Кафедра «Информационные технологии»

Введение в специальность

Курс лекций по дисциплине
«Введение в специальность»
для специальности направления
1-40 05 01-01 «Информационные системы и технологии
(в проектировании и производстве)»

Автор-составитель

Е.Г. Стародубцев, доцент, канд. физ.-мат. наук

Лекции 14, 15
Проектирование и разработка информационных систем

Проблемы разработки сложных программных систем. Блочно-иерархический подход к созданию сложных систем. Жизненный цикл и этапы разработки программного обеспечения. Эволюция моделей жизненного цикла программного обеспечения. Ускорение разработки программного обеспечения. Технология RAD. Оценка качества процессов создания программного обеспечения. Коллективная разработка ПО. Функциональные роли в коллективе разработчиков.

Основные понятия технологии проектирования информационных систем (ИС)

Еще о классификации ИС

Фактографические системы - для хранения и обработки структурированных данных в виде чисел и текстов. Над такими данными можно выполнять различные операции.

Документальные системы - информация представлена в виде документов, состоящих из наименований, описаний, рефератов и текстов. Поиск по неструктурированным данным осуществляется с использованием семантических (смысловых) признаков. Отобранные документы предоставляются пользователю, а обработка данных в таких системах практически не производится.

Еще о классификации ИС

Интегрированные (корпоративные) ИС
(пример – **SAP R/3**)

- Используются для автоматизации всех функций предприятия и охватывают весь цикл работ от планирования деятельности до сбыта продукции.
- Включают модули (подсистемы), работающие в едином информационном пространстве и поддерживающие различные направления деятельности (планирование, бухучет, склад, ...).

Типовые задачи, решаемые модулями КИС

Подсистема маркетинга	Производственные подсистемы	Финансовые и учетные подсистемы	Подсистема кадров (человеческих ресурсов)	Прочие подсистемы (например, ИС руководства)
Исследование рынка и прогнозирование продаж	Планирование объемов работ и разработка календарных планов	Управление портфелем заказов	Анализ и прогнозирование потребности в трудовых ресурсах	Контроль за деятельностью фирмы
Управление продажами	Оперативный контроль и управление производством	Управление кредитной политикой	Ведение архивов записей о персонале	Выявление оперативных проблем
Рекомендации по производству новой продукции	Анализ работы оборудования	Разработка финансового плана	Анализ и планирование подготовки кадров	Анализ управленческих и стратегических ситуаций
Анализ и установление цены	Участие в формировании заказов поставщикам	Финансовый анализ и прогнозирование		Обеспечение процесса выработки стратегических решений
Учет заказов	Управление запасами	Контроль бюджета, бухгалтерский учет и расчет зарплаты		

Анализ современного состояния рынка ИС => ***рост спроса на интегрированные ИС (КИС)***.

Автоматизация отдельной функции, например, бухгалтерского учета или сбыта готовой продукции, считается уже пройденным этапом для многих предприятий.

Перечень наиболее популярного ПО КИС

Локальные системы	Малые интегрированные системы	Средние интегрированные системы	Крупные интегрированные системы (IC)
• БЭСТ • Инотек • Инфософт • Супер-Менеджер • Турбо-Бухгалтер • Инфо-Бухгалтер	• Concorde XAL Exact • NS-2000 Platinum PRO/MIS • Scala SunSystems • БЭСТ-ПРО • 1С-Предприятие • БОСС-Корпорация • Галактика • Парус • Ресурс • Эталон	• Microsoft-Business Solutions - Navision, Axapta • J D Edwards (Robertson & Blums) • MFG-Pro (QAD/BMS) • SyteLine (СОКАП/SYMIX)	• SAP/R3 (SAP AG) • Baan (Baan) • BPCS (ITS/SSA) • OEBS (Oracle E-Business Suite)

Из истории разработки АИС

I этап (1950-1960-е гг.) - проектирование ИС по методу "***снизу-вверх***", когда ИС создавалась как набор приложений, наиболее важных в данный момент для предприятия. Основная цель - не создание тиражируемых продуктов, а обслуживание текущих потребностей конкретного учреждения.

Такой подход ("***лоскутная автоматизация***") встречается и сегодня, т.к. он обеспечивает поддержку отдельных функций предприятия (если нужно только это).

Недостатки «лоскутной автоматизации»

Практически нет стратегии развития комплексной системы автоматизации, а объединение функциональных подсистем превращается в самостоятельную и достаточно сложную проблему.

Создавая свои подразделения по автоматизации, предприятия пытались работать своими силами. Периодические изменения технологий работы и должностных инструкций, разные представления пользователей об одних и тех же данных **=>** непрерывная доработка ПО для удовлетворения новых пожеланий отдельных работников **=>** работа программистов и создаваемые ИС вызывали недовольство руководителей и пользователей ИС.

II этап (1960-1990-е гг.) – разработка стандартного ПО для автоматизации отдельных видов работ различных типов организаций.

Из всего спектра проблем разработчики выделили наиболее заметные: автоматизацию ведения бухучета и технологических процессов. ИС начали проектироваться "***сверху-вниз***", т.е. в предположении, что одна программа должна удовлетворять потребности многих пользователей.

Недостатки II этапа (1960-1990-е гг.)

Заложенные "сверху" жесткие рамки на параметры ИС не дают возможности гибко адаптировать ИС к специфике деятельности конкретного предприятия.

Решение этих задач требует серьезных доработок ИС **=>** материальные и временные затраты на внедрение/доводку ИС под требования заказчика значительно выше запланированных показателей.

Статистика: из **8380** проектов по созданию ИС (США, 1994 г.) неудачными оказались более **30 %** проектов, общая стоимость которых **80 млрд. $**. При этом были выполнены в срок лишь **16 %** от общего числа проектов, а перерасход средств составил **189 %** от запланированного бюджета.

Этап III (1980 - …) - появление **новой методологии построения ИС**.

Цель методологии **- регламентация процесса проектирования ИС и обеспечение управления этим процессом с тем, чтобы гарантировать выполнение требований как к самой ИС, так и к характеристикам процесса разработки**.

Основные решаемые задачи:

обеспечить создание КИС, отвечающих целям и задачам организации, требованиям по автоматизации деловых процессов заказчика;

гарантировать создание КИС с заданными качеством, сроками и бюджетом;

Основные решаемые задачи:

поддерживать удобную дисциплину сопровождения, модификации и наращивания ИС;

обеспечивать преемственность разработки, т. е. использование в разрабатываемой ИС существующей инфраструктуры организации (задела в области информационных технологий).

Современные требования - проектирование ИС охватывает ***3 основные области***:

1) проектирование объектов данных, которые будут реализованы в базе данных;

2) проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;

3) учет конкретной среды или технологии: топологии сети; аппаратных средств; архитектуры ИС (файл-сервер, клиент-сервер); параллельной, распределенной обработки данных и т.п.

Проектирование ИС начинается ***<u>с определения цели проекта</u>***, которую можно определить как

решение взаимосвязанных задач, обеспечивающих на момент запуска и в течение времени эксплуатации ИС:

- требуемую функциональность ИС и уровень ее адаптивности к изменяющимся условиям работы;
- требуемую пропускную способность ИС;
- требуемое время реакции ИС на запрос;
- безотказную работу ИС;
- необходимый уровень безопасности данных;
- простоту эксплуатации и поддержки ИС.

Современный подход: создание ИС – это ***построение и последовательное преобразование ряда согласованных моделей на всех этапах жизненного цикла*** (ЖЦ) ИС.

На каждом этапе ЖЦ создаются ***специальные модели:***

- ***модели организации,***
- ***модели требований к ИС,***
- ***модели проекта ИС,***
- ***модели требований к приложениям***

и т.д.

Современный подход: создание ИС – это ***построение и последовательное преобразование ряда согласованных моделей на всех этапах жизненного цикла*** (ЖЦ) ИС.

Модели формируются рабочими группами команды проекта, сохраняются и накапливаются в ***репозитории проекта***. Создание моделей, их контроль, преобразование и предоставление в коллективное пользование - с помощью специального ПО - ***CASE-средств***.

Процесс создания ИС делится на ряд ***этапов*** (***стадий***), ограниченных по времени и заканчивающихся выпуском конкретного продукта (моделей, ПО, документации и пр.).

Обычно выделяют следующие ***этапы создания ИС***:

1) формирование требований к ИС,

2) проектирование,

3) реализация,

4) тестирование,

5) ввод в действие,

6) эксплуатация,

7) сопровождение.

Модели ЖЦ ИС

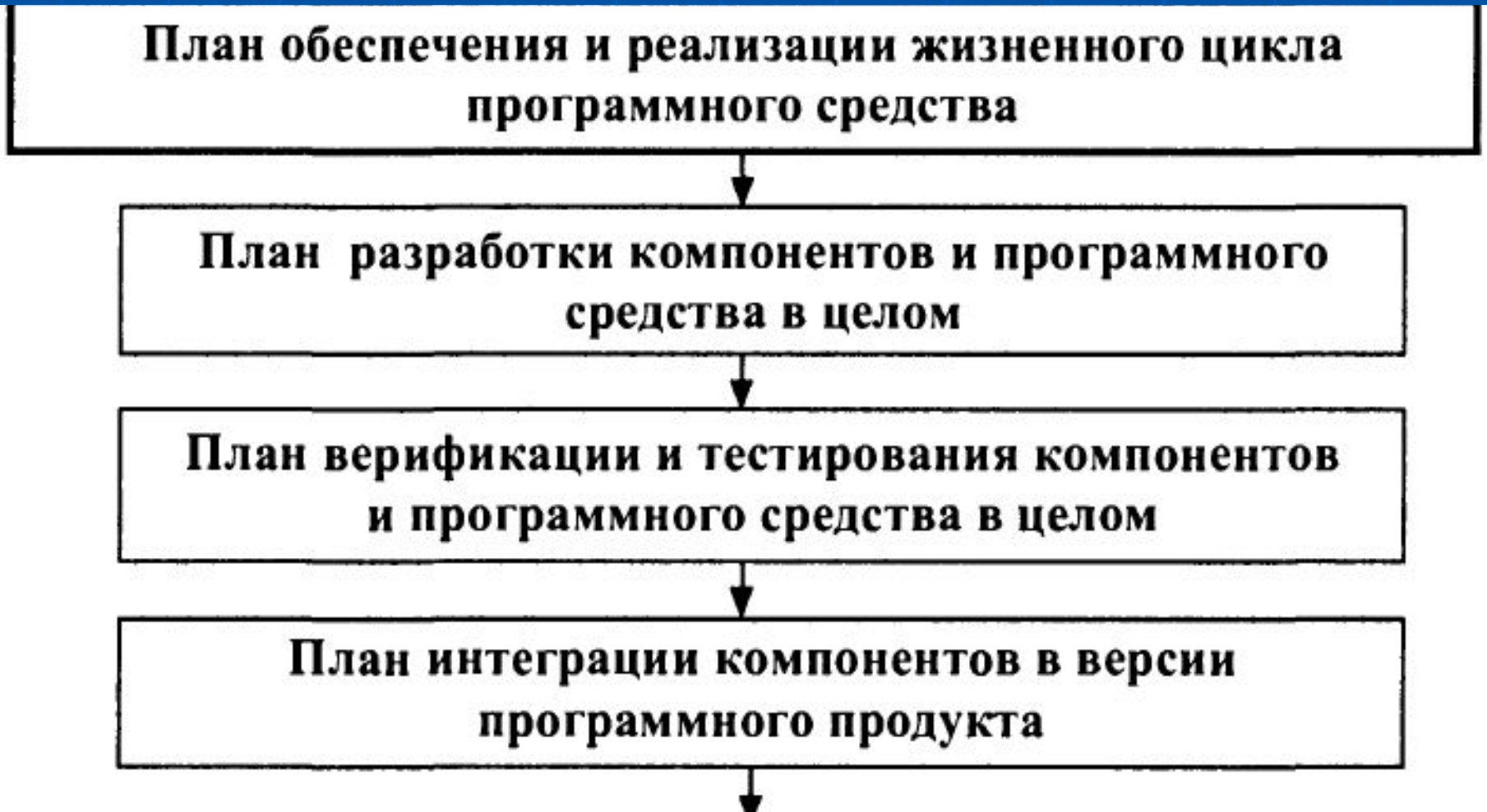
ЖЦ ИС - ряд событий, происходящих с ИС при ее создании и использовании.

