***Слайд 1***

**Лекция 3**

**Требования, цели и спецификации программного обеспечения**

***Слайд 2***

**Учебные вопросы:**

**1. Определение требований программного обеспечения.**

**2. Цели программного обеспечения.**

**3. Постановка целей для программного обеспечения.**

**4. Классификация показателей качества ПО.**

*Первые ошибки прокрадываются в программу тогда, когда определяются требования и цели. Причина большинства этих ошибок — неправильное понимание потребностей пользователя. В дальнейшем возникают и другие ошибки, когда требования и цели транслируются во внешние спецификации.*

***Слайд 3***

## 1. Определение требований программного обеспечения

Программные проекты можно разбить на три группы: ***управляемые пользователем, контролируемые пользователем и независимые от пользователя.***

В случае **управляемого пользователем** проекта требования к программному обеспечению разрабатываются непосредственно организацией-пользователем. Разработчик программного обеспечения является ее субподрядчиком, и требования представляют собой контракт или его часть.

В проекте, **контролируемом пользователем**, требования к программному обеспечению формулируются либо его разработчиком, либо совместными усилиями разработчика и организации-пользователя. В проектах такого типа организация-пользователь имеет право утверждать требования и, как правило, спецификации следующих уровней, в частности — внешние спецификации.

В **не зависимом от пользователя** проекте вся ответственность за определение требований ложится на разработчика программного обеспечения. Большинство проектов, связанных с разработкой общедоступных, поставляемых на рынок программ, относится к этой группе. Однако, организация-пользователь может быть привлечена к работе над проектом в какой-либо форме.

***Слайд 4***

**Требования** к программному обеспечению дают возможность пользователю сформулировать свои потребности в отношении конкретного программного продукта. *С этой точки зрения наиболее привлекателен проект, контролируемый пользователем.*

*Серьезное участие потенциального пользователя окончательного продукта желательно в определении требований. Но, безусловно, желательно и участие в этом процессе организации-разработчика программного обеспечения. Если в управляемом пользователем проекте разработчик программного обеспечения* не *привлечен к определению требований, это увеличивает его шансы неправильно понять или неправильно интерпретировать требования.* И управляемые пользователем, и не зависимые от пользователя проекты должны планироваться таким образом, чтобы обеспечить участие обеих сторон.

*Требования к средней или крупной системе должны разрабатываться небольшой группой. Один участник этой группы должен быть основным представителем организации-пользователя, облеченным достаточными полномочиями, чтобы принимать решения. Отметим, однако, что он обычно не является настоящим пользователем системы. Поэтому вторым членом группы должен быть человек, который действительно будет пользоваться системой. Например, при разработке требований к системе резервирования авиабилетов им должен быть опытный кассир.* *Организация-разработчик также должна быть представлена в этой группе. Одним из ее представителей должен быть человек, который в конечном счете будет играть главную роль в процессе внешнего проектирования. Другим членом группы должен быть тот, кто будет играть ключевую роль в одном из процессов внутреннего проектирования. Что касается надежности, здесь ставится цель обеспечить максимально возможную аккуратность и точность в определении требований пользователя, чтобы организация — разработчик программного обеспечения могла транслировать эти требования в проект с минимальным числом ошибок.*

Для описания требований могут применяться автоматизированные системы, «ручные» полуформальные методы.

Примером «ручных» полуформальных методов описания требований может служить метод HIPO-диаграмм (Hierarchy plus Input- Process-Output: иерархия плюс ввод-обработка-вывод). HIPO-диаграмма строится для каждой основной требуемой функции; в ней дается общая характеристика входных и выходных данных для этой функции и основных шагов обработки. Для отражения иерархической организации таких функций строится оглавление (visual table of contents).

