

Знать:

а) модели жизненного цикла ПО: типы, какие процессы происходят на каких этапах?

5.1 Заказ-Приобретение 5.2 Поставка 5.3 Разработка 5.4 Эксплуатация 5.5 Сопровождение	Основной тип
6.1 Документирование 6.2 Управление конфигурациями 6.3 Обеспечение качества 6.4 Верификация-Валидация 6.5 Аттестация 6.6 Совместный анализ 6.7 Аудит 6.8 Решение проблем	Вспомогательный тип
7.1 Управление 7.2 Создание инфраструктуры 7.3 Усовершенствование 7.4 Обучение	Организационный тип

б) типов сцепления и связности модулей:

Сцепления:

<pre>int foo(int a, int b){ return a*b; }</pre>	1) По данным, если имеют общие простые элементы данных, которые передаются от модуля к другому как параметры
<pre>int foo(int *m, int size){ int sum =0; for(int i =0; i<size; i++) sum = sum +m[i]; return sum; }</pre>	2) По образцу, если в качестве параметров используются структуры данных(например, в качестве параметра передается массив) т.е. изменения должны отображ. и в др. модуле
<pre>int foo(int *m, int size bool quadratic= false){ int sum =0; for(int i =0; i<size; i++){ int t =n[i]; if(quadratic) t = t * t sum = sum +t; } return sum; }</pre>	3) По управлению, если какой-либо из них управляет решениями внутри другого с помощью передачи флагов, переключателей.
<pre>char * foo(bool is_time, bool is_long){ if(is_long) strcpy(s, "long"); else strcpy(s, "short"); if(is_time) strcpy(s, "time"); else strcpy(s, "distance"); return s; }</pre>	<p><i>Из модуля А вызывается функция foo модуля В. Определите тип сцепления модулей в данном случае, если переменная s объявлена на уровне модуля как char s[128], а функция foo имеет вид</i></p> <p>4) По общей области данных, модули ссылаются на одну и ту же глобальную структуру данных</p>
----скорее всего не будет----	5) По содержимому, модуль ссылается на данные (содержимое) другого модуля.

Связанности:

При функциональной связи все объекты модуля предназначены для выполнения одной функции	1) Функциональная
При последовательной связности функций модуля выход одной функции служит исходными данными для другой функции	2) Последовательная
... Так связанными считают функции, обрабатывающие одни и те же данные	3) Информационная (коммуникативная)
... Так связанными функции, которые являются частями одного процесса... отдельного элемента модуля, связаны крайне слабо	4) Процедурная
... Такая связанность функции подразумевает, что эти функции выполняются параллельно или в течение некоторого периода времени	5) Временная
... Такая связь базируется на объединении данных или функций в одну логическую группу. Вызов модули связаны по управлению	6) Логическая
... Модуль, эл-ты которого связаны так, имеет самый низкий показатель технологичности, т. к. элементы объединенные в нем, не связаны	7) Случайная

В) основ тестирования (типы тестов, документы и т. п.): виды, что делают, что такое тест сценарий, планы, чем занимаются?

Проверку отдельно взятых модулей, функций или классов выполняют с помощью ... (<u>модульного</u>) тестирования	Модульное Задача: выявлении локализованных в модуле ошибок в алгоритмах или реализации алгоритмов. Требует создания тестового окружения (заглушки). Метод «белого ящика».
Проверку связи между компонентами, а также взаимодействия с различными частями системы выполняют с помощью <u>интеграционного</u> тестирования	Интеграционное Задача: поиск дефектов, связанных с ошибками в реализации и интерпретации взаимодействия между модулями. Требует создания тестового окружения (н/р внешние системы). Метод «белого ящика».
Проверку соответствия системы установленным требованиям выполняют с помощью <u>системного</u> тестирования Проверку связи между компонентами, а также взаимодействия с различными частями системы выполняется с помощью ... (<u>системное</u>) тестирования.	Системное Задача: выявление дефектов, связанных с работой системы в целом: Метод «черного ящика».
Проверку того, что существующая ранее функциональность работает как и прежде после изменений, сделанных в приложении или окружающей среде, выполняют с помощью... (<u>функционального</u>) тестирования	Функциональное
Направлено на проверку успешной установки и настройки, обновления или удаления программного обеспечения при различном программном и аппаратном окружении, а также призванное оценить работоспособность системы после завершения работы инсталлятора.	Инсталляционное
Проверку того, что существующая ранее функциональность работает как и прежде после изменений, сделанных в приложении или окружающей среде, выполняют с помощью <u>регрессионного</u> тестирования	Регрессионное (по хронологии выполнения (3))
Формальный процесс тестирования, который проверяет соответствие системы потребностям, требованиям и бизнес процессам пользователя. Проводится для вынесения решения заказчиком (внутренним или внешним) или другим уполномоченным лицом, принимается приложение или нет.	Приемочное (по хронологии выполнения (4))