Модель ЖЦ отражает различные состояния ИС, начиная с момента возникновения необходимости в данной ИС и заканчивая моментом ее полного выхода из употребления.

Модель ЖЦ - структура, содержащая ***процессы***, ***действия*** и ***задачи***, выполняемые в ходе разработки, работы и сопровождения ИС в течение всей жизни системы.

Планирование ЖЦ ИС (ПС)

Стандартами **ISO 16326** и **ISO 90003** рекомендуется при планировании ЖЦ ПС подготовить и утвердить следующие планы:



Планирование ЖЦ ИС (ПС)

План сопровождения и управления конфигурацией программного средства

План тиражирования, адаптации и внедрения программного продукта

План документирования процессов и результатов жизненного цикла программного средства

План технологического обеспечения качества и безопасности применения программного средства

Планирование ЖЦ ИС (ПС)

План подготовки и обучения пользователей применению программного продукта

План обслуживания пользователей при эксплуатации программного продукта

План организации переноса компонентов и версий программного продукта на иные платформы

Схема процессов ЖЦ ИС

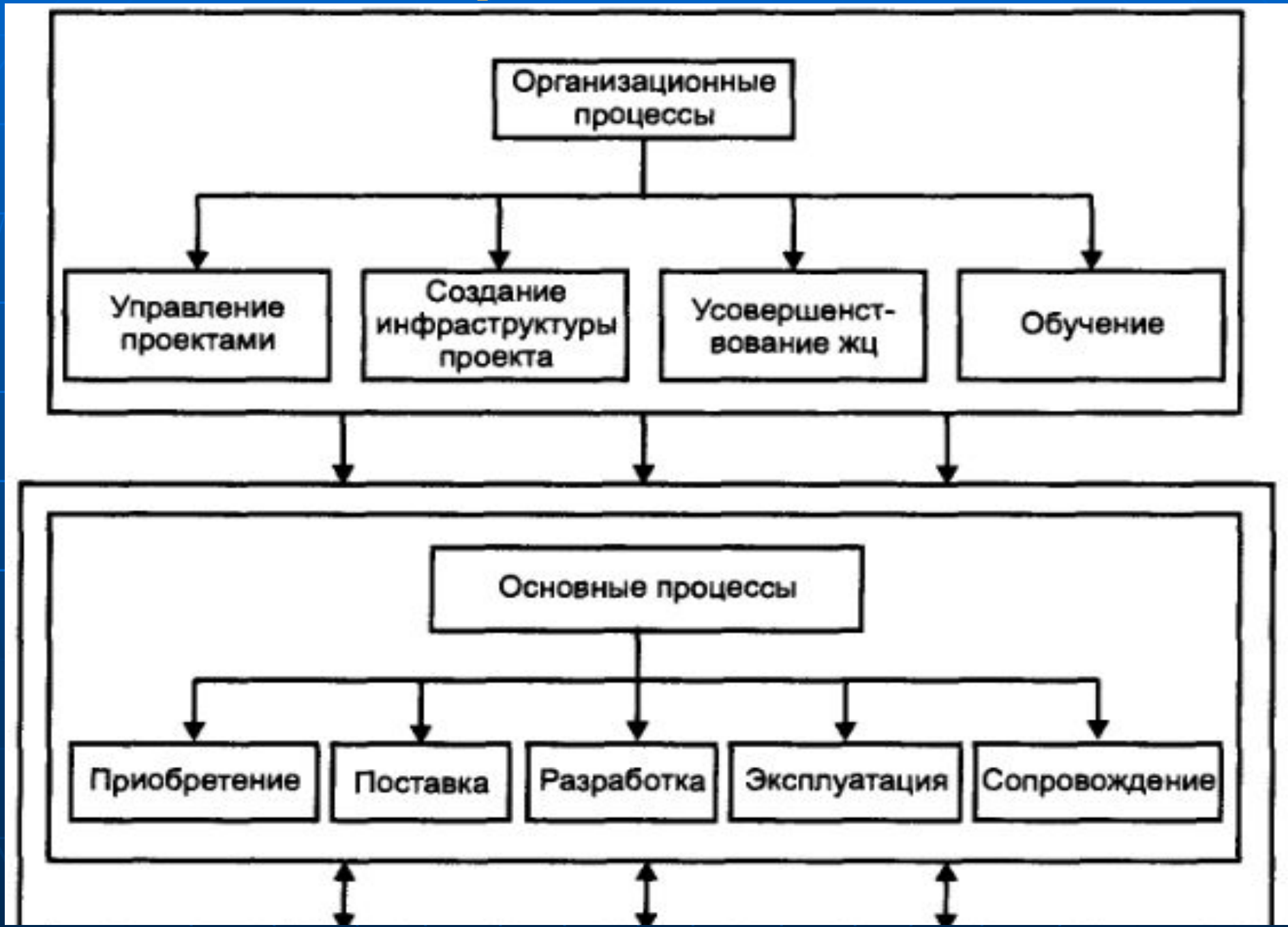
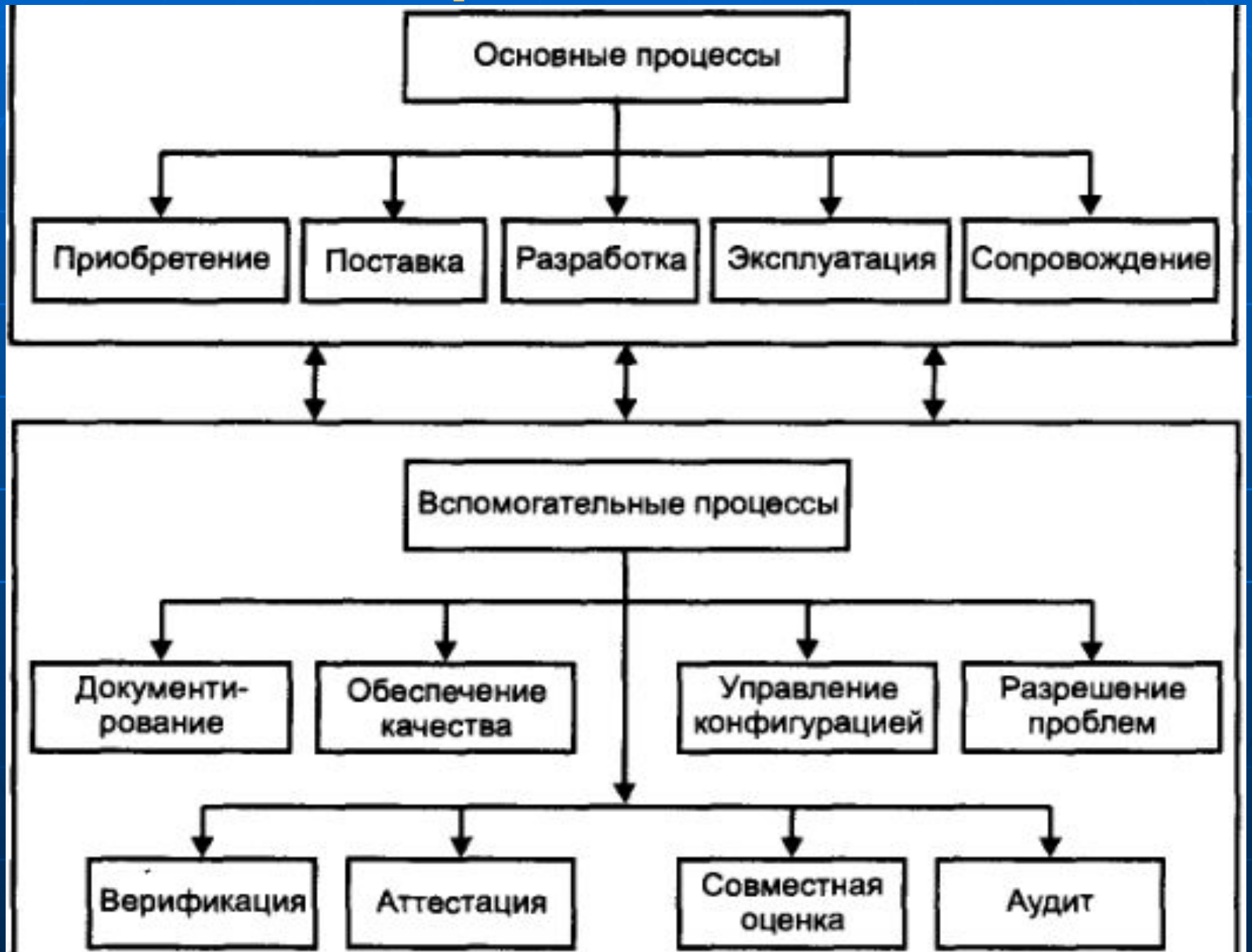


Схема процессов ЖЦ ИС



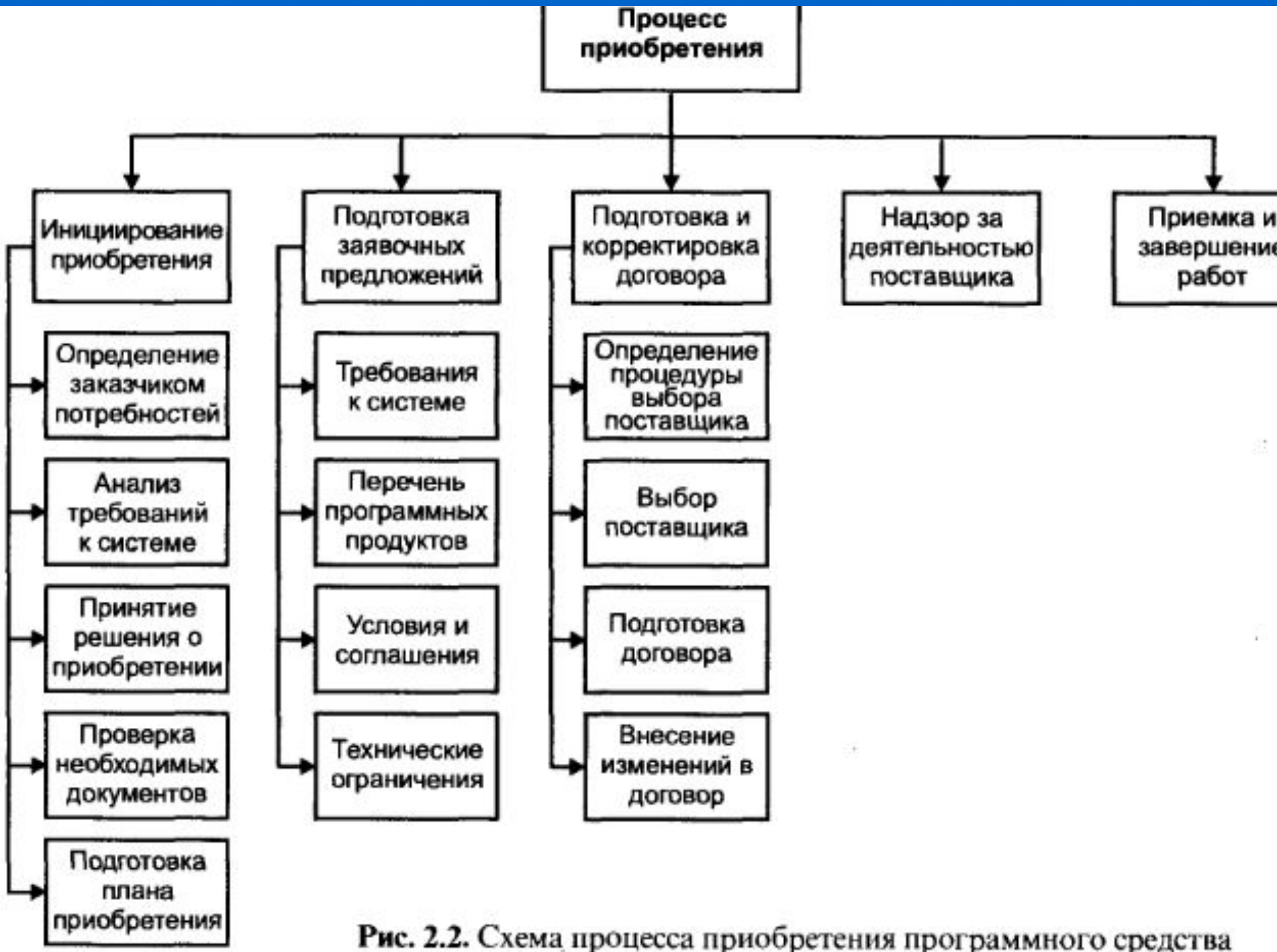


Рис. 2.2. Схема процесса приобретения программного средства

Таблица 2.1. Содержание основных процессов ЖЦ ПО ИС (ISO/IEC 12207)

