4.4 [7].

Для учета влияния на трудоемкость **С** основных факторов программного проекта можно использовать коэффициенты (рейтинги) изменения трудоемкости (КИТ) производства – $\mathbf{M_i}$, учитывающие влияние i-й составляющей совокупных затрат. Непосредственно затраты на разработку можно представить в зависимости от размера комплекса программ **П**, корректируемого произведением коэффициентов изменения трудоемкости (КИТ – $\mathbf{M_i}$**).**

Таблица 4.3

Коэффициенты моделей для оценки трудоемкости производства программных продуктов

Коэффициент **А**	Коэффициент **Е**	Модель и тип программных продуктов
2.4	1.05	Базовая – КОМОСТ
2.8 3.0	1.2 1.12	Детализированная модель КОМОСТ: • встроенное ПО • административные системы

Таблица 4.4

Количество операторов языков программирования, приходящаяся на одну функцию

Язык программирования	Количество операторов на одну функцию (FP)	Язык программирования	Количество операторов на одну функцию (FP)
Ассемблер	320	Visual C++	34
C	128	Delphi Pascal	29
Кобол	106	Smalltalk	22
Фортран	106	Perl	21
Паскаль	90	HTML3	15
C++	64	LISP	64
Java	53	Prolog	64
Ada 95	49	Miranda	40
Visual Basic	32	Haskell	38

Коэффициенты изменения трудоемкости производства программных продуктов в упрощенной предварительной модели COCOMO

Таблица 4.5.

Состав и значения факторов упрощенной предварительной модели COCOMO

Символ	**Содержание группы факторов**
	Требования к объекту разработки
$\mathbf{M_1}$	Сложность и надежность программного продукта
$\mathbf{M_2}$	Требование повторного использования компонентов
	Аппаратурно-вычислительная среда производства
$\mathbf{M_3}$	Ограничения аппаратной платформы производства и применения продукта
	Характеристики коллектива специалистов
$\mathbf{M_4}$	Квалификация специалистов и стабильность коллектива
$\mathbf{M_5}$	Опыт работы по тематике и с инструментарием
	Технологическая среда производства
$\mathbf{M_6}$	Уровень инструментальной поддержки и необходимость распределения производства
$\mathbf{M_7}$	Ограничение длительности производства

Для детальной модели КОМОСТ рекомендовано четыре группы факторов. Впоследствии на их основе было выделено семь факторов, используемых

в составе предварительной модели COCOMO. Эти факторы представлены в таблице 5.

Для выделенных семи факторов определены значения рейтингов (коэффициентов изменения трудоемкости), которые приведены в таблице 4.6.

Таблица 4.6

Коэффициенты изменения трудоемкости

Факторы	**Рейтинги оценки**					
	Очень низкий	Низкий	Номинальный	Высокий	Очень высокий	Сверх высокий
M_1	0.6	0.83	1.0	1.33	1.91	2.72
M_2		0.95	1.00	1.07	1.15	1.24
M_3		0.87	1.00	1.29	1.81	2.61
M_4	1.62	1.26	1.00	0.83	0.63	0.50
M_5	1.33	1.12	1.00	0.87	0.74	0.62
M_6	1.30	1.10	1.00	0.87	0.73	0.62
M_7	1.43	1.14	1.00	1.00	1.00	

Зависимость длительности разработки **T** от размера программы **П** значительно различаются для классов комплексов программ. Это определяется различием сложности классов программ, применяемых языков программирования и единиц измерения программных продуктов, следствием чего является различие значений размера программ при одной и той же длительности разработки. В литературе описана модель оценки длительности разработки в зависимости от трудоемкости производства. При исследовании модели было установлено, что длительность производства комплексов программ меньше подвергается изменениям при автоматизации разработки, чем трудоемкость или производительность труда. Необходимость выполнения при производстве определенной совокупности этапов и операций в заданной технологической последовательности остается более или менее постоянной при различных воздействиях на процесс разработки. Исключением является применение повторно используемых компонентов (ПИК), при котором значительно сокращаются этапы программирования и автономной отладки модулей и компонентов программ, а так же в той или иной степени длительность других этапов. Поэтому заметное сокращение длительности разработки проявляется только при создании базовой версии программного продукта практически полностью из готовых компонентов.

Зависимость длительности разработки **T** от размера программы **П** описывается зависимостью

$$T = G * C^{H} \qquad (4.2)$$

Коэффициенты модели приведены в таблице 4.7.

Таблица 4.7

Коэффициенты для оценки длительности разработки программных продуктов

Коэффициент G	**Коэффициент H**	**Модель и тип программных продуктов**
2.5	0.38	Базовая - КОМОСТ
2.5	0.32	Детализированная – КОМОСТ *встроенный
2.5	0.35	*административные системы

Оценка **требуемого среднего числа специалистов** предварительно может быть рассчитана посредством соотношения

$$N = C / T \qquad (4.3)$$

Средняя производительность труда коллектива специалистов рассчитывается посредством соотношения

$$P = П / C \qquad (4.4)$$

и может служить ориентиром при сравнении эффективности труда при создании различных продуктов на разных предприятиях.

Контрольные вопросы:

1. Какие базовые методы оценивания экономических характеристик программных продуктов вы можете назвать?
2. В чем идея экспертных оценок исходных данных проекта и каков алгоритм экспертного прогноза экономики проекта?
3. Какие модели прогнозирования экономических характеристик используются при производстве программных продуктов?
4. Как оценивается трудоемкость производства программного продукта?
5. Как оценивается продолжительность разработки программного продукта?

ГЛАВА 5. МЕТОДИКИ ОЦЕНКИ ХАРАКТЕРИСТИК ПРОГРАММНОГО ПРОЕКТА

5.1. Оценка проектных решений по показателю сложности

Конкурентоспособность программных продуктов на рынке, а также перспективы использования их в качестве инструмента информационной поддержки управления в значительной степени определяются стоимостью их разработки и испытаний. В свою очередь, стоимость разработки и испытаний определяется сложностью проектных решений. В связи с этим специалисту по IT следует знать методы оценивания сложности проектных решений, и уметь пользоваться ими на практике. В связи с этим рассмотрим методику оценки сложности проекта.

Методика оценки сложности проекта и примеры их оценивания

Число связей между частями программной системы оценивают её сложность. Части программного модуля реализуют определенные части алгоритмов обработки данных. Каждая из частей имеет связи с другими частями, причем связи по смыслу представляют собою следования, выбор, итерации (петли).

T.J.McCabe [8] предложил определять цикломатическую сложность соотношением:

$$v = e - n + 2,$$

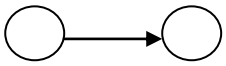
где v – показатель цикломатической сложности, e – число стрелок, n – число узлов

G.Myers [9] отмечает, что сложные предикаты следует представить в виде совокупности простых предикатов перед тем, как вычислять цикломатическую сложность. Так, решение **IF (A.AND.B.AND.C)** следует представить в виде трех независимых решений.

Рекомендуется, чтобы значение цикломатической сложности **не превышало десяти**. Исследования показали, что в модулях, у которых значение цикломатической сложности превышало эту величину, количество ошибок существенно возрастало.

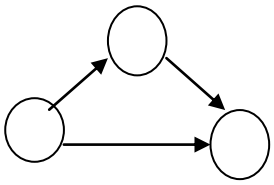
Значения показателей цикломатической сложности для типовых конструкций представлено на рис.5.1.

Последовательность



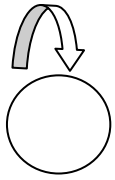
e=1; n=2; v=1

Выбор



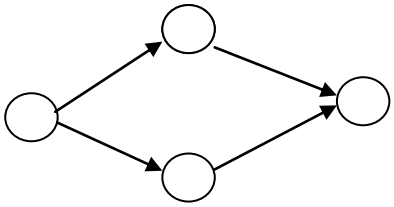
e=3; n=3; v=2

Итерация



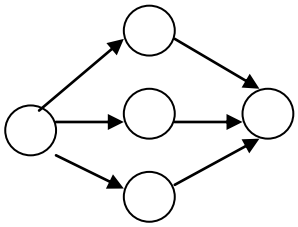
e=1; n=1; v=2

IF…THEN…ELSE…ENDIF



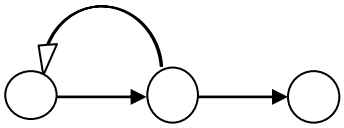
e=4; n=4; v=2

CASE



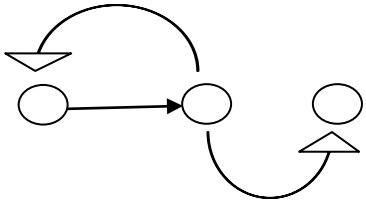
e=6; n=5; v=3

UNTIL



e=3; n=3; v=2

WHILE



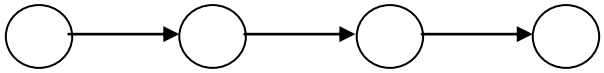
e=3; n=3; v=2

Рис.5.1. Значения показателей цикломатической сложности.
Источник [ESA-PSS-05-10]

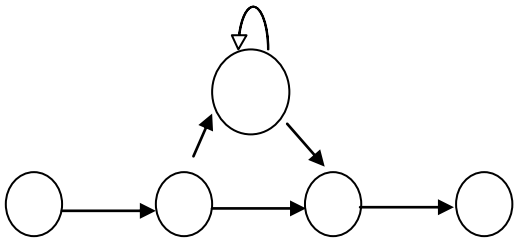
При выполнении детального проектирования следует ограничиться значением цикломатической сложности «**семь**», так как на этапе кодирования сложность увеличится. Модули, у которых превышено указанное значение цикломатической сложности, подлежат перепроектированию. Исключение составляет лишь конструкция **CASE.**

Примеры оценки сложности структурных схем:

a) e=3; n=4; v=1



b) e=6; n=5; v=3



c) e=9; n=5; v=6

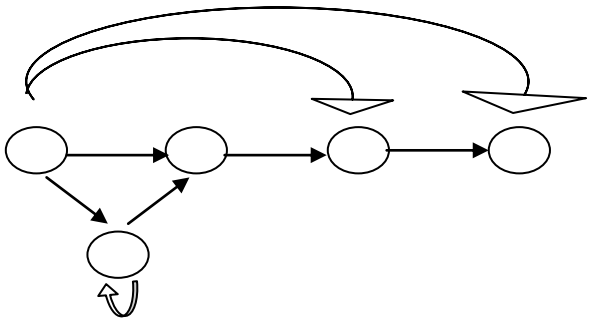


Рис.5.2. Оценка цикломатической сложности
Источник: [ESA-PSS-05-10]

Пример оценки цикломатической сложности.

Задание: разработать программу совместной обработки файлов типа F1 и F2 для формирования выходного документа типа F3, согласно исходным данным. Разработать структурные схемы программной системы, оценить цикломатическую сложность.

Исходные данные:

Таблица 5.1

Структура файла F1

Номер предприятия	Название предприятия

Таблица 5.2

Структура файла F2

Номер предприятия	Район ,где расположено предприятие	Специальность работающих	Количество рабочих мест	Количество вакансий

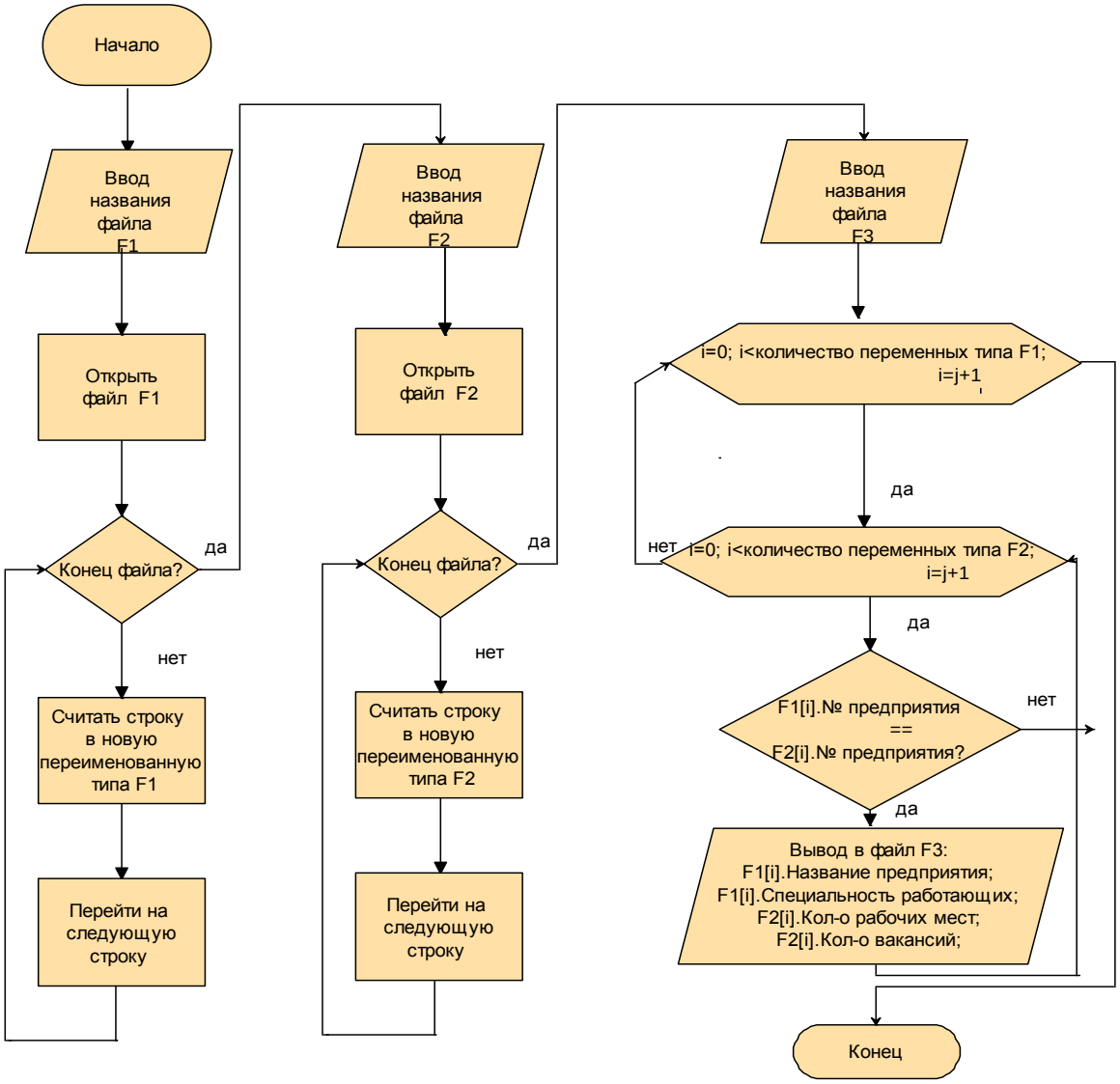
Результат обработки в виде выходного документа, представленного в таблице F3:

Таблица 5.3

Структура файла F3

Специальность работающих	Название предприятия	Район, где расположено предприятие	Количество рабочих мест	Количество вакансий

Алгоритм решения представим в виде блок-схемы:



По полученной блок-схеме составим структурную схему данного алгоритма:

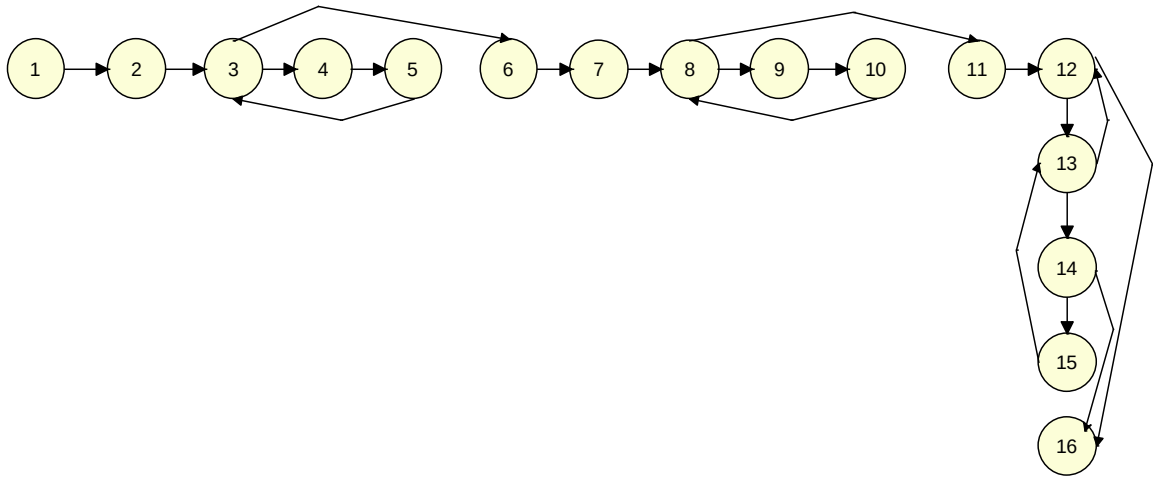


Рис.5.3. Граф структурной схемы алгоритма

Определим значение показателя цикломатической сложности алгоритма следующим образом:

$v = e – n + 2$,

где v – показатель цикломатической сло жности; e – число стрелок; n – число узлов

$n = 16$; $e = 20$,

$v = 20 – 16 + 2 = 6$.

Вывод: в ходе работы было рассчитано значение показателя цикломатической сложности структурной схемы. Его значение не перевешает «семи», т.е. он не подлежит перепроектированию.

5.2. Оценка сложности на основе структурных моделей

Необоснованная сложность проектных решений является ключевым фактором нарушения базового плана программного проекта, превышения расходования ресурсов, несоответствия полученных результатов целям проекта.

Системы ПО относятся к классу «сложных систем». Как отмечается в [10], типичная программа прошлого, написанная на КОБОЛе, была единственной сущностью, использующей подпрограммы, вызываемые по мере необходимости. Логика программы была последовательна и предсказуема. *Сложность такого ПО была следствием его размера*.

Современное объектно-ориентированное ПО является распределенным (оно может находиться во многих узлах компьютерной сети) и его выполнение случайно и непредсказуемо. Размер современного ПО – сумма размеров его компонент. Каждый компонент разработан так, чтобы быть ограниченного

управляемого размера. *В результате размер не является главным фактором в сложности современного ПО*.