***Слайд 5***

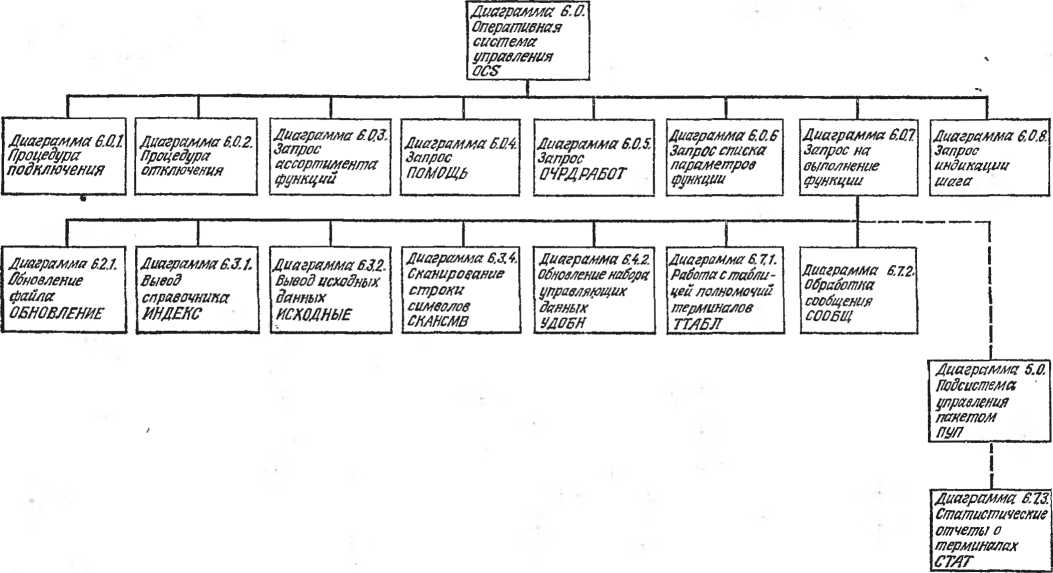


Рис. 1. Оглавление

***Слайд 6***

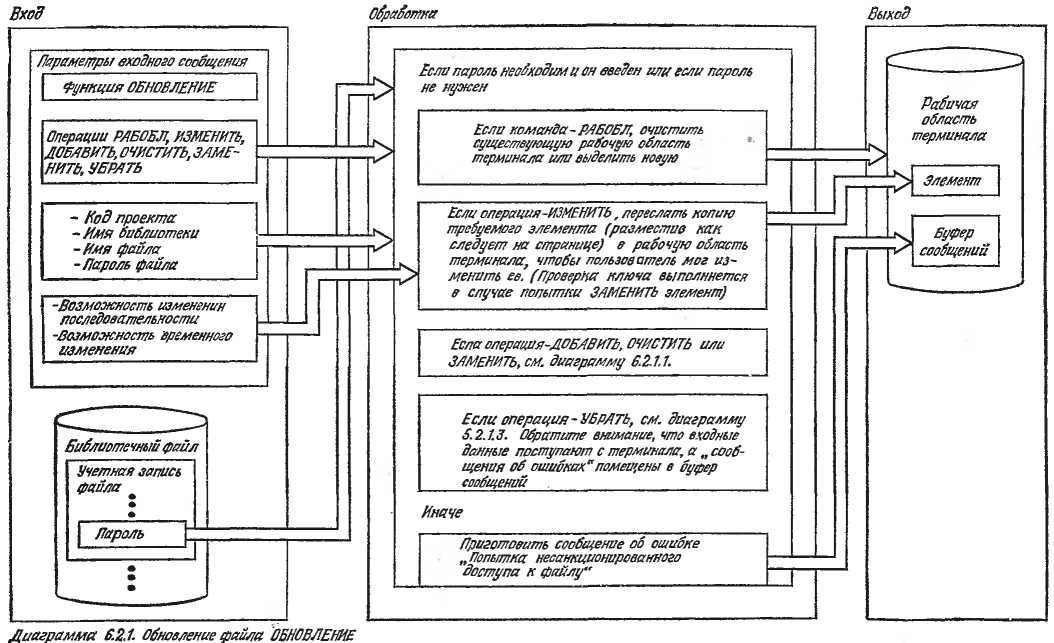


Рис. 2. HIPO-диаграмма.