Процесс (исполнитель процесса)	Действия	Вход	Результат
Приобретение (заказчик)	• Инициирование • Подготовка заявочных предложений • Подготовка договора • Контроль деятельности поставщика • Приемка ИС	• Решение о начале работ по внедрению ИС • Результаты обследования деятельности заказчика • Результаты анализа рынка ИС/ тендера • План поставки/ разработки • Комплексный тест ИС	• Технико-экономическое обоснование внедрения ИС • Техническое задание на ИС • Договор на поставку/ разработку • Акты приемки этапов работы • Акт приемно-сдаточных испытаний

Рис. 2.3. Схема процесса поставки

Таблица 2.1. Содержание основных процессов ЖЦ ПО ИС (ISO/IEC 12207)

Процесс (исполнитель процесса)	Действия	Вход	Результат
Поставка (разработчик ИС)	• Инициирование • Ответ на заявочные предложения • Подготовка договора • Планирование исполнения • Поставка ИС	• Техническое задание на ИС • Решение руководства об участии в разработке • Результаты тендера • Техническое задание на ИС • План управления проектом • Разработанная ИС и документация	• Решение об участии в разработке • Коммерческие предложения/ конкурсная заявка • Договор на поставку/ разработку • План управления проектом • Реализация/ корректировка • Акт приемно-сдаточных испытаний

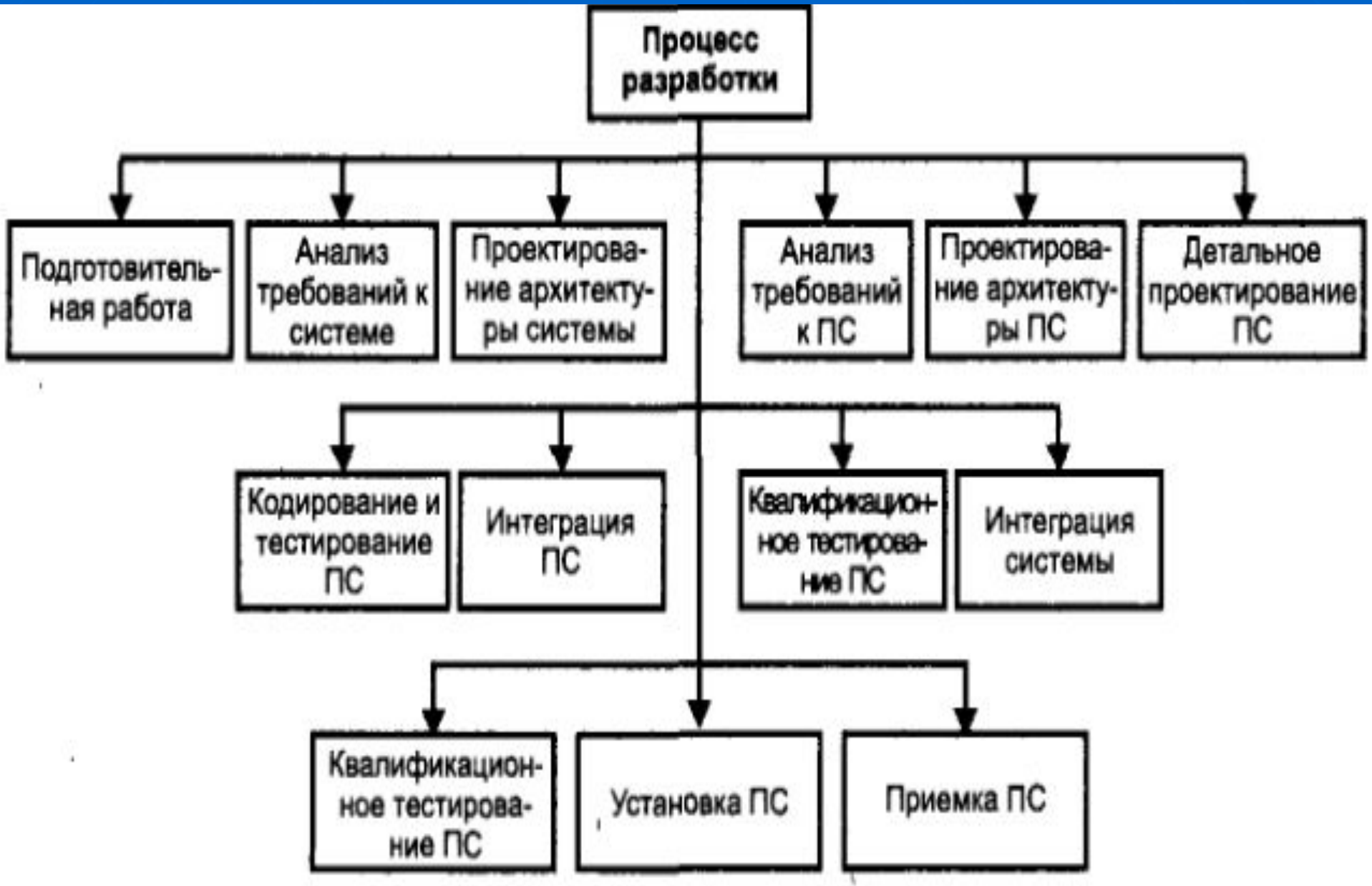


Рис. 2.4. Схема процесса разработки

Разработка (разработчик ИС)	• Подготовка • Анализ требований к ИС • Проектирование архитектуры ИС • Разработка требований к ПО • Проектирование архитектуры ПО • Детальное проектирование ПО • Кодирование и тестирование ПО • Интеграция ПО и квалификационное тестирование ПО • Интеграция ИС и квалификационное тестирование ИС	• Техническое задание на ИС • Техническое задание на ИС, модель ЖЦ • Техническое задание на ИС • Подсистемы ИС • Спецификации требования к компонентам ПО • Архитектура ПО • Материалы детального проектирования ПО • План интеграции ПО, тесты • Архитектура ИС, ПО, документация на ИС, тесты	• Используемая модель ЖЦ, стандарты разработки • План работ • Состав подсистем, компоненты оборудования • Спецификации требования к компонентам ПО • Состав компонентов ПО, интерфейсы с БД, план интеграции ПО • Проект БД, спецификации интерфейсов между компонентами ПО, требования к тестам • Тексты модулей ПО, акты автономного тестирования • Оценка соответствия комплекса ПО требованиям ТЗ • Оценка соответствия ПО, БД, технического комплекса и комплекта документации требованиям ТЗ

