**Вопросы по лекции №10.**

**Внутренние характеристики качества ПО- характеристики качества разработчика. Минимизация сложности ПО. Проектирование «сверху вниз» и «снизу вверх». Эвристические методы конструирования ПО.**

Бригада СанУ11: Ковалев К.А., Терентьев Е.А., Щекочихин Д.А., Оганисян Т.С., Шабакаев А.Р.

Выполнил: Оганисян Т.С.

Группа 6411

**1.Какие внутренние характеристики качества ПО – характеристики качества разработчика должны быть обеспечены при конструировании ПО.**

1. Минимизация сложности разработки и создание легко понимаемого кода за счет структуризации ПО - разбиения на небольшие части.

2. Безопасность многозадачности. Безопасный порядок доступа частей ПО к параллельно используемым критическим ресурсам ЦВМ.

3. «Ожидание изменений» - подготовленность к беспроблемному их проведению.

4. Устойчивость к ошибкам входных данных, устойчивость к программным ошибкам.

5. Возможность удобной проверки- контролепригодность.

6. Сокрытие информации между частями ПО.

**2.Что означает и как обеспечивается свойство ПО – устойчивость к ошибкам.**

Устойчивость к ошибкам — это свойство ПО не выдавать фатально неправильный результат и прекращать развитие ошибочной ситуации. Оно может быть реализовано, как прекращением функционирования ПО с выдачей сообщения об ошибке при её обнаружении, так и продолжением функционирования ПО с выдачей нейтрального или приближенного, но не приводящего к фатальной ситуации результата.

**3.Как обеспечивается минимизация сложности ПО.**

Минимизация сложности ПО -разделение ПО на небольшие несложные фрагменты. Несколько несложных фрагментов ПО понять проще, чем один сложный.

Правильно определенные составные части, модули, процедуры или объекты разделяют задачу так, чтобы их можно было разработать последовательно по очереди или параллельно и независимо.

Также для минимизации сложности ПО используют шлифовку кода и рефакторинг.

**4. Что такое рефакторинг?**

Рефа́кторинг - процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. Рефакторинг не предназначен для исправления ошибок и добавления новой функциональности. Он выполняется для улучшения понятности кода или изменения его структуры, для удаления «мёртвого кода», заплаток, которыми обрастает программа в процессе разработки.

Рефакторинг обычно не изменяет функции и поведение программы.

**5. Как обеспечивается приспособленность ПО к изменениям.**

- Иерархическое структурирование ПО, локализация фрагментов кода, подверженных изменениям и развитию, в отдельные элементы;

- Предотвращение дублирования кода;

- Рефакторинг кода;

- Удобство тестирования;

- Предотвращение избыточной информации в программе, локализация значений переменных в интерфейсе

**6.В чем заключается первый эвристический прием разработки ПО.**

Первый эвристический прием, позволяющий создать понятный код – описание простым русским языком то, что должна сделать программа, так же, как если бы пришлось объяснить это коллеге, отложив на «потом» описание некоторых кажущихся важными деталей. Это описание всегда становится более понятным, если удаётся скрыть детали путём введения дополнительно функций, имеющих понятное «говорящее» название.

**7. Опасности прерывания программ в «неудобное время» и защита от них**

Если после начала передачи таких данных произойдет прерывание, во время отработки которого данные могут быть обновлены, то некоторые из данных структуры будут связаны со старыми значениями, а некоторые с новыми и весь набор данных может оказаться неверным.

Также утрата целостности данных может произойти, когда переменная используется несколько раз в последовательности операторов и её правильное окончательное значение формируется после последнего оператора. В этом случае прерывание задачи более приоритетной задачей, использующей ту же переменную, может привести к ошибке – использованию в прервавшей приоритетной задаче «недообработанного» значения переменной.

Только знание возможности таких ситуаций и правильное защитное конструирование программ позволит избегать описанных ошибок. Правильное конструирование заключается в выделение подобных критических частей кода и предотвращение прерывания задачи во время исполнения критической части кода путем установки запрета прерывания или, запрещая диспетчеризацию задач на это время. Нужно также изменить имя полностью обработанной переменной.

