*Слайд 1*

# **Лекция 1**

# **Надежность программного обеспечения: основные понятия.**

*Слайд 2*

**Учебные вопросы:**

**1. Понятие надежности программного обеспечения.**

**1.1 Надежность ПО.**

**1.2 Факторы, влияющие на надежность ПО.**

**1.3 Надежность аппаратуры.**

**1.4 Высокая стоимость программного обеспечения.**

**2. Понятие ошибки в программировании. Их причины и последствия.**

**3. Разработка программного обеспечения: макро- и микро- модель перевода.**

*Одной из самых серьезных проблем в области обработки данных является проблема программного обеспечения. Внешне она проявляется в жалобах на то, что программное обеспечение, во-первых, дорого и, во-вторых, ненадежно. Большинство специалистов считает первый из недостатков во многом является проявлением второго: поскольку современное ПО по своей природе ненадежно, его тестирование и обслуживание требует существенных расходов.*

*Проблема надежности ПО рассматривалась еще на заре применения вычислительных машин. Одно из таких наблюдений было сделано английскими математиками еще в 1952 году. Хотя ошибки в программном обеспечении встречались и до 1952 г., это было первое признание проблемы надежности и того факта, что тестирование требует значительного времени и даже после его завершения некоторые ошибки в ПО остаются необнаруженными.*

*Слайд 3*

# **1. Понятие надежности программного обеспечения**

## **1.1 Надежность ПО**

**Надежность программного обеспечения** - способность программного продукта безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени с высоким уровнем вероятности.

Степень надежности характеризуется вероятностью работы программного продукта без отказа в течение определенного периода времени.

Существует *4* основные составляющие функциональной надежности программных систем:

* *безотказность* - свойство программы выполнять свои функции во время эксплуатации;
* *работоспособность* - свойство программы корректно (так как ожидает пользователь) работать весь заданный период эксплуатации;
* *безопасность* - свойство программы быть не опасной для людей и окружающих систем;
* *защищенность* - свойство программы противостоять случайным или умышленным вторжениям в нее.

*Слайд 4*

На рис. 1.1 показана модель анализа надежности программных средств.



*Рис. 1.1 Модель анализа надежности программных средств*

*Слайд 5*

## **1.2 Факторы, влияющие на надежность ПО**

К числу основных факторов, влияющих на надежность ПО отнесены:

* взаимодействие ПО с внешней средой (программно-аппаратная средства, трансляторы, ОС). *Этот фактор вносит наименьший вклад в надежность ПО при современном уровне надежности аппаратуры, ОС и компиляторов;*
* взаимодействие с человеком (разработчиком и пользователем) *(см. например метрику Холстеда)*;
* организация ПО (проектирование, постановка задачи и способы их достижения и реализации) и качество его разработки. *Этот фактор вносит наибольший вклад в надежность*;
* тестирование.

*Слайд 6*

## **1.3 Сравнительная оценка надежности аппаратуры и надежности ПО**

*Для лучшего понимания надежности программного обеспечения стоит сравнить ее с надежностью аппаратуры.*

Возможны три причины отказа некоторого устройства: ошибка проектирования, производственный дефект и сбой.

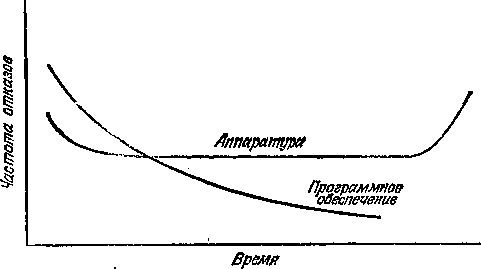
*Ошибка проектирования* — это дефект, который с самого начала присутствует в каждом экземпляре изделия. Это ошибка такого рода, когда, скажем, часть памяти ЦП оказывается недоступной из-за логической ошибки в проектировании схемы адресации.

*Производственный дефект* — это ошибка, имеющаяся в одном или нескольких экземплярах изделия из-за того, что эти конкретные экземпляры изготовлены неверно. Например, производственные дефекты могут быть вызваны плохой пайкой или неправильным соединением проводов.