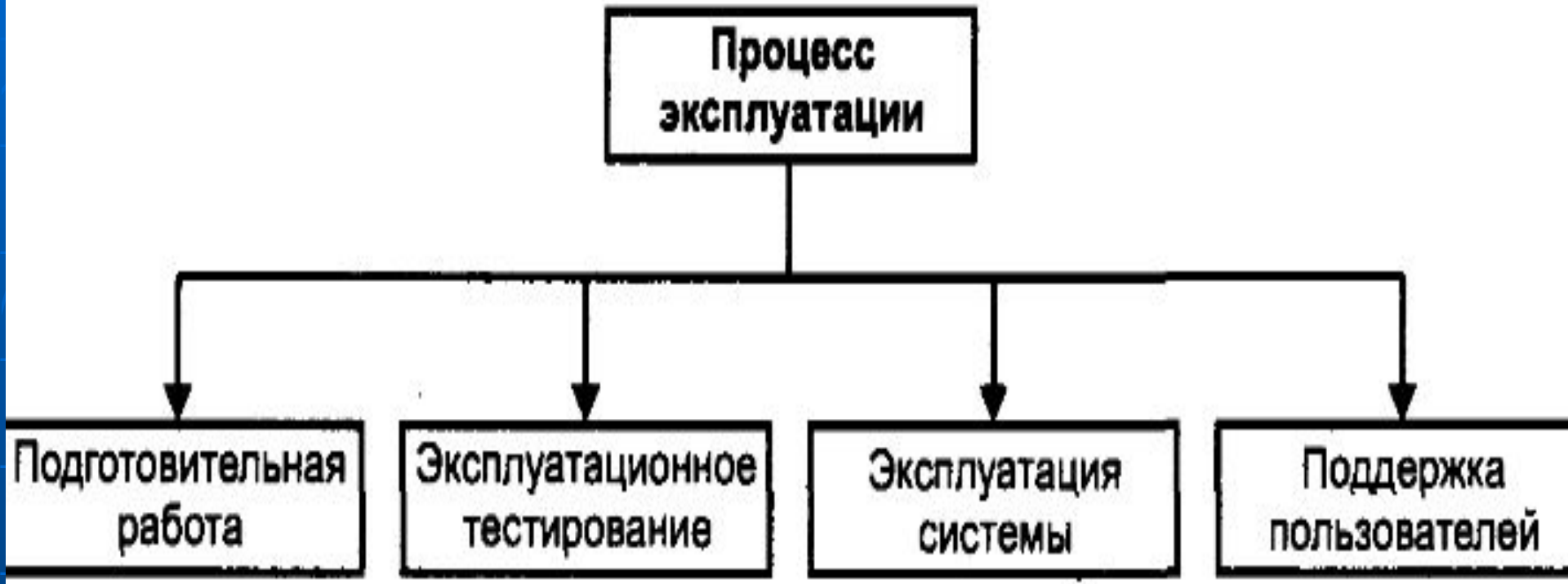


Рис. 2.5. Схема процесса эксплуатации

Рис. 2.6. Схема процесса сопровождения

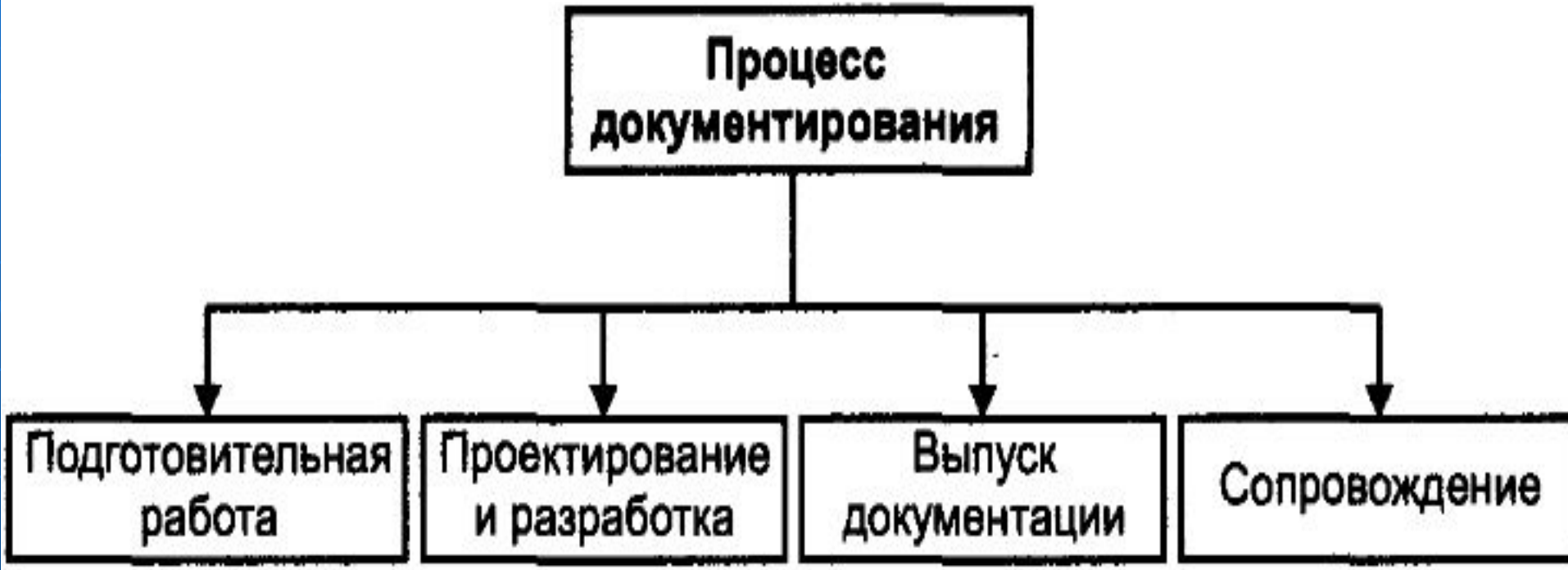


Рис. 2.7. Схема процесса документирования

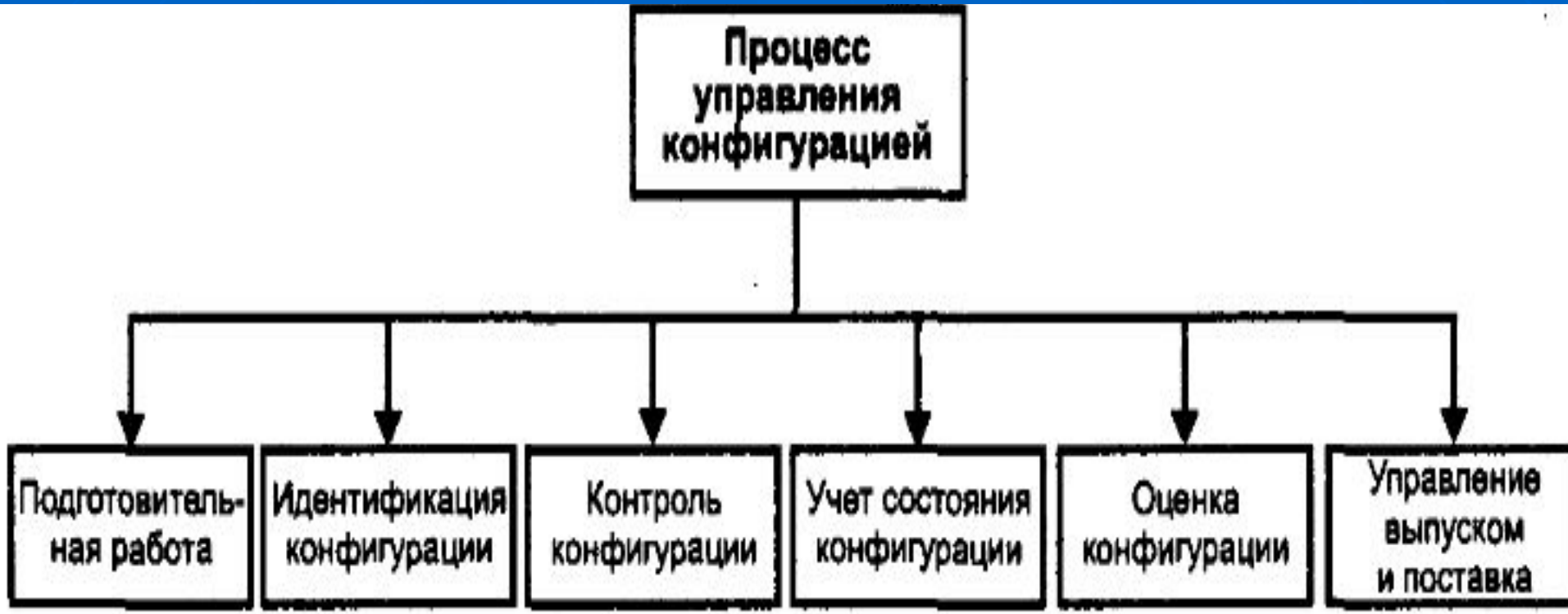


Рис. 2.8. Схема процесса управления конфигурацией

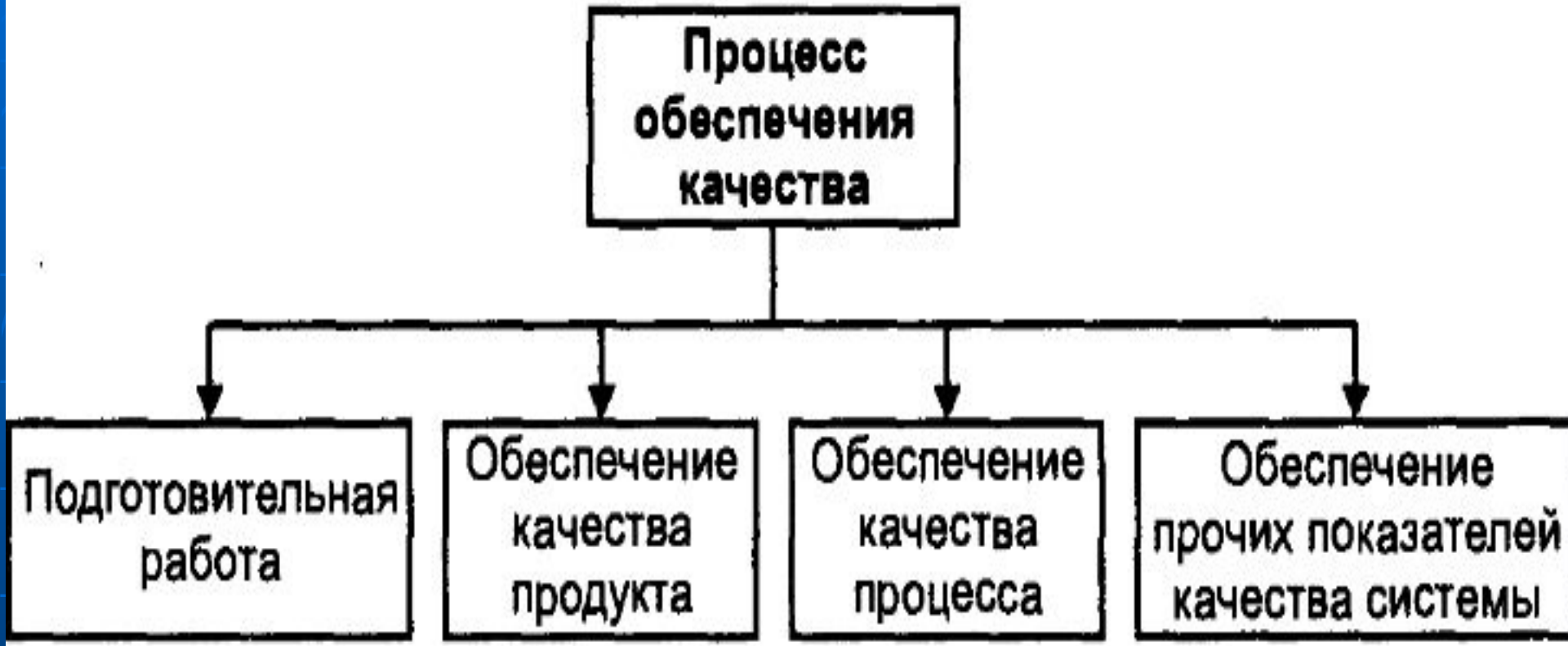


Рис. 2.9. Схема процесса обеспечения качества

Общее представление о качестве ПС стандартом **ISO 9126:1-4:2002** рекомендуется описывать тремя взаимодействующими и взаимозависимыми ***метриками характеристик качества***, отражающими:

Метрики характеристик качества отражают:

- ***внутреннее качество***, проявляющееся в процессе разработки и других промежуточных этапов ЖЦ ПС;
- ***внешнее качество***, заданное требованиями заказчика в спецификациях и отражающееся характеристиками конечного продукта;
- ***качество при использовании*** в процессе нормальной эксплуатации и результативностью достижения потребностей пользователей с учетом затрат ресурсов

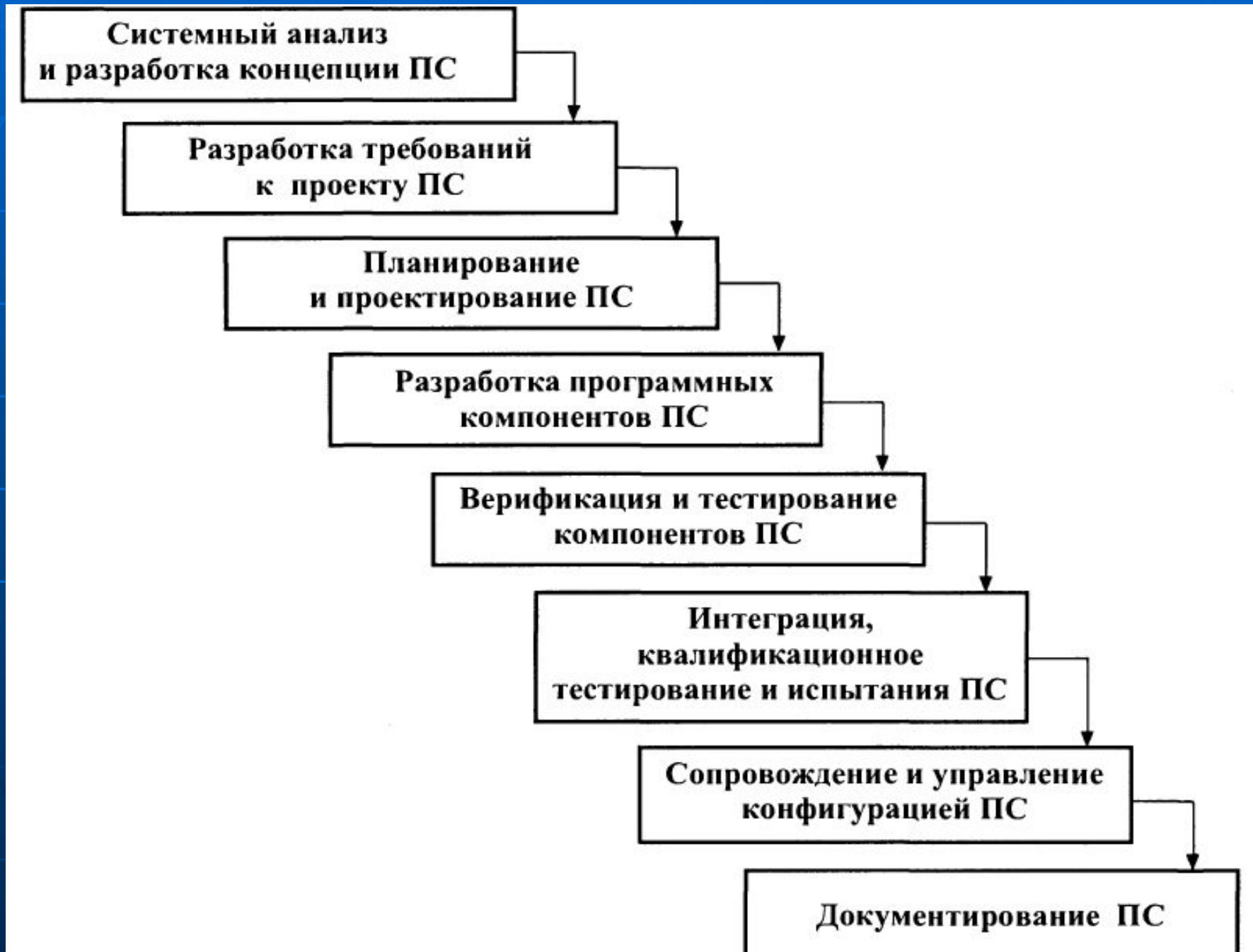
Рис. 2.18. Схема процесса обучения

Модели ЖЦ ИС

В настоящее время известны и используются ***3 модели ЖЦ***:

Каскадная модель - последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе.