Выполнил: Щекочихин Д.А.

Группа 6411

**1. Какие внутренние характеристики качества ПО – характеристики качества** разработчика должны быть обеспечены при конструировании ПО.

При конструировании программного обеспечения, кроме выполнения заданных функций должны обеспечиваться такие его внутренние свойства, которые упрощают создание ПО требуемого качества:  
1. Минимизация сложности разработки и создание легко понимаемого кода. за счет структуризации ПО - разбиения на небольшие части.  
2. Безопасность многозадачности. Безопасный порядок доступа частей ПО к параллельно используемым критическим ресурсам ЦВМ.  
3. «Ожидание изменений» - подготовленность к беспроблемному их проведению.  
4.Устойчивость к ошибкам входных данных, устойчивость к программным ошибкам.  
5.Возможность удобной проверки(тестирования) - контроле пригодность.  
6. Сокрытие информации между частями ПО по принципу «нет необходимости знать» и формирование абстракций высокого уровня.

**2. Что означает и как обеспечивается свойство ПО – устойчивость к ошибкам.** Устойчивостью к ошибкам мы будем называть свойство ПО не выдавать фатально неправильный результат и прекращать развитие ошибочной ситуации. Оно может быть реализовано, как прекращением функционирования ПО с выдачей сообщения об ошибке при её обнаружении, так и продолжением функционирования ПО с выдачей нейтрального или приближенного, но не приводящего к фатальной ситуации результата.

**3. Как обеспечивается минимизация сложности ПО.**

На уровне архитектуры ПО сложность можно снизить, разделив ПО на небольшие несложные фрагменты. Несколько несложных фрагментов ПО понять проще, чем один сложный. В разбиении сложной задачи на несколько простых и состоит цель всех методик конструирования ПО. Причем чем более независимы подсистемы, тем проще и безопаснее разработчику сосредоточится на одной подсистеме в конкретный момент времени.

Правильно определенные составные части, модули, процедуры или объекты разделяют задачу так, чтобы их можно было разработать последовательно по очереди или параллельно и независимо. Объединения частей в пакеты должно быть также простым и обеспечивать указанные преимущества при разработке очень больших задач. Такая компенсация изначальных ограничений памяти человека позволяет писать более понятный и содержащий меньшее число ошибок код.

Для проектирования с минимальным числом связей между частями ПО необходимо использовать инкапсуляцию и сокрытие информации. Это позволит максимально облегчить интеграцию, тестирование и сопровождение ПО.

**4. Что такое рефакторинг?**

Рефакторинг или реорганизация кода - процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. В основе рефакторинга лежит последовательность небольших эквивалентных (то есть сохраняющих поведение ПО) преобразований прежде всего структуры.

Рефакторинг выполняется для улучшения понятности кода или изменения его структуры, для удаления «мёртвого кода», заплаток, которыми обрастает программа в процессе разработки и т.п. — всё это для того, чтобы в будущем ПО было легче поддерживать и развивать.

**5. Как обеспечивается приспособленность ПО к изменениям.**

1. Иерархическое структурирование. Локализация поврежденных изменениям и развитию фрагментов в отдельных структурных элементах ПО.

2. Рефакторинг кода.

3. Предотвращение дублирования кода.

4. Удобство тестирования

5. Предотвращение избыточного распространения информации по программе, локализация значений входных временных в интерфейс

**6. В чем заключается первый эвристический прием разработки ПО.**

Первый эвристический прием — это значит описать простым русским языком то, что должна сделать программа, так же, как если бы пришлось объяснить это коллеге, отложив на «потом» описание некоторых кажущихся важными деталей. Это описание всегда становится более понятным, если удаётся скрыть детали путём введения дополнительно функций, имеющих понятное «говорящее» название. В дальнейшем это описание, переложенное в графическую форму, может послужить основой для создания блок-схемы программы.

В основе создания алгоритма программ обычно лежат сущности реального мира. Выделение объектов происходит во время мысленного представления процесса функционирования системы. Часто это происходит во время рассмотрения ответов на вопросы «что и кто».

