Консультация к отдельным вопросам гос. экз. бакалавриата ВМК в 2019 году

4 курс ПМИ (основная часть)

Вопрос 16. Формализация понятия алгоритма. Машины Тьюринга, нормальные алгоритмы Маркова. Алгоритмическая неразрешимость. Задача останова. Задача самоприменимости.

См. Боресков и др. Методические материалы для подготовки к государственному экзамену по прикладной математике и информатике. – М.: Издательский отдел ВМК МГУ 2004, 294 с. http://sp.cs.msu.ru/info/gos.djvu

Корухова Л.С., Шура-Бура М.Р. Введение в алгоритмы. Учебное пособие для студентов I курса. – М.: Изд. отдел ф-та ВМК МГУ, 2010 – [http://sp.cs.msu.ru/info/1/vvedalg.pdf]

Вопрос 21. Базы данных.Основные понятия реляционной модели данных. Реляционная алгебра. Средства языка запросов SQL.

План ответа по вопросу

Перечислите и определите основные понятия реляционных баз данных: тип данных, домен, атрибут, кортеж, отношение, первичный ключ. Укажите фундаментальные свойства отношений: отсутствие кортежей-дубликатов, первичный и возможные ключи, отсутствие упорядоченности кортежей и атрибутов, атомарность значений атрибутов, первая нормальная форма отношения.

Алгебра Кодда. Перечислите и определите теоретико-можественные операции: объединение, пересечение, взятие разности, взятие декартова произведения. То же сделайте для специальных операций: ограничение, проекция, соединение, деление. Дайте пояснения по совместимости операндов.

Структура языка SQL, типы данных SQL. Перечислите и поясните на примерах средства определения, изменения определения и отмены определения доменов, базовых таблиц, ограничений целостности. Дайте общую характеристику оператора SELECT, опишите организация списка ссылок на таблицы в разделе FROM, предикаты раздела WHERE, группировка и условия раздела HAVING, порождаемые и соединенные таблицы. Желательно сопроводить всё это примерами. Перечислите и поясните на примерах средства манипулирования данными (INSERT, UPDATE, DELETE)

См.:
http://citforum.ru/database/advanced_intro/10.shtml#3
http://citforum.ru/database/advanced_intro/13.shtml#4
http://citforum.ru/database/advanced_intro/46.shtml#15
http://citforum.ru/database/advanced_intro/55.shtml#17
http://citforum.ru/database/advanced_intro/58.shtml#18
http://citforum.ru/database/advanced_intro/72.shtml#21

4 курс ПМИ (дополнительная часть, III поток)

Вопрос 8. Зависимости в реляционных отношениях: функциональные, многозначные, проекции/соединения. Проектирование реляционных БД на основе принципов нормализации отношений. Нормальные формы.

План ответа по вопросу

Дайте определение функциональной зависимости. Опишите декомпозицию без потерь и функциональные зависимости. Сформулируйте теорему Хита. Расскажите о диаграммах функциональных зависимостей. Сопроводите свой ответ примерами.

Расскажите о минимальной функциональной зависимости и второй нормальной форме.
Расскажите о нетранзитивной функциональной зависимости и третьей нормальной форме.
Расскажите о перекрывающихся возможных ключах и нормальной форме Бойса-Кодда.
Сопроводите свой ответ примерами.

Дайте определение многозначной зависимости. Сформулируйте теорему Фейджина.
Расскажите о многозначных зависимостях и четвертой нормальной форме. Сопроводите свой ответ примерами.

Дайте определение зависимости проекции/соединения. Расскажите о зависимости проекции/соединения и пятой нормальной форме. Сопроводите свой ответ примерами.

Приведите примеры аномалий обновления.
См.:
http://citforum.ru/database/advanced_intro/19.shtml#7
http://citforum.ru/database/advanced_intro/22.shtml#8
http://citforum.ru/database/advanced_intro/24.shtml#9

Вопрос 12. Классификация языков, определяемых конечными автоматами, регулярными выражениями и праволинейными грамматиками. Эквивалентность и минимизация конечных автоматов.
Материал для подготовки в книге «Теория и реализация языков программирования». Авторы: Серебряков В. А., Галочкин М. П., Гончар Д. Р., Фуругян М. Г. http://sp.cs.msu.ru/info/trlp.pdf

Вопрос 13. Функции FIRST и FOLLOW. LL(l)-грамматики. Конструирование таблицы предсказывающего анализатора .
Материал для подготовки в книге Серебряков В. А., Галочкин М. П., Гончар Д. Р., Фуругян М. Г. «Теория и реализация языков программирования». http://sp.cs.msu.ru/info/trlp.pdf

