МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (СГАУ)

Современные методы и технологии тестирования программного обеспечения

Электронный учебно-методический комплекс
по дисциплине в LMS Moodle

Работа выполнена по мероприятию блока 1 «Совершенствование образовательной деятельности» Программы развития СГАУ на 2009 – 2018 годы по проекту «Разработка магистерской программы «Программное обеспечение мобильных устройств» по направлению 230100.68 – Информатика и вычислительная техника»
Соглашение № 1/12 от 3.06.2013 г.

САМАРА
2013

УДК 004.415
С 568

Автор-составитель: **Климентьев Константин Евгеньевич**

Современные методы и технологи тестирования программного обеспечения

[Электронный ресурс] : электрон. учеб.-метод. комплекс по дисциплине в LMS Moodle / Мин-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т); авт.-сост. К.Е. Климентьев. - Электрон. текстовые и граф. дан. - Самара, 2013. – 1 эл. опт. диск (CD-ROM).

В состав электронного учебно-методического комплекса входят:
1. СМИТТ ПО.Курс лекций.pdf
2. СМИТТ ПО.Методические указания и задания к лабораторным работам.pdf
3. СМИТТ ПО.Тесты к зачету.pdf
4. Рабочая программа - Современные методы и технологии тестирования программного обеспечения.pdf

УМКД «Современные методы и технологии тестирования программного обеспечения» предназначен для студентов факультета информатики, обучающихся по направлению подготовки магистров 230100.68 «Информатика и вычислительная техника» во 2-м семестре.

УМКД разработан на кафедре Информационных систем и технологий.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)

М1.В.ДВ.2.2.
Современные методы и технологии тестирования программного обеспечения (курс лекций)

Составитель: к.т.н., доц. К.Е. Климентьев

Самара 2013

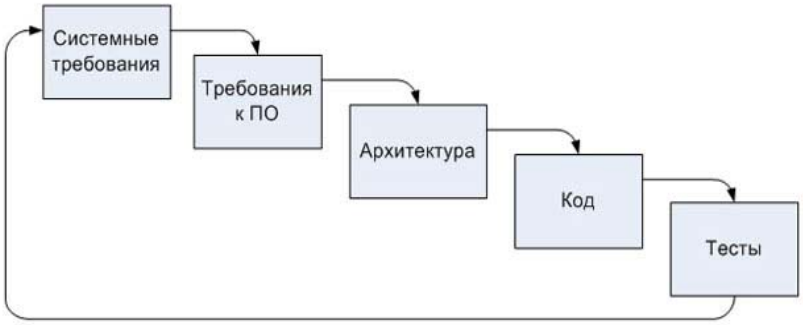
Жизненный цикл программы

В нашей стране жизненный цикл разработки ПО установлен стандартом ГОСТ 19.102-77:

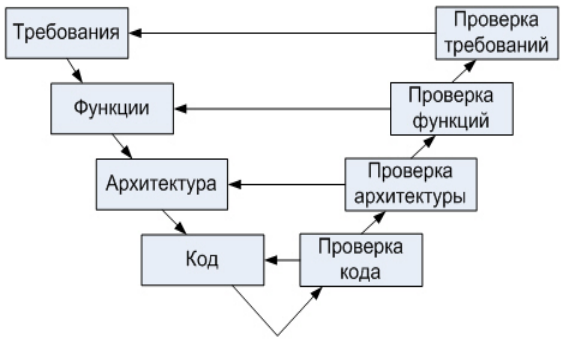
- Техническое задание (ТЗ).
- Эскизный проект (ЭП).
- Технический проект (ТП).
- Рабочий проект (РП).
- Внедрение.

На самом деле, возможны разные модели:

1. Каскадный жизненный цикл - основан на постепенном увеличении степени детализации описания всей разрабатываемой системы.



2. V-образный жизненный цикл описывает процессы двух видов - основные процессы разработки и процессы верификации.



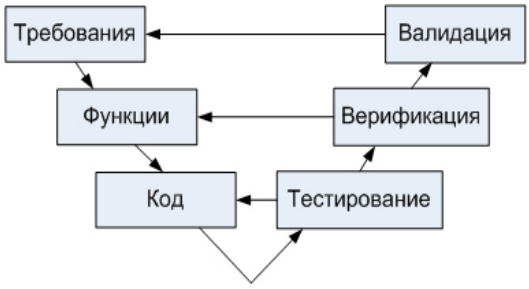
3. Спиральный жизненный цикл использует предположение о том, что на каждом этапе разрабатывается очередное полное описание системы.

4. Экстремальное программирование предусматривает быстрые жизненные циклы разработки, например (например, XP - eXtreme

Programming). Основная идея – тестирование выполняется заказчиком в процессе всей разработки.

Тестирование, верификация и валидация - различия в понятиях.

Тестирование - это управляемое выполнение программы с целью обнаружения несоответствий ее поведения и требований.



Верификация программного обеспечения - достижение гарантии того, что верифицируемый объект (требования или программный код) соответствует требованиям. Тестирование – часть верификации.

Валидация программной системы - доказательство соответствия системы ожиданиям заказчика.

В рамках ISO 9126 предложена модель качества, состоящая из 6 факторов:

1. Функциональность
2. Надежность (включая защищенность от внутренних ошибок и угроз)
3. Удобство использования
4. Удобство сопровождения
5. Производительность
6. Переносимость

Тестирование - проверка соответствия поведения программной системы требованиям, выполняемая по результатам реальной работы этой системы в некотором конечном наборе специально созданных ситуаций.

Нормативная документация по тестированию: Testing IEEE 1008-87 Standard forSoftware Unit.

Классификация по объекту тестирования:

- Модульное тестирование - позволяет проверить функционирование отдельно взятого элемента;
- Интеграционное тестирование - процесс проверки взаимодействия между программными компонентами/модулями;
- Системное тестирование охватывает целиком всю систему.

Классификация по целям тестирования:

- приёмочное тестирование - проверяет поведение системы на предмет удовлетворения требований заказчика;
- установочное тестирование - проводится с целью проверки процедуры инсталляции

- системы в целевом окружении;
- альфа (внутреннее пробное использование) и бета (пробное использование с привлечением внешних пользователей) версий.
- функциональные тесты/тесты соответствия – проверка соответствия системы поведенческим спецификациям;
- регрессионное тестирование - повторное выборочное тестирование системы или компонентов для проверки качества сделанных модификаций.
- тестирование производительности;
- нагрузочное (стрессовое) тестирование
- сравнительное тестирование - набор тестов, позволяющих сравнить две версии системы.
- восстановительное тестирование – проверка возможностей рестарта системы в случае ошибок;
- конфигурационное тестирование - проверка поведения и работоспособности системы в различных конфигурациях;
- тестирование удобства и простоты использования;
- тестирование, совмещенное с разработкой;
- тестирование безопасности (поиск уязвимостей).

Два класса методов тестирования:

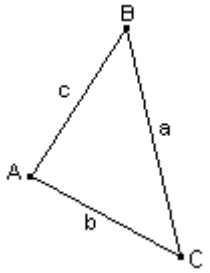
- Метод «черного ящика», когда известны только входная и выходная мпецификации;
- Метод «белого (стеклянного) ящика», когда известен алгоритм.

Тестирование МЧЯ требует полное покрытия входных данных, что почти никогда невозможно.

Тестирование МЧЯ - основные идеи:

- Эквивалентное разбиение;
- Анализ граничных значений;
- Функциональные диаграммы;
- Предположение об ошибке.

Пример тестирования МЧЯ (по Майерсу)



Правильные данные:

- Неравносторонний треугольник (2, 4, 5)

- Равнобедренный треугольник (2, 2, 3)
- Равносторонний треугольник (3, 3, 3)
- Всевозможные перестановки.

Неправильные данные:

- 1.2.1 Нарушение «суммы длин» (1, 2, 4)
- 1.2.2 Вырождение в прямую (1, 2, 3)
- 1.2.4 Вырождение в точку (0, 0, 0)
- 1.2.5. Хотя бы один 0 (0, 4, 5)
- 1.2.6. Хотя бы одно отрицательное значение (-1, 2, 4)
- 1.2.7. Совместно 0 и отрицательное значение (-1, 0, 4)
- 1.2.8. Всевозможные перестановки 1.2.1, 1.2.2, 1.2.5, 1.2.6 и 1.2.7.
- 1.2.9. Неполное или избыточное количество данных.