*Сбои* — это дефекты, первоначально не присутствующие в изделии, но возникающие в процессе его функционирования вследствие некоторых физических явлений, например, ухудшения параметров, вызванного молекулярными дефектами, нагревом, влажностью, трением, радиацией и т. д. *Примерами сбоев могут быть физический износ переключателя вследствие трения, размагничивание магнитного сердечника при перегреве или выход из строя интегральной схемы вследствие медленного ухудшения ее герметизации.*

*Слайд 7*

На рис. 2 показано изменение со временем частоты отказов типичного устройства.



*Рис. 1.2 Различие между надежностью аппаратуры и   
программного обеспечения*

*Надежность программного обеспечения существенно отличается от надежности аппаратуры. Программы не изнашиваются, поломка программы невозможна. Более того, производственные дефекты (такие, как ошибка при копировании системы на ленту во время ее переноса) не имеют значения, поскольку они относительно редки и быстро обнаруживаются.*

Таким образом, *ненадежность программного обеспечения —* следствие исключительно ошибок проектирования, т. е. ошибок, внесенных в процессе разработки. *В условиях, когда ошибки исправляются тотчас же, как только они обнаруживаются (причем они никогда не появляются повторно), изменение надежности программного обеспечения обычно соответствует кривой, изображенной на рис. 2. Подчеркнем, что эта кривая отражает предположение, что при исправлении обнаруженных ошибок не вносятся новые. Это предположение обычно не справедливо.*

*Чтобы подчеркнуть различия между ошибками в программном обеспечении и аппаратуре, рассмотрим поведение сбоев аппаратуры и ошибок в программе в зависимости от входных данных и времени. Обычно сбои аппаратуры не зависят от обрабатываемых системой данных. Если схема двоичного сумматора разрушается и вот-вот выйдет из строя, конкретные значения обрабатываемых данных не влияют на сбой. Проявление ошибок в программе, напротив, в высшей степени зависит от входных данных. Такая ошибка обнаруживается в некоторый определенный момент именно потому, что в этот момент обрабатывается ранее не встречавшаяся последовательность входных данных.*

Надежность аппаратуры определяется во многом случайными сбоями, надежность программного обеспечения — скрытыми в нем ошибками.

*Частота сбоев аппаратуры существенно зависит от времени. Все физические устройства имеют некоторый срок службы, по истечении которого эта частота начинает быстро расти. Частота, с которой обнаруживаются ошибки в программном обеспечении, хотя иногда и кажется зависящей от времени, в действительности является функцией входных данных и состояния системы.*

Большую часть времени жизни аппаратного устройства его сбои случайны и описываются пуассоновским законом распределения, на этом основаны многие исследования. Ошибки в программе проявляются как систематические, далеко не случайные события.

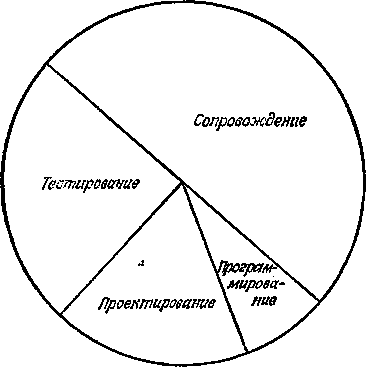
*Слайд 8*

## **1.4 Высокая стоимость программного обеспечения**

Помимо надежности ПО основной проблемой в производстве программного обеспечения — его высокая стоимость.

*Предлагаемые решения проблемы стоимости обычно сводятся к попытке поднять «производительность программиста», изобрести инструменты и методы, позволяющие ему работать быстрее.*

*Рис. 1.3 иллюстрирует относительные затраты на программное обеспечение в течение его жизненного цикла для большинства типичных крупных разработок. На долю сопровождения (устранения ошибок и внесения небольших изменений после установки системы) и тестирования приходится почти 75% затрат.*



*Рис. 1.3 Типичное распределение стоимости программного обеспечения*

Высокая стоимость программного обеспечения — во многом следствие низкой надежности. При увеличении производительности программиста (если измерять ее только скоростью разработки и кодирования программы) стоимость существенно *не* уменьшается.