Вопрос 14. Жизненный цикл программного обеспечения (ПО). Основные виды деятельности при разработке ПО. Каскадная и итерационная модели жизненного цикла.
Жизненный цикл ПО — интервал времени от появления идеи о создании этого ПО до вывода из эксплуатации последнего экземпляра.
Цикл — в связи с итеративным повторением некоторых видов деятельности: сбор и анализ требований (исходных или для изменений) – проектирование (исходное или вносимых изменений) – реализация – верификация/тестирование – ввод в эксплуатацию/развертывание.
Более 75% трудоемкости в ЖЦ сложного практически значимого ПО падает на сопровождение — исправление дефектов и внесение изменений.
Основные виды деятельности (в деталях могут быть различия в различных стандартах и описаниях процессов разработки)

- Сбор и анализ требований (анализ предметной области, выделение требований, систематизация требований, анализ области решений)
- Проектирование (проектирование архитектуры, детальное проектирование компонентов)
- Реализация (кодирование, построение ПО, отладка и исправление дефектов)
- Контроль качества (валидация и верификация, экспертизы, тестирование)
- Документирование (разработка проектной, пользовательской, эксплуатационной документации)
- Развертывание (установка, ввод в эксплуатацию, приемка)
- Управленческие виды деятельности (планирование, мониторинг и контроль работ, управление конфигурациями, управление персоналом, управление технологиями, управление рисками, информационное обеспечение)

- Поддержка разработки (создание и сопровождение репозиториев, переиспользуемых компонентов, средств разработки)

Один из основных современных стандартов на процесс разработки — ISO 12207.
Модель ЖЦ — общая схема организации и взаимосвязи работ.
Каскадная модель — виды деятельности выстроены в последовательность, от сбора требований до развертывания, каждый вид деятельности выполняется на некотором этапе проекта, пока не будет сделано все, что нужно, в требуемом качестве, возвращаться к предыдущим этапам нельзя.
Итеративная модель — этапы проекта формируют итерации, на каждой итерации происходит выполнение некоторой последовательности работ до достижения цели данной итерации, обычно в виде разработки значимой части ПО или создания значимой части проектных решений для дальнейшей реализации, по окончании очередной итерации можно вернуться к пересмотру или доработке ее решений в следующей.
Практически все сложные проекты — итеративные. Каскадная схема работоспособна для небольших систем, создаваемых за короткий период времени, во время которого не возникают значимые изменения в требованиях к создаваемым системам.
Разновидность итеративной модели — спиральная — предполагает одинаковую структуру итераций, на каждой итерации работы выстроены по практически одному и тому же сценарию.
В итеративной модели ЖЦ количество итераций устанавливается заранее при планировании ЖЦ. Этим она отличается от эволюционной модели ЖЦ, в которой вместо итераций -- периоды, завершающиеся выпусками новых версий ПО, количество которых не определено заранее. Эволюционная не относится к итеративным.
Материалы для подготовки: конспект 2-й лекции по курсу «Основы программной инженерии» http://se-course.narod.ru/Lecture02.pdf; книга Иан Соммервил. Инженерия программного обеспечения. 6-е издание. М. – СПб. – Киев: 2002. – 623 с.

Вопрос 15. Качество программного обеспечения и методы его контроля. Тестирование и другие методы верификации.

Качество ПО — его пригодность для решения тех задач, для которых оно предназначено и/или применяется на практике. Круг задач обычно четко не описан, пользователи часто придумывают новые способы как-то применить ПО. Поэтому для описания качества используют схему из набора характеристик, комбинации которых можно использовать для оценки пригодности для очередной задачи.
Современная схема — стандарт ISO 25010, выделяются следующие характеристики.

- Функциональность — какие конкретно задачи и с какими характеристиками решений ПО умеет решать.
- Надежность — насколько устойчиво ПО работает в разных условиях, насколько быстро и корректно восстанавливает работу и данные при сбоях и перебоях в работе, вызванных внешними факторами.
- Защищенность — насколько ПО предотвращает доступ и вмешательство в свою работу и данные со стороны лиц и программ, которые такого доступа должны быть лишены.
- Совместимость — насколько ПО работоспособно при использовании данных других версий или от посторонних систем, и одновременной работе других программ в той же среде.
- Переносимость — насколько ПО работоспособно без пересборки (но, возможно, с изменением каких-то конфигурационных файлов) в разных средах.
- Производительность — сколько ПО использует ресурсов разного рода (процессор, разные виды памяти, сети и пр.) при решении определенных задач.
- Удобство использования — насколько быстро и эффективно пользователи обучаются работать с ПО и могут решать определенные задачи с его помощью.
- Удобство сопровождения — насколько быстро и эффективно разработки могут анализировать ПО, находить и исправлять в нем дефекты, вносить изменения.