Для сложных программ множество входных данных велико или бесконечно. Применяется «фаззинг» - тестирование случайными наборами данных. Методы решения:

- метод эквивалентного алгоритма, который дает заведомо правильные ответы;

- метод разбиения на классы эквивалентности, например, всего на два: «заведомо неправильные» и «заведомо правильные».

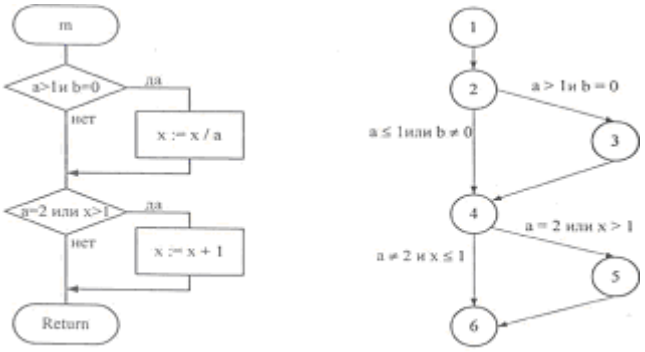
Если приходится вручную, то: проверка граничных значений.

Тестирование МБЯ, основные идеи:

- Покрытие операторов (команд)
- Покрытие решений
- Покрытие условий
- Покрытие маршрутов (решений+условий).

Тестирование МБЯ требует полного покрытия кода программы, что почти всегда возможно.

Пример тестирования МБЯ (по Майерсу).



Покрытие операторов. Критерий покрытия операторов подразумевает такой подбор тестов, чтобы каждый оператор программы выполнялся, по крайней мере, один раз. Это необходимое, но недостаточное условие для приемлемого тестирования. Получается дерево.

Можно было бы выполнить каждый оператор один раз, задав в качестве входных данных a = 2, b = O, x = 3. Но при этом из второго условия следует, что переменная x может принимать любое значение, и в некоторых версиях языка Pascal это значение проверяться не будет. Кроме того:

- если при написании программы в первом условии указано: (a > 1) or (b = 0), то ошибка обнаружена не будет;
- если во втором условии вместо x > 1 записано x > 0, то эта ошибка тоже не будет обнаружена;
- существует путь 1-2-4-6 (см. рис. 9.2, б), в котором x вообще не меняется и, если здесь есть ошибка, она не будет обнаружена.

Таким образом, для проверки программы этого явно недостаточно.

В данном случае то, что четырем тестам соответствует четыре пути, является совпадением. Представленные тесты не покрывают всех путей, например, acd. Поэтому иногда необходима реализация восьми тестов.

Итак, для программ, содержащих только одно условие на каждое решение, минимальным является набор тестов, который проверяет все результаты каждого решения и передает управление каждому оператору, по крайней мере, один раз. Для программ, содержащих вычисления, каждое из которых требует проверки более чем одного условия, минимальный набор тестов должен:

- генерировать все возможные комбинации результатов проверок условий для каждого вычисления;
- передавать управление каждому оператору, по крайней мере, один раз.

Методики и средства автоматизации тестирования

Основные виды средств автоматизации тестирования.

1. Генератор тестовых данных (ГТД). Специальная программа, с помощью которой возможно оптимизировать процесс генерации тестовых данных под особенности конкретного тестируемого приложения. Генератор может создавать данные, необходимые для использования в проверяемом продукте. Однако для эффективной работы необходимо уметь правильно

использовать ГТД. Иногда на изучение генератора может понадобиться так много времени, что разумнее будет создать тестовые данные вручную.

2. Система учета и отслеживания дефектов. Для учета и отслеживания найденных ошибок используют специальную программу – багтрекер. С помощью системы отслеживания багов тестировщик может заводить дефекты тестируемого продукта, а заказчик имеет возможность проследить за процессом устранения ошибок.

3. Система анализа тестового покрытия. Чем выше уровень тестового покрытия, тем больше тестируемых классов покрыто и тем больше ошибок может обнаружить тестировщик. Тестовое покрытие позволяет оценить качество тестирования. Оценка представляет собой плотность покрытия тестами требований либо исполняемого кода. Чем выше уровень тестового покрытия, тем больше тестов будет выбрано для проверки тестируемых требований или исполняемого кода. Следует учитывать тот факт, что метод покрытия требований может оставить непроверенными некоторые участки кода.

4. Система выявления ошибок времени выполнения. Инструменты данной системы позволяют определить ошибки, которые возникают во время выполнения программы. Для разработчика это одни из самых тяжелых ошибок, которые могут возникать по абсолютно разным причинам. Чаще всего причиной возникновения ошибки времени выполнения становятся неправильные исходные данные. Инструменты системы выявления ошибок времени выполнения позволяют определить, какие ошибки могут возникнуть при работе с программой и определить возможность решения данных ошибок.

5. Система тестирования пользовательского интерфейса. Данный инструмент обладает свойствами записи и воспроизведения. Позволяет определить, удобен ли продукт для предполагаемого применения.

Такие средства помогают программистам провести анализ программ большого объема, что в противном случае было бы неосуществимо. Однако не стоит забывать, что любое средство автоматизированного тестирования является всего лишь инструментом, который не будет за вас писать тест-план. Оно способно лишь выполнять то, что вы укажете ему делать.

Инструментарий тестирования.

1. Для Java
   - JUnit JUnit.org
   - TestNG testNG.org
   - JavaTESK UniTESK.ru
   - NUnit — для языков платформы .NET: C#, Visual Basic .NET и др.
2. Для C
   - CUnit cunit
   - CTESK UniTESK.ru
   - cfix cfix

- API Sanity Autotest — для динамических C/C++ библиотек в Unix-подобных ОС.

3. Для Objective-C
   - OCUnit
4. Для C++
   - CPPUnit
   - Boost Test
   - Google C++ Testing Framework
   - Symbian — Фреймворк для Symbian OS всех версий.
   - API Sanity Autotest — для динамических C/C++ библиотек в Unix-подобных ОС.
5. Для Delphi
   - DUnit
6. Для PHP
   - SimpleTest
   - PHPUnit
7. Для Python
   - PyUnit
   - PyTest
   - Nose
8. Visual Basic
   - vbUnit
9. JavaScript
   - JsUnit.

Среды комплексного тестирования.

Разработчик	Функциональное	Нагрузочное	Качество кода	Управление тестами
IBM	+	+	+	+
Borland	+	+	-	+
AutomatedQA	+	+	-	+
HP	+	+	+	+
Open-source	Abbot, Selenium, Watir	Grinder, Jmeter, OpenSTA	GCT, NCover, Cobertura	FitNesse, TestLink

Пример среды автоматизации.

(По материалам: Дмитрий Карбасов, Константин Пасевич «Тестирование ПО с использованием инструментов HP Mercury». http://software-testing.ru/library/vendors/162-hp-mercury).

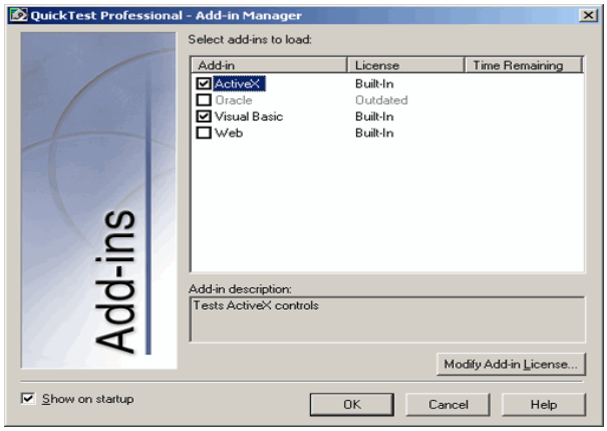
QuickTest Professional (QTP) — программный продукт, предназначенный для автоматизации функционального и регрессионного тестирования, обладающий следующими особенностями:

- создание сложных наборов тестов с минимальным обучением;

- быстрое изолирование дефектов;
- гарантирование правильного функционирования приложения во всех средах, при любых наборах данных, и в любых бизнес-процессах;
- полное документирование и копирование дефектов для разработчиков;
- легкая реализация регрессионного тестирования;
- предоставление возможности организации поставлять программные изделия высокого качества.