*Наилучший путь решительного сокращения стоимости программного обеспечения — в уменьшении стоимости его тестирования и сопровождения. Это может быть достигнуто не за счет инструментов, призванных увеличить скорость программирования, а лишь в результате разработки средств, повышающих корректность и четкость при создании программного обеспечения.*

*Слайд 9*

# **2. Понятие ошибки в программировании. Их причины и последствия.**

*При анализе надежности ПО требуется определить, что такое ошибка в программном обеспечении и что такое надежность программного обеспечения.*

*В существующей литературе дается несколько определений, что такое ошибка в ПО. Рассмотрим некоторые из определений ошибки.*

***Программное обеспечение содержит ошибку, если его поведение не соответствует спецификациям.*** *Недостаток данного определения - неявно предполагается, что спецификации корректны, однако подготовка спецификаций – один из источников ошибок.*

***Программное обеспечение содержит ошибку, если его поведение не соответствует спецификациям при использовании в установленных при разработке пределах.*** *В действительности, если система случайно используется в непредусмотренной ситуации, ее поведение должно оставаться разумным, иначе она содержит ошибку.*

*Например: Рассмотрим авиационную систему диспетчеризации, которая прослеживает и координирует движение самолетов над некоторым географическим районом. Предположим, что, согласно спецификациям, система должна управлять движением до 200 самолетов одновременно. Но однажды по непредвиденным обстоятельствам в районе появился 201 самолет. Если поведение системы неразумно — скажем, она забывает об одном из самолетов или выходит из строя,— система содержит ошибку, хотя она используется вне пределов, установленных при проектировании.*

***Ошибка имеется тогда, когда ПО ведет себя не в соответствии с официальной документацией.*** *Это определение неудачно, т.к. ПО может вести себя в соответствии с официальной документацией, но ошибки имеются, т.к. содержатся они и в программе, и в документации.*

*Например: Имеется руководство для пользователей системы разделения времени, в котором говорится: «Чтобы дать новую команду, нажмите один раз клавишу - «внимание» и напечатайте эту команду». Предположим, что пользователь случайно нажимает клавишу «внимание» дважды и система программного обеспечения выходит из строя, потому что ее разработчики не предусмотрели такой ситуации. Система, очевидно, содержит ошибку, но мы не можем утверждать, что она ведет себя не в соответствии с публикациями.*

***Ошибка определяется как неспособность системы действовать в соответствии с исходным контрактом или перечнем требований пользователя.*** *Такое определение также содержит недостатки, т.к. если согласно требованиям пользователя, система должна обеспечивать среднее время между отказами из-за ошибки в ПО на уровне 100 часов, а для действующей системы этот показатель равен 150 часов, система все же имеет ошибки, даже несмотря на то, что она превышает требования пользователя. Практически невозможно детализировать действия пользователя настолько, чтобы описывать желаемое поведение ПО при всех мыслимых обстоятельствах.*

В соответствии с ГОСТ 27.002 – 89 «Надежность в технике. Основные понятия. Термины и определения.», под **ошибкой подразумевается неправильность, погрешность или неумышленное искажение объекта или процесса, что может быть причиной ущерба - риска при функционировании или применении программы**.