*На рис. 1 и 2 изображены оглавление и одна из диаграмм для проекта оперативной библиотеки программ. Каждый прямоугольник на рис. 1 представляет крупную функцию, необходимую пользователю. Рис. 2 — это HIPO-диаграмма для одного такого прямоугольника.*

*При использовании этого метода диаграммы обычно готовит организация-разработчик, а затем предъявляет их пользователю для проверки. Диаграммы не содержат никаких предположений о конечной структуре программы и не должны быть написаны на формальном языке программирования. Их задача — дать пользователю наглядное представление о будущем продукте, и убедится в правильном понимании требований.*

*При использовании этого метода важно понимать, что HIPO-диаграммы выражают* ***только функциональные требования****.*

***Другие требования*** *(например, надежность, производительность, стоимость)* ***должны быть описаны отдельно.***

***Подведя итог, о методах проверки правильности требований можно сказать лишь то, что пользователь несет ответственность за проверку требований на полноту и точность, а разработчик — за проверку осуществимости и понятности. В процессе проверки требований желательно установить приоритеты для каждого из них, чтобы помочь разработчику программного обеспечения принимать компромиссные решения на следующих этапах проектирования.***

***Слайд 7***

## 2. Цели программного обеспечения

Второй процесс разработки программного обеспечения — постановка целей. Цели — это конкретные ориентиры для программного продукта. Процесс их постановки — прежде всего процесс принятия компромиссных решений.

При постановке целей распространены следующие ошибки.

1. Цели не формулируются явно.
2. Составляется беглый набросок списка целей, причем жизненно важные цели в него не включаются.
3. Цели формулируются, но таким образом, что они конфликтуют друг с другом. В результате каждый программист, работающий над проектом, разрешает эти конфликты сам, причем все разрешают их по-своему. В проекте с участием ста человек это приведет к тому, что компромиссные решения будут принимать по крайней мере сотней различных путей, что дает совершенно непредсказуемый результат.
4. Необходимость конкретных целей признается, но цели формулируются только для продукта, а не для проекта.

***Слайд 8***

Цели программного обеспечения можно разбить на десять больших групп, одну из которых представляет интересующая нас надежность.

****

***Слайд 9***

**1 группа – Общность.**

**Общность** характеризуется числом, мощью и областью действия предоставляемых пользователю функций.

Требования надежности и общности конфликтуют между собой, поскольку «обобщенные» системы обычно больше и сложнее. Конфликт можно сгладить, избегая обобщений в тех аспектах, которые не очень или вовсе не важны для пользователя. *Каждая функция программного обеспечения должна быть оценена с точки зрения реальной ее выгоды для пользователя и ее влияния на надежность.*

**2 группа - Психологические факторы.**

**Психологические факторы готовой системы** — это мера легкости ее понимания и удобства использования, защищенности от неправильного употребления и, как результат, частоты ошибок пользователя.

*Хотя «гуманизация» взаимодействия с пользователем может увеличить сложность системы и, таким образом, отрицательно влияет на надежность, психологические факторы и надежность, вообще говоря, не находятся в конфликте.* *Например, когда неожиданно меняются условия функционирования операционных систем, в них проявляется множество «затаившихся» ошибок.* *Эти новые условия часто являются результатом неожиданных действий пользователя.* Хороший учет психологических факторов позволяет свести к минимуму возможность таких неожиданных действий, что уменьшает и возможность проявления «затаившихся» ошибок.

**3 группа – Адаптируемость.**

**Адаптируемость** — это мера легкости расширения продукта, например, добавления еще одной потребовавшейся пользователю функции. Требования адаптируемости и надежности согласуются между собой.