Обеспечение качества — предотвращение, поиск и исправление ошибок. Поиск ошибок, он же контроль качества, разделяется на валидацию и верификацию.
Валидация — поиск ошибок, опирающийся на понимание разработчиков, пользователей и экспертов того, как ПО должно работать и как должны выглядеть проектные решения.
Верификация — поиск ошибок, опирающийся на выявление несоответствий между двумя или более артефактами разработки (документированными требованиями, проектными решениями, моделями работы, прототипами, кодом, тестами и пр.) и/или реальной работой ПО.
Методы верификации.
• Экспертиза — привлечение людей (разработчиков и экспертов) для выявления несоответствий.
Инспекция кода или проектных документов, экспертиза защищенности или удобства использования, экспертные оценки надежности и пр.
• Статические методы — автоматизированный поиск ошибок по некоторым шаблонам и правилам без выполнения ПО.
Применение разнообразных инструментов, наиболее эффективные техники анализа включаются в семантику языков и компиляторы.
• Динамические методы — поиск ошибок в результатах реальной работы ПО (или в рамках некоторой симуляции).
Мониторинг, профилирование, тестирование.
• Формальные методы — формализация проверяемых артефактов в виде математических моделей (спецификации требований и реализации) и формальная проверка определенного математического соответствия между ними.
Дедуктивная верификация — модели логико-алгебраические, соответствие означает выводимость спецификации из реализации. Проверка моделей — обычно спецификация логико-алгебраическая, реализация исполнимая или автоматная, соответствие означает выполнимость спецификации на реализации.
Тестирование — проверка соответствия реальной работы системы требованиям на заранее выбранном наборе ситуаций. Наиболее широко используемый метод контроля качества. Основная проблема — выбор небольшого набора ситуаций, достаточного, чтобы более-менее адекватно судить о поведении системы в целом. Выбранные ситуации преобразуются в тесты — каждый тест содержит способ формирования нужной ситуации и проверку корректности работы системы в ней. Выделяются виды тестирования
• По проверяемым свойствам (тестирование функциональности, надежности, производительности, и т.д., на соответствие, сертификационное, регрессионное)
• По источнику данных для тестов (тестирование черного ящика, структурное тестирование, тестирование на отказ, включая тестирование работоспособности, нагрузочное и стрессовое)
• По уровню проверяемой части системы (модульное, компонентное, интеграционное, тестирование характеристик, системное)
• По исполнителю (тестирование при разработке, альфа, бета, независимое, приемочное).
Материалы для подготовки: лекция 5 по курсу «Основы программной инженерии» http://se-course.narod.ru/Lecture05.pdf исключая описания стандартов ISO 9000, 9001, 9003, 9004, 90003 и раздел об ошибках. Обзор методов верификации http://www.ict.edu.ru/ft/005645/62322e1-st09.pdf, разделы 1, начало раздела 3 (до 3.1), разделы 3.2, 3.3.5, 3.3.6, начало 3.4 (до 3.4.1) и 3.4.2-3.4.5.

Вопрос 22. Синхронизация в распределенных системах. Синхронизация времени. Логические часы. Выборы координатора. Взаимное исключение. Координация процессов.
При подготовке ответа следует воспользоваться текстами лекций: В.А. Крюков, В.А. Бахтин. «Распределенные системы» [http://sp.cs.msu.su/courses/os/distr-systems-2018.zip]. Файл 3-4-MPI-Sync.doc Раздел 4

Вопрос 23. Отказоустойчивость в распределенных системах. Типы отказов. Фиксация контрольных точек и восстановление после отказа. Репликация и протоколы голосования. Надежная групповая рассылка.
При подготовке ответа следует воспользоваться текстами лекций: В.А. Крюков, В.А. Бахтин. «Распределенные системы» [http://sp.cs.msu.su/courses/os/distr-systems-2018.zip]. Файл 7-Fault_T.doc Раздел 7

Вопрос 24. Распределенные файловые системы. Доступ к директориям и файлам. Семантика одновременного доступа к одному файлу нескольких процессов. Кэширование и размножение файлов.
При подготовке ответа следует воспользоваться текстами лекций: В.А. Крюков, В.А. Бахтин. «Распределенные системы» [http://sp.cs.msu.su/courses/os/distr-systems-2018.zip]. Файл 5-DistrFS.doc Раздел 5

Вопрос 25. Промежуточные представления программы: абстрактное синтаксическое дерево; последовательность трехадресных инструкций. Базовые блоки и граф потока управления.
[...]