*При этом предполагается, что известно правильное, эталонное состояние объекта или процесса по отношению к которому может быть определено наличие отклонения.* Исходным эталоном для любого программного обеспечения являются спецификации требований заказчика или потенциального пользователя, предъявляемых к программам и ожидаемый пользователем или заказчиком эффект от использования программного обеспечения. *Важной особенностью при этом является отсутствие полностью определенной программы - эталона, которой должны соответствовать текст и результаты функционирования разрабатываемой программы. Поэтому определить качество программного обеспечения и наличие ошибок в нем путем сравнения разрабатываемой программы с эталонной программой невозможно.*

*Риски проявляются как негативные последствия проявления ошибок в программном обеспечении в ходе его применения и функционирования, которые могут нанести ущерб системе, в которой применяется это программное обеспечение, внешней среде или пользователям этой системы в результате отклонения характеристик программного обеспечения заданных или ожидаемых пользователем или заказчиком.*

*Ошибки в ПО* ***не являются*** *внутренним его свойством. Это значит, что как бы долго и пристально не тестировалась программа, невозможно рассчитывать и найти их все.*

Одной из причин появления ошибок является сложность ПО.

В борьбе со **сложностью ПО** используются две концепции**:**

**Иерархическая структура**. *Иерархия позволяет разбить систему по уровням понимания (абстракции, управления). Концепция уровней позволяет анализировать систему, скрывая несущественные для данного уровня детали реализации других уровней.* Иерархия позволяет понимать, проектировать и описывать сложные системы.

**Независимость.** В соответствии с этой концепцией, для минимизации сложности, необходимо максимально усилить независимость элементов системы. *Это означает такую декомпозицию системы, чтобы её высокочастотная динамика была заключена в отдельных компонентах, а межкомпонентные взаимодействия (связи) описывали только низкочастотную динамику системы.*

*Слайд 10*

*Методы обнаружения ошибок*, которые базируются на введении в ПО системы различных видов избыточности:

**Временная избыточность**. Использование части производительности ЭВМ для контроля исполнения и восстановления работоспособности ПО после сбоя.

**Информационная избыточность.** Дублирование части данных информационной системы для обеспечения надёжности и контроля достоверности данных.

**Программная избыточность** включает в себя: взаимное недоверие - компоненты системы проектируются, исходя из предположения, что другие компоненты и исходные данные содержат ошибки, и должны пытаться их обнаружить; немедленное обнаружение и регистрацию ошибок; выполнение одинаковых функций разными модулями системы и сопоставление результатов обработки; контроль и восстановление данных с использованием других видов избыточности.

*Слайд 11*

***Задачи обеспечения ПО устойчивости к ошибкам*** направлены на применение методов минимизации ущерба, вызванного появлением ошибок, и включают в себя:

* обработку сбоев аппаратуры;
* повторное выполнение операций;
* динамическое изменение конфигурации;
* сокращенное обслуживание в случае отказа отдельных функций системы;
* копирование и восстановление данных;
* изоляцию ошибок.

Известны *4* группы *принципов обеспечения надежности ПО от влияния ошибок*:

* предупреждение ошибок;
* обнаружение ошибок;
* исправление ошибок;
* обеспечение устойчивости к ошибкам.

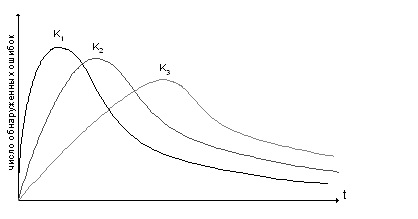
*Слайд 12*

Действия, направленные на минимизацию ошибок и сбоев:

* предотвращение ошибок за счет структурного программирования;
* сокрытие информации или дозированный доступ к данным со стороны программных средств и объектов в объектно-ориентированном программировании;
* отладка;
* устойчивость к сбоям;
* обработка исключительных ситуаций (перехват ошибок, например, деление на ноль) и локализация ошибок и сбоев;
* восстановление программы после сбоя.

В соответствии с ГОСТ 19.004-80 «ЕСПД» различают следующие *виды работ, направленные на устранение ошибок в ПО*: проверка, отладка и испытание программы.

Чем интенсивнее использование программного изделия (ПИ), тем быстрее выявляются в нем ошибки. На рисунке 4 приведена зависимость числа обнаруженных ошибок от числа использующих ПИ пользователей:



где *K* – число пользователей, *K1 > K2 > K3*.

Рис.1.4 Интенсивность обнаружения ошибок от интенсивности использования

*Слайд 13*

Процентные частоты появления ошибок в ПО по типам ошибок представлены в таблице 1.1.