**4 группа - Удобство сопровождения.**

**Удобство сопровождения** — это мера затрат времени и средств на исправление ошибки в работающей системе. Это требование согласуется с требованием надежности, поскольку оно тесно связано с адаптируемостью. *Кроме того, такие методы обеспечения надежности, как обнаружение и изоляция ошибок, положительно влияют на удобство сопровождения системы.*

***Слайд 10***

**5 группа – Безопасность.**

**Безопасность** — это мера вероятности того, что один пользователь системы может случайно или намеренно обратиться к данным, являющимся собственностью другого пользователя, разрушить их или помешать работе системы. Например: *Средства защиты (обеспечения безопасности) включают тщательную изоляцию данных и программ разных пользователей друг от друга и от операционной системы.* Поскольку они схожи со средствами изоляции ошибок, они обычно согласуются со стремлением к надежности.

**6 группа – Документация.**

Этот вопрос касается качества и количества публикаций для пользователя. Цели здесь аналогичны тем, которые касаются психологических факторов — они тоже связаны с легкостью понимания и использования продукта. Поэтому цели документирования не конфликтуют со стремлением к надежности.

**7 группа - Стоимость продукта.**

Стоимость программного продукта включает затраты на первоначальную разработку плюс сопровождение продукта. Ранее   
(**в лекции 1**) было рассмотрено соотношение между надежностью и возрастающей стоимостью программного обеспечения; было показано, что рост стоимости во многом вызывается возрастанием числа ошибок. Считается, что стремление к высокой надежности и минимальной стоимости разработки и сопровождения не противоречат друг другу.

**8 группа - Календарный план.**

Еще одна ключевая цель всякого проекта — получить результат к определенному сроку. Выгоды от создания продукта обычно существенно зависят от того, когда он станет доступным.

Многие соотношения, связывающие надежность и стоимость программного обеспечения, имеют место также между надежностью и календарным планом. ***Одна из главных причин срывов графика — ненадежность создаваемого продукта.*** *Например, имеется тенденция чрезвычайно сильно недооценивать время, необходимое для тестирования. Действительно, требующееся на тестирование время тесно связано с числом ошибок, остающихся в программном обеспечении после завершения его проектирования, и это время можно сократить, уменьшая число ошибок, порождаемых в процессах проектирования. Две тенденции: обеспечить надежность и сократить затрачиваемое на разработку время — вполне согласуются между собой при условии, конечно, что сроки не сокращены до такой крайности, когда на надлежащее проектирование просто не остается времени.*

***Слайд 11***

**9 группа – Эффективность.**

Соотношение между эффективностью, или производительностью, и надежностью крайне сложны.

Так, средства обеспечения надежности могут увеличить число команд, которые придется выполнять (например, для обнаружения и регистрации ошибок), увеличить накладные расходы из-за иерархической структуры программы, что может увеличить время выполнения программы и необходимый объем памяти. С другой стороны, о ненадежной системе нельзя утверждать, что она «эффективна», независимо от того, как быстро она работает.

**10 группа - Скорость выполнения.**

**Скорость выполнения** может быть охарактеризована двумя мерами: мгновенной скоростью продвижения (временем ответа терминала или временем обработки сообщения) и пропускной способностью (полной работой, выполняемой за определенный период времени). Надежность в определенном смысле конфликтует со скоростью продвижения и в то же время согласуется с пропускной способностью (последняя зависит от многих факторов, в их числе — процент времени, когда система работоспособна).

***Слайд 12***

## 3. Постановка целей для программного обеспечения

*При правильной постановке целей для программного обеспечения не делается никаких предположений о конкретной реализации, но при этом указывается, каким образом на последующих этапах проектирования следует принимать компромиссные решения.*

При разработке программного обеспечения необходимы два набора целей: **цели продукта,** т. е. определение целей окончательного результата с точки зрения пользователя, и **цели проекта,** такие, как график, стоимость, степень тестированности и т. д.