QTP позволяет тестировать стандартные Windows приложения, Web приложения, управляющие элементы ActiveX, Visual Basic приложения и мультимедийные объекты на Web страницах.

Можно также приобрести дополнительные модули (add-in) для некоторых специфических рабочих сред, таких как Java, Oracle, SAP solutions, .NET Windows и Web Forms, Siebel, Web services, PeopleSoft и terminal emulator.



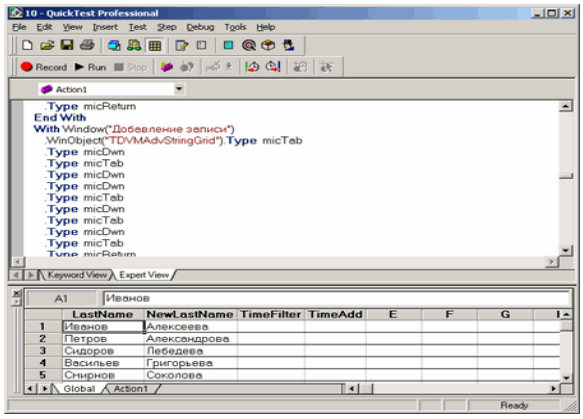
QuickTest Professional позволяет даже новичку создавать test-case (test-case — тест-кейсы, тестовые модули) очень быстро. Вы можете создать test-case просто нажав кнопку Record и используя приложение, чтобы выполнить типичный бизнес-процесс. Например, ведение журнала, составление справочников и т.д.

QuickTest Professional может автоматически ставить checkpoints (checkpoints — контрольные точки), чтобы проверить прикладные свойства и функциональные возможности приложения. Для каждого шага в Tree View (Tree View — дерево представления), есть ActiveScreen (ActiveScreen — окно, показывающее текущее состояние теста), в котором представлен вид приложения в данный момент времени. Вы можете также добавить несколько типов checkpoints для любого объекта, проверить, что компоненты ведут себя так, как ожидается, просто, нажимая на объект в ActiveScreen.

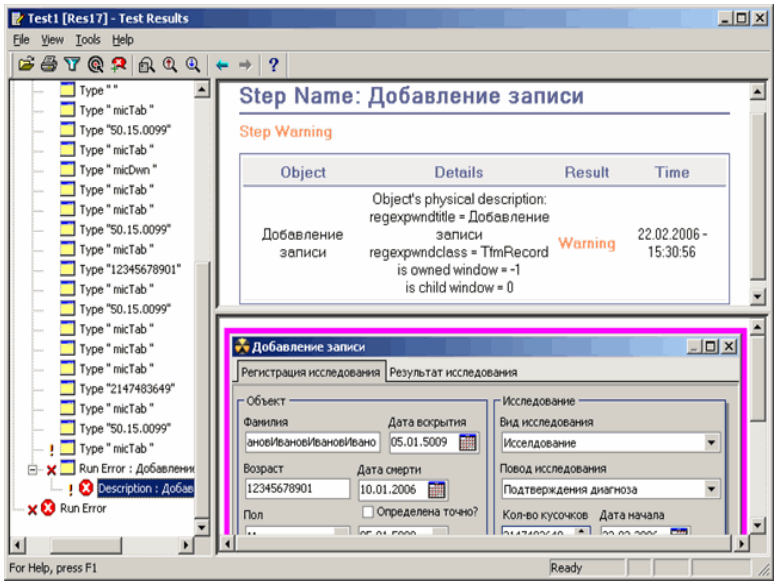
Также вы можете переключиться на Data Table (Data Table — таблица данных), включающую таблицу Excel для работы с данными (Рис. 2), и быстро создавать многократные повторения, чтобы расширить охват test-case.

Опытные пользователи могут рассматривать и редактировать test-case в Expert View (Expert View — поле кода), с помощью VBScript (Visual Basic

Script), которые QuickTest Professional автоматически записывает (Рис. 2). Любые изменения, сделанные в Expert View автоматически синхронизируются с Tree View.



Как только test-case выполнен, TestFusion показывает все аспекты испытания: краткий обзор результатов, Tree View тестируемого приложения, определяющее точно, где произошли ошибки (отказы), используемые испытательные данные, screenshots (screenshots — скриншоты, снимки экрана) для каждого шага, которые четко показывают любые несоответствия, и детальные пояснения для каждого прохода checkpoint ’ а



Поддерживаемые среды

Core Environments • Windows applications (MFC) • Visual Basic • Java • ActiveX **Enterprise Applications** • SAP • Oracle • PeopleSoft • Siebel **Operating Systems** • Windows XP • Windows 2000 • Windows 98 • Windows NT • Windows ME **Web Technologies** • HTML • DHTML • JavaScript **Browsers** • Internet Explorer • Netscape • AOL	**Emerging Technologies** • .Net Winforms, Webforms, Web services • J2EE Web services • XML, WSDL, UDDI **Terminal Emulators** • 3270 • 5250 • VT100 **Server Technologies** • Oracle • Microsoft • IBM • BEA • ODBC • COM/COM+ **Multimedia** • RealAudio/RealVideo • Windows Media Player **Languages** • European • Japanese • Chinese (traditional and simplified) • Korean

Тестирование с помощью QuickTest Professional включает три основных этапа.

Создание Тестов или Компонентов

Перед тем как описывать процедуру создания, определим, что такое тест и компонент. Итак, тест — выполняемая тестовая процедура с конкретными входными данными, начальными условиями и ожидаемым результатом, разработанными для определенной цели, такой, как проверка отдельной программы или верификация соответствия на определенное требование. Компонент — составная часть, элемент чего-либо. В нашем случае это набор функций, который будет использоваться многократно в различных тестах.

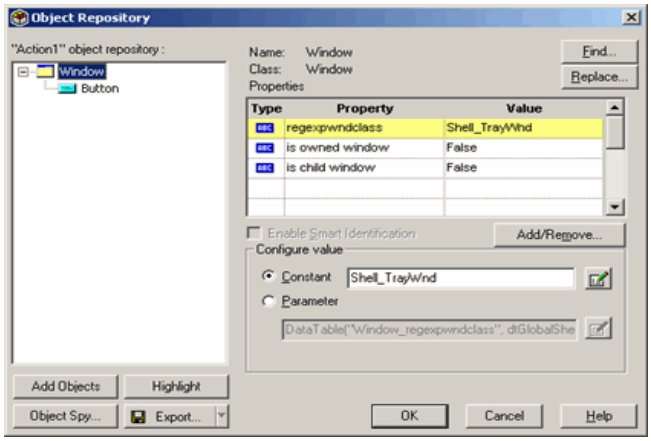
Создание тестов или компонентов происходит путем записи сеанса работы с приложением или Web сайтом. Также существует возможность ручного создания тестового скрипта, используя в Keyword View (Keyword

View — поле ключевых слов) ключевые слова из предварительно созданного object repository (object repository — хранилище объектов).

Чтобы создать тест или компонент необходимо написать (записать) скрипт выполнения данного теста.

Существует два варианта записи скрипта:

1. Автоматически записать сеанс работы с приложением или сайтом.
2. Сформировать object repository и использовать эти объекты для добавления шагов вручную в Keyword View или Expert View (Expert View — поле кода).



Включить checkpoints (checkpoints — контрольные точки) в тест или компонент.

Checkpoints проверяют специфические величины, характеристики страницы или объекта. Они позволяют понять правильно ли приложение функционирует.

Заменить фиксированные величины параметрами (произвести параметризацию).

При тестировании сайта или приложения, можно параметризовать тест или компонент, чтобы проверить, как приложение выполняет те же операции с другими данными. В данном случае можно воспользоваться Data Table (Data Table — таблица данных (Рис.3)) или генератором случайных чисел. Каждый запуск сеанса, который использует параметризацию, называется итерация.