Сложность современного ПО заключается в «проводах», то есть в связях и коммуникационных путях между компонентами. Связи между компонентами создают зависимости между распределенными компонентами, которые могут быть сложными для понимания и управления. Трудность усугубляется тем, что компоненты часто разрабатываются и управляются людьми и коллективами, данными не известными друг другу. На рис. 5.3 приведен пример системы, в которой объекты различных компонентов связаны без разбора. Это создает сеть внутренней связи объектов. Сложность в пределах отдельных компонентов все еще управляема из-за ограниченного размера компонентов. Однако зависимости, созданные между пакетами, будут расти по экспоненте с добавлением новых пакетов. *Кто должен обеспечить управление такими зависимостями не всегда ясно, поскольку обязанности обеспечения управления для разных компонентов остаются за разными коллективами*. Не менее важно и то, что любой объект в одном пакете может связываться с любым объектом в другом пакете. Это создает *потенциальные зависимости* между всеми объектами в системе. Это означает, что изменение какого-либо объекта может потенциально повлиять (вызвать *эффект ряби*) на любой другой объект в системе.

Более формально: совокупной мерой зависимости объектов с неограниченными межкомпонентными коммуникационными связями является число различных комбинаций пар объектов. Формула для вычисления совокупной зависимости класса (CCD – cumulative class dependency) в системе с n классами объектов имеет вид:

$$CCD = \frac{n!}{2!(n-2)!}$$

Расчеты по этой формуле дают оценку максимальной сложности, когда каждый объект связан со всеми другими объектами. Для пяти классов *CCD* равно десяти. Для 57 классов *CCD* равняется 1596. Такой рост сложности *становится неприемлемым*.

Системы, допускающие произвольную сеть связи объектов, подобно изображенной на рис. 5.4, считаются неприемлемыми с точки зрения системной инженерии. Они непонятны, неремонтируемые, нерасширяемые.

Решение проблемы находят в замене сетей объектов иерархиями (древовидными структурами) объектов. На рис. 5.5 показано, как можно уменьшить

сложность системы, допуская единственный канал связи между компонентами.

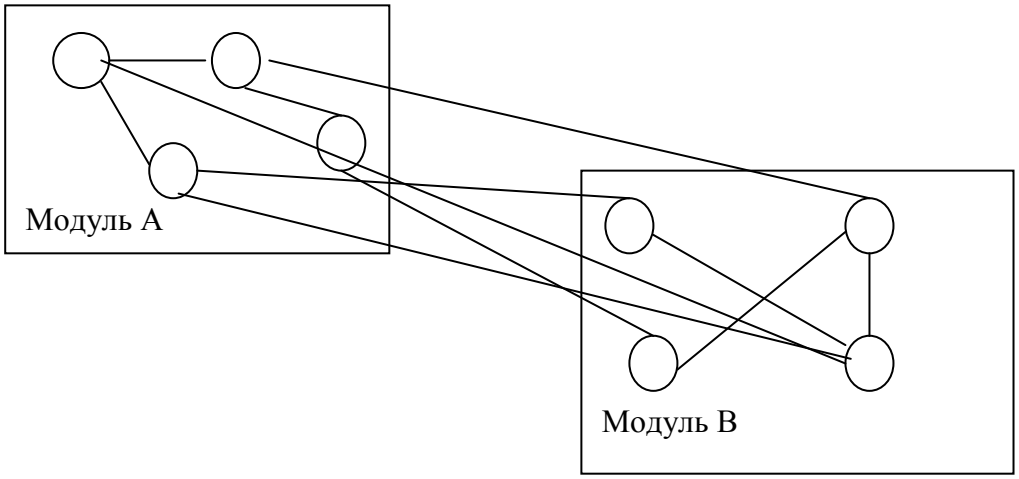


Рис. 5.4. Сложность в «проводах»

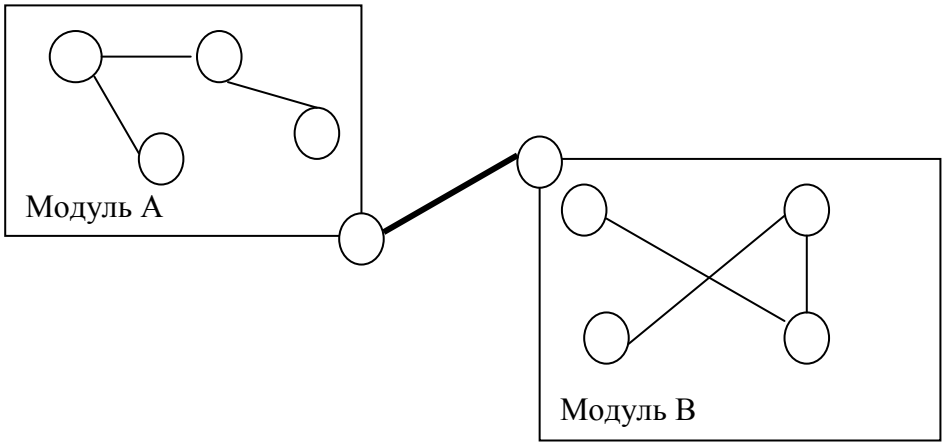


Рис. 5.5. Понижение сложности добавлением интерфейсов между компонентами

Каждый компонент определяет интерфейсный объект (так называемый доминантный класс), через который осуществляется связь между объектами. Несмотря на добавление двух дополнительных объектов, сложность системы на рис. 5.5 меньше, нежели системы, изображенной на рис. 5.4.

В работе Е.С. Вентцель [3] на примере решения задачи динамического программирования показано, что при исследовании сложных систем лучше много раз решить простую задачу, чем один раз сложную. Декомпозиция является одним из инструментов системного анализа, направленного на уменьшение сложности исследуемых объектов. Вместе с тем, возникает вопрос: как оценить качество моделей, получаемых в результате представления сложного объекта в виде иерархической совокупности более простых конструкций.

Показатель цикломатической сложности. В [ESA PSS-05-10] утверждается, что общая цикломатическая сложность программы рассчитывается на основе значений цикломатических сложностей, входящих в её состав модулей. Согласно T.J.McCabe полная цикломатическая сложность определяется соотношением

$$v = e - n + 2 * p \ ,$$

где p – число модулей, e – число ветвей, n – число узлов графа.

Предполагается, что каждому из модулей ставится в соответствие собственный управляющий граф.

В приведенном ниже примере:

число узлов n=14, ребер $e = 18$, модулей $p = 3$.

$$v = 18 - 14 + 2 * 3 = 10.$$

Суммарная сложность компонентов:

$$v_{\Sigma} = v_A + v_B + v_C = 1 + 3 + 6 = 10 .$$

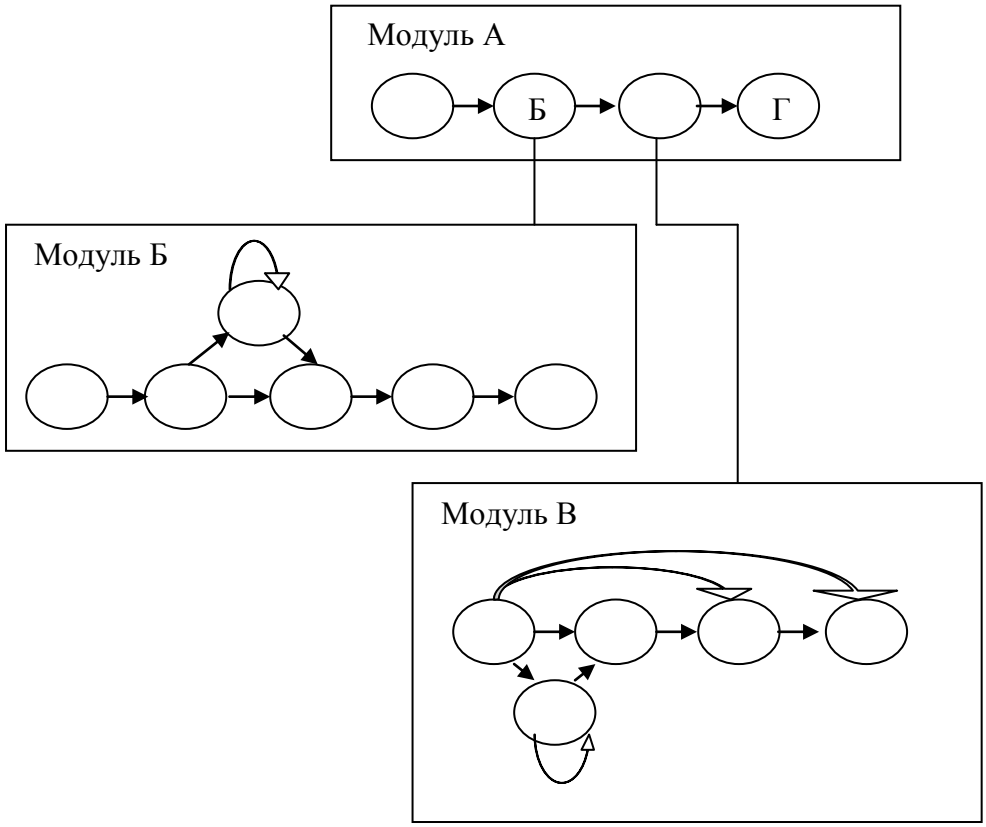


Рис.5.6. Программа разбита на модули
Источник [ESA-PSS-05-10]

Объединение модулей, представленных на рис.5.6, в один даст значение цикломатической сложности равное восьми (рис.5.7).

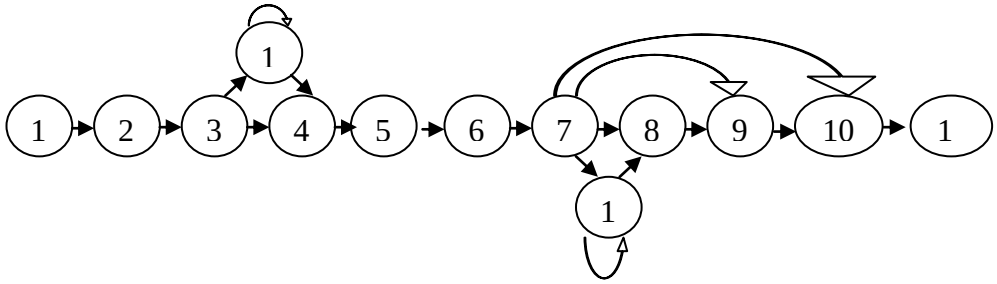


Рис 5.7. Граф системы без разделения на компоненты
Источник: [ESA-PSS-05-10]

Имеем $e = 18; \quad n = 13; \quad \mathrm{v} = 18 - 13 + 2 = 8.$

Общая сложность конструкции оказалась выше, нежели сложность отдельных модулей.

Полезным правилом декомпозиции при решении проектных задач является следующее: декомпозицию следует продолжать до тех пор, пока значение показателя сложности для каждого модуля не окажется меньше десяти.

Дополнительную информацию по упомянутой теме можно найти в работе [13].

Метод базовой линии. Этот метод используется в структурном тестировании при принятии решения о том, какой путь следует выбрать для проверки каждой ветви. Проектировщик тестов должен проверить основные функции модуля и определить «базовый путь», при использовании которого функции будут проверяться.

Пример.

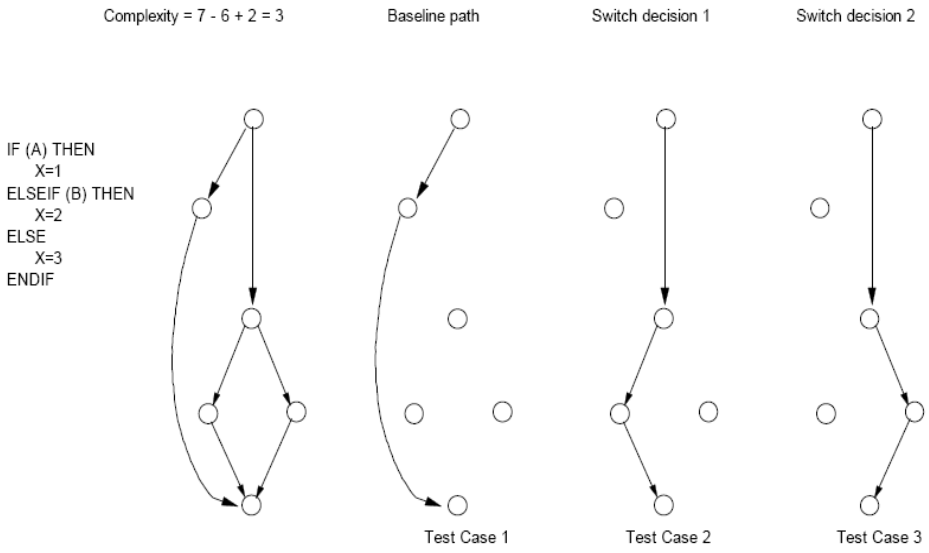


Рис. 5.8. Пример «Газового пути»
Источник [ESA-PSS-05-10]

Цикломатический показатель сложности имеет значение v=3, поэтому необходимо три теста для выполнения полного покрытия всех ветвей. Тест 1 покрывает базовый путь, тесты 2 и 3 покрывают пути, определенные положением переключателя. Значение переключателя и связанное с этим отклонение от базовой линии фиксируется в контрольном листе. В общем случае метод базовой линии не гарантирует тестирования всех возможных путей. Генерация тестов для исследования пропущенных путей может осуществляться другими способами.

Структурное интеграционное тестирование. В основе методов структурного интеграционного тестирования лежат следующие положения:

(1) улучшение тестируемости за счет ограничения сложности при принятии проектных решений;

(2) генерация тестов в процессе интеграционного тестирования.

Трассируемость . В рамках методов структурного интеграционного тестирования предлагается три метрики измерения тестируемости:

1. Сложность модуля;
2. Сложность конструкции;
3. Сложность интеграции.

В качестве метрики сложности выступает предложенное McCabe цикломатическое число.

Сложность конструкции оценивается на основе графа управления модулями, в котором выделяются узлы, из которых происходит обращение к другим модулям. После этого происходит «редуцирование» графа в соответствии с правилами, определенными ниже.

Правило 1. Выделение узлов, в которых происходит принятие решений (они не могут быть удалены).

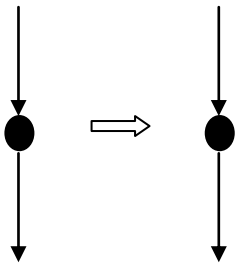


Рис.5.9

Правило 2.Удаление невыделенных узлов, в которых не происходит принятия решений.

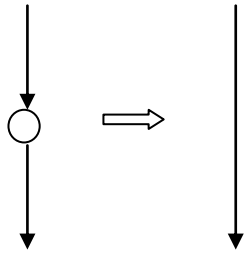


Рис.5.10

Правило 3.Удаление стрелок, соответствующих контурам, содержащих невыделенные узлы

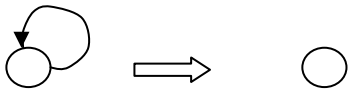


Рис.5.11

Правило 4. Стрелки, начинающиеся в операторе выбора и не содержащие выделенных узлов, удаляются

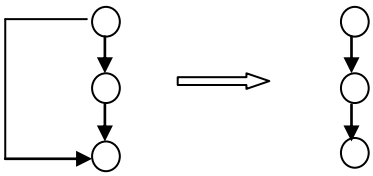


Рис.5.12

Пример редукции 1

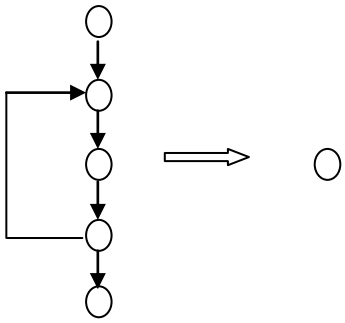


Рис.5.13

Пример редукции 2

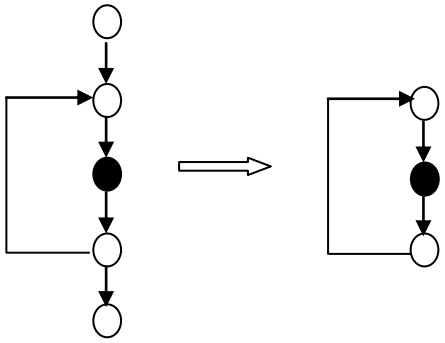


Рис.5.14

После того, как в графе управления будут удалены все «дублирующие» дуги, рассчитывается величина цикломатической сложности редуцированного графа.