Каскадная модель – более детальный пример



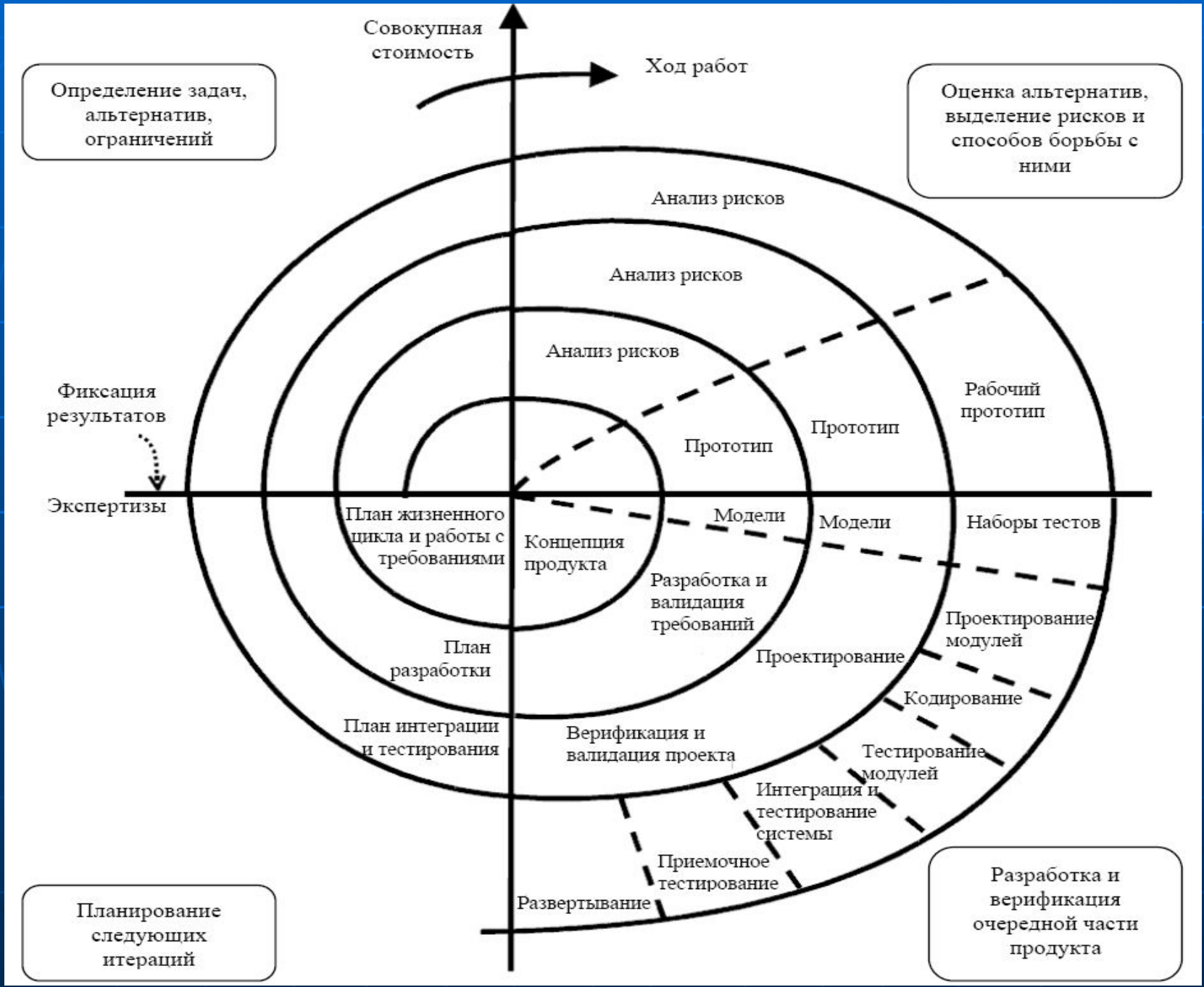
Поэтапная модель с промежуточным контролем. Разработка ИС ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки.

Спиральная модель. На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка.

Особое внимание - начальным этапам - ***анализу*** и ***проектированию***, где реализуемость технических решений проверяется и обосновывается путем создания прототипов (макетов).

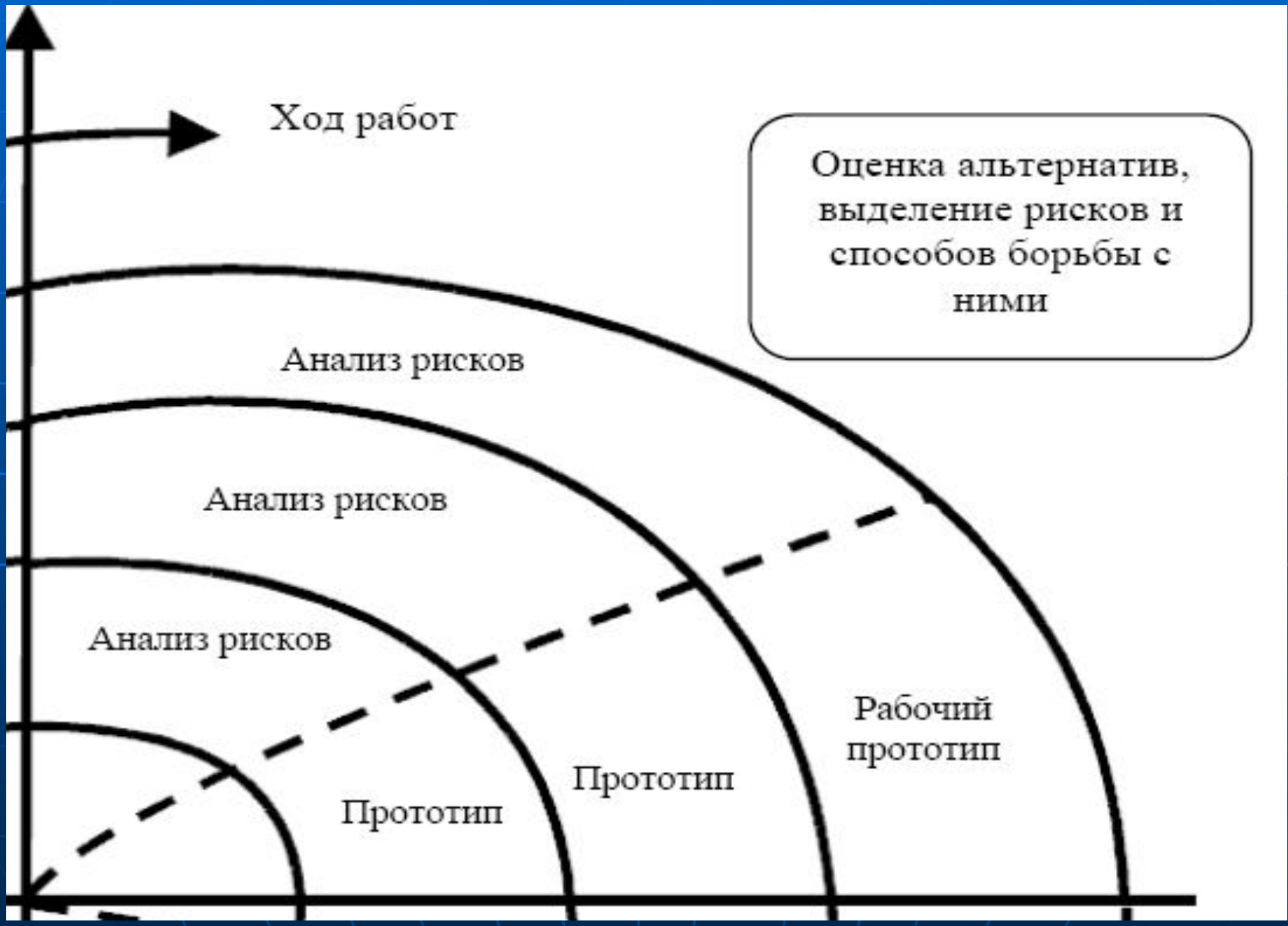
Спиральная модель

Спиральная модель – более подробно



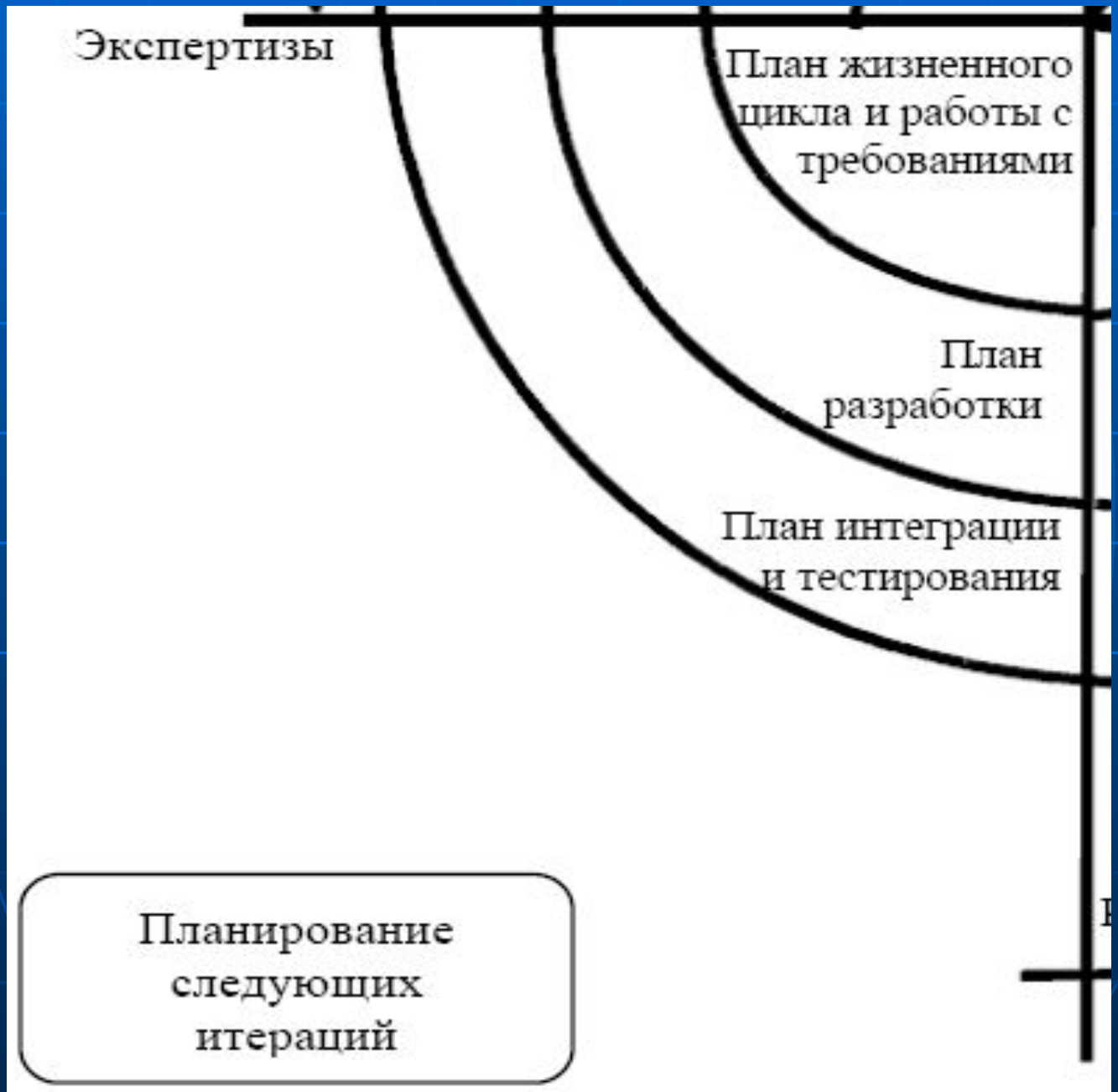
Спиральная модель – более подробно

Спиральная модель – более подробно



Спиральная модель – более подробно

Спиральная модель – более подробно



На практике наибольшее распространение – ***две основные модели ЖЦ***:

каскадная модель (характерна для периода 1970-1985 гг.);

спиральная модель (характерна для периода после 1986 г.).

В ранних проектах относительно простых ИС каждое приложение было единым, функционально и информационно независимым блоком. Для разработки таких приложений эффективен каскадный способ. Каждый этап завершался после полного выполнения и документального оформления всех предусмотренных работ.

Преимущества каскадного подхода

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям ***полноты*** и ***согласованности***;
- выполняемые в логической последовательности этапы работ позволяют ***планировать сроки*** завершения всех работ и соответствующие затраты.