**3.1 Цели продукта**

Это цели с точки зрения пользователя. Должна быть представлена следующая информация.

1. Резюме. Вначале нужно коротко сформулировать общее назначение разрабатываемого продукта.
2. Определение пользователя. Если разрабатывается большая система, с разными группами пользователей, должны быть определены роли различных пользователей.
3. Подробное перечисление функций. Здесь с точки зрения пользователя следует обрисовать функции, которые должны обеспечиваться системой.
4. Публикации. Должны быть определены цели для документации, поставляемой пользователям, в том числе типы документации и предполагаемый круг читателей для каждого типа.
5. Эффективность. Сюда относятся все цели, касающиеся эффективности, или производительности, такие, как временные характеристики, пропускная способность, использование ресурсов, а также необходимые средства измерения производительности и средства настройки.

***Слайд 13***

1. Совместимость. Если конкретный программный продукт должен быть совместим с другими, эти цели указываются здесь. Следует- указать также относящиеся к делу международные и государственные стандарты и внутренние стандарты компании.
2. Конфигурации. Здесь указываются различные конфигурации аппаратуры и программного обеспечения, в которых система может работать, и другие программные продукты, от которых она зависит, а также дополнительные возможности выбора отдельных частей системы, если это осуществимо.
3. Безопасность. Сюда относится описание целей в отношении, обеспечения безопасности. Если система связана с финансовой деятельностью (например, игорная система или система управления запасами), должны быть указаны средства надзора.
4. Обслуживание. Здесь намечаются стоимость и время исправления ошибок, а также необходимые для достижения намечаемых параметров программные средства, например диагностические программы.
5. Установка. Сюда относятся методы и средства настройки системы на конкретные условия эксплуатации.
6. Надежность. Цели в области надежности, как и другие цели, существенно зависят от конкретного типа системы.

***Слайд 14***

При определении **целей продукта с точки зрения его** надежности, следующие вопросы подлежат обязательному рассмотрению:

1. Для каждого типа отказов должно быть определено среднее время между отказами (отказ системы, ошибка пользователя, отказы конкретных функций) с учетом серьезности такого отказа.
2. Среднее время восстановления системы после отказа.
3. Ориентиры в отношении числа ошибок в программном обеспечении, расклассифицированные по серьезности и времени обнаружения.
4. Последствия отказов системы или отказа наиболее важных функций.
5. Допустимый объем данных, утрачиваемых в случае отказа.
6. Жизненно важная информация, которая должна быть защищена от разрушения.
7. Функции, необходимые для обнаружения ошибок.
8. Функции, необходимые для исправления ошибок.
9. Функции, необходимые для обеспечения устойчивости к ошибкам.
10. Возможности обнаруживать ошибки пользователя и сбои аппаратуры, а также восстанавливать работоспособность.

***Слайд 15***

3.2 Цели проекта.

Здесь описываются цели, которые касаются только самого процесса разработки и не проявляются в окончательном результате работы. *Они должны быть установлены официально, т.к. в случаях, когда программисты не имеют списка целей проекта, они получают противоречивые и неожиданные результаты.*

**Цели проекта** должны давать ответы на следующие вопросы:

1. Ориентировочная стоимость каждого процесса.
2. Календарный план проекта.
3. Цели для каждого процесса тестирования.
4. Цели в области адаптируемости, указывающие степень адаптируемости, или расширяемости, которая должна быть достигнута.
5. Вопросы сопровождения создаваемой системы, которые необходимо учитывать при разработке.
6. Уровни надежности, которые должны быть достигнуты на каждом этапе разработки для достижения заданной надежности продукта.
7. Внутренняя документация при работе над проектом.
8. Критерии для оценки готовности продукта к использованию.