Пример расчета сложности конструкции. Сложность конструкции, определяемая McCabe как S_0 оценивается по формуле

$$S_0 = \sum_i iv_j$$

Общая интегральная сложность конструкции, обозначаемая McCabe как «S_1», рассчитывает число путей передачи потоков в графе. Значение общей (интегральной) сложности определяется соотношением

$$S_1 = S_0 - N + 1$$

На рисунках 5.15, 5.16 приведены примеры упрощения структур

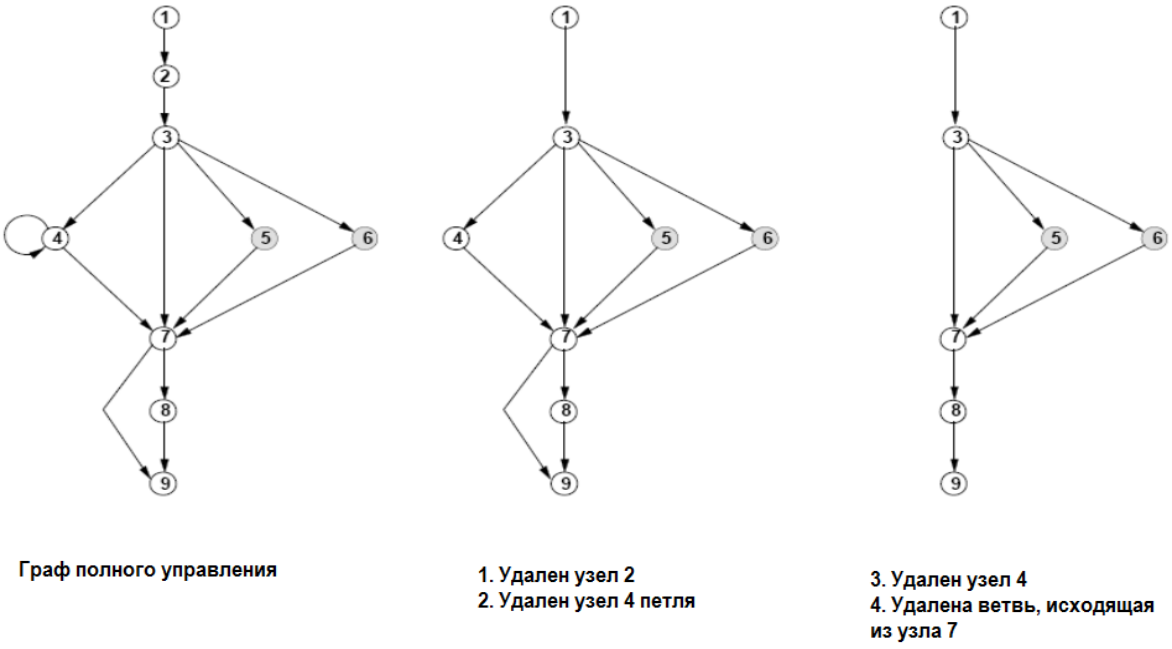


Рис.5.15. Пример 1 упрощения структуры.
Источник [ESA-PSS-05-10]

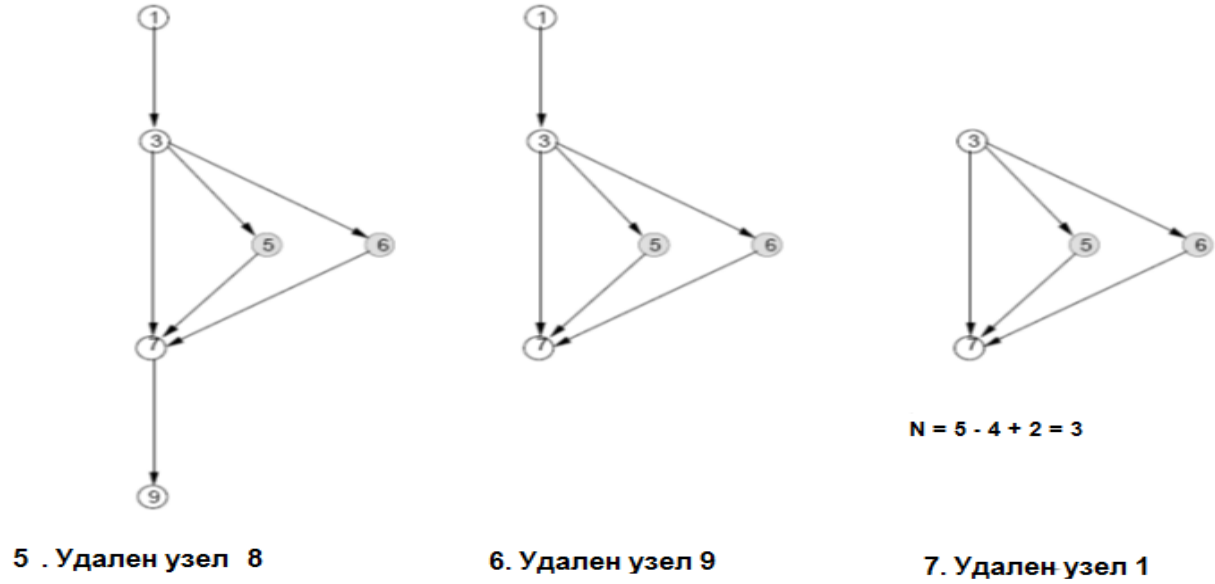


Рис.5.16 Пример 2 упрощения структуры
Источник: [ESA-PSS-05-10]

Тестируемость конструкции измеряется их интегральной сложностью. Формально значение показателя интегральной сложности зависит от значения показателя сложности iv_j $j-$ х модулей, входящих в конструкцию. Во время проектирования архитектуры полный граф управления, соответствующий каждому из модулей, обычно не известен. Вместе с тем, необходима информация, позволяющая оценить сложность конструкции, не зная внутреннего устройства модулей. Ниже представлена схема рис. 5.17, показывающая, как S_1 может быть рассчитано на основе информации о том, как осуществляется активация каждой из компонент, т.е. знаний о потоках управления. Прямоугольники на схеме соответствуют проектируемым компонентам, а стрелки обозначают передачу потоков управления. Ромбы подчеркивают, что передача управления является условной.

Потоки управления имеют следующий смысл:

- компонент A вызывает либо компонент B, либо компонент C;
- компонент B последовательно вызывает компонент D, затем компонент E;
- компонент C вызывает либо компонент E, либо компонент F, либо компонент G, либо не вызывает никакого компонента.

Значение интегральной оценки сложности равно пяти.

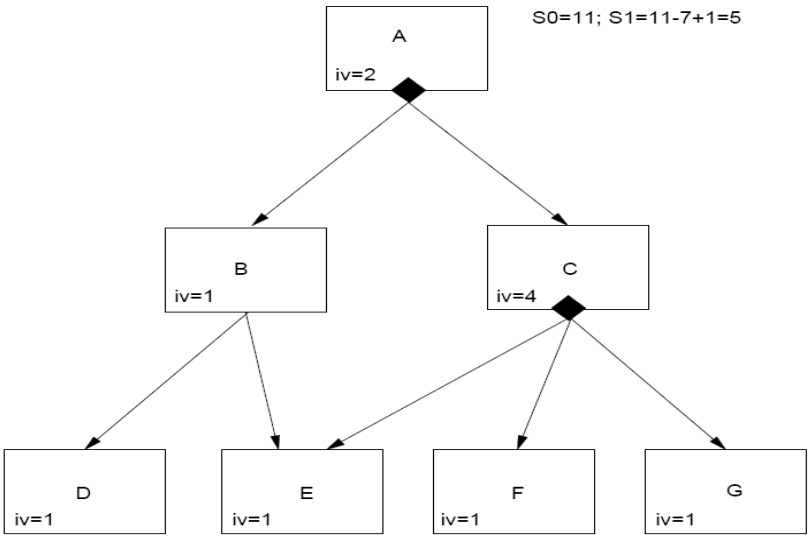


Рис.5.17 Схема активации компонентов.
Источник [ESA-PSS-05-10]

Показатель цикломатической сложности определяет число тестовых наборов, необходимых для покрытия каждой ветви в модуле. Интегральный показатель сложности определяет число тестов, необходимых для покрытия всех потоков управления. Структурное интеграционное тестирование может быть использовано для того, чтобы верифицировать, что все потоки управления, определенные на этапе проектирования архитектуры, выполнены.

Проектирование интеграционных тестов. Метод проектирования интеграционных тестов позволяет проектировщику тестов определить, какой из путей передачи потоков управления следует протестировать. Проектировщик тестов должен:

- оценить значение интеграционного показателя сложности S1 для конструкции, содержащей N модулей;
- сформировать бланк матрицы размера S1 X N, именуемую интеграционной матрицей тестирования путей;
- пометить каждую колонку именем модуля, добавляя «P» для предикатов с указанием номера предиката;
- заполнить матрицу «0», либо «1» для демонстрации того, вызывается или нет модуль в тесте.

Для приведенного выше рисунка матрица имеет вид:

Таблица 5.1. Интеграционный тест

Case	A	P1 B	P2 C	D	P3 E	P4 F	P5 G	Control flow path
1	1	1	0	1	1	0	0	A calls B; B calls D and E
2	1	0	1	0	0	0	0	A calls C and then returns
3	1	0	1	0	1	0	0	A calls C and then C calls E
4	1	0	1	0	0	1	0	A calls C and then C calls F
5	1	0	1	0	0	0	1	A calls C and then C calls G

Здесь P_1, P_2 – предикаты, связанные с модулем A;
P_3, P_4, P_5 – предикаты, связанные с модулем C.

Задание

1. Разработать структурные схемы программной системы, реализующие задание, указанное в папке «Задание». Номер варианта совпадает с порядковым номером студента в групповом журнале.
2. Разработать структурные схемы для случая построения системы с использованием модульных конструкций, а также для случая построения системы в виде монолитной конструкции.
3. Рассчитать значения цикломатической сложности для разработанных структурных схем.
4. Рассчитать сложность конструкции по McCabe для схем с использованием модульных конструкций, общую интегральную сложность, построить матрицу интегральных тестов.

   Пример выполнения задания приведен в приложении 5.

5.3. Методика системы сетевого планирования

В настоящее время исследования в области оценки программных проектов сосредоточены на совершенствовании методов оценки, чтобы передовые организации, занятые реализацией сложных больших проектов, могли прогнозировать результат с точностью $\pm 5\%$ вместо $\pm 10\%$. (Заметим, что «большое» и «сложное» применительно к программным системам не одно и то же). В этих методах задействованы интенсивные вычисления. Их понимание требует хорошей математической подготовки, а обработка данных выходит далеко за рамки того, что можно сделать на ручном калькуляторе. Такие методы лучше всего работают в виде специализированных коммерческих пакетов оценки

программ и называются *научными методами оценки*. На практике значительно чаще приходится реализовывать малые, либо средние проекты. Это, в определенной степени, обусловлено тем, что в системе IT давно уже определилась тенденция оттока специалистов из области по разработке программных систем в область их сопровождения (см., например, работу В.В. Липаева). В подобной ситуации вовсе не стремятся к получению оценок с точностью $\pm 5\%$. Достаточно избежать ошибок порядка $\pm 100\%$ и более. Поэтому можно заключить, что в простые методы оценок также имеют право на жизнь.

Замечание. Не нужно противопоставлять разные методы друг другу. Нужно стремиться к тому, чтобы подобрать из множества доступных тот, который наиболее адекватен текущей ситуации.

Простые методы оценок не обеспечивают результата с точностью $\pm 5\%$, но они позволяют сократить ошибки оценок до $\pm 25\%$ и менее. Для большинства проектов этого вполне достаточно.

Методика решения задач и примеры их решения. Рассмотрим простую базовую систему сетевого планирования, называемую стрелочным графиком или методом стрелочных диаграмм. Стрелочный график демонстрирует как последовательность действий по проекту, так и связь между ними. Действия обозначены стрелками, расположенными слева направо. Начало и конец действия – стрелка – помечена кружком (узлом). Как будет показано далее, узлы используются для накопления информации о продолжительности отдельных действий и проекта в целом.

На приведенном ниже рисунке 5.18 представлен пример стрелочного графика, на котором действия помечены цифрами 1,2,3,4,5, а узлы, которые также называются событиями, помечены буквами а, б, в, г, д, е.

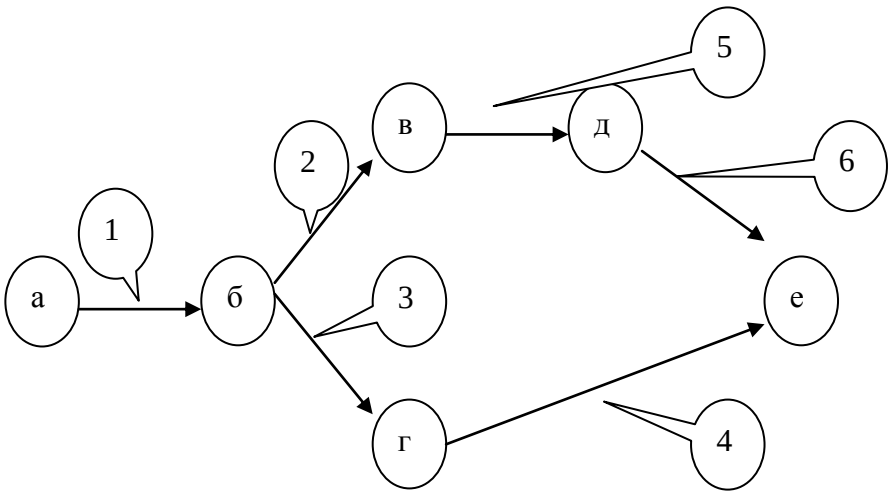


Рис.5.18. Пример сетевого графика

На основе оставления подобных сетевых графиков лежат простые правила:

- завершение одного действия является началом другого, за исключением конца графика;
- действия могут осуществляться одновременно.

Одно и то же событие может послужить началом для разных действий или же разные действия могут слиться в одно явление.

Однако в том виде, как он представлен в приведенном примере, график не дает представления о продолжительности действий и не позволяет ответить на вопросы, есть ли между ними взаимосвязь и каков критический путь.

Замечание. Критический путь – последовательность действий по проекту, определяющая его общую продолжительность.

Для ответа на сформулированные вопросы необходимо знать конкретную продолжительность действий и их взаимосвязи. В приведенной ниже таблице приведены данные о продолжительности действий, а также подчеркнуто то обстоятельство, что действие 6 может быть выполнено только после выполнения действий 3&5. Например, глубина испытаний программного продукта может быть оценена лишь после определения класса программного продукта (в составе системы реального времени, либо автоматизированная офисная система), а также размера бюджета проекта.

Таблица 5.2

Взаимосвязь действий и их продолжительность

Действие	Продолжительность (в часах)	Может быть завершено только после действия
1	2	
2	2.75	1
3	3.5	1
4	4	3
5	1	2
6	2	3&5

После включения этих данных в сетевой график, он приобретет вид, показанный на рис.5.19.

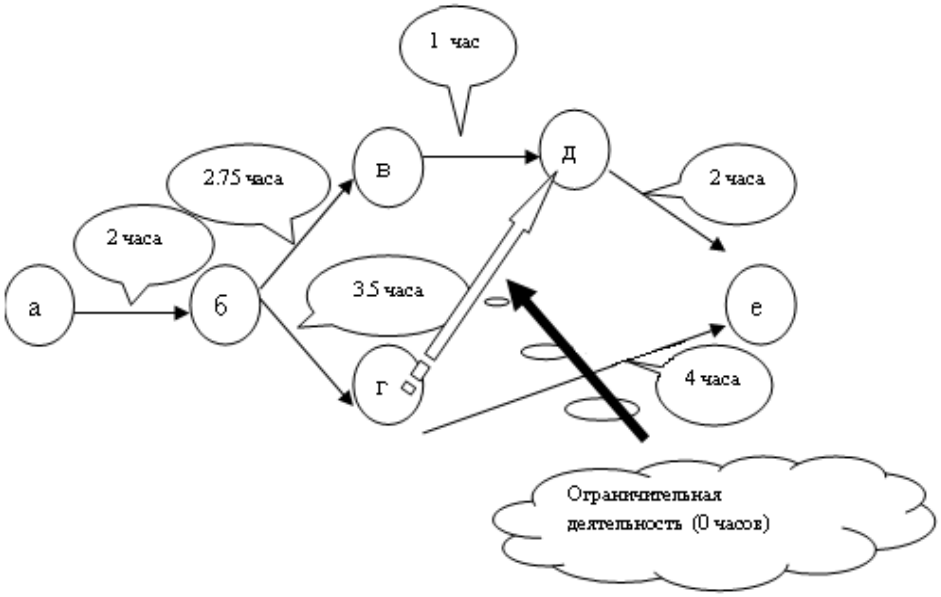


Рис. 5.19. Сетевой график с указанной продолжительностью деятельности

Стрелка с незаштрихованным указателем (ограниченная деятельность) показывает, что действие 6 не может быть начато до тех пор, пока не будут завершены действия 3 и 5. Определим теперь, когда может произойти то или инее событие (наиболее ранний срок события –НРСС), и наиболее поздний возможный срок, когда может произойти событие (наиболее поздний срок – НПСС). Для включения этой информации, необходимо изменить кружки (узлы), как показано на рис.5.20.

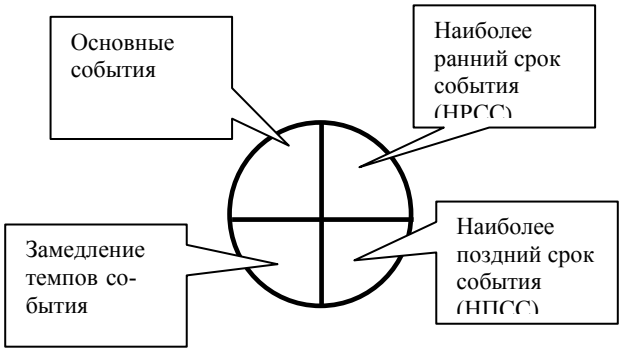


Рис.5.20. Узел сетевого графика

Расчет НРСС и НПСС для каждого узла осуществляется следующим образом.

- Начните с первого события сетевого графика и напишите цифру 0 там, где обозначен НРСС.
- Переходя к событию «б», добавьте продолжительность действия 1, которое составляет 2 часа, к НРСС события «а», которое равно 0; в результате получается 2 часа, которое следует вписать там, где обозначен НРСС события «б».

- Пройдите так по всему графику, добавляя продолжительность каждого действия к предыдущему НРСС и вписывая полученные результаты там, где обозначен НРСС узла, соответствующего завершению соответствующего действия.
- Когда достигается событие, где происходит слияние двух действий, как в случае «д» и «е», запишите полученное максимальное значение НРСС. Обратите внимание, что ограниченная деятельность имеет нулевую продолжительность.
- Продолжайте следовать по графику до последнего события и запишите полученные значения НРСС и НПСС в соответствующем месте на последнем узле.
- Начиная с события «е» пройдите по сетевому графику в обратном направлении, определите продолжительность каждого последующего действия. Исходя из значения НПСС каждого действия, запишите полученную цифру там, де обозначен НПСС на узле начала события.
- Когда достигается событие, где происходит слияние двух действий, как в случае «б» и «г», запишите наименьшее полученное значение НПСС.
- Продолжайте движение по графику до первого события, где значения НПСС и НРСС для данного события должны совпасть. Если они не совпадут, значит, допущена ошибка.

Сетевой график с внесенными в него цифрами будет выглядеть, как показано на рис.5.21.

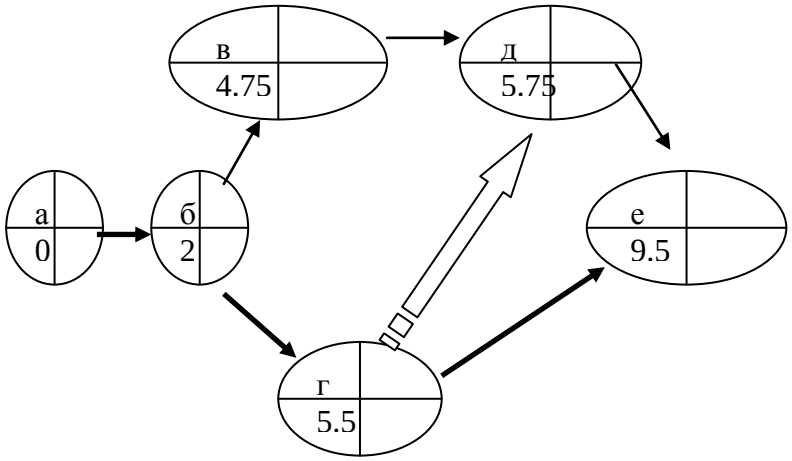


Рис.5.21. Расчет НРСС и НПСС (в часах)

Некоторые линии на сетевом графике проведены толстыми линиями. Они обозначают последовательность действий, определяющих продолжительность всего проекта, то есть критический путь проекта.