Недостаток каскадного подхода

реальное создание ИС никогда полностью не укладывается в жесткую схему, постоянно возникает ***потребность в:***

возврате к предыдущим этапам

уточнении или пересмотре ранее принятых решений.

Преимущества спиральной модели

На этапах анализа и проектирования реализуемость и правильность технических решений проверяется путем создания прототипов.

Каждый виток спирали - ***создание работоспособного фрагмента или версии ИС***.

Это позволяет уточнить требования, цели и характеристики проекта, определить качество разработки, спланировать работы следующего витка спирали **=>** последовательно уточняются детали проекта и выбирается обоснованный вариант, который удовлетворяет требованиям заказчика и доводится до реализации.

Преимущества спиральной модели

Итеративная разработка отражает реальный спиральный цикл создания сложных систем.

Она позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем этапе и решить главную задачу –

быстрее показать пользователям ИС работоспособный продукт,

тем самым активизируя процесс уточнения и дополнения требований.

Основная проблема спирального цикла -

определение момента перехода на следующий этап. Для ее решения вводятся временные ограничения на каждый из этапов ЖЦ, и переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена.

Планирование производится на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Проблемы использования моделей ЖЦ ИС

Несмотря на рекомендации экспертов в области разработки ИС, многие компании продолжают использовать ***каскадную модель*** вместо какого-либо варианта ***итерационной модели***.

Основные причины, по которым каскадная модель сохраняет свою популярность:

Привычка - многие ИТ-специалисты получали образование в то время, когда изучалась только каскадная модель, поэтому она используется ими и в наши дни.

Иллюзия снижения рисков заказчика и исполнителя

Каскадная модель - разработка на каждом этапе:

технического задания,
технического проекта,
ПО,
пользовательской документации.

Эта документация определяет: требования к ПО следующего этапа, обязанности сторон, объем работ и сроки, при этом окончательная оценка сроков и стоимости проекта производится на начальных этапах, после завершения обследования.

Иллюзия снижения рисков заказчика и исполнителя - *Каскадная модель*

Если требования к ИС ***меняются в ходе реализации проекта*** (обычная ситуация), то использование каскадной модели создает лишь иллюзию определенности и на деле увеличивает риски, уменьшая лишь ответственность участников проекта.

Проблемы внедрения при использовании итерационной модели

Спиральная модель не может применяться там, где невозможно использование / тестирование продукта, обладающего ***неполной функциональностью***
(военные разработки,
атомная энергетика и т.д.).

Проблемы внедрения при использовании итерационной модели

Поэтапное итерационное внедрение ИС для бизнеса возможно, но сопряжено с дополнительными организационными сложностями:

- ***перенос данных***,
- ***интеграция систем***,
- ***изменение бизнес-процессов***,
- ***обучение пользователей***.

Проблемы внедрения при использовании итерационной модели

Трудозатраты при поэтапном итерационном внедрении оказываются значительно выше, а управление проектом - сложнее.

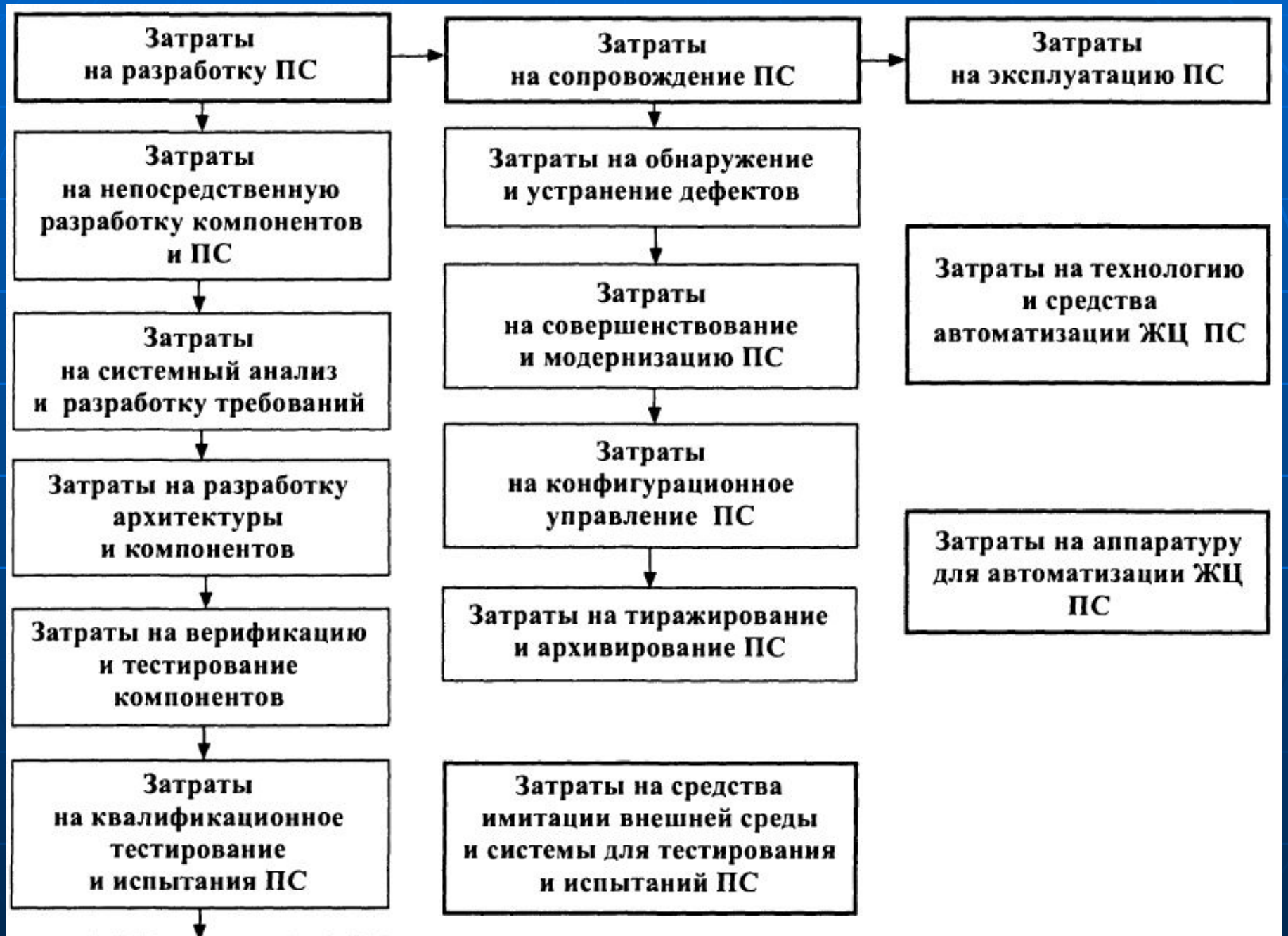
В этих случаях заказчики выбирают каскадную модель, чтобы "внедрять систему один раз".

Затраты ресурсов в ЖЦ ПС -

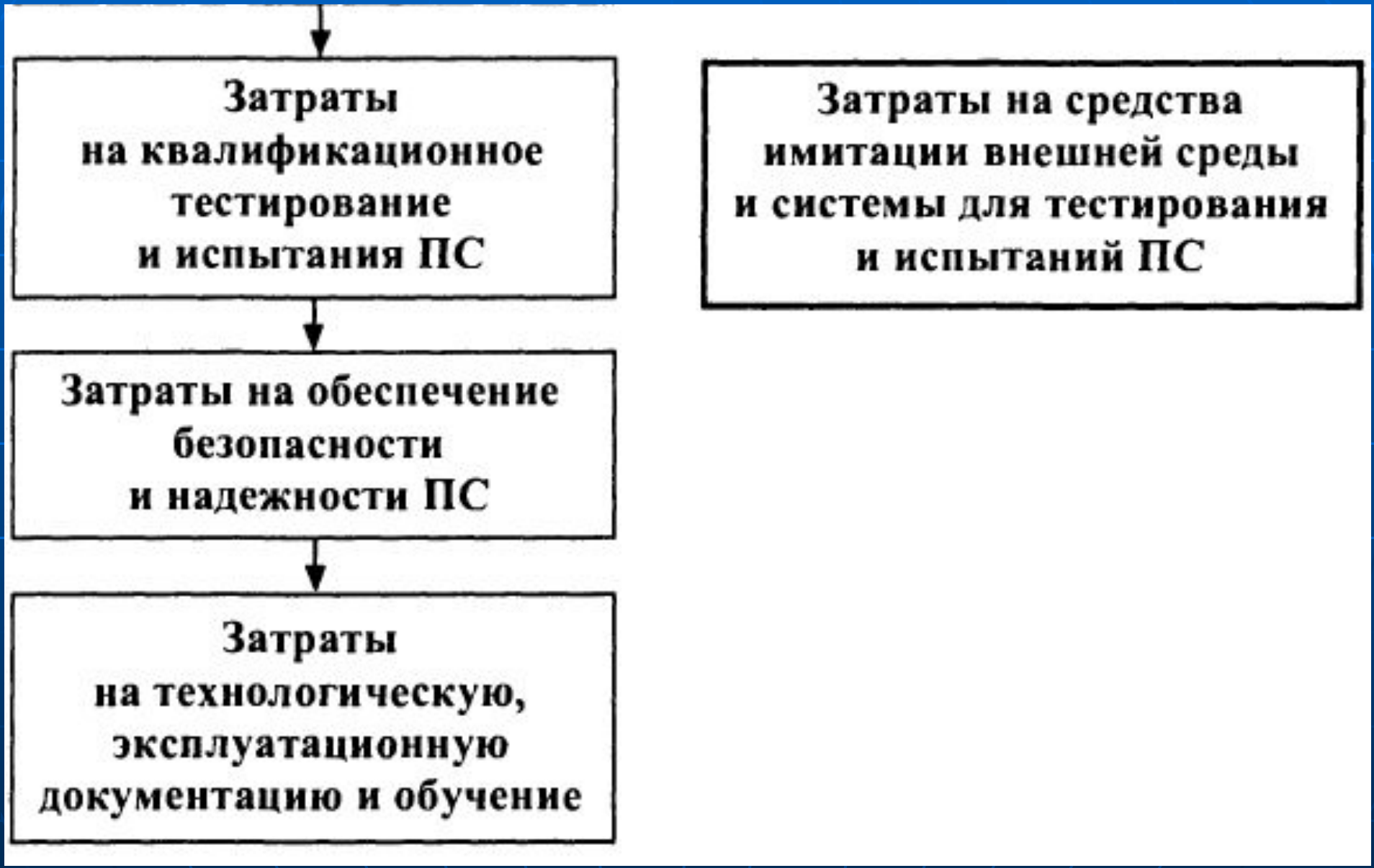
определяются не только этапами разработки, но и этапами эксплуатации и сопровождения.

Затраты на этих этапах могут значительно превышать затраты при разработке и характеризуются своими закономерностями. Эффективность разработки ПС невозможно определять без учета эффективности последующей эксплуатации, а для долго модифицируемых программ - без оценки эффективности их сопровождения.

Затраты ресурсов в ЖЦ ПС



Затраты ресурсов в ЖЦ ПС



Основы RAD-технологий

RAD (от англ. *rapid application development* — быстрая разработка приложений) — концепция создания средств разработки программных продуктов, уделяющая особое внимание быстроте и удобству программирования, созданию технологического процесса, позволяющего программисту максимально быстро создавать компьютерные программы. С конца XX века RAD получила широкое распространение и одобрение. Концепцию RAD также часто связывают с концепцией визуального программирования.

Основы RAD-технологий

Основные принципы RAD [прави

- Инструментарий должен быть нацелен на минимизацию времени разработки.
- Создание *прототипа* для уточнения требований заказчика.
- Цикличность разработки: каждая новая версия продукта основывается на оценке результата работы предыдущей версии заказчиком.
- Минимизация времени разработки версии, за счёт переноса уже готовых модулей и добавления функциональности в новую версию.
- Команда разработчиков должна тесно сотрудничать, каждый участник должен быть готов выполнять несколько обязанностей.
- Управление проектом должно минимизировать длительность цикла разработки.