Таблица 1.1 – Процентные частоты появления ошибок в ПО

| ТИП ОШИБКИ | ЧАСТОТА ПОЯВЛЕНИЯ, % |
| --- | --- |
| Не полная или ошибочная спецификация | 28 |
| Отклонение от спецификации | 12 |
| Пренебрежение правилами программирования | 10 |
| Ошибочная выборка данных | 10 |
| Ошибочная логика или последовательность операций | 12 |
| Ошибочные арифметические операции | 9 |
| Нехватка времени для решения | 4 |
| Ошибка обработки прерываний | 4 |
| Ошибка в исходных данных | 3 |
| Неточная запись | 8 |

Как видно из таблицы 1.1 основное количество ошибок делается из-за неверной спецификации или ТЗ.

*Слайд 14*

Ошибки спецификации, в свою очередь, могут быть разделены на следующие категории:

Таблица 1.2 – Категории ошибок спецификации в ПО

| ПРИЧИНА ОШИБКИ | ЧАСТОТА ПОЯВЛЕНИЯ, % |
| --- | --- |
| Ошибки в числовых значениях | 12 |
| Недостаточные требования к точности | 4 |
| Ошибочные символы или знаки | 2 |
| Ошибки оформления | 15 |
| Неправильное описание или требование к аппаратуре | 2 |
| Исходные данные для разработки неполные, неточные или ошибочные | 52 |
| Двусмысленность требований | 13 |

*Анализ таблиц показывает, на что нужно обращать особое внимание при проведении* ***валидации*** *и* ***верификации ПО*** *(****верификация*** *отвечает на вопрос, правильно ли и качественно ли создана программа, а* ***валидация*** *(или аттестация) - на вопрос правильно ли работает программа).*

*Слайд 15*

В соответствии с **ГОСТ 27.310 - 95** (Надежность в технике. Анализ видов, последствий и критичности отказов.) и ГОСТ 51901.12 - 2007 Менеджмент риска. Метод анализа видов и последствий отказов) для программного обеспечения, создаваемого для систем управления, потеря работоспособности которых может повлечь за собой катастрофические последствия, возможные категории тяжести ошибок приведены в таблице 1.3.

**Таблица 1.3. Категории тяжести ошибки в программном обеспечении, нарушение работоспособности которого могут привести к катастрофическим последствиям**

|  |  |  |
| --- | --- | --- |
| **Номер**  **категории**  **ошибки** | **Наименование категории тяжести ошибки** | **Описание последствий проявления ошибки** |
| **IV** | Катастрофическая | проявление ошибки с высокой вероятностью влечет за собой прекращение функционирования программного обеспечения (его отказ) и может вызвать повреждение системы и окружающей среды и/или гибель и тяжелые травмы людей |
| **III** | Критическая | проявление ошибки с высокой вероятностью влечет за собой прекращение функционирования программного обеспечения (его отказ), может вызвать повреждение системы и окружающей среды, но не угрожает жизни и здоровью людей |
| **II** | Существенная | проявление ошибки влечет за собой снижение эффективности функционирования программного обеспечения и может вызвать прекращение функционирования программного обеспечения (его отказ) без заметного повреждения системы и угрозы жизни и здоровью людей |
| **I** | Несущественная | проявление ошибки может повлечь за собой снижение эффективности функционирования программного обеспечения и практически не приводит к возникновению отказа в нем (вероятность возникновения отказа очень низкая) |

*Слайд 16*

*В соответствии с теми же нормативными документами для программного обеспечения общего применения или программного обеспечения систем, нарушение работоспособности которых не представляет угрозы жизни людей и не приводит к разрушению самой системы, возможные категории тяжести приведены в таблице 1.4.*