Поскольку событие имеет НРСС и НПСС, равные 9.5, ясно, что для завершения проекта потребуется 9.5 часов. Но если посмотреть внимательнее, то можно заметить, что действие 2, которое начинается спустя 2 часа после

начала проекта, может быть закончено через 2+2,75 = 4,75 часа, но нет необходимости заканчивать его раньше, чем через 6,5 часов. Точно так же действие 5 может быть закончено через 4,75+1=5,75 часов. Разницу между временем требуемым и временем, имеющимся для выполнения действия, называют «ЗАМЕДЛЕНИЕМ ТЕМПОВ» или «СПАДОМ» данного действия.

«Замедление темпов», или «спад», могут использоваться при планировании времени выполнения действий в рамках проекта. Например, можно отложить действие 5 до НПСС (6,5 часов) узла, с которого начинается его выполнение, или начать в исходное время – 4,75 часов – и увеличить за счет сокращения объема ресурсов или использования других методов максимум до 7,5 – 4,75 = 2,75. В полностью заполненном сетевом графике замедление темпов показано в узлах, в результате получается график, представленный на рис. 5.21.

До сих пор мы исходили из того, что точно известна продолжительность действий. Но так бывает не всегда. Например, может быть известна «наименьшая», «наибольшая» и «наиболее вероятная» продолжительность действия. В таком случае можно рассчитать так называемую «предполагаемую продолжительность» каждого действия и на основе полученных результатов составить сетевой график, используя следующее уравнение:

Предполагаемая продолжительность действия

$$T = \frac{(a + 4m + b)}{6},$$

где a – наименьшая предполагаемая продолжительность действия;

b – наибольшая предполагаемая продолжительность действия;

m – наиболее вероятная продолжительность действия.

Сетевые графики в компьютерной или рукописной форме могут успешно применяться в малых проектах.

5.4. Пузырьковая диаграмма как способ представления информации

Одним из ключевых положений проектирования сложных программных систем является комплексный подход к реализации программных проектов, что, в частности, предполагает выполнение сравнительного анализа альтернативных вариантов построения сложных информационных систем

Общие сведения. Пузырьковые диаграммы – это способ представления информации, который наглядно демонстрирует ключевые параметры проекта. Оси *X* и *У* представляют два множества ключевых аспектов (параметров). Местоположение пузырька на диаграмме указывает положение проекта в пространстве этих двух параметров, а его размер и цвет соответствуют дополни-

тельным характеристикам (доля выполненных работ, тип проекта и т.д.). Распределенные по площади диаграммы пузырьки помогают руководителю проекта понять, правильно ли расположены проекты в пространстве ключевых параметров.

Выбор типа диаграммы. Определение типа диаграммы начинается с определения параметров, которые будут представлены по осям диаграммы. Вариантов здесь множество:

- фазы жизненного цикла;
- дата завершения;
- стратегическое соответствие или важность (низкая, средняя, высокая).
- качество ресурсов (низкое, среднее, высоко) ;
- стоимость проекта (рубли, часы работы ресурсов) ;
- вероятность технического или коммерческого успеха;
- легкость исполнения (диапазон от «трудно» до «легко») ;
- категории проектов (НИОКР, производство, инжиниринг, маркетинг, сопровождение и т.д.) ;
- сегменты рынка (рынок А, рынок Б и т.д.).

Совершенно очевидно, что никакая отдельно взятая диаграмма не может всесторонне охарактеризовать портфель проектов. Возникает закономерный вопрос: сколько же диаграмм требуется? И каких? Вам может понадобиться множество диаграмм, но не следует бесконтрольно увеличивать их число и информационную нагрузку. Рекомендуется отобрать необходимый минимум (несколько диаграмм), который сможет точно отразить основные стратегические требования портфеля проектов.

Чтобы графически представить проблемы, связанные с портфелем проектов, менеджеру понадобятся пузырьковые диаграммы, отражающие следующую информацию:

- типы проектов в сопоставлении с сегментом рынка;
- даты завершения в сопоставлении с фазами жизненного цикла;
- риск (вероятность успеха) в сопоставлении с отдачей.

Кроме того, для выявления несбалансированности ресурсов полезна диаграмма, отражающая результаты сравнения между требуемыми и доступными ресурсами.

Преимущества использования пузырьковой диаграммы

- Простота. Их дружественность по отношению к пользователям и легкость применения столь высоки, что специалисты, впервые с ними работающие, нуждаются в минимальной предварительной подготовке или вовсе не нуждаются в ней.

- Основанность на фактических данных. Процедура построения пузырьковой диаграммы полностью формализована, поэтому представление информации не искажается чьим-либо субъективным мнением.

К недостаткам использования этого инструмента следует отнести следующее.

1. Эти диаграммы выступают в качестве средства информационного представления портфеля проектов, а не в качестве модели принятия решений. Как следствие, они не содержат механизма, помогающего принимать решения. Они скорее создают стартовую точку, отталкиваясь от которой руководители будут принимать необходимые действия.

2. Пузырьковые диаграммы не показывают правильный баланс. Менеджер сам должен определить такой баланс и сравнить его с балансом, фактически имеющим место, а также решить, когда изменять баланс, прекращать те или иные проекты, или добавлять новые.

3. Хотя процедура построения диаграммы полностью формализована, подбор исходных данных, построение шкал и выбор критериев целиком зависят от исследователя.

Методика решения задач и примеры их решения. Одна из областей применения пузырьковых диаграмм – управление несколькими проектами. Сталкиваясь с необходимостью управлять шестью или семью проектами одновременно, менеджеры проектов обращаются к пузырьковым диаграммам, чтобы контролировать работу в координатах «размер проекта/фаза проекта», не допуская наличия двух больших проектов, находящихся, например, на стадии планирования. Примеры пузырьковых диаграмм приведены на рис. 5.22 и рис. 5.23.

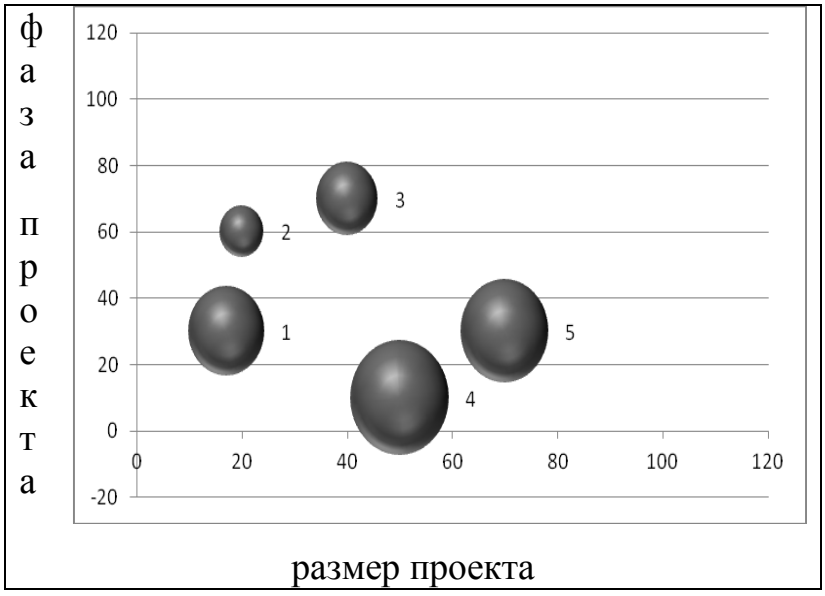


Рис.5.22. Пузырьковая диаграмма в координатах «размер проекта/фаза проекта», числа около пузырьков соответствуют номеру проекта, диаметры пузырьков соответствуют рискам проекта

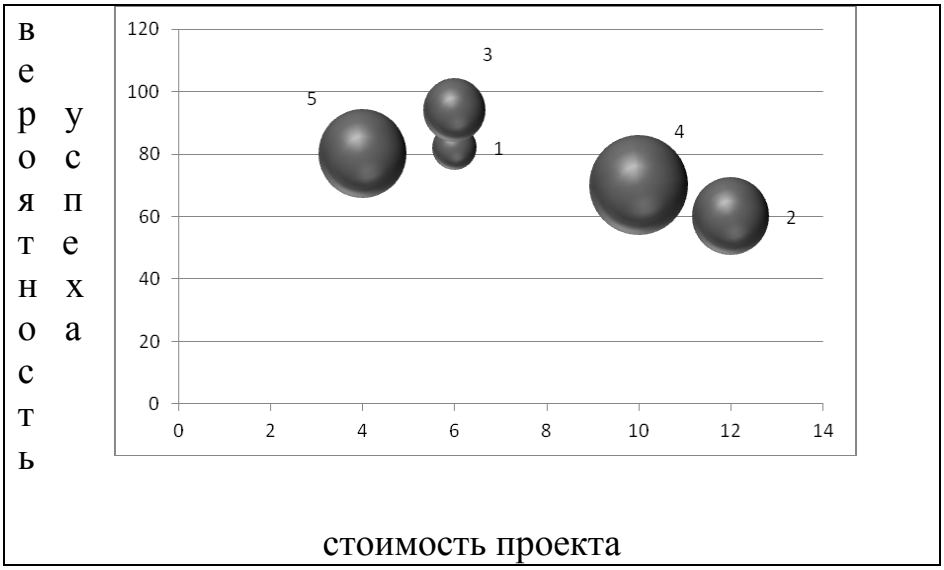
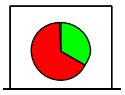
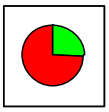
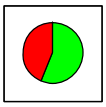
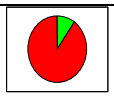


Рис.5.23. Пузырьковая диаграмма в координатах «стоимость проекта (кол-во часов работы)/вероятность успеха», числа около пузырьков соответствуют номеру проекта, диаметры пузырьков соответствуют числу дней, оставшихся до сдачи проекта

Использование таблиц, подобных приведенной ниже, способствует, во-первых, мониторингу за состоянием разных проектов, находящихся на разных стадиях разработки. Во-вторых, рациональному распределению ресурсов, доступных менеджеру портфеля, с учетом стадии проектов, состояния и бюджета проектов.

Таблица 5.3

Состояние портфеля проектов

Концепция				
Планирование				
Выполнение				
Закрытие				

5.5. Методики моделирования трендов состояния сложных объектов

Математико-статистические методы являются основным инструментом исследования текущего и оценивания прогнозируемого состояния сложных систем, к числу которых относятся сложные информационные и программные системы. Одним из классов информационных задач, решаемых при управлении состоянием программных проектов математико-статистическими методами, является анализ тенденций и трендов изменения состояния программных (сложных) систем. Этот класс задач входит в множество задач, связанных с контролем выполнения опорного плана проекта. Например, необходимо оценить, насколько эффективной является выбранный метод проведения совещаний, целью которых является согласование требований к программному продукту с учетом точек зрения разных групп правообладателей [17]. Либо, исходя из динамики выявления количества ошибок в программных модулях по результатам тестирования, оценить предполагаемую дату завершения тестирования и сопоставить эту оценку со значениями планового показателя. Приведенные примеры относятся к числу обязательных задач управления программными проектами, выделенных в рамках группы стандартов ESA-PSS-05-XX.

Под тенденцией понимают некоторое общее направление развития, долговременную эволюцию. Тенденцию ряда динамики представляют в виде гладкой кривой (траектории), которая аналитически выражается некоторой функцией времени, называемой трендом, который характеризует основную закономерность движения во времени, свободную в основном от случайных воздействий.

Анализ тенденций изменения состояния программных систем, в зависимости от стадии жизненного цикла программного продукта, приходится оценивать либо на основе качественных оценок экспертов, либо на основе измерительных данных. Этим обосновывается целесообразность изучения двух нижеприведенных методик.

5.5.1. Оценка характера тенденций на основе качественных исходных данных

При анализе случайных данных часто возникают ситуации, при которых требуется выяснить, являются ли наблюдения статистически независимыми, или они подвержены тренду. Поскольку на начальной стадии программного проекта в силу высокой степени неопределенности состояния проекта получить количественные значения параметров состояния невозможно (см. модель «Конус неопределенности), выносить заключения приходится на основе экс-

пертных оценок. В этом случае исследования удобно проводить на основе *непараметрических методов*, в которых относительно функции распределения исследуемых данных не делается никаких предположений. Критерий серий представляет собою пример таких процедур, весьма полезных для установления статистической независимости и выявления тренда.

Критерий серий. Рассмотрим последовательность *N* наблюдений значений случайной величины x, причем каждое наблюдение отнесено к одному из двух взаимно исключающих классов, которые можно обозначить просто как (+) или (-). Простейший пример дает последовательность заключений эксперта относительно того, является ли предъявляемая ему система требований полной (обоснование того, что мнение эксперта относительно качества требований может меняться во времени, приводится, например, в [4CIO]).

В рассматриваемом случае образуется последовательность вида

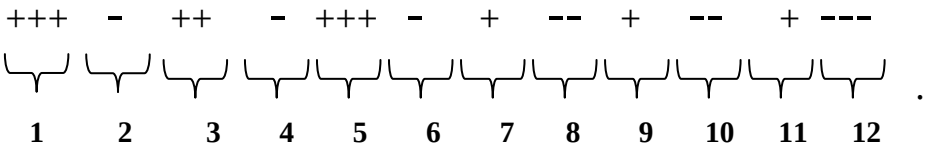


Рис. 5.24.

Серией называется последовательность однотипных наблюдений, перед и после которой следуют наблюдения противоположного типа или же вообще нет никаких наблюдений. В приведенном примере в последовательности из N = 20 наблюдений имеется r = 12 серий.

Число серий, появившихся в последовательности наблюдений, позволяет выяснить, являются ли отдельные результаты независимыми наблюдениями одной и той же случайной величины. Именно, если последовательность N наблюдений состоит из независимых исходов одной и той же случайной величины, т.е. если вероятность отдельных исходов ((+) или (-)) не меняется от наблюдения к наблюдению, то выборочное распределение числа серий в последовательности является случайной величиной r со средним значением и дисперсией

$$\mu_r = \frac{2 * N_1 * N_2}{N} + 1, \qquad (5.1)$$

$$\sigma_r^2 = \frac{2 * N_1 * N_2 (2 * N_1 * N_2 - N)}{N^2 * (N - 1)} \qquad (5.2)$$

Здесь N_1 – число исходов (+), а N_2 – число исходов (-). В частном случае $N_1 = N_2 = N/2$ и выражения (5.1) и (5.2) принимают вид

$$\mu_r = \frac{N}{2} + 1. \qquad (5.3)$$

Самое непосредственное применение критерий серий находит в задачах оценки данных, связанных с выявлением тренда в анализируемой последовательности. Предположим, есть основание подозревать, что в последовательности наблюдений имеется тренд, т.е. есть основания считать, что вероятности появления (+) или (-) меняются от наблюдения к наблюдению. Существование тренда можно проверить следующим образом. Примем в качестве гипотезы, что тренда нет, т.е. предположим, что N наблюдений являются независимыми исходами одной и той же случайной величины. Предположим также, что число исходов (+) равно числу исходов (-). Тогда число серий в последовательности будет иметь выборочное распределение, протабулированное в табл.1. Для проверки гипотезы с любым требуемым уровнем значимости α надо сравнить наблюдаемое число серий с границами области принятия гипотезы, равными $r_{n;1-\alpha/2}$ и $r_{n;\alpha/2}$, где $n = N / 2$. Если наблюдаемое число серий окажется вне этой области, то гипотеза должна быть отвергнута с уровнем значимости α. В противоположном случае гипотезу можно принять.

5.5.2. Оценка характера тенденций на основе количественных измерений

Методы моделирования одномерных временных рядов

Динамика рядов показателей состояния программных систем в общем случае складывается из четырех компонентов:

- тенденции, характеризующие долговременную основную закономерность развития исследуемого явления;
- периодического компонента, связанного с влиянием сезонности развития явления;
- циклического компонента, характеризующего циклические колебания, свойственные явления;
- случайного компонента как результат влияния множества случайных факторов.

В большинстве случаев полученная траектория связывается исключительно со временем. Предполагается, что рассматривая любое явление как функцию времени, можно выразить влияние всех основных факторов. Механизм их влияния в явном виде не учитывается. В связи с этим под трендом обычно понимают регрессию на время. Отклонение от тренда – есть некоторая случайная составляющая, характеризующая влияние случайных факторов. Исходя из этого, уровни временного ряда описываются следующим уравнением:

$$y_t = f(t) + \varepsilon_t, \qquad (5.4)$$

где *f(t)* – систематическая составляющая, характеризующая основную тенденцию явления во времени;

ε_t – случайная составляющая.

Во временных рядах можно наблюдать тенденцию трех видов:

- тенденции среднего уровня;
- тенденции дисперсии;
- тенденции автокорреляции.

Тенденцию среднего уровня наглядно можно представить графиком временного ряда. Аналитически она выражается в виде функции f(t), которой варьируются фактические значения изучаемого явления.

Тенденция дисперсии – это изменения отклонений эмпирических значений временного рода от значений, вычисленных по уравнению тренда.

Тенденция автокорреляции – это тенденция изменения связи между отдельными уровнями временного ряда.

Рассмотрим два метода выявления тенденции по результатам анализа временных рядов.

Один из способов проверки основан на проверке средних уровней ряда [6]. Временной ряд разбивают на две примерно равные части по числу членов, каждая из которых рассматривается как некоторая самостоятельная выборочная совокупность, имеющая нормальное распределение. Если временной ряд имеет тенденцию, то среднее, вычисленное для каждой совокупности, должно существенно (значимо) различаться между собой. Если же расхождение будет незначительным (случайным), то временной ряд не имеет тенденции. Таким образом, проверка наличия тренда в исследуемом ряду сводится к проверке гипотезы о равенстве средних двух нормально распределенных совокупностей.

Необходимым элементом анализа тенденций является проверка гипотезы о равенстве дисперсий. Равенство дисперсий свидетельствует о том, что степень влияния различных факторов, определяющих значения параметров состояния, за исследуемый период времени остается неизменной. Иными словами, выполнение такой проверки обосновывает однородность анализируемых данных, т.е. обосновывает допустимость использования математико-статистического подхода.