A49 Обручёв

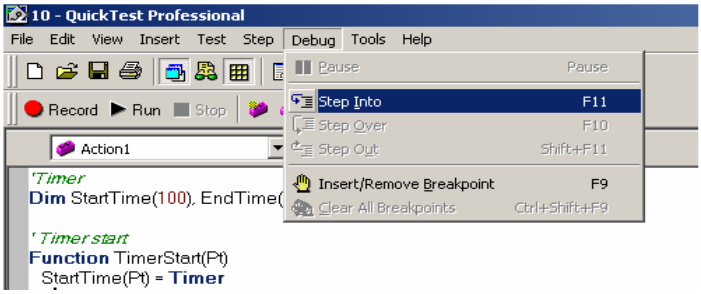
	LastName	NewLastName	TimeFilter	TimeAd
49	Обруччёв	Балмашёва		
50	Обыдённов	Балдачёва		
51	Обыдёнов	Балдёнкова		
52	Овсёнов	Баличёва		
53	Огарёв	Обёртышева		
54	Огнёв	Обойдёнова		
55	Озёркин	Обручёва		

Также можно использовать и выходные величины, чтобы извлекать данные из теста или компонента. Это позволит использовать извлеченные данные в течение исполнения сеанса в других частях теста или компонента.

Выполнение тестов или компонентов

Тест или компонент можно запустить двумя способами:
1.Запуск теста или компонента для проверки сайта или приложения.
2. Запуск теста или компонента для его отладки.

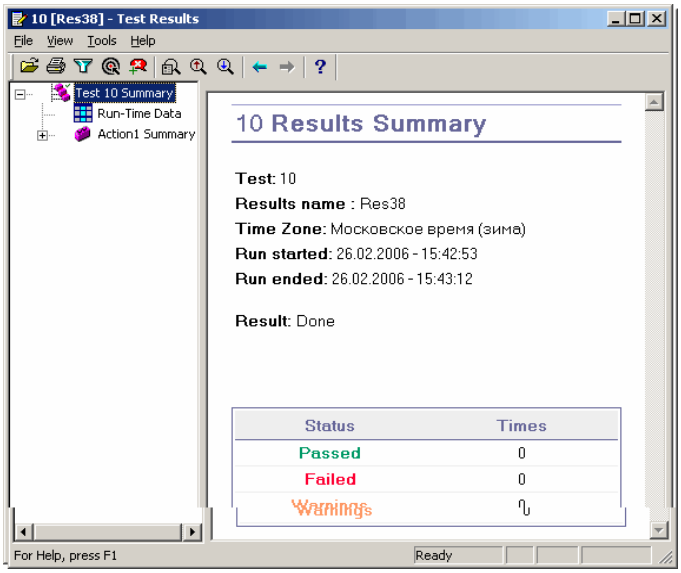
Можно использовать Step Into (выполнение по шагам всех строк кода внутри функции), Step Over (переход в конец функции — строки кода внутри функции пошагово не обрабатываются) и Step Out (повтор предыдущей строки кода в тесте) для запуска тестов или компонентов в режиме step by step (пошаговом режиме) для отладки. Также можно установить break points, чтобы останавливать тест или компонент в нужных точках.



Анализ результатов

После исполнения теста или компонента, можно рассмотреть результаты его работы.

В данном окне можно рассмотреть результаты прогона, а также подробное сообщение о характере ошибки, если таковая имела место.



Отчет о дефектах, обнаруженных в течение сеанса.

С помощью Quality Center (в частности, TestDirector) можно передать дефекты в общую базу данных. QuickTest Professional может автоматически передавать результаты каждого непройденного теста или Вы можете передать данную информацию вручную из окна Результатов Теста.

Понимание объектной модели теста в QuickTest Professional

Как QuickTest Professional определяет объекты

QuickTest Professional исследует объект, действие над которым Вы записываете, и хранит его как объект анализа, определяя затем к какому классу он относится. QuickTest может, например, классифицировать объект анализа как стандартный диалоговый блок Windows (Dialog), кнопку Web (WebButton), или объект Visual Basic ScrollBar (VbScrollBar).

Для каждого класса объекта анализа, QuickTest имеет список обязательных свойств, которые он может определить. Когда Вы записываете действия над объектом, QuickTest всегда определяет эти заданные по умолчанию свойства, и затем просматривает аналогичные свойства других объектов определяя уникальны ли эти свойства, возможно ли однозначно идентифицировать объект. Если нет, то QuickTest добавляет дополнительные свойства в описание, пока это описание не станет уникальным. Если никакие дополнительные свойства не доступны, или доступны, но не достаточны, чтобы создать уникальное описание, QuickTest добавляет специальный порядковый идентификатор, например, позицию объекта на странице или в исходном коде.

Как Mercury QuickTest идентифицирует объекты в течение выполнения сессии

В течение выполнения сеанса, QuickTest ищет объект, который точно соответствует описанию объекта анализа, созданному при записи. Описание, узнанное в течение записи, почти всегда достаточно для QuickTest, чтобы он однозначно идентифицировал объект. Но ваше приложение может включать объекты, которые трудно идентифицировать в течение последующих сеансов.

QuickTest использует метод исключения, чтобы идентифицировать объект, даже когда записанное описание больше точное. Даже если свойства объекта анализа изменяются, технология TestGuard опознает объект, используя Smart Identification (Smart Identification — интеллектуальную идентификацию).

Если QuickTest активизирует механизм Smart Identification в течение запуска сеанса (поскольку он не смог идентифицировать объект по записанному описанию), из этого следуют следующие действия для идентификации:

QuickTest «забывает» записанное описание объекта анализа и создает новый список возможных объектов, содержащий объекты (в пределах родительского объекта), соответствующие всем свойствам, определенным в базовом списке свойств фильтра.

Из этого списка объектов QuickTest убирает объект, который не соответствует первому списку свойств в списке Optional Filter Properties (Optional Filter Properties — свойства дополнительного фильтра).

QuickTest просматривает новый список возможных объектов:

Если новый список возможных объектов все еще имеет более одного объекта, QuickTest использует новый (меньший) список возможных объектов, чтобы повторить шаг 2 для следующих дополнительных свойств фильтра.

Если новый список возможных объектов пуст, QuickTest игнорирует это дополнительное свойство фильтра, возвращается к предыдущему списку возможных объектов и повторяет шаг 2 для следующего дополнительного свойства фильтра.

Если список возможных объектов содержит один объект, тогда QuickTest решает, что идентификация выполнена.

QuickTest продолжает выполнение процесса, описанного в шагах 2 и 3 до того, пока идентифицирует объект или испытает недостаток дополнительных свойств фильтра для идентификации.

Если идентификация не удалась, то QuickTest использует записанное описание плюс порядковый идентификатор, чтобы идентифицировать объект.

Если и комбинированное описание (описание и порядковый идентификатор) не достаточны, чтобы идентифицировать объект, тогда QuickTest останавливает выполнение сеанса и отображает сообщение об ошибке.

Создание тестов или компонентов в QuickTest Professional

Прежде чем Вы начнете записывать, Вы должны спланировать ваш тест или компонент:

Определите функциональное назначение. Короткие тесты или компоненты, которые проверяют специфические функции приложения лучшие чем те, что выполняют сразу несколько задач.

Решите какую информацию Вы хотите проверить в течение теста или компонента. Checkpoint (контрольная точка) может проверить наличие ошибок.

Оцените типы событий, которые Вам нужно записывать. Если Вам нужно записывать события, отличающиеся от обычных (заданных по умолчанию), Вы можете сконфигурировать события, которые Вы хотите записать.

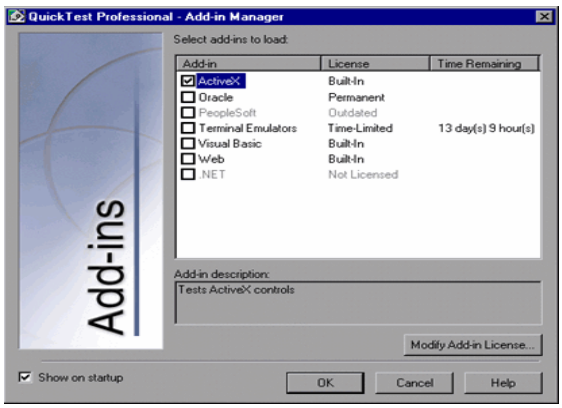
Замените фиксированные величины параметрами, чтобы обеспечить гибкость вашего теста или компонента. Когда Вы параметризируете ваш тест или компонент, Вы можете проверить, как выполняются те же операции с другими данными.

Решите, как Вы будете организовать Object Repository (хранилище объектов) — индивидуально для каждого теста или вести общее хранилище.

Используйте действия, чтобы придать гибкость процессу тестирования.