**7. Опасности прерывания программ в «неудобное время» и защита от них.**

Проблема целостности данных возникает тогда, когда данные, связанные в массив, таблицу или другую структуру данных передаются во время исполнения задачи. Если после начала передачи таких данных произойдет прерывание, во время отработки которого данные могут быть обновлены, то некоторые из данных структуры будут связаны со старыми значениями, а некоторые с новыми и весь набор данных может оказаться неверным.

Следующая утрата целостности данных может произойти, когда переменная используется несколько раз в последовательности операторов и её правильное окончательное значение формируется после последнего оператора. В этом случае прерывание задачи более приоритетной задачей, использующей ту же переменную, может привести к ошибке – использованию в прервавшей приоритетной задаче «недообработанного» значения переменной.

Беда в том, что вероятность попадания прерывания на неудачный момент обычно очень мала и при отладке не реализуется, тем более что при автономной отладке вообще всё работает правильно (прерываний нет по определению).

Знание возможности возникновения таких ситуаций и правильное защитное конструирование программ позволит избегать описанных ошибок. Правильное конструирование заключается в выделение подобных критических частей кода и предотвращение прерывания задачи во время исполнения критической части кода путем установки запрета прерывания или, запрещая диспетчеризацию задач на это время. Нужно также изменить имя полностью обработанной переменной.

Выполнил: Шабакаев А.Р.

Группа 6411

**1. Какие внутренние хар-ки качества ПО – хар-ки качества разработчика должны быть обеспечены при конструировании ПО.**

- Минимизация сложности ПО;

- Безопасность многозадачности;

- Приспособленность ПО к изменениям;

- Устойчивость к ошибкам входных данных, устойчивость к программным ошибкам;

- Контролепригодность;

- Сокрытие информации между частями ПО.

**2. Что означает и как обеспечивается свойство ПО – устойчивость к ошибкам.**

Устойчивость к ошибкам – это не выдавать фатально неправильный результат и прекращать развитие ошибочной ситуации.

Устойчивость можно обеспечить прекращением работы ПО с выводом сообщения об ошибке при её обнаружении, так и продолжением работы ПО с выводом нейтрального или приближенного результата.

Для реализации свойства устойчивости необходимы средства контроля правильности входных данных и результатов его функционирования.

**3. Как обеспечивается минимизация сложности ПО.**

Минимизацию можно обеспечить разделением ПО на несколько небольших частей, сокрытием избыточной информации ПО, инкапсуляцией и рефакторингом.

**4. Что такое рефакторинг?**

Рефакторинг - процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения с целью облегчить понимание её работы.

**5. Как обеспечивается приспособленность ПО к изменениям.**

- Иерархическим структурированием ПО, локализацией фрагментов кода, подверженных изменениям и развитию, в отдельные элементы;

- Предотвращением дублирования кода;

- Рефакторингом кода;

- Удобством тестирования;

- Предотвращение избыточной информации в программе, локализация значений переменных в интерфейсе;

**6. В чем заключается первый эвристический прием разработки ПО.**

Описание простым языком того, что должна делать программа, скрыв описание деталей.

**7. Опасности прерывания программ в «неудобное время» и защита от них**

Опасность заключается в нарушении целостности набора данных, что может привести к ошибке. Также эту проблему трудно обнаружить, тк прерывание программы в «правильное время» не будет обнаружено, но может проявиться на этапе эксплуатации ПО из-за своей асинхронности относительно процесса решения задачи.

Защититься от этой проблемы поможет знание о возможности таких ситуаций и правильное защитное конструирование программ, которое заключается в выделении критических частей кода и предотвращение прерывании на время их исполнения путем установки запрета прерывания либо запрета диспетчеризации задач на это время.

Выполнил: Терентьев Е.А.

Группа 6411

**1. Какие внутренние характеристики качества ПО – характеристики качества разработчика должны быть обеспечены при конструировании ПО.**

1. Минимизация сложности разработки и создание легко понимаемого кода

2. Безопасный порядок доступа частей ПО к параллельно используемым критическим ресурсам ЦВМ

3. Подготовленность к беспроблемному проведению изменений.

4. Устойчивость к ошибкам входных данных, устойчивость к программным ошибкам.

5. Возможность удобной проверки(тестирования) - контролепригодность.