**Таблица 1.4. Категории тяжести ошибки в программном обеспечении, нарушение работоспособности которого не приводят к катастрофическим последствиям**

|  |  |  |
| --- | --- | --- |
| **Номер**  **категории**  **ошибки** | **Наименование категории тяжести ошибки** | **Описание последствий проявления ошибки** |
| **III** | Критическая | проявление ошибки с высокой вероятностью влечет за собой прекращение функционирования программного обеспечения (его отказ) |
| **II** | Существенная | проявление ошибки влечет за собой снижение эффективности функционирования программного обеспечения и может вызвать прекращение функционирования программного обеспечения (его отказ) |
| **I** | Несущественная | проявление ошибки может повлечь за собой снижение эффективности функционирования программного обеспечения и практически не приводит к возникновению отказа в нем (вероятность возникновения отказа очень низкая) |

*Одной из основных причин ошибок в программном обеспечении* — ***неправильный перевод информации (из одного представления в другое)****.*

***Программное обеспечение*** *— это совокупность информационных элементов (но не физических объектов), описывающих решение задачи. Создание программного обеспечения в этом случае — совокупность процессов трансляции, т. е.* ***перевода исходной задачи в различные промежуточные решения****, до тех пор пока будет получен подробный набор машинных команд.*

*Понимание того, что именно ошибки перевода являются причиной ошибок в программе, чрезвычайно важно, так как это — ключ к пониманию проблемы надежности.*

*Слайд 17*

# **3. Разработка программного обеспечения: макро- и микро- модель перевода**

*Прежде чем приступить к ее анализу, следует сделать несколько замечаний. В эту модель не включен процесс тестирования программы. Представлены только два шага проектирования, хотя в действительности их обычно бывает (и должно быть) больше. Задача модели — не в том, чтобы перечислить все рекомендуемые действия, а показать лишь основные типы возникающих процессов перевода.*

Модель разработки программного обеспечения изображена на рис. 1.5.

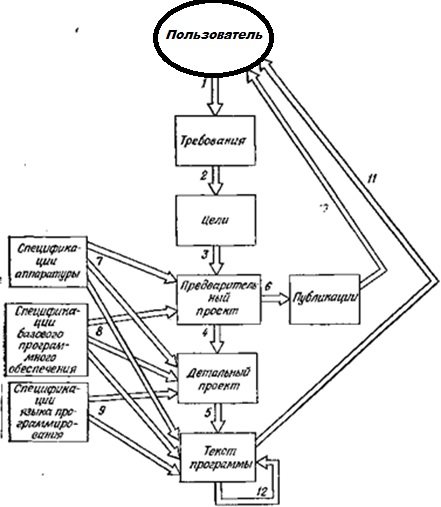


Рис. 1.5 Модель разработки программного обеспечения

Чтобы подробнее исследовать проблему ошибок в программном обеспечении, рассмотрим различные типы процессов перевода при его создании.

Процессы перевода

Рассмотрим каждый из процессов перевода

1. ***Требования:*** *Этот процесс начинается с разработки описания решаемой задачи.* Такое описание имеет вид перечня требований пользователя. *В некоторых случаях пользователь составляет этот перечень сам. В других случаях это делает разработчик программного обеспечения либо в результате бесед с пользователем, либо исследуя его потребности, либо самостоятельно оценивая эти потребности в будущем, либо комбинируя перечисленные методы.*

Очевидно, здесь имеются обширные **возможности** для **ошибок**, например, *пользователь может не суметь адекватно выразить свои потребности, они могут быть неверно поняты либо учтены не в полном объеме. Ясно, что ошибки, возникающие на этом уровне, обходятся чрезвычайно* ***дорого****.*