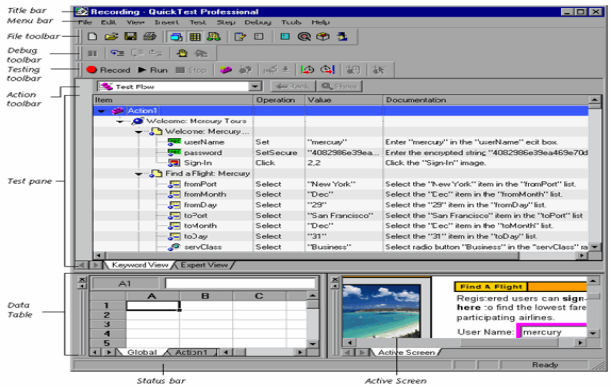
Запуск QuickTest

Перед запуском QuickTest Professional открывается диалоговое окно Add-in Manager (Add-in Manager — менеджер расширений, дополнений (Рис. 1)). Если Вы устанавливали расширения QuickTest, то можно определить какие из расширений будут загружены в этом сеансе.



Щелкните OK . Откроется окно QuickTest Professional. Оно содержит следующие элементы:

- QuickTest title bar — показывает имя открытого теста или компонента;
- Menu bar — меню QuickTest;
- File toolbar — содержит кнопки для управления тестом или компонентом;
- Testing toolbar — содержит кнопки для тестирования;
- Debug toolbar — содержит кнопки для отладки (по умолчанию не показаны);
- Action toolbar — содержит кнопки и выпадающий список действий;
- Test pane — содержит Keyword View (Keyword View — поле ключевых слов) и Expert View (Expert View — поле кода);
- Active Screen — окно, показывающее текущее состояние теста;
- Data Table — таблица данных (используется для параметризации тестов);
- Status bar — статус приложения QuickTest.

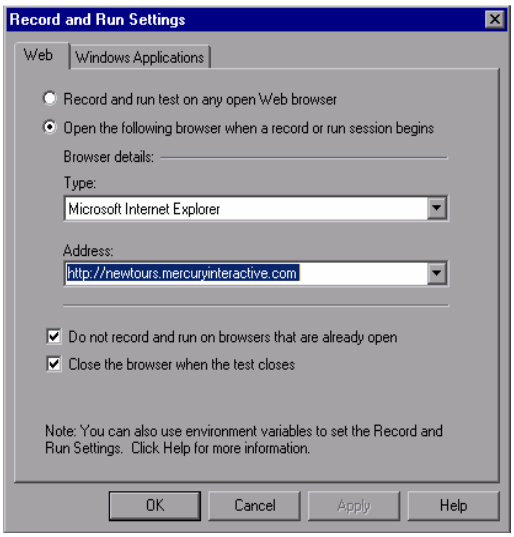


Запись теста или компонента

Создание теста происходит путем записывания типичных действий, которые выполняют пользователи. QuickTest записывает действия, которые Вы выполняете, отображая их как шаги в Keywords View (Keywords View — поле ключевых слов) и генерируя код в сценарии.

Чтобы записать тест:

- Откройте QuickTest. Более подробно, смотрите Запуск Mercury QuickTest Professional .
- Создайте тест. Для того, чтобы создать новый тест, щелкните кнопку New или выберите Файл > Новый.
- Щелкните кнопку Запись или выберите Тест > Запись. Если Вы записываете новый тест и еще не установили свойства записи и запуска, откроется диалоговое окно Свойства записи и запуска.



Диалоговое окно Свойства записи и запуска содержит страницы, соответствующие загруженным расширениям.

Сделайте настройки в данном диалоговом окне и щелкните OK, чтобы начать записывать тест.

Используйте тестируемое программное обеспечение, чтобы записать тест.

Для того, чтобы определить, что приложение работает правильно можно включить checkpoints (контрольные точки).

Вы можете параметризовать ваш тест, чтобы проверить, как он выполняет те же операции с многочисленными вариантами данных, или с данными из внешнего источника.

Щелкните кнопку Стоп или выберите Тест > Стоп для остановки записи.

Для сохранения теста щелкните кнопку Сохранить или выберите Файл -> Сохранить.

Организация процедуры тестирования (на примере Aplana Software services).

Целью тестирования является:

- выявление проблем, связанных с несоответствием разрабатываемого программного продукта – требованиям к нему;
- учет статуса проблем;
- снижение рисков проекта, связанных с качеством разрабатываемого продукта.

В соответствии с моделью рабочей группы в проекте разработки программного обеспечения участвуют следующие ключевые роли: менеджер проекта, аналитик, конструктор, разработчик, тестировщик, технический писатель, интегратор.

Распределение ответственности

Менеджер проекта – ключевая роль рабочей группы, несет ответственность за обеспечение ресурсами процесса тестирования, координацию взаимодействия работ по тестированию и исправлению выявленных дефектов и организацию разрешения спорных вопросов по проблемам.

Разработчик, Технический писатель – ключевые роли рабочей группы, несут ответственность за исправление выявленных ошибок в рамках выделенных ресурсов.

Конструктор – ключевая роль рабочей группы, несет ответственность за контроль целостности проектных решений в процессе исправления разработчиками выявленных дефектов и формирование способов исправления ошибок в сложных или неоднозначных ситуациях.

Интегратор – ключевая роль рабочей группы, несет ответственность за контроль и выпуск версий разрабатываемого программного обеспечения в соответствии с согласованными критериями тестирования.

Аналитик – ключевая роль рабочей группы, несет ответственность за установку приоритетов, связанных с необходимостью и срочностью исправления выявленных ошибок.

Тестировщик – ключевая роль рабочей группы, несет ответственность за процесс тестирования в целом.

Ключевая роль тестировщика

В каждом проекте разработки программного обеспечения в рабочей группе должна быть заполнена ключевая роль тестировщика, на которую возлагается ответственность за обеспечение процесса тестирования в целом.

В зависимости от масштабов проекта роли могут совмещаться по правилам, описанным в Положении о планировании или, для больших проектов, - расширяться до образования ролевой группы.

Для малых проектов роль тестировщика в первую очередь рекомендуется совмещать с ролями: аналитика и документатора. Допускается совмещение роли тестировщика с ролью интегратора. Не допускается совмещение роли тестировщика с ролями разработчика, конструктора и менеджера проекта (см. Положение о планировании).

Для больших проектов роль тестировщика заполняется несколькими специалистами с образованием группы тестирования. При этом должна обеспечиваться следующая структура группы тестирования:

Руководитель группы тестирования (Test manager) – представляет ключевую роль тестировщика в рабочей группе, несет ответственность за организацию процесса тестирования в проекте, планирование и контроль действий по тестированию.

Тест аналитик (Test analyst) – несет ответственность за формирование тестовых спецификаций, и анализ итогов тестирования.

Тест разработчик (Test developer) – несет ответственность за разработку автоматизированных тестов, предусмотренных в плане тестирования, установку и сопровождение инфраструктуры тестирования, создание стенда для проведения тестирования в соответствии с планом тестирования.

Исполнитель тестов (Test operator) - несет ответственность за фактическое исполнение тестов и документирование выявленных дефектов.

Приведенные роли могут совмещаться внутри группы тестирования. Роль руководителя группы тестирования должна быть заполнена от начала до завершения проекта. Остальные роли могут привлекаться в ходе проекта по мере необходимости.

Документирование

Процесс тестирования ПО должен быть документированным. Действующие версии документов должны быть оформлены, согласованы и утверждены в соответствии с настоящим положением и опубликованы для сведения рабочей группы проекта, ГКК, руководства организации и заказчика, по его требованию. Выявленные в ходе тестирования дефекты должны быть полностью описаны и документированы. Полная документация, созданная в ходе тестирования, сохраняется и сдается в архив по завершении проекта.

Контрольные вопросы к зачету

1. Каковы особенности тестирования методом «черного ящика»?
2. Какие категории ошибок выявляет тестирование методом «черного ящика»?
3. Какие достоинства имеет тестирование методом «черного ящика»?
4. 1. Каковы особенности тестирования методом «белого ящика»?
5. Поясните суть способа разбиения по эквивалентности.
6. Что такое класс эквивалентности?
7. Какие правила формирования классов эквивалентности вы знаете?
8. Как выбирается тестовый вариант при тестировании по способу разбиения по эквивалентности?
9. Поясните суть способа анализа граничных значений.
10. Чем способ анализа граничных значений отличается от разбиения по эквивалентности?
11. Поясните правила анализа граничных значений.
12. Что такое дерево разбиений? Каковы его особенности?
13. В чем суть способа диаграмм причин-следствий?
14. Что такое причина и что такое следствие?
15. Дайте общую характеристику графа причинно-следственных связей.
16. Какие функции используются в графе причин и следствий?
17. Какие ограничения используются в графе причин и следствий?
18. Поясните шаги способа диаграмм причин-следствий.
19. Какую структуру имеет таблица решений в способе диаграмм причин-следствий?
20. Как таблица решений преобразуется в тестовые варианты?