6 Сокрытие информации между частями ПО

**2.Что означает и как обеспечивается свойство ПО – устойчивость к ошибкам.**

Устойчивость к ошибкам означает свойство ПО не выдавать фатально неправильный результат и прекращать развитие ошибочной ситуации. Оно может быть реализовано, как прекращением функционирования ПО с выдачей сообщения об ошибке при её обнаружении, так и продолжением функционирования ПО с выдачей нейтрального или приближенного, но не приводящего к фатальной ситуации результата.

**3.Как обеспечивается минимизация сложности ПО.**

На уровне архитектуры ПО сложность можно снизить, разделив ПО на небольшие несложные фрагменты. Правильно определенные составные части, модули, процедуры или объекты разделяют задачу так, чтобы их можно было разработать последовательно по очереди или параллельно и независимо.

Правильное конструирование ПО предусматривает использование внутренних стандартов. соглашений и процедур, которые могут быть созданы внутри организации или проектной команды. Эти стандарты минимизируют сложность при взаимодействии членов проектной команды.

Для минимизации сложности также используется рефакторинг кода, шлифовка кода, улучшения его структуры

**4. Что такое рефакторинг?**

Рефа́кторинг или реорганизация кода - процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. Рефакторинг не изменяет функции и поведение программы. Он выполняется для улучшения понятности кода или изменения его структуры, для удаления «мёртвого кода», заплаток, которыми обрастает программа в процессе разработки и т.п.

**5. Как обеспечивается приспособленность ПО к изменениям.**

- Иерархическое структурирование

- Рефакторинг кода

- Предотвращение дублирования кода

- Удобство тестирования

- Предотвращение избыточного распространения информации по программе, локализация значений входных переменных в интерфейсе

Разделение ПО на части, приводящие к иерархической структуре, позволяет разрабатывать ее составные части, модули, процедуры или объекты ПО последовательно по очереди или параллельно и независимо друг от друга. Это позволяет легко вносить дальнейшие изменения только в один из модулей, не затрагивая работу других.

**6. В чем заключается первый эвристический прием разработки ПО.**

Первый эвристический прием, позволяющий создать понятный код – описать простым русским языком то, что должна сделать программа, отложив «на потом» описание некоторых кажущихся важными деталей. Для такого описания требуется выделить главные моменты и важные концепции, скрыв или отложив объяснение деталей. Это позволит понять основную мысль и представить себе её более четко.

**7. Опасности прерывания программ в «неудобное время» и защита от них**.

Проблема целостности данных возникает тогда, когда данные, связанные в массив, таблицу или другую структуру данных передаются во время исполнения задачи. Если после начала передачи таких данных произойдет прерывание, во время отработки которого данные могут быть обновлены, то некоторые из данных структуры будут связаны со старыми значениями, а некоторые с новыми и весь набор данных может оказаться неверным.

Следующая утрата целостности данных может произойти, когда переменная используется несколько раз в последовательности операторов и её правильное окончательное значение формируется после последнего оператора. В этом случае прерывание задачи более приоритетной задачей, использующей ту же переменную, может привести к ошибке – использованию в прервавшей приоритетной задаче «недообработанного» значения переменной. Мы не можем ни предсказать, ни управлять моментом прихода прерывания.

Знание возможности возникновения таких ситуаций и правильное защитное конструирование программ позволит избегать описанных ошибок. Правильное конструирование заключается в выделение подобных критических частей кода и предотвращение прерывания задачи во время исполнения критической части кода путем установки запрета прерывания или, запрещая диспетчеризацию задач на это время. Нужно также изменить имя полностью обработанной переменной

Выполнил: Ковалев К.А.

Группа 6411

**1. Какие внутренние характеристики качества ПО – характеристики качества разработчика должны быть обеспечены при конструировании ПО.**

При конструировании программного обеспечения, кроме выполнения заданных функций должны обеспечиваться такие его внутренние свойства, которые упрощают создание ПО требуемого качества:  
1. Минимизация сложности разработки и создание легко понимаемого кода. за счет структуризации ПО - разбиения на небольшие части.

2. Безопасность многозадачности. Безопасный порядок доступа частей ПО к параллельно используемым критическим ресурсам ЦВМ.