Расчеты средних $\hat{\Pi}_j$ и эмпирического среднеквадратического отклонения S^2_j (j = 1; 2) осуществляется на основе соотношений

$$\widehat{П}_j = \frac{\sum_{i=1}^{k_j} П_{ij}}{n_j}; \qquad S_j^2 = \frac{\sum_{i=1}^{n_j} (П_{ij} - \widehat{П}_j)^2}{n_j - 1} \qquad (5.5)$$

Здесь *j* – номер части временного ряда;

Пij – i-е значение временного ряда в *j*-й части;

n_j -число членов в *j*-й части временного ряда.

$$F_{расч} \prec F_{кр}(\alpha, m_1, m_2) \quad (5.6)$$

Проверка гипотезы о равенстве дисперсий Но: $S_1^2=S_2^2$ при уровне значимости α основана на проверке соблюдения неравенства

$$F_{расч} = \frac{\max(s_1^2, s_2^2)}{\min(s_1^2, s_2^2)} \quad (5.7)$$

Здесь α – уровень значимости, а F*расч* и m_j (j=1,2) определяется с помощью соотношения (5.7):

Число степеней свободы эмпирических оценок среднеквадратического отклонения m_k (k=1,2) определяется по правилу :

$m_k = n_k - 1$,

причём k=1 соответствует большей из оценок { S_1^2 ; S_2^2 }, а k=2 соответствует меньшей из оценок { S_1^2 ; S_2^2 }.

Табличные значения $F_{кр}(\alpha, m_1, m_2)$ приводятся в приложении 2.

Проверка основной гипотезы (о равенстве средних частей исходного ряда)

$$H_0: \quad \widehat{П}_1 = \widehat{П}_2; \quad H_1: \widehat{П}_1 \neq \widehat{П}_2$$

основана на проверке выполнения неравенства

$$T_{расч} < t_{кр}(\alpha; m_1 + m_2) \qquad (5.8)$$

Если неравенство соблюдается, то гипотеза об отсутствии тенденций не противоречит фактическим данным. Если же не соблюдается, то гипотеза о наличии тенденций согласуется с фактическими данными.

Величина $T_{расч}$ определяется с помощью соотношения

$$T_{расч} = \frac{|\widehat{П}_1 - \widehat{П}_2|}{\sqrt{(n_1 - 1) * s_1^2 + (n_2 - 1) * s_2^2}} * \sqrt{\frac{n_1 * n_2 * (n_1 + n_2 - 2)}{n_1 + n_2}} \quad (5.9)$$

Значение $t_{кр}(\alpha; m_1+m_2)$ приводится в приложении 3.

Характер тренда (в случае его наличия) определяется значениями $\widehat{П}_1; \widehat{П}_2$. Если $\widehat{П}_1 > \widehat{П}_2$, то тренд убывающий. При $\widehat{П}_1 < \widehat{П}_2$ имеет место возрастающий тренд.

Рассмотрим еще один метод анализа наличия тенденций, предложенный Ф. Фостером и А. Стюартом. Этот метод основан на том, что по данным исследуемого ряда определяются величины U_t и l_t путем последовательного сравне-

ния уровней ряда. Если какой-либо уровень ряда превышает по своей величине каждый из предыдущих уровней, то величине U_t присваивается значение 1, в остальных случаях она равна 0.

Таким образом,

$$U_t = \begin{cases} 1, & если П_t > П_{t-1}, П_{t-2}, ..., П_1 \\ 0 & в \quad остальных \quad случаях \end{cases} \qquad (5.10)$$

И наоборот, если уровень ряда меньше всех предыдущих, то величина l_t равна 1, в остальных случаях она равна 0, т.е.

$$l_t = \begin{cases} 1, & если П_t < П_{t-1} < П_{t-2}, ..., П_1 \\ 0 & в \quad остальных \quad случаях \end{cases} \qquad (5.11)$$

Затем находятся еще две величины s и d:

$$S = \sum S_t \text{, где } S_t = U_t + l_t;$$

$$d = \sum d_t \text{, где } d_t = U_t - l_t \qquad (5.12)$$

Суммирование проводится по всем членам ряда.

Величины S и d асимптотически нормальны и имеют независимые распределения. С помощью S можно проверить, существует ли тенденция изменения в дисперсиях, а d позволяет обнаружить тенденцию в средней. С этой целью проверяются две гипотезы о том, существенно ли отличаются d от 0 и S от μ, где μ – математическое ожидание S. Эти гипотезы проверяются с помощью случайных величин:

$$T_1 = \frac{d-0}{\sigma_2}; \qquad T_2 = \frac{S-\mu}{\sigma_1}, \qquad (5.13)$$

где σ_1- средняя квадратическая ошибка S; σ_2- средняя квадратическая ошибка d. Значения μ, σ_1,σ_2 табулированы для различных длин временных рядов n (приложение 4). Величины T_1 и T_2 имеют распределения Стьюдента с k=n-1 степенями свободы. Значения T_1 и T_2, рассчитанные по (5.13), сравниваются с табличными, найденными по таблице критических точек распределения Стьюдента с

$k=n-1$ степенями свободы при заданном уровне значимости α (приложение 3).

Если $T_{1расч} > t_{кр(\alpha, k)}$, то делается заключение о том, что гипотеза о наличие тенденции в средней соответствует фактическим данным (тенденция есть), в противном случае нет основания отвергать гипотезу об отсутствии тенденции (тенденции нет). Аналогично, если $T_{2расч} > t_{кр(\alpha, k)}$, то гипотеза о наличии тенденции дисперсии не противоречит фактическим данным, если же $|T_{2расч}| < t_{кр(\alpha, k)}$, то нет оснований отвергать гипотезу об отсутствии тенденции в дисперсиях.

5.5.4. Методики оценки состояния программной системы

Пример 1. Методика применения ***критерия серий***.

Пусть имеется следующая последовательность N=20 наблюдений, которым соответствует последовательность вида

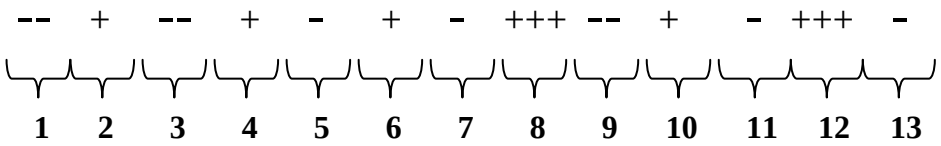


Рис.5.25

В приведенной последовательности из 20 наблюдений имеется 13 серий. Предположим, что наблюдения независимы. Область принятия этой гипотезы имеет вид

$[r_{10;1-\alpha/2} < r \le r_{10;\alpha/2}]$.

Из таблицы. 1 для $\alpha = 0.05$ находим

$r_{10;1-\alpha/2} = r_{10;0.975} = 6, \quad r_{10;\alpha/2} = r_{10;0.025} = 15.$

Гипотеза принимается, т.к. r=13 попадает в интервал, заключенный между 6 и 15. Это означает, что нет оснований сомневаться в независимости наблюдений, т.е. свидетельства в пользу тренда нет.

Пример 2. В табл.5.4 приведены данные, характеризующие значения параметра состояния программного продукта.

Разбив ряд на две равные части по 13 элементов (в первую вошли значения ряда с 1 по 13 включительно; во вторую – с 14 по 26 включительно) на основе (5.5) рассчитываются эмпирические дисперсии, значение которых составляет:

S_1^2=37310.52; S_2^2=11762.08

Значения m_1 и m_2 равны двенадцати. Значение $F_{расч}$ составляет $F_{расч}$=3,17. При *α=0,05* значение $F_{кр}$ составляет 2,69. Неравенство $F_{расч} < F_{кр}$ не соблюдается, следовательно можно предположить, что с течением времени степень влияния различных факторов, определяющих значения параметра $П_1$ изменилась. Например, изменилась внешняя среда программного проекта.

Таблица 5.4

Значение временного ряда

Номер измерения	1	2	3	4	5	6	7	8	9	10	11	12	13
Значение параметра состояния	73,9	118,0	695,7	180,1	241,9	180,9	340,4	163,8	324,8	202,2	167,0	630,5	114,5
Номер измерения	14	15	16	17	18	19	20	21	22	23	24	25	26
Значение параметра состояния	119,5	425,7	63,7	24,18	70,8	29,7	21,8	31,75	46	47,89	44,18	15,36	4942

Пример 3. В табл. 5.5 приведены данные, характеризующие значения параметра состояния программной системы.

Таблица 5.5

Номер измерения	1	2	3	4	5	6	7	8	9	10	11	12	13
Значение параметра состояния	255,0	212,1	306,4	260,8	173,3	121,1	121,1	231,1	181,3	253,4	723,1	557,5	462,5
Номер измерения	14	15	16	17	18	19	20	21	22	23	24	25	26
Значение параметра состояния	154,9	93,1	116,5	485,7	429,2	125,0	123,0	77,38	231,8	116,86	119,1	99,81	65,28

Разбив ряд на два разные части, рассчитаем эмпирических дисперсий:

S_1^2=31976.26; S_2^2=17811.3

Значение $F_{расч}$ составляет $F_{расч}$ = 1.80. При α=0.05 неравенство (5.6) соблюдается, следовательно можно переходить к проверке основной гипотезы. Среднее значение уровней заболеваемости, соответствующие различным частям временного ряда, составляют:

$\widehat{\Pi}_1 = 296.88; \quad \widehat{\Pi}_2 = 172{,}13$

В соответствии с (5.8) значение $T_{расч}$ составляет $T_{расч}$=2.0160. Значение tкр(α, $m_1 + m_2$) при α=0.05 составляет 2.064; неравенство (5.9) не соблюдается, следовательно гипотеза об отсутствии тенденции не противоречит фактическим данным

Пример 4. Метод анализа наличия тенденций, предложенный Ф. Фостером и А. Стюартом.

В таблице 5.6 приведены значения параметра состояния сложной системы, соответствующие различным временным срезам.

Значения параметра состояния.

Таблица 5.6

Номер измерения	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Значение параметра состояния	14.1	9.3	19.4	19.7	5.4	24.2	13.8	24.5	14.7	16.6	5.6	16.2	25.3	11.9	18.5
U	-	0	1	1	0	1	0	1	0	0	0	0	1	0	0
l	-	1	0	0	1	0	0	0	0	0	0	0	0	0	0

В соответствии с (12) значения S = 5+2=7; d = 5-2=3; μ = 4.636; σ_1 = 1.521; σ_2=2.153 при n=15.

Тогда $T_{1расч} = \frac{3-0}{2.153} = 1.39$, $T_{2расч} = \frac{7-4.636}{1.521} = 1.55$

По данным таблицы, приведенной в приложении 3, находим $t_{кр}(0.01;14)$=2.62 (обратите внимание на то, что в таблице приведены значения уровня значимости α двусторонней критической области. Т.е. при *α=0.01*, двусторонняя критическая область составит 0.02).

$T_{1расч} < t_{кр}(0.01, 14)$, т.е. нет оснований отвергать гипотезу об отсутствии тенденции в средней. $T_{2расч} < t_{кр}(0.01, 14)$, т.е. гипотеза об отсутствии тенденции изменения в дисперсиях не противоречит фактическим данным.

Контрольные вопросы:

1. Для чего необходима оценка сложности структурных схем на этапе детального проектирования?
2. Какое граничное значение показателя цикломатической сложности?
3. Какое рекомендуемое значение показателя цикломатической сложности?
4. Для решения каких информационных задач предназначены пузырьковые диаграммы?
5. Что такое портфель проектов?
6. Ограничения инструментального средства «пузырьковая диаграмма».

ЛИТЕРАТУРА

Литература к главе 1

1. Dhillon B.S. Engineering and technology management tools and applications. 2002. P.380
2. ESA PSS-05-08, Выпуск 1, Редакция 1, Март 1995. Руководство по программному обеспечению и управлению проектом. Европейское космическое агентство. Rue Mario-Nikis, 75738 PARIS CEDEX, France.
3. Function Point Counting Practices Manual, Release 4.2, IFPUG, 2004.
4. PMBOK. Руководство к Своду знаний по управлению проектами, 3-е изд., PMI, 2004.
5. Архипенков С. Руководство командой разработчиков программного обеспечения. Прикладные мысли. – М.: 2008
6. Архипенков С.. Лекции по управлению программными проектами. – М.: 2009/ [Электронный ресурс] http://citforum.ru /SE/project/ arkhipenkov _lectures/8.shtml. Дата просмотра 18.12.2014
7. Арчибальд, Р. Управление высокотехнологичными программами и проектами / Рассел Д. Арчибальд; ред. А. Д. Баженов, А. О. Арефьева; пер. с. англ. Е. В. Мамонтова. – 3-е изд., перераб. и доп. – М.: ДМКПресс; Компания АйТи, 2006. – 472 с.
8. Боэм Барри У. Инженерное проектирование программного обеспечения. – М.: Изд-во: Радио и связь, 1985. – 512 с.
9. Боэм Б., Браун Дж., Каспар Х. и др. Характеристики качества программного обеспечения. – М.: Мир, 1981. – 208 с.
10. Брукс Фредерик. Мифический человеко-месяц, или Как создаются программные системы. – М.: Символ-Плюс, 2010. – 304 с.
11. Гвоздев, В.Е., Колоденкова, А.Е. Системные вопросы проектирования программных продуктов. Учебное пособие. – Уфа: АН РБ Изд-во «Гилем», 2010. - 188с.
12. Гецци, К., Джазайери, М., Мандриоли, Д. Основы инженерии программного обеспечения. – СПб. : БХВ-Петербург, 2005.
13. Гради Буч. Объектно-ориентированное проектирование. – 3-е издание. – М.: Бином, 1998. – 560 с.
14. Гузаиров, М.Б. и др. Элементы системной инженерии: методологические основы разработки программных систем на основе V-модели жизненного цикла. Монография / М.Б.Гузаиров, В.Е.Гвоздев, Б.Г.Ильясов, О.Я.Бежаева. – М.: Машиностроение, 2013. – 180с.
15. Гультяев, А.К. MS PROJECT 2002. Управление проектами. Русская версия; Практическое пособие. – Спб.: КОРОНА, 2003. –592 с.

16. Джалота Панкаж. Управление программным проектом на практике. Software Project Management in Practice. – М.: Изд-во «Лори», 2005 г. – с.224.
17. Книги по управлению проектами [Электронный ресурс]: http://www.pmwebinars.ru/blog/knigi-po-upravleniyu-proektami-kotoryie-m Дата просмотра 18.12.2014
18. Липаев В.В. Программная инженерия. Методологические основы. Учебник. – М.: ГУ-ВШЭ. 2006.
19. Липаев В.В. Технико-экономическое обоснование проектов сложных программных средств. – М.: СИНТЕГ. 2004.
20. Липаев В.В. Экономика производства программных продуктов. Издание второе. – М.: СИНТЕГ. 2011.
21. Липаев, В.В. Методы обеспечения качества крупномасштабных программных средств. – М.: СИНТЕГ.– 2003.– 520 с.
22. Макконнелл С. Остаться в живых. Руководство для менеджеров программных проектов. – СПб: Питер, 2006.
23. Макконнелл С. Сколько стоит программный проект. – СПб: Изд-во «Питер», Русская Редакция, 2007. – 297 с.
24. Мацяшек Л. А., Лионг Б. Л. Практическая программная инженерия на основе учебного примера [Текст] / пер. с англ. А. М. Епанешникова и В. А. Епанешникова. – М. : Бином. Лаб. знаний, 2009. – 956 с.
25. Мацяшек Л.А. Анализ требований и проектирование систем: Разработка информационных систем с использованием UML: Пер. с англ. – М.: Вильямс, 2002. – 428 с.
26. Николаев В.И., Брук В.М. Системотехника: методы и приложения. Л.: Машиностроение, 1985.-199 с.
27. Новиков Ф.А., Опалева Э.А. и др. Управление проектами и разработкой программного ПО. - СПб: СПбГУ ИТМО, 2008. - 256 с.
28. Орлов, С.А. Технологии разработки программного обеспечения. Учебник для вузов. – СПб: Питер, 2002. – 463с.
29. Расмуссон Дж. Гибкое управление IT-проектами. Руководство для настоящих самураев. – Питер, 2012. — 272 с.
30. Саати Т. Принятие решений. Метод анализа иерархий. – М.: Радио и связь, 1993. – 278с.
31. Солдатов В.П. Управление программными проектами. – М.: Изд-во: Бином, 2010. – 384с.
32. Список литературы по управлению ИТ-проектами. http://www. koltunova. com /seminars-training/it-project-management /literatura- it- project-management /ozhno-skachat.html. Дата просмотра 18.12.2014

Литература к главе 2

1. Albrecht, A. J. “Measuring Application Development Productivity,” Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, October 14–17, IBM Corporation (1979), pp. 83–92.
2. Albrecht, A. Software Function, Source Lines of Code, and Development Effort Estimation - A Software Science Validation. 1983.
3. Boehm B.W., Abts C., Brown A.W., at al. Software Cost Estimation with COCOMO II. – Prentice Hall, 2000. – 256 p.
4. Boehm B.W., Lane J., Koolmanojwong S., Turner R. The Incremental Commitment Spiral Model: Principles and Practices for Successful Systems and Software. Addison-Wesley Professional, 2014.
5. Demark D.A., McGowan R.L. SADT: Structured Analysis and Design Technique. New York: McCray Hill, 1988.– 378 с.
6. Function Point Counting Practices Manual, Release 4.2, IFPUG, 2004.
7. Pfleeger S.L. Software Engineering. Theory and Practice.- Prentice Hall, 1998. – 576p.
8. Барри У. Боэм Инженерное проектирование программного обеспечения. – М.: Изд-во: Радио и связь, 1985. – 512 с.
9. Боэм Б., Браун Дж., Каспар Х. и др. Характеристики качества программного обеспечения. – М.: Мир, 1981. – 208 с.
10. Арчибальд, Р. Управление высокотехнологичными программами и проектами / Рассел Д. Арчибальд; ред. А. Д. Баженов, А. О. Арефьева; пер. с. англ. Е. В. Мамонтова. – 3-е изд., перераб. и доп. – М.: ДМКПресс; Компания АйТи, 2006. – 472 с.
11. Бабенко Л.П., Лаврищева Е.М. Основы программной инженерии. – М.: Изд-во «Знание», 2001. – 269с.
12. Басс Л., Клементс П., Кацман Р. Архитектура программного обеспечения на практике. – 2-еизд. – СПб.: Питер, 2006. – 575 с.
13. Брауде Э. Технология разработки программного обеспечения – СПб.: Питер, 2004. – 655 с. (Заполнитель1)
14. Буч Г. Объектно-ориентированное проектирование с примерами применения: Пер. с англ. – М.: Конкорд, 1992.- 519 с.
15. Венчковский Л.Б. Разработка сложных программных изделий. Учебное пособие для вузов. – М.:ЗАО “Финстатинформ”, 1999. – 109 с.
16. Гультяев А.К. MS PROJECT 2002. Управление проектами. Русская версия; Практическое пособие. – СПб.: КОРОНА, 2003. –592 с.