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (СГАУ)

РАБОТА с HP QuickTest Pro и JUnit

(методические указания к лабораторным работам)

Составитель: К.Е. Климентьев

Самара 2013

Введение

В настоящих методических указаниях приводятся примеры работы со средствами тестирования программного обеспечения:

- Mercury QuickTest Pro;
- Junit.

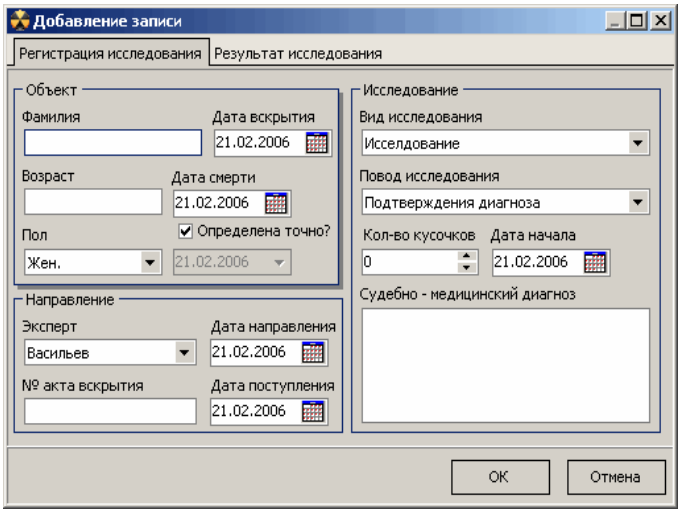
Так же приводятся индивидуальные задания для лабораторных работ.

Лабораторная работа 1. Mercury QuickTest Professional. Создание простейшего функционального теста

За основу для создания функционального теста будет взято заполнение журнала в приложении «Судмедэкспертиза». Данное приложение написано на Delphi 2005 под Firebird 1.5. Наша задача состоит в том, чтобы всесторонне проверить функцию «Добавление записи в журнал».

Основные направления проверки

Отразим основные направления проверки:
Проверка вызова окна «Добавление записи».



Данную операцию можно выполнить двумя способами:
— с помощью кнопки на панели инструментов;
— нажав Insert;
Проверка заполнения данными окна «Добавление записи». Из теории тестирования — для проверки нам необходимо проверить реакцию приложения на правильные, неправильные и граничные значения.

Граничные значения:

- Фамилия — символьный тип (макс. 35);

- Возраст — символьный тип (макс. 10);
- № акта вскрытия — символьный тип (макс. 10);
- Кол-во кусочков — целый тип (макс. 2147483648);
- Судебно-медицинский диагноз — символьный тип (макс. 1000);
- Гистологический диагноз — символьный тип (макс. 1000);
- Количество препаратов — целый тип (макс. 2147483648);
- Примечание — символьный тип (макс. 1000);
- Проверка работы кнопок ОК и Отмена.

План тестирования

Создадим план данного теста:

1. Тестирование открытия и закрытия окна «Добавление записи»:
   - нажать кнопку на панели инструментов, затем нажать Отмена;
   - нажать кнопку на панели инструментов, затем нажать кнопку закрытия окна;
   - нажать на клавиатуре Insert, затем кнопку закрытия окна;
   - нажать на клавиатуре Insert, затем кнопку Esc;

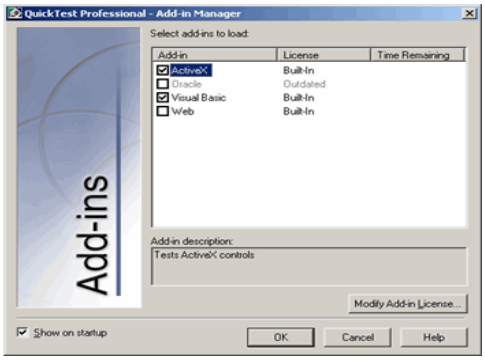
2. Тестирование заполнения разделим на четыре этапа:
   - ввод заведомо правильных данных;
   - ввод заведомо неправильных данных;
   - ввод данных верхней границы диапазона правильных значений;
   - ввод данных нижней границы диапазона правильных значений;

Составим наборы данных для каждого этапа. Заметим сразу, что поля Пол, Эксперт, Вид исследования, Повод исследования, Врач, Лаборант и Причина смерти представляют собой выпадающий список. Ввод данных не предусмотрен.

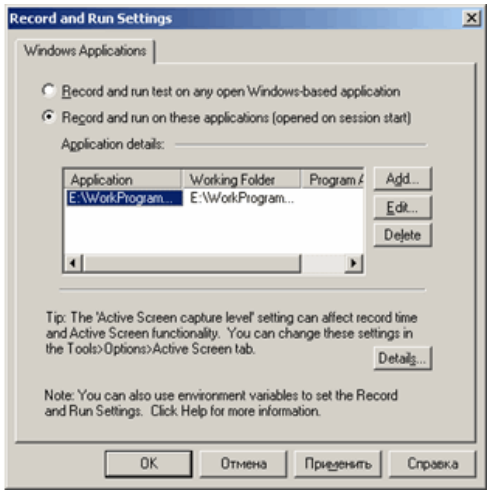
Поле	Правильные данные	Неправильные данные	Граничные значения	
			Верхняя граница	Нижняя граница
Фамилия	Иванов	ИвановИванов-Иванов Иванов-ИвановИванов (36 символов)	ИвановИванов-Иванов Иванов-ИвановИвано (35 символов)	0 символов
Дата вскрытия	10.01.2006	50.15.0099	31.12.9999	01.01.0100
Возраст	56	Сто пять (или 12345678901)	1234567890	0 символов
Дата смерти	10.01.2006	50.15.0099	31.12.9999	01.01.0100
Определена точно	Да (или нет и 01.01.2006)	Да и 50.15.0099	Да и 31.12.9999	Да и 01.01.0100
Пол	-	-	-	-
Эксперт	-	-	-	-
Дата направления	10.01.2006	50.15.0099	31.12.9999	01.01.0100
№ акта вскрытия	56	Сто пять (или 12345678901)	1234567890	
Дата поступления	10.01.2006	50.15.0099	31.12.9999	01.01.0100
Вид исследования	-	-	-	-
Повод исследования	-	-	-	-
Кол-во кусочков	15	-15 или 2147483649 (больше чем тип integer)	100	0
Дата начала	10.01.2006	50.15.0099	31.12.9999	01.01.0100
Судебно-медицинский диагноз	умер	1001 символ	1000 символов	0 символов
Врач	-	-	-	-
Лаборант	-	-	-	-
Дата окончания	10.01.2006	50.15.0099	31.12.9999	01.01.0100
Гистологический диагноз	умер	1001 символ	1000 символов	0 символов
Причина смерти	-	-	-	-
Количество препаратов	15	-15 или 2147483649 (больше чем тип integer)	1000	0
Примечание	умер	1001 символ	1000 символов	0 символов

3. Проверка работоспособности кнопок ОК и Отмена для каждого этапа.
Запись test-case (тестового модуля)

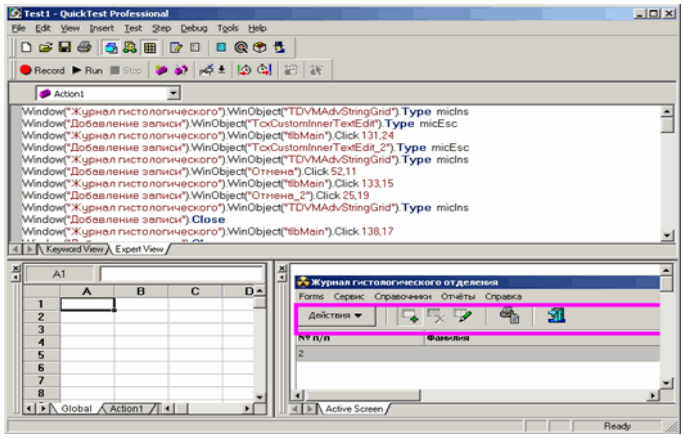
Откройте QuickTest Professional, отметьте в Add-in Manager ActiveX и Visual Basic и нажмите ОК.