Основы RAD-технологий

История [править]

Концепция RAD стала ответом на неуклюжие методы разработки программ 1970-х и начала 1980-х годов, такие как «модель водопада» (англ. *Waterfall model*). Эти методы предусматривали настолько медленный процесс создания программы, что зачастую даже требования к программе успевали измениться до окончания разработки. Основателем RAD считается сотрудник IBM Джеймс Мартин, который в 1980-х годах сформулировал основные принципы RAD, основываясь на идеях Барри Бойма и Скотта Шульца. А в 1991 году Мартин опубликовал известную книгу, в которой детально изложил концепцию RAD и возможности её применения. В настоящее время RAD становится общепринятой схемой для создания средств разработки программных продуктов. Именно средства разработки, основанные на RAD, имеют наибольшую популярность среди программистов.

Основы RAD-технологий

Среды разработки, использующие принципы RAD

- Borland Delphi
- Borland C++ Builder
- IBM Lotus Domino Designer
- Microsoft Visual Studio
- Macromedia Flash
- Macromedia Authorware
- Macromedia Director
- Omnis Studio
- Visual DataFlex
- IntraWeb
- Miracle
- MonoDevelop
- WxDev-C++
- Lazarus
- QDevelop (в связке с Qt-Designer)
- Code::Blocks
- wxFormBuilder

Основы RAD-технологий

Бы́строе прототипи́рование, сокр. **БП** — технология быстрого «макетирования», быстрого создания опытных образцов или работающей модели системы для демонстрации заказчику или проверки возможности реализации. Прототип позже уточняется для получения конечного продукта.

Термин используется как в информационных технологиях для обозначения процесса быстрой разработки программного обеспечения (см. RAD), так и в технологиях, связанных с изготовлением физических прототипов деталей.

Стандарты, регламентирующие ЖЦ ПО и процессы разработки

Значительный вклад в теорию проектирования и разработки ИС - компания IBM, предложила еще в середине 1970-х годов методологию ***BSP*** (Business System Planning - ***методология организационного планирования***).

Метод структурирования информации с использованием матриц пересечения бизнес-процессов, функциональных подразделений, функций систем обработки данных (ИС), информационных объектов, документов и баз данных, предложенный в BSP, используется до сих пор в CASE-системах.

Основные шаги процесса BSP, их последовательность:

- ***получить поддержку высшего руководства***,
- ***определить процессы предприятия***,
- ***определить классы данных***,
- ***провести интервью***,
- ***обработать и организовать данные интервью,***

можно встретить практически во всех формальных методиках, а также в проектах, реализуемых на практике.

ПРИМЕРЫ ПЛАНИРОВАНИЯ РАБОТ ПО РАЗРАБОТКЕ ПРОГРАММНЫХ СРЕДСТВ

Постановка задачи: нормально работающая, полностью загруженная компания – разработчик ПО получила заказ, от которого по разным причинам невозможно отказаться.

Каким может быть план этих работ (часть ЖЦ) ???

ПРИМЕРЫ ПЛАНИРОВАНИЯ РАБОТ ПО РАЗРАБОТКЕ ПРОГРАММНЫХ СРЕДСТВ

1) начало работы;
2) коллектив сформирован, рабочие места подготовлены;
3) проектирование завершено;
4) программирование завершено;
5) комплексная отладка завершена;
6) оборудование закуплено;

ПРИМЕРЫ ПЛАНИРОВАНИЯ РАБОТ ПО РАЗРАБОТКЕ ПРОГРАММНЫХ СРЕДСТВ

7) группа технических писателей получила описание проекта и пояснения от проектировщиков;

8) то же для ПО, разработка проектной документации завершена;

9) группа технических писателей получила всю необходимую информацию об интерфейсах с пользователем, разработка программной документации завершена;

ПРИМЕРЫ ПЛАНИРОВАНИЯ РАБОТ ПО РАЗРАБОТКЕ ПРОГРАММНЫХ СРЕДСТВ

10) группа оценки качества (Quality Assurance – QA) разработала тесты;

11) группа QA оценила проект положительно;

12) группа QA завершила автономное тестирование;

13) группа QA завершила комплексное тестирование, получила всю документацию и действующий вариант системы;

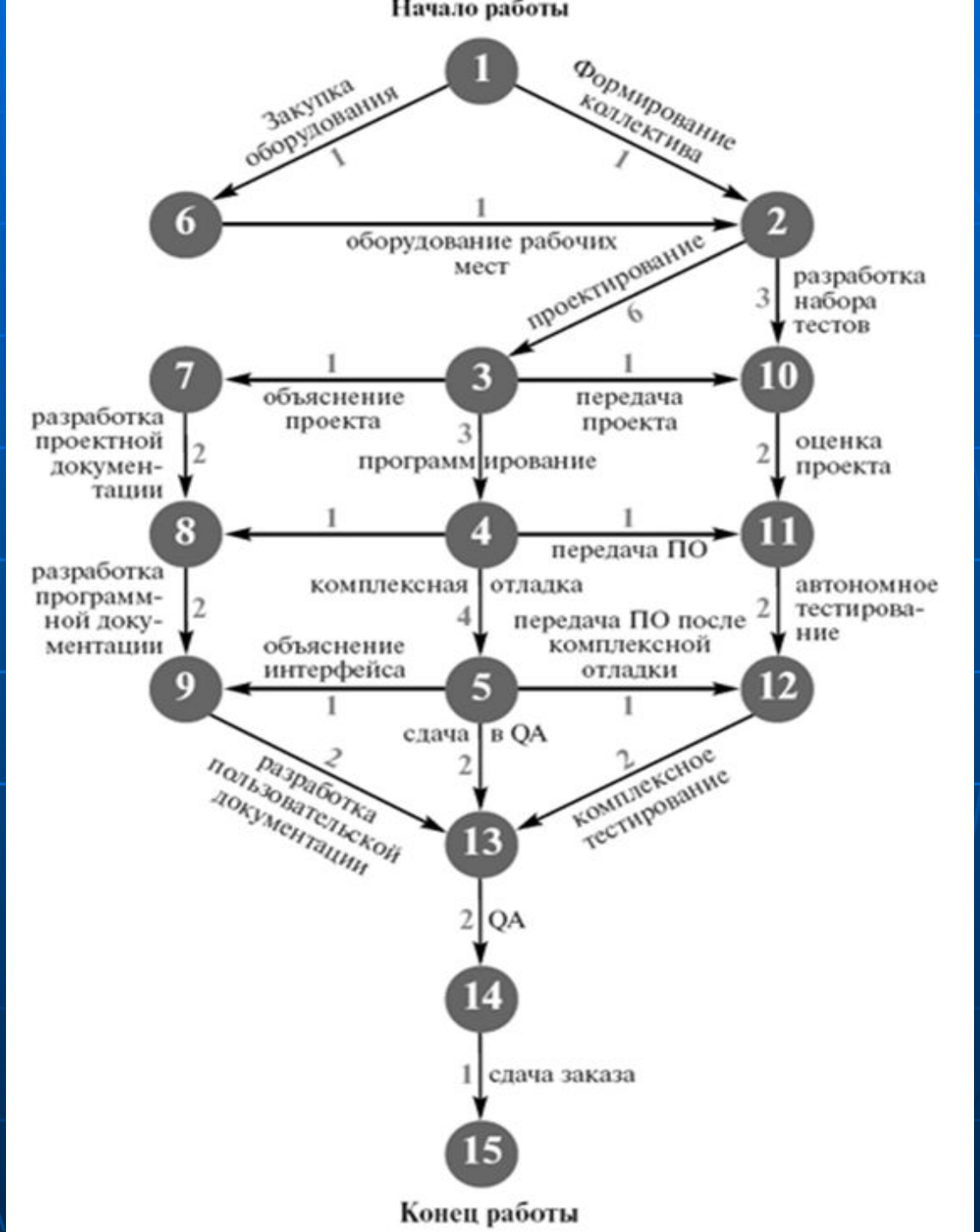
ПРИМЕРЫ ПЛАНИРОВАНИЯ РАБОТ ПО РАЗРАБОТКЕ ПРОГРАММНЫХ СРЕДСТВ

14) проверка качества завершена;

15) конец работы

(конечно, это не конец, будет еще сопровождение, но пример-то надо закончить ☺).

Сетевой график – ориентированный граф с двумя выделенными вершинами – *начало* и *конец работы*



Сетевой график

Вершины графа – ***события*** (пункты плана), а ребра – ***работы***. События выражаются глаголами совершенного вида: "тексты программ переданы в БД исходников", "все тесты пропущены", но уж никак не "выполняется прогон тестов".

Ребра *нагружаются оценками длительности работ*, например, в днях или неделях.

Сетевой график – удобный инструмент:

1) видна зависимость работ друг от друга,
2) можно вычислить длительность всей работы,
3) пользуясь результатами этих расчетов, можно оптимизировать длительность и затраты на всю работу.

Сетевой график

Длительность вычисляется следующим образом. Суммируем длительности работ по всем возможным путям в графе. Тот путь от начала к концу, который является самым длинным, объявляется ***критическим***, потому что задержка любой работы, лежащей на этом пути, приводит к задержке всей работы в целом.

Критических путей (с одинаковой длительностью) может быть несколько.

Сетевой график – детали

Сетевой график – детали

Сетевой график – замечания по примеру

Критические - пути 1-6-2-3-4-5-9-13-14-15 и 1-6-2-3-4-5-12-13-14-15, т.е. вся работа не выполняется быстрее 21 недели.

Для оптимальной загрузки коллектива лучше, чтобы все пути в графе от начала к концу имели близкую длительность - уменьшается длина критического пути. Например, есть соблазн заставить группу QA проводить даже начальное тестирование, уменьшив нагрузку на программистов, работа которых находится на критическом пути. Но тогда очень трудно определить границы ответственности, программисты начинают «халтурить» и в результате сроки даже удлиняются.

Сетевой график – замечания по примеру

В реальных проектах, где работ много, возможно перераспределением работ улучшать сетевой график.

Неудачно спланированы работы между событиями 1, 2, 6. Коллектив сформирован за 1 неделю, а рабочие места еще не готовы. Группа технических писателей начинает работать на 6 недель позже проектировщиков, а группа QA имеет трехнедельный перерыв перед завершением проектирования и т.д.