17. Лаврищева Е.М. , Петрухин В.А. Методы и средства инженерии программного обеспечения. – М.: МФТИ (государственный университет), 2006.
18. Липаев В.В. Технико-экономическое обоснование проектов сложных программных средств. – М.: СИНТЕГ. 2004.
19. Липаев В.В. Экономика производства программных продуктов. Издание второе. – М.: СИНТЕГ. 2011.
20. Макконнелл С. Сколько стоит программный проект. – СПб: Изд-во «Питер», Русская Редакция, 2007. – 297 с.
21. Мацяшек Л. А., Лионг Б. Л. Практическая программная инженерия на основе учебного примера [Текст] / пер. с англ. А. М. Епанешникова и В. А. Епанешникова. – М. : Бином. Лаб. знаний, 2009. – 956 с.
22. Мацяшек Л.А. Анализ требований и проектирование систем: Разработка информационных систем с использованием UML: Пер. с англ. – М.: Вильямс, 2002. – 428 с.
23. Милошевич Д. Набор инструментов для управления проектами / Драган З. Милошевич; Пер. с англ. Мамонтова Е.В.; Под ред. Неизвестного С.И. — М.: Компания АйТи; ДМК Пресс, 2008. — 729 с.
24. Орлов С.А. Технологии разработки программного обеспечения. Учебник для вузов. – СПб: Питер, 2002. – 463с.
25. Саати Т. Принятие решений. Метод анализа иерархий. – М.: Радио и связь, 1993. – 278с.

Литература к главе 3

1. UML. Специальный выпуск. Питер.–Санкт–Петербург–Москва–Харьков–Минск, 2002. –552с.
2. Бабенко Л.П., Лаврищева Е.М. Основы программной инженерии. – М.: Изд-во «Знание», 2001. – 269с.
3. Гецци, К., Джазайери, М., Мандриоли, Д. Основы инженерии программного обеспечения. – СПб. : БХВ-Петербург, 2005.
4. Горбаченко В.И., Убиенных Г.Ф., Бобрышева Г.В. Проектирование информационных систем с CA ERwin Modeling Suite 7.3: учеб. пособие. – Пенза: Изд-во ПГУ, 2012. – 154 с.
5. Калянов Г.Н. CASE – технологии. Консалтинг в автоматизации бизнес процессов. – 3 –изд. – М.: Горячая Линия- Телеком, 202. – 320с.
6. Лаврищева, Е.М., Петрухин, В.А. Методы и средства инженерии программного обеспечения: учебник. – М.: МФТИ (государственный университет), 2006. – 304с.

7. Маклаков, С.В. BPWin и ERWin – CASE – средства разработки информационных систем. – М.: Диалог-МИФИ, 1999.
8. Орлов С.А. Технологии разработки программного обеспечения. Учебник для вузов. СПб: Питер, 2002. – 463с.
9. Трофимов, С.А. CASE-технологии: практическая работа в Rational Rose. – 2–е изд. – М.: Бином–Пресс, 2002.–288с.
10. Уилсон, С.Ф., Мєйлс, Б., Ленгрейв, Т. Принципы проектирования и разработки программного обеспечения: учебный курс MCSD, пер. с англ. – М.: Из–во торговый дом «Русская редакция», 2000. – 608с.
11. Уэнди Боггс, Майкл Боггс. UML и Rational Rose. – М.: Изд-во: Лори, 2000.
12. Федотова, Д.Э., Семенов, Ю.Д., Чижик, К.Н. CASE-технологии: практикум. – М.: Горячая линия-Телеком, 2003. – 160 с.

Литература к главе 4

1. Гультяев А.К. MS PROJECT 2002. Управление проектами. Русская версия; Практическое пособие. – СПб.: КОРОНА, 2003. –592 с.
2. Липаев В.В. Экономика производства сложных программных продуктов. – М.: СИНТЕГ, 2008. -432с.
3. Липаев В.В. Экономика производства программных продуктов. Издание второе. – М.: СИНТЕГ. 2011.
4. Фаулер М., Скотт К.. UML в кратком изложении. Применение стандартного языка объектного моделирования: Пер. с англ. – М.: Мир, 1999.
5. Макконнелл С. Сколько стоит программный проект. – СПб: Изд-во «Питер», Русская Редакция, 2007. – 297 с.
6. Милошевич Д. Набор инструментов для управления проектами – М.: Компания АйТи: ДМК Пресс, 2008. – 729с.
7. Орлов С.А. Технологии разработки программного обеспечения: учеб. пособие. – СПб.: Питер, 2003. – 480с.
8. Промыслов В.Г., Жарко Е.Ф., Промыслова О.А. Практические аспекты сопровождения и модификации сложных программных систем. Труды IV Международной конференции "Идентификация систем и задачи управления" SICPRO '05 Москва 25-28 января 2005 г. Приложение 1. Цикломатическая сложность
9. Arthur H.Watson, Thomas J. McCabe, “Structured Testing: A Testing Methodology Using Cyclomatic Complexity Metric”, NIST Special Publication 500-235, 1996 (http://hissa.nist.gov/HHRFdata/Artifacts/ITLdoc/235/title.htm)
10. T.J. McCabe, "A complexity measure," IEEE Transactions on Software Engineering, vol. SE-2, no. 4, pp. 308-320, December, 1976.

11. Myers, G., “An Extension to the Cyclomatic Measure of Complexity”, SIGPLAN Notices, October 1977

Литература к главе 5

1. Guide to software verification & validation ESA PSS-05-10.
2. Бэгьюли Ф. Управление проектом /Фил Бэгъюли. – Пер. с англ. – М.: ФАИР-ПРЕСС, 2002. – 208 с.
3. Вентцель Е.С. Элементы динамического программирования DJVU. – М.: Наука, 1964. – 176с.
4. Гвоздев В.Е., Бежаева О.Я, Ефремова О.А., Таназлы Г.И. Программные проекты: базовые термины и определения.: учеб. Пособие / В.Е.Гвоздев и др.; Уфимск. Гос. Авиац. Техн. Ун-т. – Уфа, УГАТУ, 2011. – 218 с.
5. Гвоздев В.Е., Колоденкова А.Е. Системные вопросы проектирования программных продуктов: учеб. пособие / В.Е.Гвоздев, А.Е.Колоденкова; Уфимск. Гос. Авиац. Техн. ун-т. – Уфа: Изд-во «Гилем», 2010.
6. Гвоздева Т. В. Проектирование информационных систем: учеб. пособие / Т. В. Гвоздева, Б. А. Баллод .— Ростов-на-Дону : Феникс, 2009 . – 509 с.
7. Гранберг А.С., Горбачев НН, Бондаренко АС Информационные технологии управления. – М.: ЮНИТИ-ДАНА, 2004. – 479 с.
8. Дуброва Т.А. Статистические методы прогнозирования в экономике. - М.: Московский международный институт эконометрики, информатики, финансов и права, 2003. – 50 с.
9. Информационные системы: [учебное пособие для студентов вузов, обучающихся по направлению подготовки дипломированных специалистов "Информатика и вычислительная техника"] / Ю. С. Избачков [и др.] . – 3-е изд. – СПб: Питер, 2011 . – 544 с.
10. Липаев В.В. Сопровождение и управление конфигурацией сложных программных средств. – М.: СИНТЕГ, 2006. – 372с.
11. Липаев В.В. Экономика производства программных продуктов. Издание второе. – М.: СИНТЕГ. 2011.
12. Макконнелл С. Сколько стоит программный проект. – М.: «Русская редакция». - СПб.: Питер, 2007. – 297с.
13. Мацяшек Л. А., Лионг Б. Л. Практическая программная инженерия на основе учебного примера [Текст] / пер. с англ. А. М. Епанешникова и В. А. Епанешникова. – М. : Бином. Лаб. знаний, 2009. – 956 с.
14. Мацяшек Л.А. Анализ требований и проектирование систем: Разработка информационных систем с использованием UML: Пер. с англ. – М.: Вильямс, 2002. – 428 с.

15. Милошевич Д. Набор инструментов для управления проектами – М.: Компания АйТи: ДМК Пресс, 2008. – 729с.
16. Садовникова Н.А., Шмойлова Р.А. Анализ временных рядов и прогнозирование: учеб. пособие. – М.: МЭСИ, 2001 г. – 67с.
17. Учебник 4CIO. Версия 1.0. – М.: 2011. – 381с.

Приложение 1

Таблица 1. Процентные точки распределения серий

Значения $r_{n;r}$ такие, что Prob[$r_n > r_{n;\alpha}$] = α, где $n = N_1 = N_2 = N$

n=N/2	***α***					
	0.99	**0.975**	**0.95**	**0.05**	**0.025**	**0.01**
5	2	2	3	8	9	9
6	2	3	3	10	10	11
7	3	3	4	11	12	12
8	4	4	5	12	13	13
9	4	5	6	13	14	15
10	5	6	6	15	15	16
11	6	7	7	16	16	17
12	7	7	8	17	18	18
13	7	8	9	18	19	20
14	8	9	10	19	20	21
15	9	10	11	20	21	22
16	10	11	11	22	22	23
18	11	12	13	24	25	26
20	13	14	15	26	27	28
25	17	18	19	32	33	34
30	21	22	24	37	39	40
35	25	27	28	43	44	46
40	30	31	33	48	50	51
45	34	36	37	54	55	57
50	38	40	42	59	61	63
55	43	45	46	65	66	68
60	47	49	51	70	72	74
65	52	54	56	75	77	79
70	56	58	60	81	83	85
75	61	63	65	86	88	90
80	65	68	70	91	93	96
85	70	72	74	97	99	101
90	74	77	79	102	104	107
95	79	82	84	107	109	112
100	84	86	88	113	115	117

Приложение 2

Критические точки распределения F Фишера-Снедекора
(k_1 - число степеней свободы большей дисперсии,
k_2 - число степеней свободы меньшей дисперсии)

Уровень значимости α=0.01												
$к_2/к_1$	1	2	3	4	5	6	7	8	9	10	11	12
1	4025	4999	5403	5625	5764	5889	5928	5981	6022	6056	6082	6106
2	98,49	99,01	90,17	99,25	99,33	99,30	99,34	99,36	99,36	99,40	99,41	99,42
3	34,12	30,81	29,46	28,71	28,34	27,91	27,67	27,39	27,34	27,23	27,13	27,05
4	21,20	18	16,69	15,98	15,52	15,21	14,98	14,8	14,66	14,54	14,45	14,37
5	16,26	13,27	12,06	11,39	10,97	10,67	10,45	10,27	10,15	10,05	9,96	9,89
6	13,74	10,92	9,78	9,15	8,75	8,47	8,26	8,1	7,98	7,87	7,79	7,72
7	12,25	9,55	8,45	7,85	7,46	7,19	7	6,84	6,71	6,62	6,54	6,47
8	11,26	8,65	7,59	7,01	6,63	6,37	6,19	6,03	5,91	5,82	5,74	5,67
9	10,56	8,02	6,99	6,42	6,06	5,8	5,62	5,47	5,35	5,26	5,18	5,11
10	10,04	8,56	6,55	5,99	5,64	5,39	5,21	5,06	4,95	4,85	4,78	4,71
11	9,86	7,20	6,22	5,67	5,32	5,07	4,88	4,74	4,63	4,54	4,46	4,40
12	9,33	6,93	5,95	5,41	5,06	4,82	4,65	4,50	4,39	4,30	4,22	4,16
13	9,07	6,7	5,74	5,2	4,86	4,62	4,44	4,3	4,19	4,10	4,02	3,96
14	8,86	6,51	5,56	5,03	4,69	4,46	4,28	4,12	4,03	3,94	3,86	3,8
15	8,68	6,36	5,42	4,89	4,56	4,32	4,14	4,00	3,89	3,8	3,73	3,67
16	8,53	6,23	5,29	4,77	4,44	4,20	4,03	3,89	3,78	3,69	3,61	3,55
17	8,4	6,11	5,18	4,67	4,44	4,10	3,93	3,79	3,68	3,59	3,52	3,45

Уровень значимости α=0.05												
$к_2\backslash к_1$	1	2	3	4	5	6	7	8	9	10	11	12
1	161	200	216	225	230	234	237	239	241	242	243	244
2	18,51	19,00	19,16	19,25	19,3	19,33	19,36	19,37	19,38	19,39	19,4	19,41
3	10,13	9,55	9,28	9,12	9,01	8,94	8,88	8,84	8,81	8,78	8,76	8,74
4	7,71	6,94	6,59	6,39	6,26	6,16	6,09	6,04	6,00	5,96	5,93	5,91
5	6,61	5,79	5,41	5,19	5,05	4,95	4,88	4,82	4,78	4,74	4,7	4,68
6	5,99	5,14	4,76	4,53	4,39	4,28	4,21	4,15	4,10	4,06	4,03	3,00
7	5,59	4,74	4,35	4,12	3,97	3,87	3,79	3,73	3,68	3,63	3,6	3,57
8	5,32	4,46	4,07	3,84	3,69	3,58	3,5	3,44	3,39	3,34	3,31	3,28
9	5,12	4,26	3,876	3,63	3,48	3,37	3,29	3,23	3,18	3,13	3,10	3,07
10	4,96	4,10	3,71	3,48	3,33	3,22	3,14	3,07	3,02	12,97	2,94	2,91
11	4,48	3,98	8,59	3,36	3,20	3,09	3,01	2,95	2,90	2,86	2,82	2,79
12	4,75	3,8	3,49	3,26	3,11	3,0	2,92	2,85	2,8	2,76	2,72	2,69
13	4,67	3,80	3,11	3,18	3,02	2,92	2,84	2,77	2,72	2,67	2,63	2,60
14	4,6	3,74	3,34	3,11	2,96	2,85	2,77	2,7	2,65	2,6	2,56	2,53
15	4,54	3,68	3,29	3,06	2,9	2,79	2,70	2,64	2,59	2,55	2,51	2,48
16	4,49	3,63	3,24	,301	2,85	2,74	2,66	2,59	2,54	2,49	2,45	2,12
17	4,45	3,59	3,2	2,96	2,91	2,7	2,62	2,55	2,5	2,45	2,41	2,38

Приложение 3

Критические точки распределения Стьюдента

	Уровень значимости α (двухсторонняя критическая область)					
Число степеней свободы	0,1	0,05	0,02	0,01	0,002	0,001
1	6,31	12,7	31,82	63,7	318,2	637,0
2	2,92	4,3	6,97	9,92	22,33	31,6
3	2,35	3,18	4,54	5,84	10,22	12,9
4	2,13	2,78	3,75	4,6	7,17	8,61
5	2,01	2,57	3,37	4,03	5,89	6,86
6	1,94	2,45	3,14	3,71	5,21	5,96
7	1,89	2,36	3,00	3,5	4,79	5,4
8	1,86	2,31	2,9	3,36	4,5	5,04
9	1,83	2,26	2,82	3,25	4,3	4,78
10	1,81	2,23	2,76	3,17	4,14	4,59
11	1,8	2,2	2,72	3,11	4,03	4,44
12	1,78	2,18	2,68	3,05	3,93	4,32
13	1,77	2,16	2,65	3,01	3,85	4,23
14	1,76	2,14	2,62	2,98	3,79	4,14
15	1,75	2,13	2,6	2,95	3,73	4,07
16	1,75	2,12	2,58	2,92	3,69	4,01
17	1,74	2,11	2,57	2,9	3,65	3,96
18	1,73	2,10	2,55	2,88	3,61	3,92
19	1,73	2,09	2,54	2,86	3,58	3,88
20	1,73	2,09	2,53	2,85	3,55	3,85
21	1,72	2,08	2,52	2,83	3,53	3,82
22	1,72	2,07	2,51	2,82	3,51	3,79
23	1,71	2,07	2,5	2,81	3,49	3,77
24	1,71	2,06	2,49	2,8	3,47	3,74
25	1,71	2,06	2,49	2,79	3,45	3,72
26	1,71	2,06	2,48	2,78	3,44	3,71
27	1,71	2,05	2,47	2,77	3,42	3,69
28	1,7	2,05	2,46	2,76	3,4	3,66
29	1,7	2,05	2,46	2,76	3,4	3,66
30	1,7	2,01	2,46	2,75	3,39	3,65
40	1,68	2,02	2,42	2,7	3,31	3,55
60	1,67	2,00	2,39	2,66	2,23	3,46
120	1,66	1,98	2,36	2,62	3,17	3,37
	1,64	1,96	2,33	2,58	3,09	3,29
	0,05	0,025	0,01	0,005	0,001	0,0005

Приложение 4

Значение средней μ и стандартных ошибок σ_1 и σ_2
для n от 10 до 50

N	μ	σ_1	σ_2
10	3.858	1.288	1.961
15	4.636	1.521	2.153
20	5.195	1.677	2.279
25	5.632	1.791	2.373
30	5.990	1.882	2.417
35	6.294	1.956	2.509
40	6.557	2.019	2.561
45	6.790	2.072	2.666
50	6.998	2.121	2.615

Приложение 5

Лабораторная работа

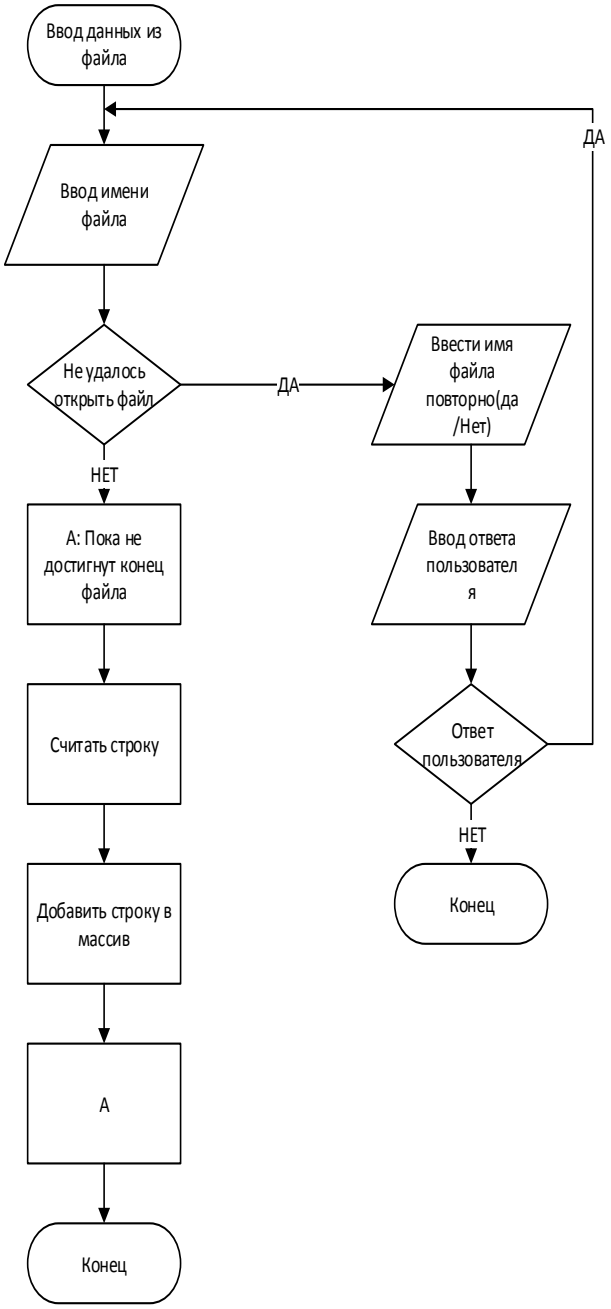
Оценка сложности проекта

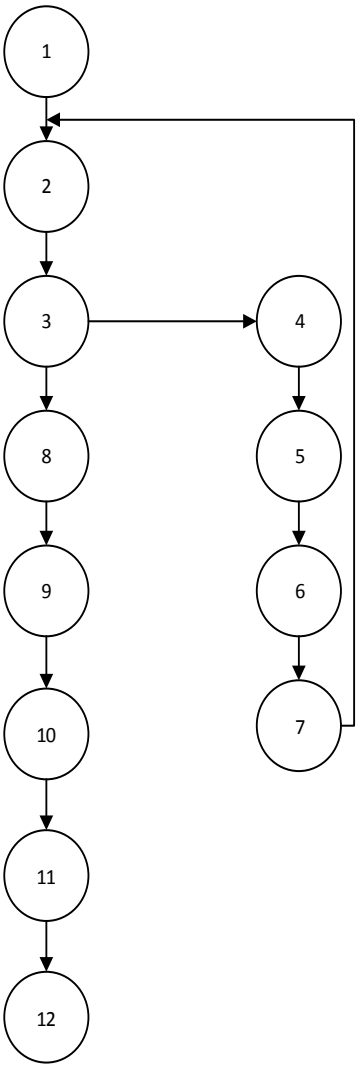
Рассмотрим пример оценки цикломатической сложности в виде лабораторной работы согласно заданию п.5.1.