**Оценка целей**

Правило «n плюс минус один» предлагает привлечь к оценке целей автора требований и проектировщика исходных внешних спецификаций. *Однако, необходимо привлечение дополнительных сил, т.к. цели – самый важный аспект проекта. К оценке должны быть привлечены представители организации-заказчика, а также те, кто будет заниматься проектированием, тестированием, сопровождением и подготовкой публикаций. Первый шаг – сопоставить цели с требованиями, чтобы убедится, что все требования переведены на язык целей.*

***Слайд 16***

**ВНЕШНЕЕ ПРОЕКТИРОВАНИЕ**

Внешнее проектирование — это процесс описания ожидаемого поведения разрабатываемого продукта с точки зрения внешнего по отношению к нему наблюдателя. Цель этого процесса — «конструирование» внешних взаимодействий будущего продукта (обычно с пользователем) без конкретизации его внутреннего устройства.

*Внешний проект выражается в форме внешних спецификаций, предназначенных для широкой аудитории, включающей пользователя (для проверки и одобрения), авторов документации для пользователя, всех участвующих в проекте программистов, а также всех тех, кто будет заниматься тестированием продукта.*

*Подготовка полных и правильных внешних спецификаций — одна из важных задач в разработке программного обеспечения, так как внешние спецификации участвуют в большем числе процессов перевода, чем любой другой проектный документ.*

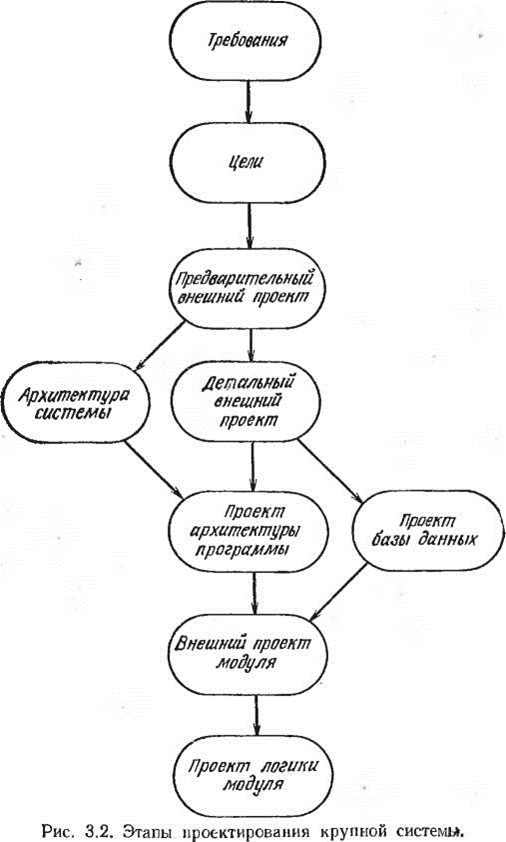
В явном виде методология внешнего проектирования отсутствует, но при разработке внешних спецификаций используется принцип концептуальной целостности.

Концептуальная целостность — это согласованность между внешними функциями системы.

ПОДГОТОВКА ВНЕШНИХ СПЕЦИФИКАЦИЙ

Правильно составленные внешние спецификации — объемистый документ, организованный иерархически.

В лекции 2, при рассмотрении ***Модели процессов проектирования*** внешнее проектирование представлено в виде двух процессов: предварительного и детального внешнего проектирования.