Вопрос 26. Локальная оптимизация при компиляции программы. Ориентированный ациклический граф и метод нумерации значений.
[...]

Вопрос 27. Глобальная оптимизация при компиляции программы. Построение передаточных функций базовых блоков. Монотонные и дистрибутивные передаточные функции. Метод неподвижной точки и его применение для нахождения достигающих определений.
[...]

4 курс ФИИТ (основная часть)

Вопрос 18. Понятие алгоритма. Нормальные алгоритмы Маркова. Алгоритмически неразрешимые проблемы.
См. Боресков и др. Методические материалы для подготовки к государственному экзамену по прикладной математике и информатике. – М.: Издательский отдел ВМК МГУ 2004, 294 с. http://sp.cs.msu.ru/info/gos.djvu
Корухова Л.С., Шура-Бура М.Р. Введение в алгоритмы. Учебное пособие для студентов I курса. – М.: Изд. отдел ф-та ВМК МГУ, 2010 – [http://sp.cs.msu.ru/info/1/vvedalg.pdf]

Вопрос 27. Базы данных.Основные понятия реляционной модели данных. Реляционная алгебра. Средства языка запросов SQL.
План ответа по вопросу
Перечислите и определите основные понятия реляционных баз данных: тип данных, домен, атрибут, кортеж, отношение, первичный ключ. Укажите фундаментальные свойства отношений: отсутствие кортежей-дубликатов, первичный и возможные ключи, отсутствие упорядоченности кортежей и атрибутов, атомарность значений атрибутов, первая нормальная форма отношения.

Алгебра Кодда. Перечислите и определите теоретико-можественные операции: объединение, пересечение, взятие разности, взятие декартова произведения. То же сделайте для специальных операций: ограничение, проекция, соединение, деление. Дайте пояснения по совместимости операндов.

Структура языка SQL, типы данных SQL. Перечислите и поясните на примерах средства определения, изменения определения и отмены определения доменов, базовых таблиц, ограничений целостности. Дайте общую характеристику оператора SELECT, опишите организация списка ссылок на таблицы в разделе FROM, предикаты раздела WHERE, группировка и условия раздела HAVING, порождаемые и соединенные таблицы. Желательно сопроводить всё это примерами. Перечислите и поясните на примерах средства манипулирования данными (INSERT, UPDATE, DELETE)
См.:
http://citforum.ru/database/advanced_intro/10.shtml#3
http://citforum.ru/database/advanced_intro/13.shtml#4
http://citforum.ru/database/advanced_intro/46.shtml#15
http://citforum.ru/database/advanced_intro/55.shtml#17
http://citforum.ru/database/advanced_intro/58.shtml#18
http://citforum.ru/database/advanced_intro/72.shtml#21

4 курс ФИИТ (дополнительная часть)

Вопрос 15. Использование разных типов грамматик (по Хомскому) в реализации современных систем программирования.
При подготовке может быть полезна книга Серебряков В. А., Галочкин М. П., Гончар Д. Р., Фуругян М. Г. «Теория и реализация языков программирования». http://sp.cs.msu.ru/info/trlp.pdf

Вопрос 19. Жизненный цикл программного обеспечения (ПО). Основные виды деятельности при разработке ПО. Каскадная и итерационная модели жизненного цикла.
Жизненный цикл ПО — интервал времени от появления идеи о создании этого ПО до вывода из эксплуатации последнего экземпляра.
Цикл — в связи с итеративным повторением некоторых видов деятельности: сбор и анализ требований (исходных или для изменений) – проектирование (исходное или вносимых изменений) – реализация – верификация/тестирование – ввод в эксплуатацию/развертывание.
Более 75% трудоемкости в ЖЦ сложного практически значимого ПО падает на сопровождение — исправление дефектов и внесение изменений.
Основные виды деятельности (в деталях могут быть различия в различных стандартах и описаниях процессов разработки)