1. **Второй процесс** *состоит в переводе требований пользователя в цели программы. Объем перевода невелик, но требуется оценить компромиссные решения.* **Ошибки на этом шаге возникают, когда неверно интерпретируются требования пользователя**.
2. **Третий шаг** связан с преобразованием целей программы в ее внешние спецификации, т. е. *точное описание поведения всей системы с точки зрения пользователя.* *По* ***объему перевода это самый значительный этап разработки*** *программного обеспечения.* **Больше всего подвержен ошибкам**.
3. **Четвертый шаг** представляет собой несколько процессов перевода — *от внешнего описания готового продукта до получения детального проекта*, *описывающего множество составляющих программу предложений, выполнение которых должно обеспечить поведение системы, соответствующее* ***внешним спецификациям****.* Этот шаг включает такие процессы, *как перевод внешнего описания в структуру компонент программы* (например, модулей) и *перевод каждой из этих компонент в описание процедурных шагов* (например, в блок-схемы). Шансы внесения ошибок становятся чрезвычайно высокими.
4. **Последний (пятый шаг) процесс проектирования** — перевод описания логики программы в предложения языка программирования. *Хотя на этом шаге часто делается много ошибок, они обычно относительно мелкие, легко обнаруживаются и корректируются.*

На этом шаге возникает также и другой **процесс трансляции**: *перевод представления программы на языке программирования в объектный (выполняемый машиной) код (компилятором).* *Иногда и компиляторы содержат ошибки, вследствие чего ошибки могут появиться и в объектной программе.*

1. *В результате работы над программным проектом возникают как само программное обеспечение, так и* **документы, описывающие его использование**. *Последние обычно имеют вид печатных руководств, хотя их можно хранить и в самой вычислительной системе и выдавать, например, на терминал пользователя. Эти руководства обычно получаются переводом внешних спецификаций в материалы, ориентированные на конкретные группы пользователей.*

**Публикации** определенным образом **влияют на надежность программного обеспечения. *Если ошибка возникает при их подготовке, они не будут точно описывать поведение программы.***

1. *Спецификации аппаратуры являются источником информации во время разработки. Неправильное истолкование материалов может привести к ошибкам в программном обеспечении.*
2. Обычно **прикладные программы взаимодействуют с базовым программным обеспечением вычислительной системы**, обращаясь к нему с *заказами на ввод-вывод и динамическое распределение ресурсов*. *Неправильное понимание документации по базовому программному обеспечению служит еще одним источником ошибок.*
3. *Готовая программа состоит из предложений по крайней мере одного языка программирования.* ***Непонимание синтаксиса и семантики языка также является причиной ошибок в программном обеспечении.***
4. Пользователь взаимодействует с готовой программой двумя путями – через руководства, описывающие ее использование, и непосредственную работу с ней. *Необходимо изучение пользователем руководств и перевод их содержания в его понимание того, как он желает применять программу.*

*Попытки использовать систему незапланированным способом часто увеличивают вероятность натолкнуться ошибки в программном обеспечении. Таким образом, качество документации для пользователя может влиять на ее надежность.*

1. Этот шаг представляет собой ***непосредственное взаимодействие пользователя с программным обеспечением.*** *Если человеческие факторы учтены слабо (т. е. диалог человек — машина разработан плохо), вероятность ошибки пользователя увеличивается.*
2. Значительная доля затрат на программное обеспечение в мире приходится на изменения существующих программ. *Важный дополнительный шаг, возникающий в процессе изменения — чтение и понимание текста существующей программы для выяснения того, что и как надо изменить. Этот шаг появляется при сопровождении (исправлении ошибок) и модификации (добавлении новых функций существующим программам). Ошибки могут возникать и на этом этапе.*

Макромодель перевода описывает происхождение большинства ошибок в программном обеспечении.

**Макромодель перевода важна, поскольку она описывает причины, лежащие в основе ненадежности. Она позволяет определить перечень подлежащих решению задач.**

*Слайд 18*

Микромодель перевода

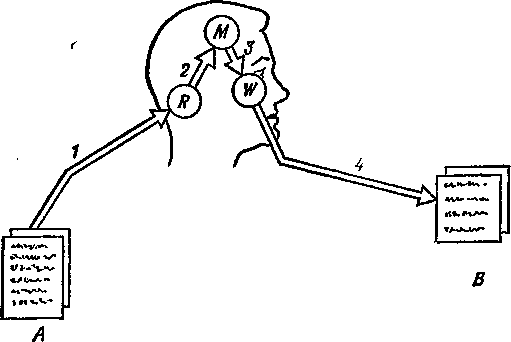


Рис. 6. Микромодель перевода.