*В случае если спецификации организованы иерархически, двум этим процессам соответствуют* ***контрольные точки*** *в процессе нисходящего проектирования системы в соответствии с этой иерархией.*

***Первый шаг*** *— обрисовать основные компоненты (или то, что находится на первом уровне),* ***второй шаг*** *— проектирование компонент,* ***третий шаг*** *— внешние функции (функции пользователя) и* ***четвертый шаг*** *- детали всех функций пользователя.*

***Предварительное внешнее проектирование*** включает три первых шага. *Система проектируется до такого уровня, когда уже выделены все функции пользователя, но точные их синтаксис, семантика и выходные результаты остаются еще неопределенными.*

*Этим преследуются* ***две цели****: во-первых, внутри продолжительного процесса внешнего проектирования устанавливается контрольная точка для руководства и, во-вторых, становится возможной проверка правильности промежуточного уровня проекта и сопоставление его с поставленными целями.*

***Слайд 17***

**Детальный внешний проект** каждой функции пользователя должен освещать следующие вопросы.

1. Описание входных данных. Точное описание синтаксиса (например, формат, допустимые значения, области изменения) и семантики всех данных, вводимых пользователем. Этими «данными» могут быть приказ, управляющая карта, входной документ, ответ на подсказку или аналоговый сигнал.
2. Описание выходных данных. Точное описание всех результатов функции (например, реакция терминала, сообщения об ошибках, отчеты, аналоговые управляющие сигналы). Должна быть описана функциональная связь входных данных с выходными; это значит, что читатель спецификаций должен быть в состоянии представить себе выходные данные, порождаемые каждым конкретным вариантом входных. Для каждой функции должны быть указаны также результаты для всех неправильных входных данных.
3. Преобразования системы. Многие внешние функции не только порождают выходные данные, но изменяют также состояние системы. Здесь должны быть описаны все такие преобразования системы, но при этом следует помнить, что речь идет о внешних спецификациях, поэтому преобразования должны быть описаны с точки зрения пользователя.
4. Характеристики надежности. Описание воздействия всех возможных отказов функций на саму систему, файлы и пользователя.
5. Эффективность. Описание всех требований, которые предъявляются к эффективности функции, таких, как затрачиваемое время и используемая память. *Эффективность редко удается указать в абсолютных терминах, поскольку она зависит от конфигурации аппаратуры, скорости линий связи, эффективности всех остальных выполняемых параллельно программ, числа активных пользователей терминалов и т. д. Чтобы справиться с этой проблемой, в спецификациях можно описать несколько стандартных конфигураций и уровней нагрузки, а затем можно указать эффективность отдельных функций по отношению к ним.*
6. Замечания по программированию. Внешние спецификации должны описывать продукт с точки зрения пользователя и избегать ограничений на внутреннее устройство системы.

*Внешние спецификации описывают все возможные входные данные системы (как допустимые, так и недопустимые) и соответствующую реакцию системы. Внешние спецификации должны точно и полно описывать внешние сопряжения при минимальных предположениях о внутреннем устройстве системы (или совсем без них).*

***Слайд 18***

## Литература

1. Орлов С.А. Технологии разработки программного обеспечения : учеб. для вузов. - 3-е изд СПб.: Питер, 2004
2. Аллен Э. Типичные ошибки проектирования : [Пер. с англ.] СПб.: Питер, 2003
3. Фатрелл Р.Т. Управление программными проектами: достижение оптимального качества при минимуме затрат : [Пер. с англ.] М.: Вильямс, 2003
4. Липаев В.В. Надежность программных средств./ В.В. Липаев — М.: СИНТЕГ,1998
5. Майерс Г. Надежность программного обеспечения: Пер. с англ/ Г. Майерс.. - М.: Мир, 1980.

***Слайд 19***

# Контрольные вопросы

1. На какие группы можно разделить программные проекты при определении требований?
2. Для чего при определении требований используются HIPO-диаграммы?
3. Какие ошибки характерны при постановке целей, как они влияют на надежность ПО?
4. На какие группы разбивают цели программного обеспечения, кратко охарактеризуйте их?
5. Для чего необходимы цели продукта?
6. Дайте характеристику целей проекта.
7. Какие вопросы должны быть рассмотрены при определении целей продукта с точки зрения его надежности.
8. Как осуществляется оценка целей?
9. Для чего необходимы внешние спецификации?
10. Что указывается в детальном внешнем проекте?