Документ, на основании которого определяются требования к компоненту или системе, и базируются тестовые сценарии называют тестовым <u>базисом</u>	Базис
Набор входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанный для определенной цели или тестового условия, таких как выполнения определенного пути программы или же для проверки соответствия определенному требованию называют тестовым ... (<u>сценарием</u>)	Сценарий
Процесс определения полноты соответствия заданных требований и созданной системы их конкретному функциональному назначению называется ... (<u>аттестация</u>)	Аттестация
Документ, описывающий цели, подходы, ресурсы и график запланированных тестовых активностей называется тестовым ... (<u>планом</u>)	План
Для защиты заголовочного файла от повторного включения можно использовать директиву #pragma ... (<u>once</u>)	once
Документ, на основании которого определяются требования к компоненту или системе, и базируются тестовые сценарии называют тестовым ... (<u>базисом</u>)	Базис
Как должен называться файл в репозитории, который содержит имена и шаблоны имен файлов, которые не должны попадать во внешний репозиторий? <u>.gitignore</u>	.gitignore
Использование модулей в программировании имеет следующие преимущества:	возможность создания библиотек функций и классов возможность разделения интерфейса и реализации возможность раздельной компиляции
Процесс внесения изменений в ПО в целях исправления ошибок, повышения производительности или адаптации к изменившимся условиям работы или требованиям называется <u>сопровождением</u>	Сопровождение

Г) основ документирования (команды Doxygen):

... служит для вставки ...Описание конкретного файла	@file
... служит для вставки ...Указывает автора	@author
... служит для вставки ...Указывает версию	@version
... служит для вставки ...Указывает дату разработки	@date
... служит для вставки ...Используемая лицензия	@copyright
... служит для вставки ...Предупреждение	@warning
... служит для вставки ... Полное описание файла	@details
В Doxygen команда ... (@brief) служит для вставки краткого описания	@brief
В Doxygen команда ... (@return) служит для вставки описания возвращения значения функцией или методом	@return
В Doxygen команда @throw служит для вставки описания исключения, возбуждаемого функцией или методом.	@throw
В Doxygen команда @param служит для вставки описания параметра функции или метода	@param
...служит для вставки...Перечисления известных ошибок	@bug

Уметь:

- а) работать с git (команды): необходимо будет написать команду для выполнения той или иной операции.

Что делает?	Команда
Инициализация нового репозитория осуществляется командой git ... (init)	git init
Добавление новых файлов в индекс репозитория осуществляется командой git ... (add)	git add hello.txt git add . (тут пробел перед точкой)
Сохранить изменения в репозитории необходимо командой git ... (commit) //Флажок -m задаст commit message - комментарий разработчика	git commit -m 'Add some code'
Клонирование репозитория осуществляется командой git ... (clone)	\$ git ... clone https://github.com/ ... /git
Для передачи изменений локального репозиторий используется команда git ... (push)	\$ git push origin master
Запрос обновления локального репозитория из внешнего репозитория осуществляется командой git ... (pull)	git pull origin master
Запрос обновления локального репозитория из внешнего репозитория осуществляется командой	\$ git ... (update)
Отследить интересующие вас операции в списке изменений, можно по хэшу коммита, при помощи команды	git show hash_commit
Ну а если вдруг нам нужно переделать commit message и внести туда новый комментарий	git branch -M main
status — это еще одна важная команда, которая показывает информацию о текущем состоянии репозитория	\$ git status

- б) работать с make (конфигурационные файлы): в файле будет пропуск необходимо его найти.

Построение простого Makefile

То есть, правило make это ответы на следующие вопросы:

{Из чего делаем?(реквизиты)} → [как делаем?(команды)] → что делаем?(цели)}

По умолчанию make станет выполнять самое первое правило, если цель выполнения не была явно указана при вызове: \$ make <цель>

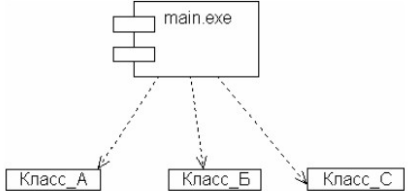
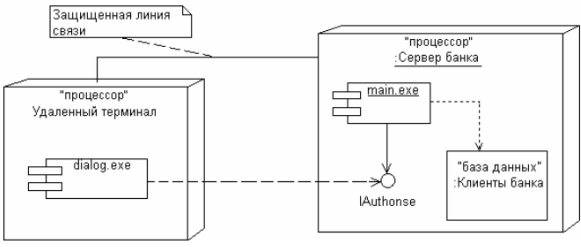

all — является стандартной целью по умолчанию. При вызове make ее можно явно не указывать. clean — очистить каталог от всех файлов полученных в результате компиляции. install — произвести инсталляцию uninstall — и деинсталляцию соответственно.

Для того чтобы make не искал файлы с такими именами, их следует определить в Makefile, при помощи директивы .PHONY.

– \$@ - имя цели; – \$< - имя первой зависимости – \$? - имена всех зависимостей, которые новее чем цель; – \$^ - имена всех зависимостей цели

в) работать с UML (типы диаграмм)

	<p>диаграмме вариантов использования (Use case diagram)</p>
	<p>диаграмме классов (Class diagram)</p>
	<p>Диаграммы последовательностей (Sequence diagram)</p>
	<p>диаграмма состояний (Statechart diagrams)</p>
	<p>Диаграмма деятельности (Activity diagram)</p>

 <p>Графическое изображение зависимости между компонентом и классами</p>	<p>Диаграмма компонентов (Component diagram)</p>
	<p>диаграмме развертывания (Deployment diagram)</p>
	<p>Диаграмма кооперации (collaboration diagram)</p>

Владеть:

- а) разрабатывать тестовые сценарии модульный
- б) разрабатывать тестовые сценарии системный