Цель работы: изучение методов оценивания сложность проектных решений на основе исследования структурных схем.

Рассчитать цикломатические показатели сложности для модулей проекта «Программа совместной обработки файлов типа F1 и F2 для формирования выходного документа типа F3».

1. Модуль «Ввод данных из файла»



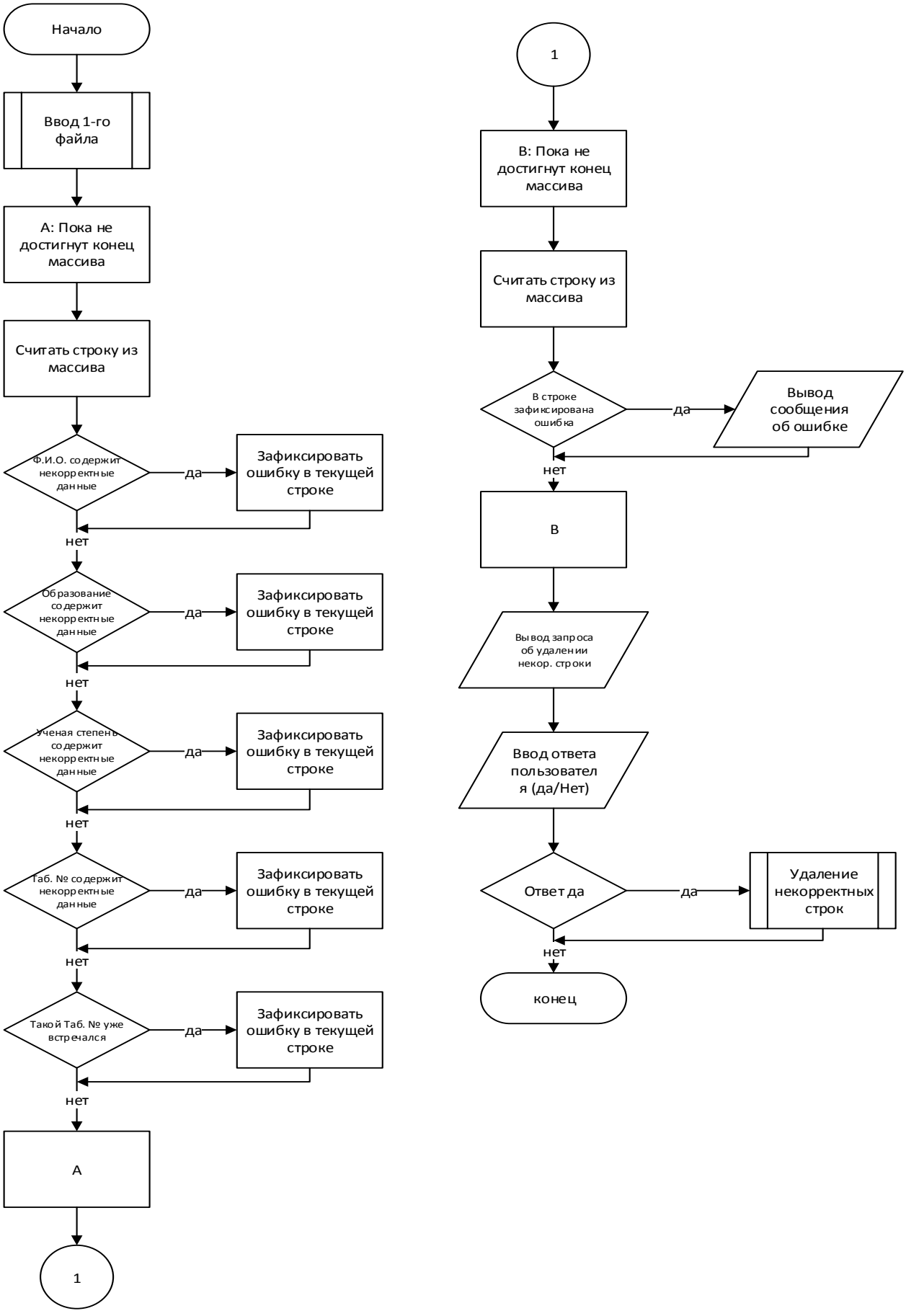


Сформируем на основе функциональной схемы граф (справа):
Рассчитаем показатель цикломатической сложности данного модуля по McCa-be через соотношение:

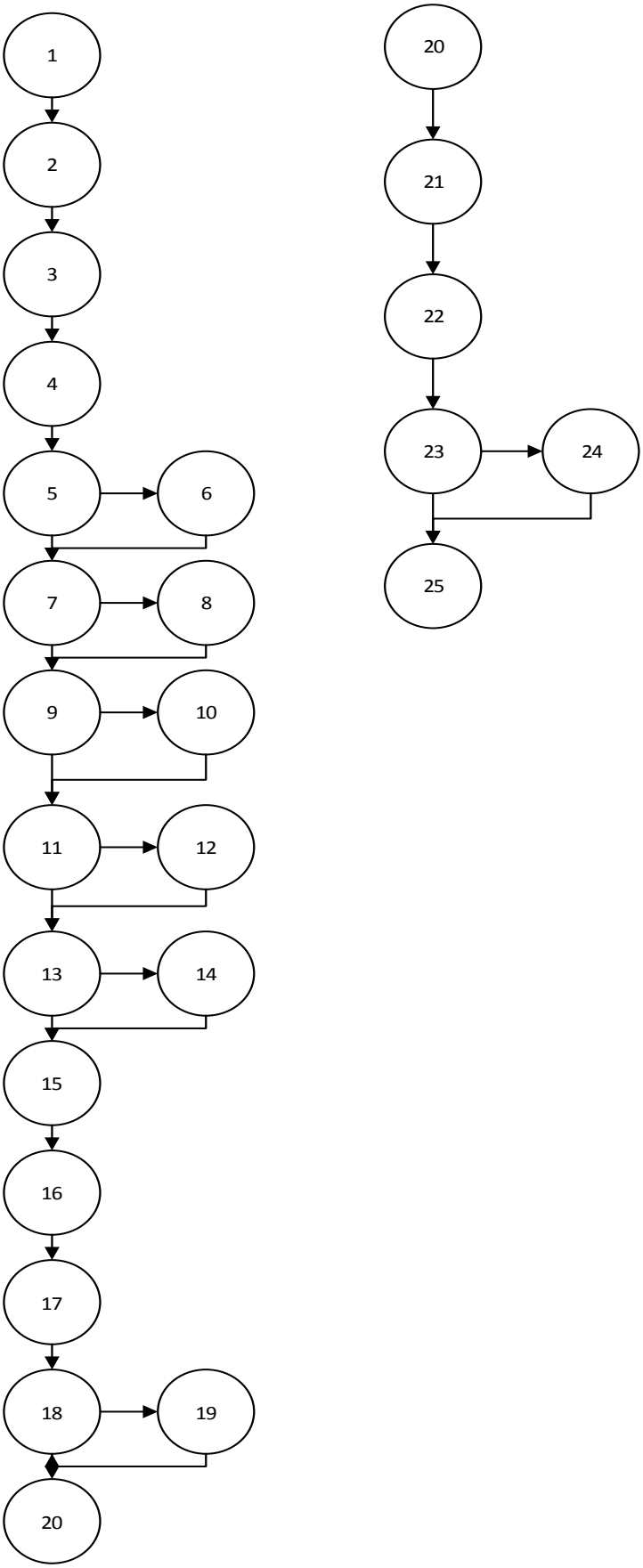
$$v = e - n + 2$$

$e=12$ $n=12$ $v= 12-12+2=2$

2. Модуль «Проверка на корректность содержимого файла F1»



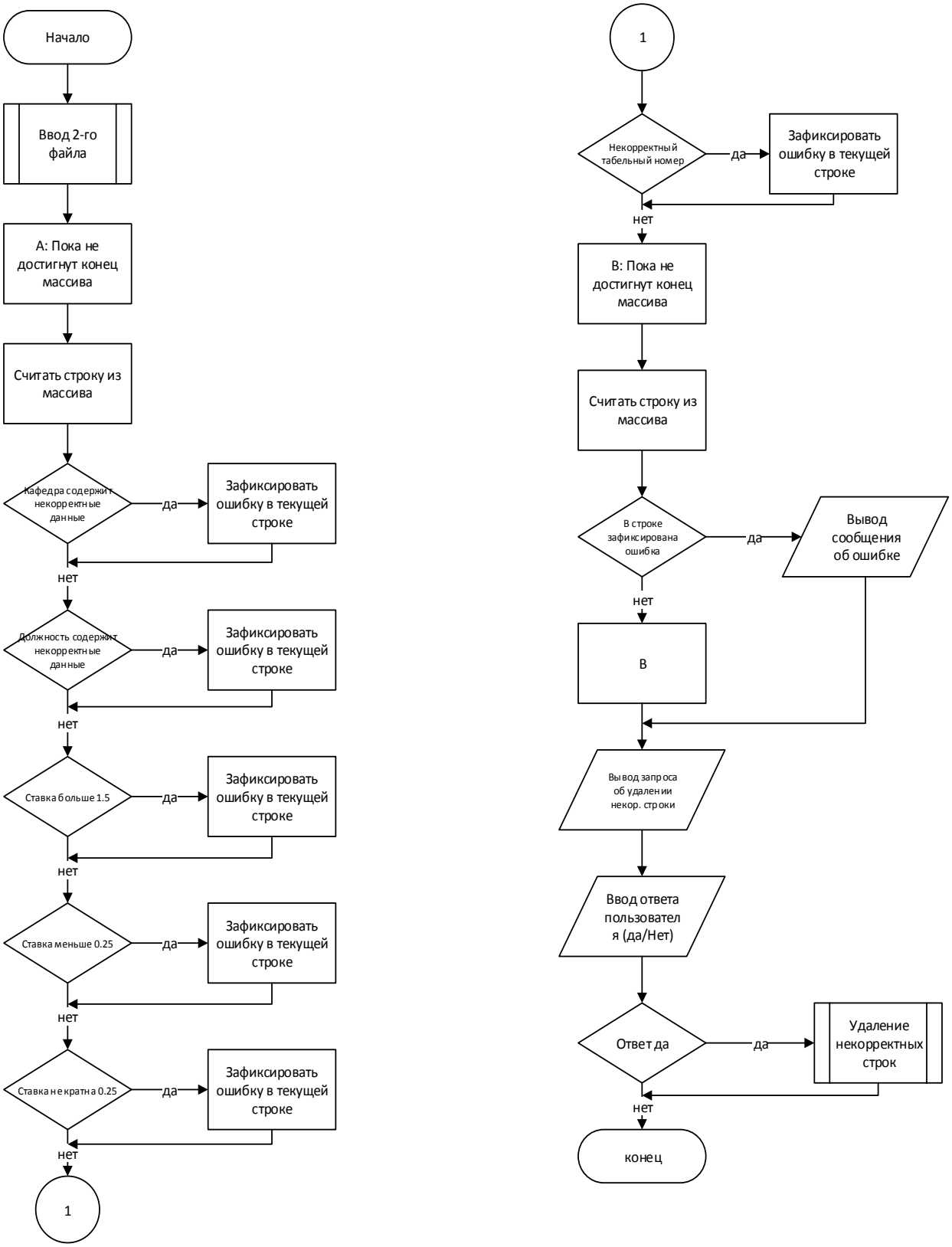
Построим граф для данной схемы:



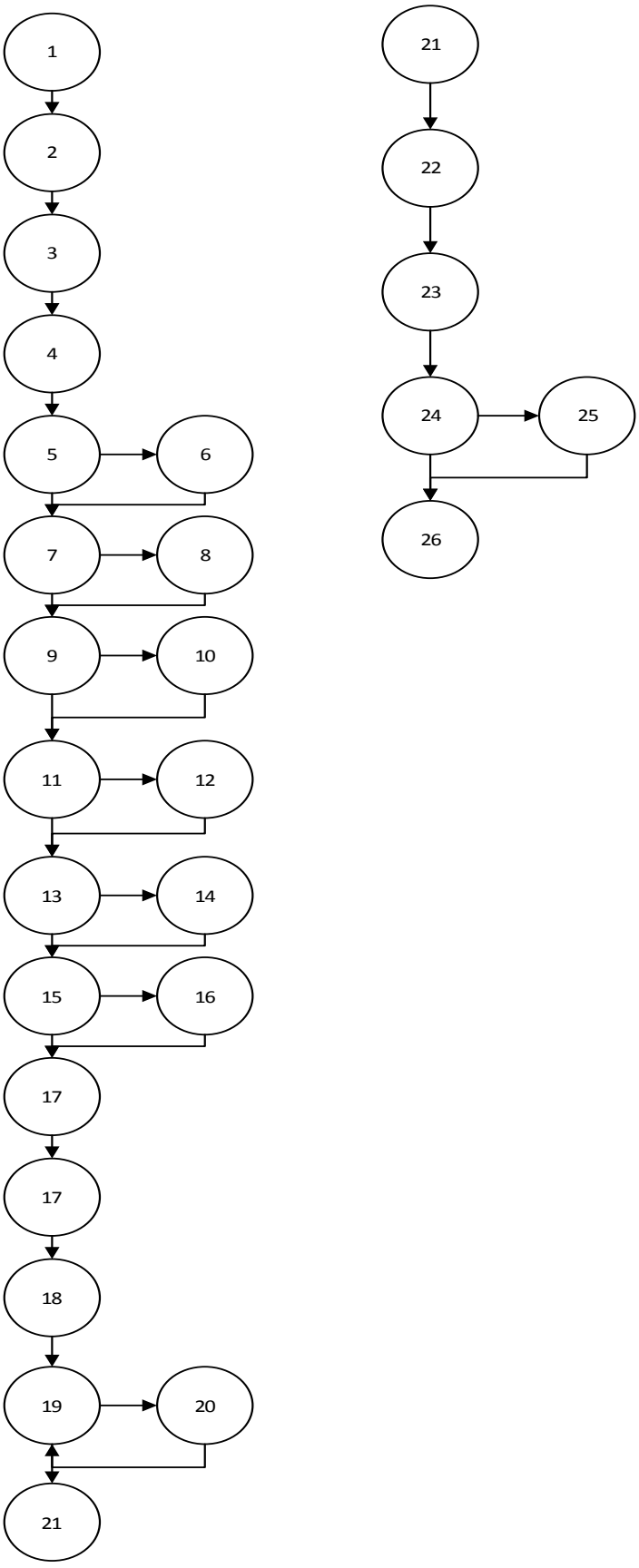
Рассчитаем показатель цикломатической сложности данного модуля

e=31 n=25 v=31-25+2=8

3. Модуль «Проверка на корректность содержимого файла F2»