На данной модели **человек — любое лицо**, описываемое макромоделью: ***пользователь, системный аналитик, проектировщик или программист***. Этот человек пытается переводить информацию из формы *А* в форму *В.* Для этого ему следует совершить четыре основных шага.

1. Получить информацию с помощью своего читающего механизма *R* (областей мозга, управляющих зрением и слухом).
2. Запомнить информацию в своей памяти *М.*
3. Выбрать из памяти эту информацию, а также информацию, описывающую процесс перевода, выполняет перевод и посылает результат своему пишущему механизму *W* (областям мозга, управляющим речью или движением рук).
4. Информация физически распространяется с помощью письма, печати на терминале или речи.

Даже такая упрощенная модель перевода позволяет исследовать возникающие на каждом из этапов ошибки:

1. *Человек понимает входную информацию, сопоставляя ее с набором хранимых в памяти образцов и жизненным опытом. Эти способности в некоторых случаях вызывают неточности в процессе перевода, являются причиной ошибок в программном обеспечении.* Ошибки возникают тогда, когда при чтении документа *А* человек видит то, что он ожидает или хочет увидеть, а не то, что написано, когда он пытается восстановить недостающие факты или просто не замечает существенной информации.
2. *В большинстве случаев, для того чтобы правильно запомнить информацию, нам надо ее понять.* На этом этапе ошибки в программном обеспечении появляются в результате неправильной интерпретации входной информации. Она может быть слишком сложной или двусмысленной, квалификация человека может оказаться недостаточной.
3. Основной источник ошибок на этом этапе — забывчивость: человек может забыть входную информацию либо точные правила выполнения перевода.
4. На этом этапе человек может допускать ошибки, потому что не умеет ясно писать и выражать свои мысли — это затемняет смысл сообщений (например, используя сокращения). *Это увеличивает вероятность того, что следующий участник процесса разработки при переводе совершит ошибки.*

*Средства представления проекта на различных уровнях (например, языки описаний и программирования) влияют на процесс перевода. Процессы проектирования и свойства самого проекта (например, его сложность) также существенно влияют на ошибки.*

*Слайд 19*

# **Литература:**

1. Казарин О. В. Надежность и безопасность программного обеспечения: учебное пособие для бакалавриата и магистратуры /О. В. Казарин, И. Б. Шубинский. — М.: Издательство Юрайт, 2018. — 342 с.
2. Орлов С.А. Технологии разработки программного обеспечения : учеб. для вузов. - 3-е изд СПб.: Питер, 2004
3. Аллен Э. Типичные ошибки проектирования : [Пер. с англ.] СПб.: Питер, 2003
4. Фатрелл Р.Т. Управление программными проектами: достижение оптимального качества при минимуме затрат : [Пер. с англ.] М.: Вильямс, 2003
5. Липаев В.В. Надежность программных средств./ В.В. Липаев — М.: СИНТЕГ,1998
6. Майерс Г. Надежность программного обеспечения: Пер. с англ/ Г. Майерс.. - М.: Мир, 1980.

*Слайд 20*

# **Контрольные вопросы:**

1. Что понимают под надежностью программного обеспечения (ПО)? Какие известны составляющие функциональной надежности программных систем?
2. В чем разница между надежностью программного обеспечения и надежностью аппаратуры?
3. Что вносит наибольший вклад в стоимость ПО, почему?
4. Какие концепции используются в борьбе со сложностью ПО?
5. Какие методы обнаружения ошибок, которые базируются на введении в ПО системы различных видов избыточности известны?
6. Какие существуют группы принципов обеспечения надежности ПО от влияния ошибок?
7. Как интенсивность обнаружения ошибок ПО зависит от интенсивности его использования?
8. Дайте характеристику макромодели перевода.
9. Что в себя включают второй и третий этап микромодели перевода и как они влияют на появление ошибок в ПО?