- Сбор и анализ требований (анализ предметной области, выделение требований, систематизация требований, анализ области решений)
- Проектирование (проектирование архитектуры, детальное проектирование компонентов)
- Реализация (кодирование, построение ПО, отладка и исправление дефектов)
- Контроль качества (валидация и верификация, экспертизы, тестирование)
- Документирование (разработка проектной, пользовательской, эксплуатационной документации)
- Развертывание (установка, ввод в эксплуатацию, приемка)
- Управленческие виды деятельности (планирование, мониторинг и контроль работ, управление конфигурациями, управление персоналом, управление технологиями, управление рисками, информационное обеспечение)
- Поддержка разработки (создание и сопровождение репозиториев, переиспользуемых компонентов, средств разработки)

Один из основных современных стандартов на процесс разработки — ISO 12207.
Модель ЖЦ — общая схема организации и взаимосвязи работ.

Каскадная модель — виды деятельности выстроены в последовательность, от сбора требований до развертывания, каждый вид деятельности выполняется на некотором этапе проекта, пока не будет сделано все, что нужно, в требуемом качестве, возвращаться к предыдущим этапам нельзя.
Итеративная модель — этапы проекта формируют итерации, на каждой итерации происходит выполнение некоторой последовательности работ до достижения цели данной итерации, обычно в виде разработки значимой части ПО или создания значимой части проектных решений для дальнейшей реализации, по окончании очередной итерации можно вернуться к пересмотру или доработке ее решений в следующей.
Практически все сложные проекты — итеративные. Каскадная схема работоспособна для небольших систем, создаваемых за короткий период времени, во время которого не возникают значимые изменения в требованиях к создаваемым системам.
Разновидность итеративной модели — спиральная — предполагает одинаковую структуру итераций, на каждой итерации работы выстроены по практически одному и тому же сценарию.
В итеративной модели ЖЦ количество итераций устанавливается заранее при планировании ЖЦ. Этим она отличается от эволюционной модели ЖЦ, в которой вместо итераций -- периоды, завершающиеся выпусками новых версий ПО, количество которых не определено заранее. Эволюционная не относится к итеративным.
Материалы для подготовки: конспект 2-й лекции по курсу «Основы программной инженерии» http://se-course.narod.ru/Lecture02.pdf; книга Иан Соммервил. Инженерия программного обеспечения. 6-е издание. М. – СПб. – Киев: 2002. – 623 с.

Вопрос 20. Архитектура программного обеспечения, методы ее описания (язык UML) и анализа. Архитектурные образцы.

При ответе следует дать определение понятию архитектура ПО, перечислить основные виды UML-диаграмм (д. вариантов использования, д. классов, д. объектов, д. компонентов, д. развертывания, д. последовательности, д. деятельности, д. состояний), кратко рассказать, зачем они нужны, какие аспекты архитектуры изображаются на диаграммах каждого вида. Следует привести пример UML-диаграммы какого-либо вида и дать пояснение по элементам и связям на ней. Следует перечислить четыре вида архитектурных образцов (конвейер, хранилище, вызов-возврат и интерактивная система), а также может рассказать про основные характеристики (структурную организацию и типичные сценарии взаимодействия) примеров образцов каждого вида (например, "каналы и фильтры" из конвейерных, "репозитория" из основанных на хранилище, "многоуровневой системы" из вызов-возвратных и Model-View-Controller из интерактивных). Рассказать, чем они отличаются. Желательно уметь приводить примеры известных систем, построенных по различным образцам.
Материал для подготовки в конспектах 7-й и 8-й лекций по курсу «Основы программной инженерии» http://se-course.narod.ru/Lecture07.pdf, раздел про архитектурные стили и далее,
http://se-course.narod.ru/Lecture08.pdf, раздел про MVC (там - "данные-представление-обработка"). Также полезна глава 2 книги F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. Pattern-Oriented Software Architecture. A System of Patterns. Wiley, 2002 https://wiki.sch.bme.hu/images/9/98/Sznikak_jegyzet_Pattern-Oriented-SA_vol1.pdf и статья
David Garlan, Mary Shaw. An Introduction to Software Architecture. CMU-CS-94-166.
или в сборнике Advances in Software Engineering and Knowledge Engineering, Volume I, eds V.Ambriola and G.Tortora, 1993. http://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/intro_softarch.pdf
Справочником по UML является книга Фаулер М. UML. Основы. 3-е издание.: Пер. с англ. – СПБ: Символ-Плюс, 2004. - 192 с. Может быть полезно англоязычное пособие: http://www.uml-diagrams.org

Вопрос 21. Качество программного обеспечения и методы его контроля. Тестирование и другие методы верификации.