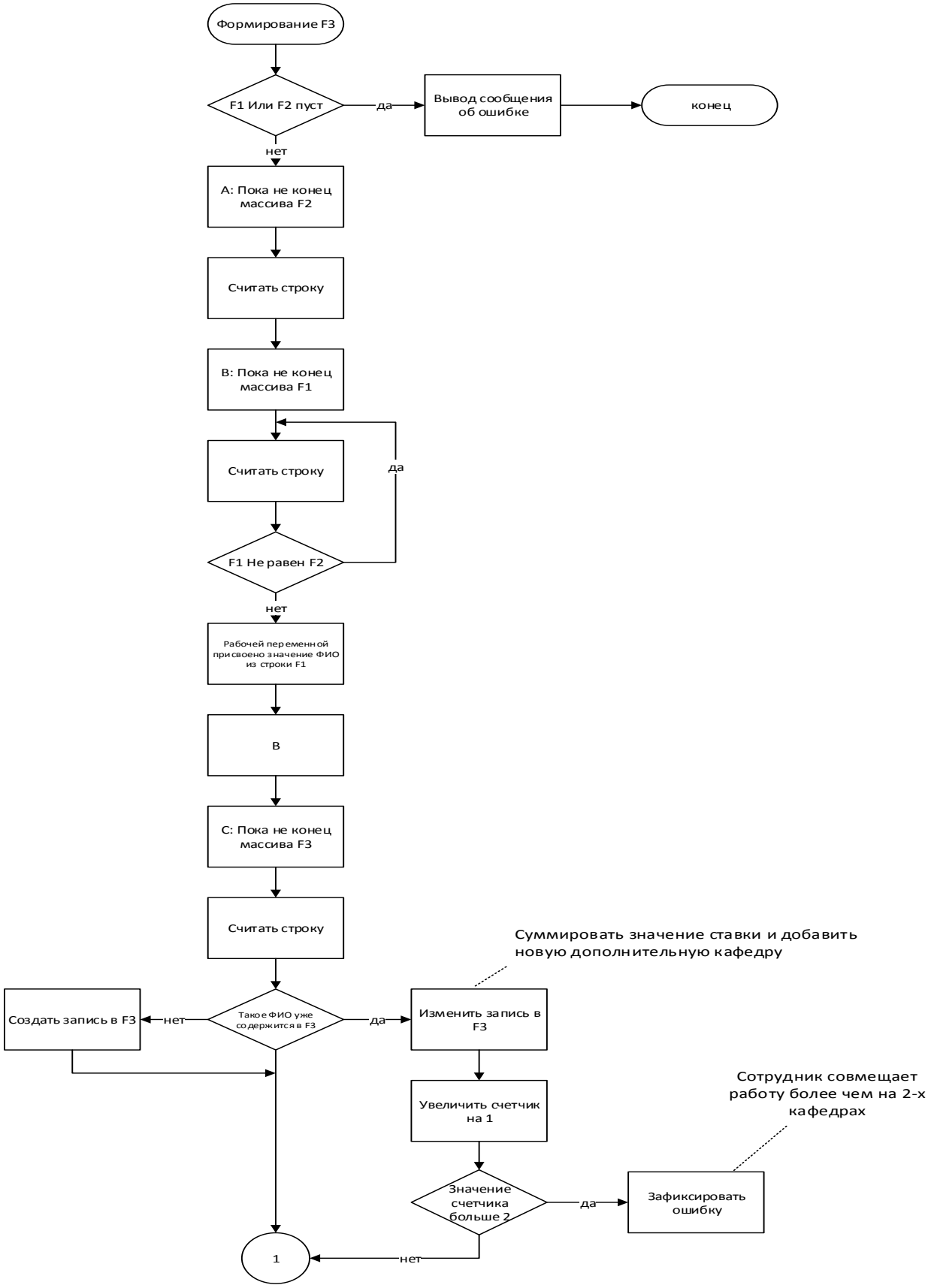
Построим граф данной функциональной схемы:

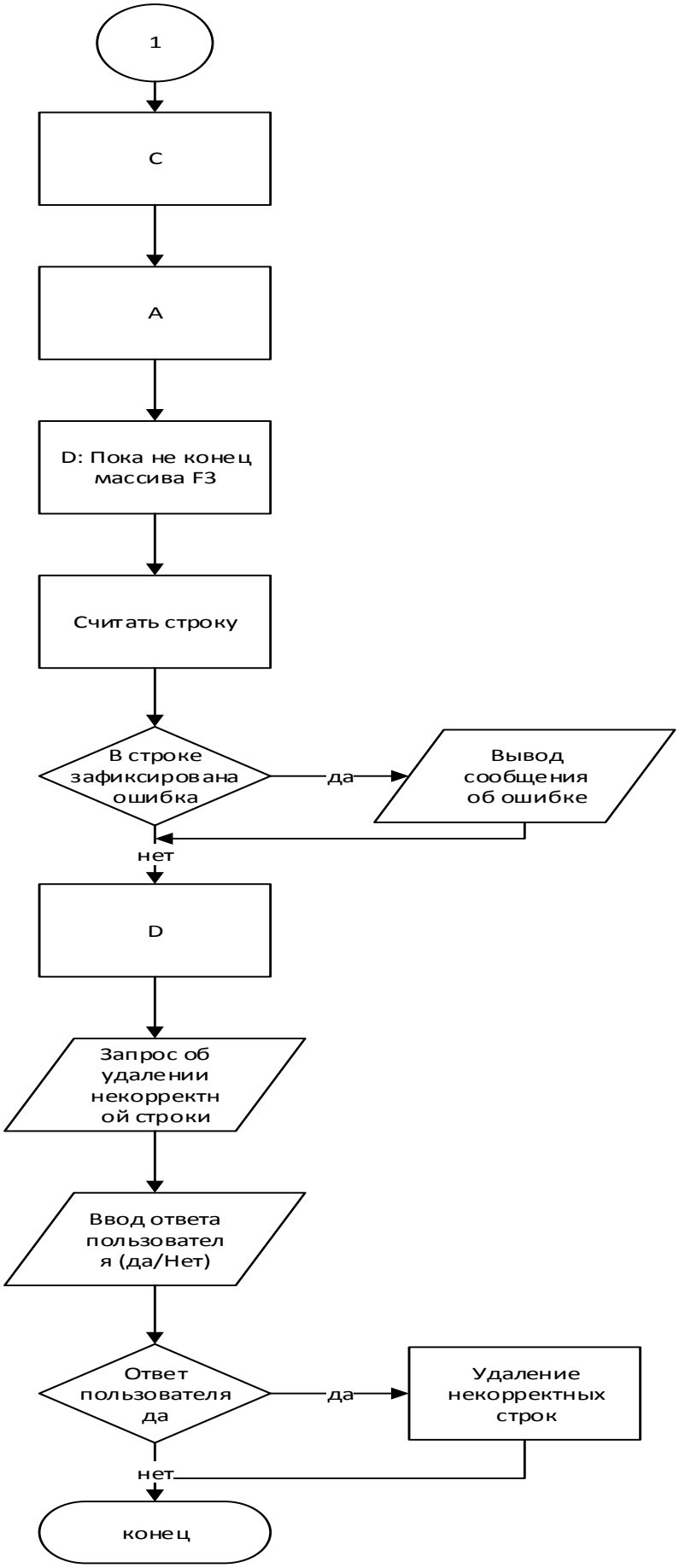


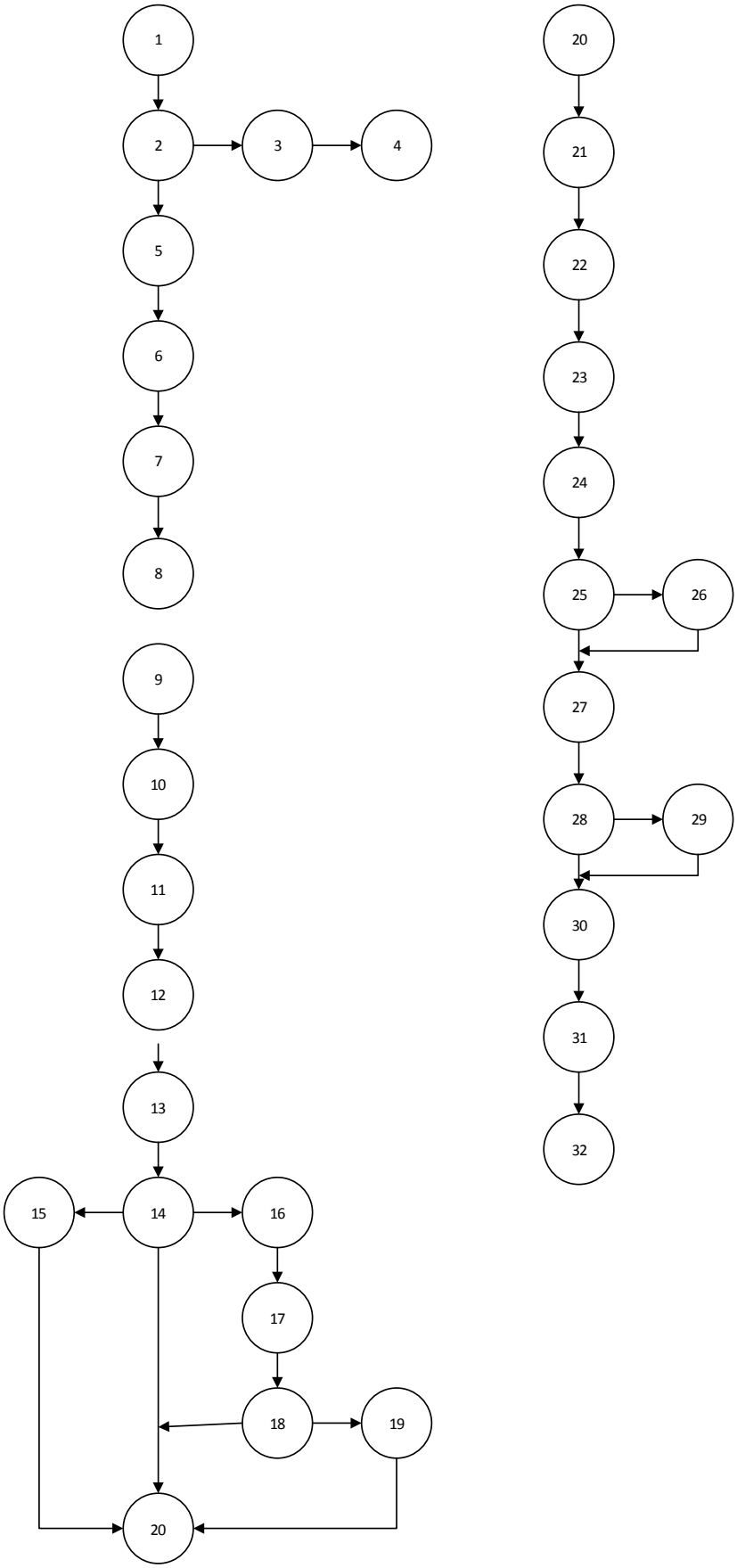
Рассчитаем показатели цикломатической сложности данного модуля

e=33 n=26 v=33-26+2=9

4. Модуль «Формирование перечня»



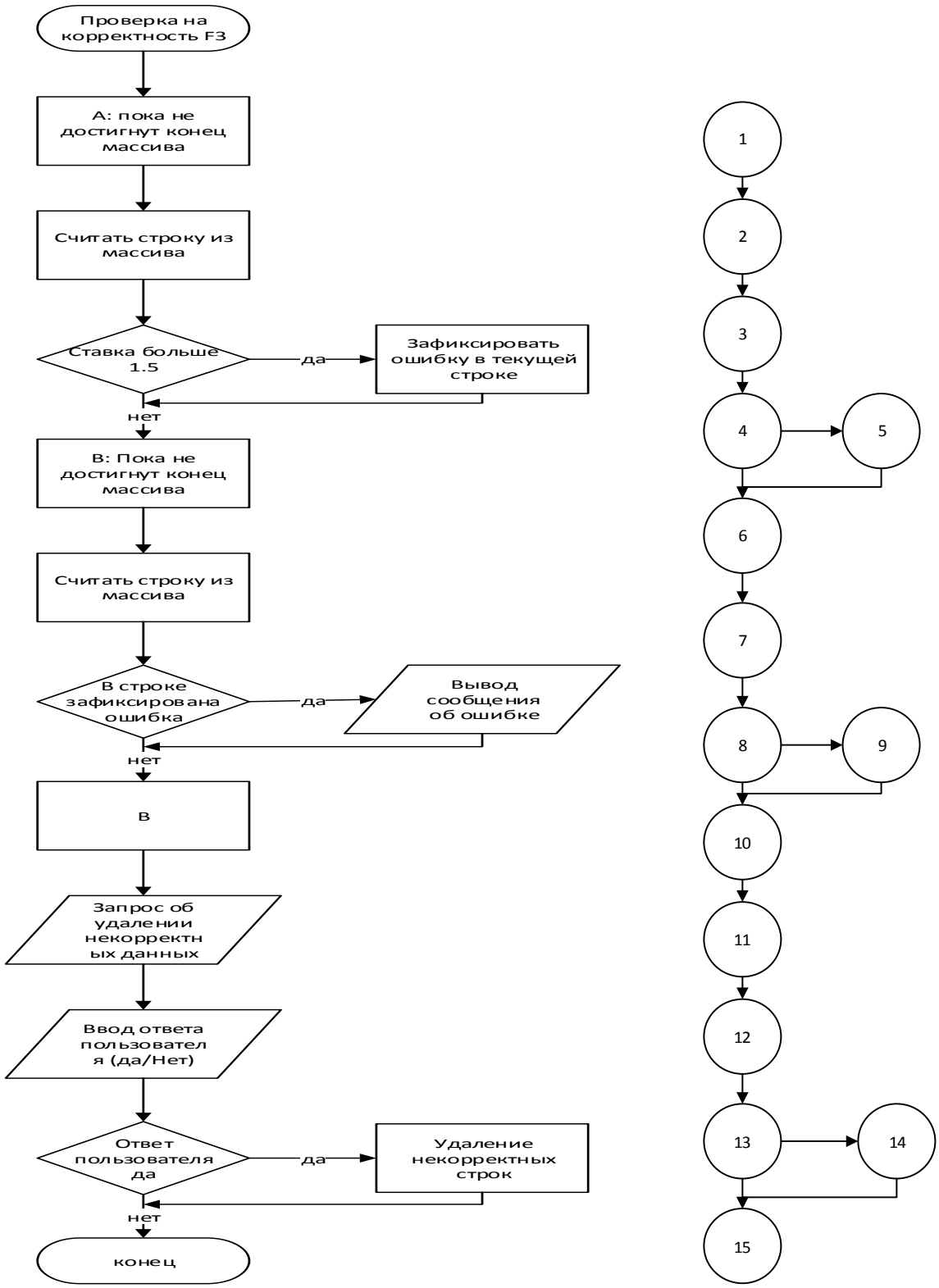

1
C
A
D: Пока не конец массива F3
Считать строку
В строке зафиксирована ошибка
да
Вывод сообщения об ошибке
нет
D
Запрос об удалении некорректной строки
Ввод ответа пользователя (да/Нет)
Ответ пользователя да
да
Удаление некорректных строк
нет
конец



Показатель цикломатической сложности данного модуля

e=37 n=32 v=37-32+2=7

5. Проверка модуля на корректность содержимого файла F3

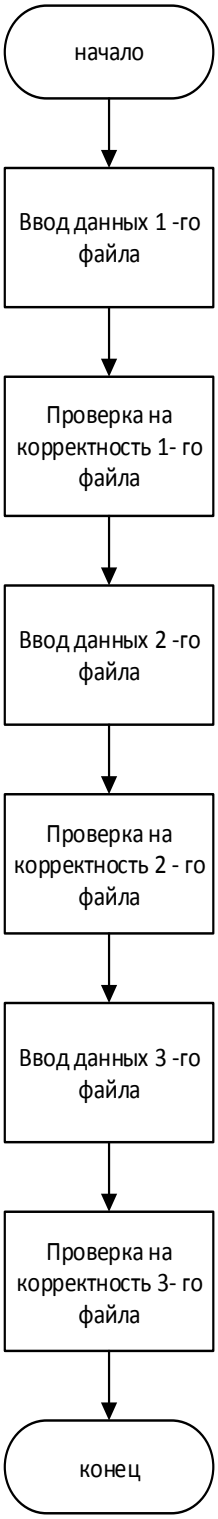


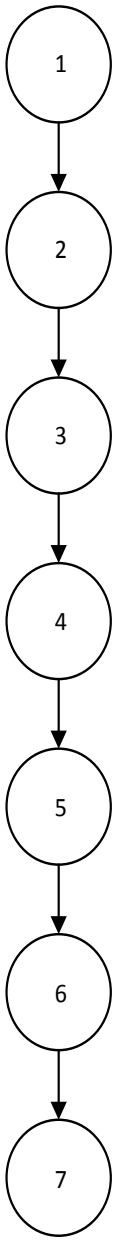
Построим граф

Цикломатическая сложность данного модуля

e=17 n=14 v=17-14+2=5

Структура общей программы





Цикломатическая сложность общей программы

e=7 n=8 v=7-8+2=1

Если при оценке сложности цикломатический показатель превышает граничное значение 10, то в модуле программного проекта с большой вероятностью будет обнаружено существенное количество ошибок, которые повлекут за собой неверную работу модуля, так как на дальнейших этапах проектирования сложность будет только увеличиваться. Поэтому в этих случаях необходимо выявить подобные моменты и перепроектировать модули в которых цикломатическая сложность превышает рекомендованное значение показателя цикломатической сложности – 7.

В данной лабораторной работе были проведены расчеты показателя цикломатической сложности заданного проекта. Для модулей 2, 3 показатель цикломатической сложности превышает рекомендованное значение ($8 > 7$, $9 > 7$), поэтому данные модули подлежат перепроектированию, в остальных модулях показатель не превышает рекомендованное значение. Все модули проекта имеют показатель цикломатической сложности не превышающий граничные значения (10).

Список сокращений

CIO (Chief Informational Officer) –главный руководитель компании в области информационных технологий

ESA – European Space Agency

PSS – Procedures Specifications and Standards

WBS (Work Breakdown Structure) – иерархически организованная совокупность взаимосвязанных пакетов работ

SR – проектирование системных требований

AD – проектирование архитектуры

DD – детальное проектирование

URD – спецификация требований пользователей

SRD – спецификация системных требований

ADD – архитектура программной системы

SPMP (Software Project Management Plan) - план управления программным проектом)

SCMP(Software Configuration Management Plan) – план управления конфигурацией

SVVP (Software Verification & Validation Plan) – верификация и валидация программного продукта

SQAP (Software Quality Assurance Plan) – план обеспечения качества программных продуктов

ERD – Entry-Relationship Diagrams – диаграммы «сущность-связь»

CAD – Computer Aided Design

CASE – Computer Aided Software Engineering – программная инженерия с компьютерной поддержкой

DFD – Data Flow Diagrams

IEC – International Electro technical Commission – Международная комиссия по электротехнике

ISO – International Organization of Standardization – Международная организация по стандартизации

RAD – Rapid Application Development

UML - Unified Modeling Language – универсальный язык моделирования

SADT – Structured Analysis and Design Technique

SWEBOK – Software Engineering of Body Knowledge

STD – State Transition Diagrams - диаграмма переходов состояний

SCM – Software Configuration Management – управление конфигурацией ПО

VRML – Virtual Reality Modeling Language

XP – eXtreme Programming

наиболее ранний срок события – НРСС

наиболее поздний срок события – НПСС

Для замечаний

Учебное издание

Михаил Асхатович Абдрафиков
Владимир Ефимович Гвоздев
Рамиль Фарукович Маликов
Алмаз Раилевич Исхаков

Управление программными проектами: теория и практика

Редактор Т.В. Подкопаева
Технический редактор И.В. Пономарев

Лиц. на издат. деят. Б848421 от 03.11.2000 г. Подписано в печать 17.10.2015.
Формат 60Х84/16. Компьютерный набор. Гарнитура Times New Roman.
Отпечатано на ризографе. Усл. печ. л. – 7,9 Уч.-изд. л. – 7,7.
Тираж 100 экз. Заказ №

ИПК БГПУ 450000, г.Уфа, ул. Октябрьской революции, 3а