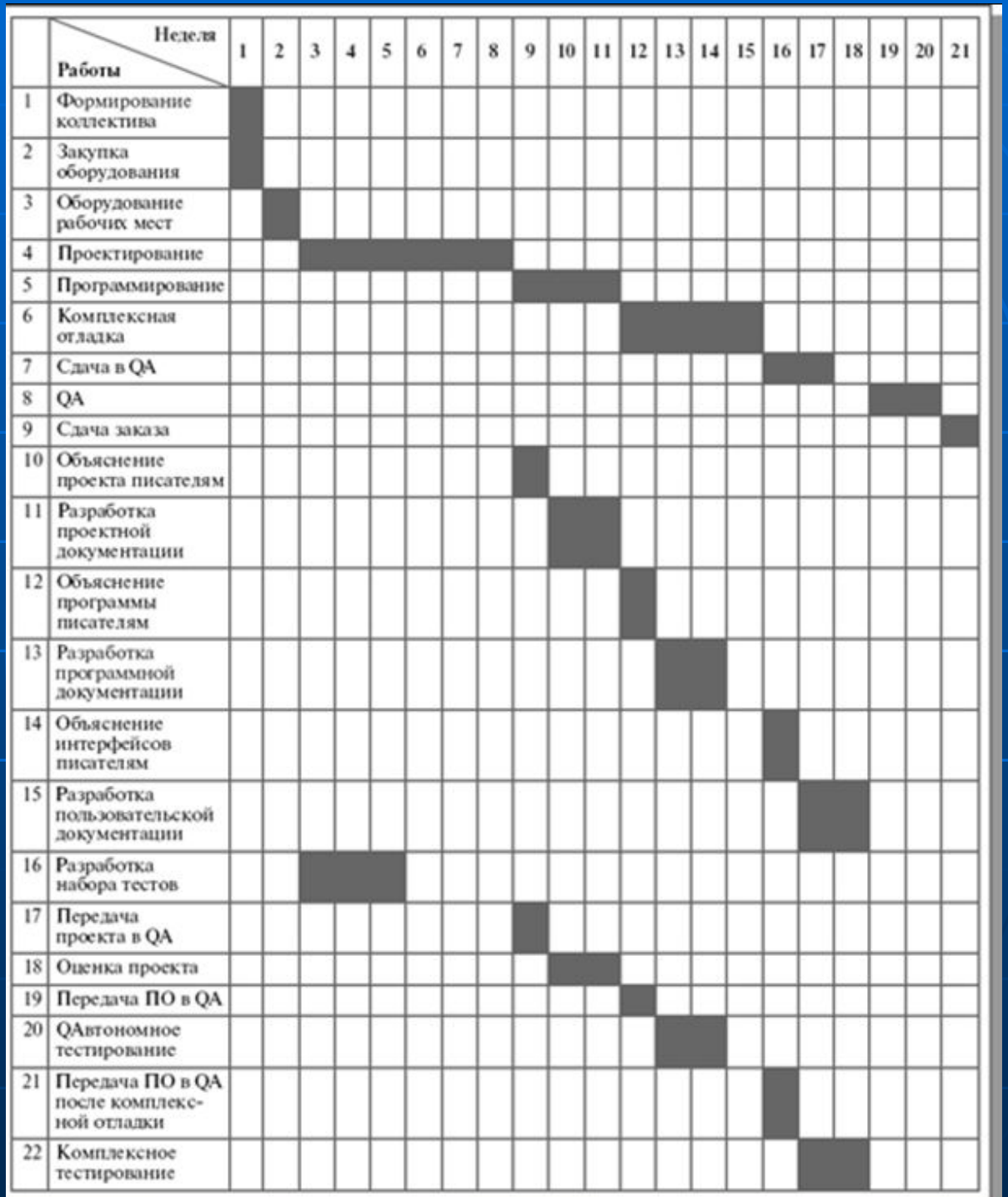
Эти проблемы трудны, решаются по-разному, например, можно использовать одну и ту же группу QA или технических писателей для нескольких групп разработчиков.

Еще одной популярной формой графического представления плана работ (реализации ЖЦ) является ***диаграмма Ганта*** (Gantt).

Диаграмма Ганта - прямоугольник: слева направо равномерно отсчитываются периоды времени (недели, месяцы), сверху вниз перечисляются работы, причем каждая работа - отрезок, начало и конец которого размещаются в соответствующем периоде.

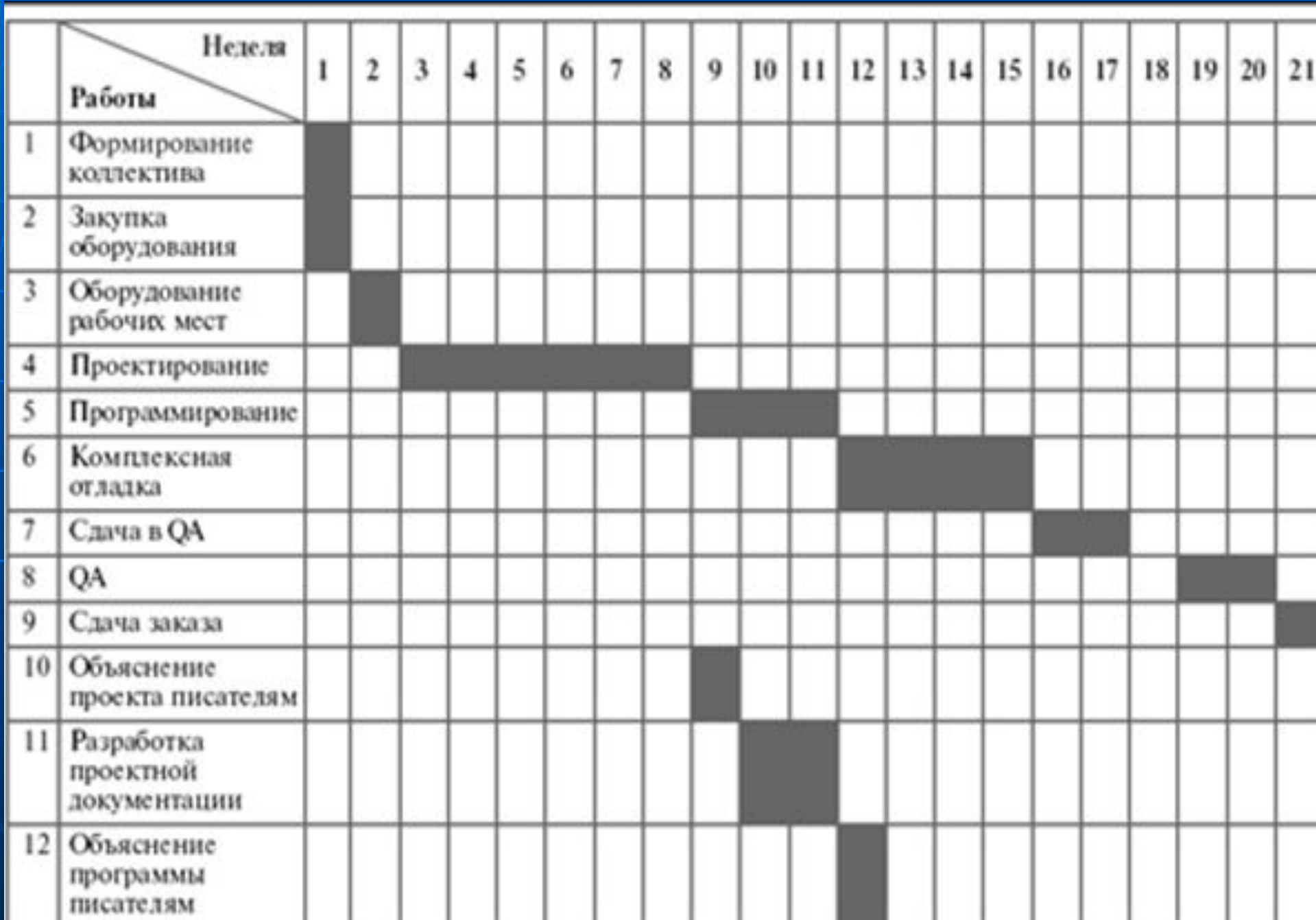
Для сравнения с сетевым графиком нарисуем диаграмму Ганта для того же примера:

Диаграмма Ганта



	Неделя / Работы	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	Формирование коллектива	■																				
2	Закупка оборудования	■																				
3	Оборудование рабочих мест		■																			
4	Проектирование			■	■	■	■	■	■													
5	Программирование									■	■	■										
6	Комплексная отладка												■	■	■	■						
7	Сдача в QA																■	■				
8	QA																			■	■	
9	Сдача заказа																					■
10	Объяснение проекта писателям									■												
11	Разработка проектной документации										■	■										
12	Объяснение программы писателям												■									
13	Разработка программной документации													■	■							
14	Объяснение интерфейсов писателям																■					
15	Разработка пользовательской документации																	■	■			
16	Разработка набора тестов			■	■	■																
17	Передача проекта в QA									■												
18	Оценка проекта										■	■										
19	Передача ПО в QA												■									
20	QAвтономное тестирование													■	■							
21	Передача ПО в QA после комплекс-ной отладки																■					
22	Комплексное тестирование																	■	■			

Диаграмма Ганта



	Неделя / Работы	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	Формирование коллектива																					
2	Закупка оборудования																					
3	Оборудование рабочих мест																					
4	Проектирование																					
5	Программирование																					
6	Комплексная отладка																					
7	Сдача в QA																					
8	QA																					
9	Сдача заказа																					
10	Объяснение проекта писателям																					
11	Разработка проектной документации																					
12	Объяснение программы писателям																					

Диаграмма Ганта

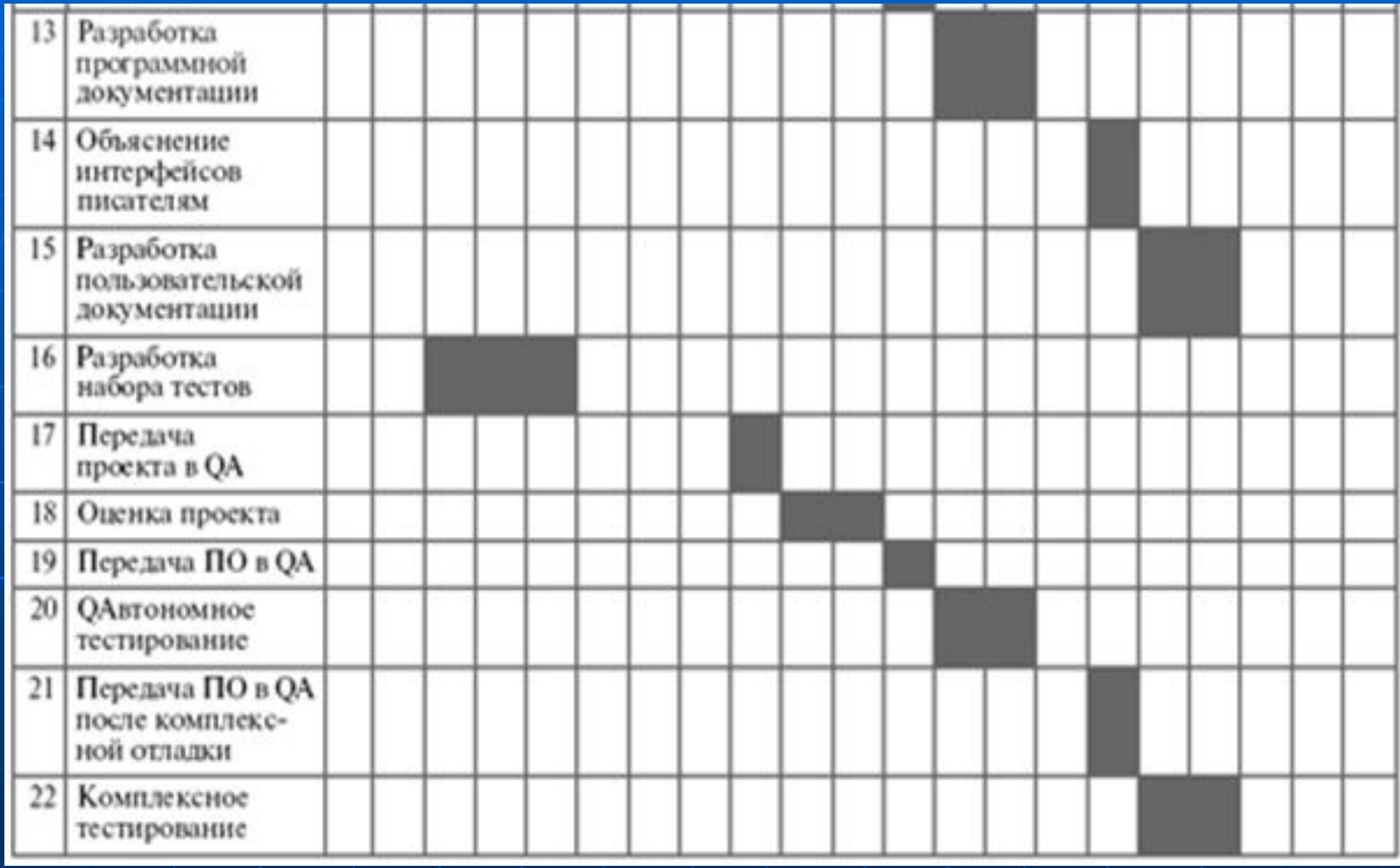


Диаграмма Ганта

Если в сетевом графике видны зависимости работ друг от друга (работа 12-13 может начаться только после завершения работ 5-12 и 11-12), то в диаграмме Ганта основной упор делается на то, что происходит в каждую конкретную неделю (видно, что группа QA имеет перерыв в работе в 2 недели между работами 11-12 (автономное тестирование) и 12-13 (комплексное тестирование).

Удобно для руководства: в конце недели можно провести вертикальную линию и сразу увидеть, какие работы велись, что должно закончиться, а что начаться.

Технические менеджеры предпочитают сетевые графики, так как им важнее информация о том, что от чего зависит, да и пересчитывать критические пути им приходится практически каждую неделю.