3. «Ожидание изменений» - подготовленность к беспроблемному их проведению.

4.Устойчивость к ошибкам входных данных, устойчивость к программным ошибкам.

5.Возможность удобной проверки(тестирования) - контроле пригодность.

6. Сокрытие информации между частями ПО по принципу «нет необходимости знать» и формирование абстракций высокого уровня.

**2. Что означает и как обеспечивается свойство ПО – устойчивость к ошибкам.**

Устойчивостью к ошибкам мы будем называть свойство ПО не выдавать фатально неправильный результат и прекращать развитие ошибочной ситуации. Оно может быть реализовано, как прекращением функционирования ПО с выдачей сообщения об ошибке при её обнаружении, так и продолжением функционирования ПО с выдачей нейтрального или приближенного, но не приводящего к фатальной ситуации результата.

**3.Как обеспечивается минимизация сложности ПО.**

Главной целью конструирования ПО должна быть минимизация его сложности. На уровне архитектуры ПО сложность можно снизить, разделив ПО на небольшие несложные фрагменты. Несколько несложных фрагментов ПО понять проще, чем один сложный. В разбиении сложной задачи на несколько простых и состоит цель всех методик конструирования ПО. Причем чем более независимы подсистемы, тем проще и безопаснее разработчику сосредоточится на одной подсистеме в конкретный момент времени.

Правильно определенные составные части, модули, процедуры или объекты разделяют задачу так, чтобы их можно было разработать последовательно по очереди или параллельно и независимо. Объединения частей в пакеты должно быть также простым и обеспечивать указанные преимущества при разработке очень больших задач. Такая компенсация изначальных ограничений памяти человека позволяет писать более понятный и содержащий меньшее число ошибок код.

Для проектирования с минимальным числом связей между частями ПО необходимо использовать инкапсуляцию и сокрытие информации. Это позволит максимально облегчить интеграцию, тестирование и сопровождение ПО.

**4. Что такое рефакторинг?**

Рефа́кторинг (англ. refactoring) или реорганизация кода – процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. В основе рефакторинга лежит последовательность небольших эквивалентных (то есть сохраняющих поведение ПО) преобразований прежде всего структуры. Рефакторинг выполняется для улучшения понятности кода или изменения его структуры, для удаления «мёртвого кода», заплаток, которыми обрастает программа в процессе разработки.

**5. Как обеспечивается приспособленность ПО к изменениям.**

1) Иерархическим структурированием ПО, локализацией фрагментов кода, подверженных изменениям и развитию, в отдельные элементы;

2) Предотвращением дублирования кода;

3)Рефакторингом кода;

4)Удобством тестирования;

5) Предотвращение избыточной информации в программе, локализация значений переменных в интерфейсе;

**6. В чем заключается первый эвристический прием разработки ПО.**

Первый эвристический прием — описать простым русским языком то, что должна сделать программа, так же, как если бы пришлось объяснить это коллеге, отложив на «потом» описание некоторых кажущихся важными деталей. Это описание всегда становится более понятным, если удаётся скрыть детали путём введения дополнительно функций, имеющих понятное «говорящее» название.

**7. Опасности прерывания программ в «неудобное время» и защита от них**

Если после начала передачи таких данных произойдет прерывание, во время отработки которого данные могут быть обновлены, то некоторые из данных структуры будут связаны со старыми значениями, а некоторые с новыми и весь набор данных может оказаться неверным.

Также утрата целостности данных может произойти, когда переменная используется несколько раз в последовательности операторов и её правильное окончательное значение формируется после последнего оператора. В этом случае прерывание задачи более приоритетной задачей, использующей ту же переменную, может привести к ошибке – использованию в прервавшей приоритетной задаче «недообработанного» значения переменной.

Только знание возможности таких ситуаций и правильное защитное конструирование программ позволит избегать описанных ошибок. Правильное конструирование заключается в выделение подобных критических частей кода и предотвращение прерывания задачи во время исполнения критической части кода путем установки запрета прерывания или, запрещая диспетчеризацию задач на это время. Нужно также изменить имя полностью обработанной переменной.