Затем создайте новый тест, щелкните кнопку Запись. В Свойствах записи и запуска (Рис. 3)укажите приложение, которое будете тестировать (в нашем случае — Expert.exe). Более подробно операция создания теста описана в предыдущем разделе.

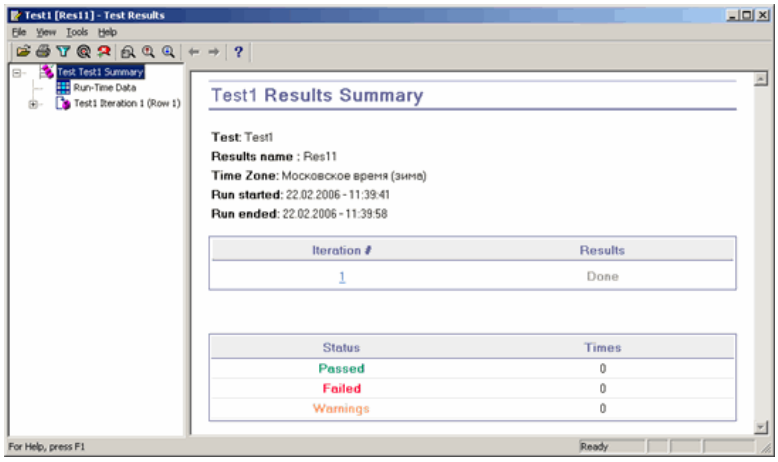


При работе с приложением QuickTest Professional запишет все действия пользователя. По окончании теста нажимаем Stop.

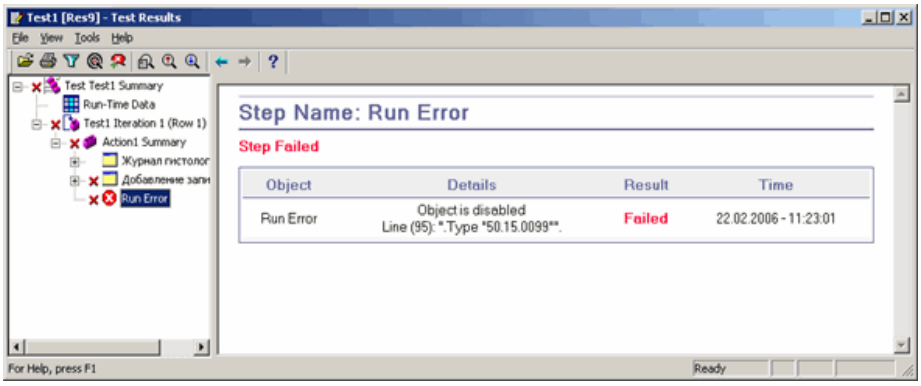


Т.к. в плане тестирования указано, что наш тест будет содержать четыре этапа, и этапы отличаются только набором данных, мы запишем только один из них. Затем, после записи, скопировав записанный тест, создав процедуры и вставив его с остальными вариантами данных, мы получим полный тест с четырьмя этапами.

Тест на открытие/закрытие окна был пройден. Тест с правильными данными также пройден без ошибок.

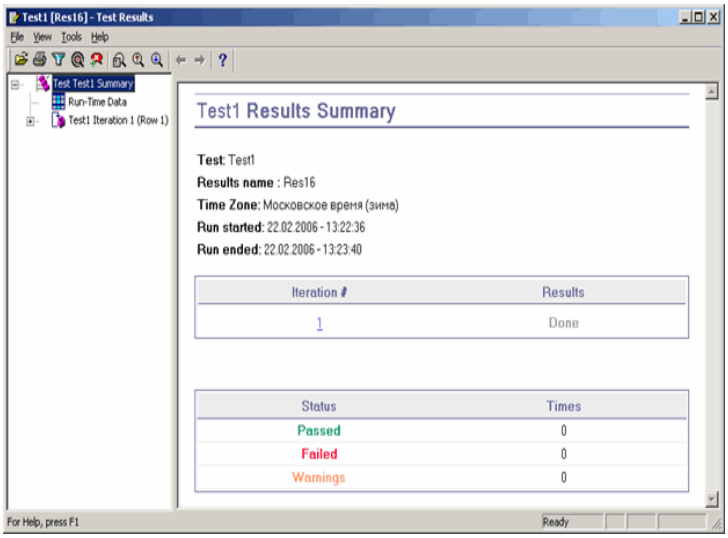


Как и ожидалось, тест с неправильными данными не прошел.



Также приложение выдало следующую ошибку:

Тесты с граничными значениями данных прошли успешно.



Итак, мы видим, что тестирование с правильными данными и граничными значениями правильных данных прошло успешно. Тестирование же с неверными данными приводит к ошибке в программе, непонятной пользователю.

Для дальнейшего изучения влияния наборов данных на приложение необходимо параметризовать данный тест. Также можно расставить транзакции для измерения временной характеристики выполнения теста. Данные вопросы будут рассмотрены в следующей статье.

Литература

1. Дмитрий Карбасов, Константин Пасевич. Тестирование ПО с использованием инструментов HP Mercury. http://software-testing.ru/library/vendors/162-hp-mercury

Лабораторная работа 2. Работа с JUnit

Создадим в директории test класс test и создадим в нем публичный void метод testing и добавим к нему аннотацию @Test. Данная аннотация всегда обозначает тестовые методы. Все методы тестирования обязательно являются public void. Будем использовать обычную операцию сравнения assertEquals.

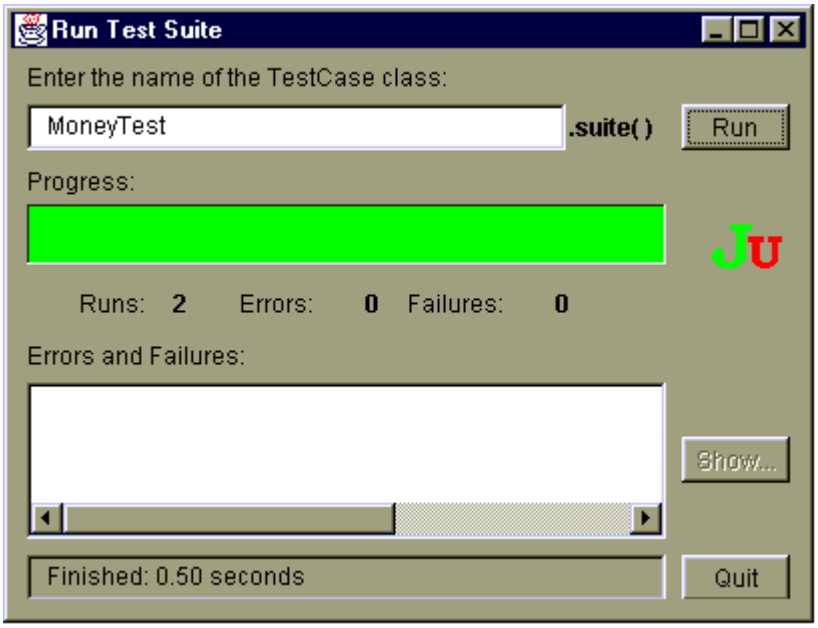
```
import static org.junit.Assert.assertEquals;

import org.junit.Test;
public class test {

@Test
public void testing(){
String str = "This is string";
assertEquals("This is string",str);
}

}
```

Далее делаем: Run as jUnit test, и видим зеленую полоску, то есть тест завершился успешно.



Как мы можем видеть, все просто и легко. Теперь создадим класс Maths в нашей привычной директории src. Код класса приведен ниже:

```
public class Maths {

private int varible = 0;

public int getVarible() {
return varible;
}

public void setVarible(int varible) {
this.varible = varible;
}

public void plusVarible(int val){
this.setVarible(val+varible);
}
}
```

Типичный класс, имеющий переменную с методами get и set для нее и метод для суммирования переменной и переданного значения.