Качество ПО — его пригодность для решения тех задач, для которых оно предназначено и/или применяется на практике. Круг задач обычно четко не описан, пользователи часто придумывают новые способы как-то применить ПО. Поэтому для описания качества используют схему из набора характеристик, комбинации которых можно использовать для оценки пригодности для очередной задачи.

Современная схема — стандарт ISO 25010, выделяются следующие характеристики.

- Функциональность — какие конкретно задачи и с какими характеристиками решений ПО умеет решать.
- Надежность — насколько устойчиво ПО работает в разных условиях, насколько быстро и корректно восстанавливает работу и данные при сбоях и перебоях в работе, вызванных внешними факторами.
- Защищенность — насколько ПО предотвращает доступ и вмешательство в свою работу и данные со стороны лиц и программ, которые такого доступа должны быть лишены.
- Совместимость — насколько ПО работоспособно при использовании данных других версий или от посторонних систем, и одновременной работе других программ в той же среде.
- Переносимость — насколько ПО работоспособно без пересборки (но, возможно, с изменением каких-то конфигурационных файлов) в разных средах.
- Производительность — сколько ПО использует ресурсов разного рода (процессор, разные виды памяти, сети и пр.) при решении определенных задач.
- Удобство использования — насколько быстро и эффективно пользователи обучаются работать с ПО и могут решать определенные задачи с его помощью.
- Удобство сопровождения — насколько быстро и эффективно разработки могут анализировать ПО, находить и исправлять в нем дефекты, вносить изменения.

Обеспечение качества — предотвращение, поиск и исправление ошибок. Поиск ошибок, он же контроль качества, разделяется на валидацию и верификацию.

Валидация — поиск ошибок, опирающийся на понимание разработчиков, пользователей и экспертов того, как ПО должно работать и как должны выглядеть проектные решения.

Верификация — поиск ошибок, опирающийся на выявление несоответствий между двумя или более артефактами разработки (документированными требованиями, проектными решениями, моделями работы, прототипами, кодом, тестами и пр.) и/или реальной работой ПО.

Методы верификации.

- Экспертиза — привлечение людей (разработчиков и экспертов) для выявления несоответствий.

Инспекция кода или проектных документов, экспертиза защищенности или удобства использования, экспертные оценки надежности и пр.

- Статические методы — автоматизированный поиск ошибок по некоторым шаблонам и правилам без выполнения ПО.

Применение разнообразных инструментов, наиболее эффективные техники анализа включаются в семантику языков и компиляторы.

- Динамические методы — поиск ошибок в результатах реальной работы ПО (или в рамках некоторой симуляции).

Мониторинг, профилирование, тестирование.

- Формальные методы — формализация проверяемых артефактов в виде математических моделей (спецификации требований и реализации) и формальная проверка определенного математического соответствия между ними.

Дедуктивная верификация — модели логико-алгебраические, соответствие означает выводимость спецификации из реализации. Проверка моделей — обычно спецификация логико-алгебраическая, реализация исполнимая или автоматная, соответствие означает выполнимость спецификации на реализации.

Тестирование — проверка соответствия реальной работы системы требованиям на заранее выбранном наборе ситуаций. Наиболее широко используемый метод контроля качества. Основная проблема — выбор небольшого набора ситуаций, достаточного, чтобы более-менее

адекватно судить о поведении системы в целом. Выбранные ситуации преобразуются в тесты — каждый тест содержит способ формирования нужной ситуации и проверку корректности работы системы в ней. Выделяются виды тестирования

- По проверяемым свойствам (тестирование функциональности, надежности, производительности, и т.д., на соответствие, сертификационное, регрессионное)
- По источнику данных для тестов (тестирование черного ящика, структурное тестирование, тестирование на отказ, включая тестирование работоспособности, нагрузочное и стрессовое)
- По уровню проверяемой части системы (модульное, компонентное, интеграционное, тестирование характеристик, системное)
- По исполнителю (тестирование при разработке, альфа, бета, независимое, приемочное).

Материалы для подготовки: лекция 5 по курсу «Основы программной инженерии» http://se-course.narod.ru/Lecture05.pdf исключая описания стандартов ISO 9000, 9001, 9003, 9004, 90003 и раздел об ошибках. Обзор методов верификации http://www.ict.edu.ru/ft/005645/62322e1-st09.pdf, разделы 1, начало раздела 3 (до 3.1), разделы 3.2, 3.3.5, 3.3.6, начало 3.4 (до 3.4.1) и 3.4.2-3.4.5.