Теперь напишем тест для нашего класса:

```
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertNotSame;
import static org.junit.Assert.assertSame;

import org.junit.Test;
```

```
public class test {

@Test
public void testing(){
Maths m1 = new Maths();
Maths m2 = new Maths();
Maths m3 = new Maths();
m1.plusVarible(10); //10
m2.plusVarible(10); //10
m3.plusVarible(15); //15

assertFalse(m1.getVarible() == m3.getVarible()); // Если true - то тест
завален
assertEquals(m1.getVarible(), m2.getVarible()); // Если не равны - тест
завален
assertNotNull(m1); // Если null - тест завален
assertNotSame(m1, m2); // Если оба объекта являются одинаковыми(не одно и
то же, что равны) - тест завален
assertSame(m1, m1); // Если оба объекта не являются одинаковыми - тест
завален

}

}
```

Если выполнить этот тест, то он завершится успехом.

Индивидуальные задания к лабораторным работам

ВАРИАНТ 1
Разработать класс, представляющий студента. Студент характеризуется именем, фамилией, группой и набором экзаменов, которые он сдавал. Экзамен характеризуется названием предмета, оценкой студента по нему и датой сдачи (год, семестр). Группа характеризуется курсом и факультетом.

Операции:

- узнать наивысшую оценку среди всех экзаменов по данному предмету
- добавить ему оценку по экзамену
- удалить для него оценку по экзамену;
- если он такой экзамен не сдавал - сгенерировать исключение
- узнать число экзаменов, которые он сдал с указанной оценкой
- узнать его средний балл за указанный семестр;

ВАРИАНТ 2
Разработать класс, представляющий общежитие. Общежитие характеризуется улицей, номером дома, факультетом и набором комнат. Комната характеризуется номером, вместимостью и числом занятых мест. Факультет характеризуется институтом, названием и числом студентов.

Операции:

- открыть комнату для заселения

- занять комнату
- в случае если она уже занята сгенерировать исключение
- освободить комнату
- вернуть число свободных комнат (полностью/частично)
- узнать, какой процент студентов института живет в общежитии

ВАРИАНТ 3
Разработать класс, представляющий факультет. Факультет характеризуется полным и кратким названием, институтом, к которому он относится и списком учебных групп. Институт характеризуется названием и адресом. Группа характеризуется названием, числом студентов и средним баллом по итогам последней сессии.

Операции:
- узнать средний балл по всем группам с числом студентов выше заданного
- добавить группу (начало первого курса)
- удалить группу (после диплома)
- если группы нет - сгенерировать исключение
- узнать число студентов данной группы
- вернуть группу с наивысшим средним баллом

ВАРИАНТ 4
Разработать класс, представляющий преподавателя. Преподаватель характеризуется именем, фамилией, кафедрой, списком предметов, которые он читает. Кафедра относится к факультету и имеет имя. Предмет характеризуется названием, количеством часов и средним баллом, выставленным преподавателем на последней сессии по всем группам.

Операции:
- узнать среднее количество часов для всех предметов со средним баллом не ниже данного
- добавить предмет (внесли в учебный план)
- удалить предмет (исключили из учебного плана)
- если предмета нет - сгенерировать исключение
- найти самый низший средний балл по всем предметам
- узнать, есть ли предмет с заданным количеством часов

ВАРИАНТ 5
Разработать класс, представляющий методическое пособие. Пособие характеризуется автором, названием, предметом и списком выдач экземпляров на руки. Выдача характеризуется датой и именем студента. Автор характеризуется именем, фамилией и названием кафедры.

Операции:

- узнать среднее число выдач для всех студентов
- выдать пособие на руки
- получить пособие обратно
- если пособие не выдавалось - сгенерировать исключение
- найти студента с наибольшим числом выдач
- узнать число выдач для заданной даты
- если выдач нет - сгенерировать исключение

ВАРИАНТ 6
Разработать класс, представляющий книгу в библиотеке. Книга характеризуется списком авторов, названием и темой. Тема характеризуется кодом темы и названием темы. Автор характеризуется именем, фамилией, предпочтительной темой и числом написанных книг.

Операции:
- узнать среднее число написанных книг для всех авторов
- добавить автора в книгу
- вычеркнуть автора из книги
- если автора нет - сгенерировать исключение
- найти автора с максимальным числом написанных книг
- найти число авторов, чья предпочтительная тема совпадает с темой книги

ВАРИАНТ 7
Разработать класс, представляющий студенческую группу. Группа характеризуется факультетом, именем и списком студентов. Студент характеризуется именем, фамилией и средним баллом за последнюю сессию. Факультет характеризуется именем и профилем деятельности.

Операции:
- узнать средний балл, рассчитанный по всем студентам с данным именем
- принять студента в группу
- исключить студента из группы
- найти число студентов с данным средним баллом
- найти средний балл указанного студента
- если студент не существует - сгенерировать исключение

ВАРИАНТ 8
Разработать класс, представляющий университет. Университет характеризуется названием, улицей, номером дома, списком факультетов и ректором. Ректор характеризуется именем, фамилией и ученой степенью. Факультет характеризуется названием, профилем и числом студентов.

Операции:
- узнать число студентов данного факультета

- открыть новый факультет
- закрыть факультет
- если факультета нет - сгенерировать исключение
- найти среднее число студентов по всем факультетам
- найти общее число студентов всех факультетов данного профиля

ВАРИАНТ 9
Разработать класс, представляющий лекционный курс. Лекционный курс характеризуется названием, преподавателем, который его читает, группой и списком проведенных пар. Пара характеризуется датой и числом студентов, посетивших ее. Группа характеризуется названием и числом студентов.

Операции:
- найти средний процент посещаемости по всем парам
- провести пару
- вычеркнуть проведенную пару
- найти число пар, на которые пришло заданное число студентов
- узнать, была ли стопроцентная посещаемость, если была, то когда в
- первый раз, если нет - сгенерировать исключение

ВАРИАНТ 10
Разработать класс, представляющий учебный корпус института. Учебный корпус относится к институту и характеризуется названием и списком аудиторий. Институт характеризуется адресом и названием. Аудитория характеризуется номером, вместимостью и занятостью (занята/не занята).

Операции:
- найти среднюю вместимость свободных аудиторий
- добавить аудиторию (открыть ее для занятий)
- удалить аудиторию (закрыть на ремонт)
- если аудитории нет - сгенерировать исключение
- узнать, занята ли данная аудитория
- найти суммарную вместимость корпуса

Тесты к зачету по курсу
«Современные методы и технологии тестирования программного обеспечения»

Составитель: к.т.н., доц. К.Е. Климентьев

1. Что входит в жизненый цикл разработки ПО в соответствие с ГОС 19.102-77.
А) Техническое задание
Б) Эскизный проект
В) Технический проект
Г) Рабочий проект
Д) Внедрение.
Е) Эксплуатация
Ж) Утилизация

2. Какие типы жизненного цикла бывают
А) Каскадный
Б) V-образный
В) Квадратный
Г) Спиральный

3. Что такое «экстремальное программирование»
А) Технология разработки ПО
Б) Тип жизненного цикла ПО
В) Программирование задач экстремальной сложности

4. Что такое «управляемое выполнение программы с целью обнаружения несоответствий ее поведения и требований»
А) Валидация
Б) Тестирование
В) Сертификация
Г) Верификация

5. Что такое «достижение гарантии того, что объект (требования или программный код) соответствует требованиям»
А) Валидация
Б) Тестирование
В) Сертификация
Г) Верификация

6. Что такое «доказательство соответствия системы ожиданиям заказчика»
А) Валидация
Б) Тестирование
В) Сертификация
Г) Верификация

7. Какие факторы входят с определение качества продукции в соответствии с ISO 9126
А) Функциональность
Б) Надежность (включая защищенность от внутренних ошибок и угроз)
В) Распределенность
Г) Безопасность
Д) Быстродействие
Е) Переносимость

8. В классификацию по объекту тестирования входит:
А) Локальное тестирование
Б) Модульное тестирование - позволяет проверить функционирование отдельно взятого элемента;
В) Интеграционное тестирование - процесс проверки взаимодействия между программными компонентами/модулями;
Г) Системное тестирование охватывает целиком всю систему.
Д) Глобальное тестирование

9. Тестирование при известных входной и выходной спецификации это
А) Метод «черного ящика»
Б) Метод «белого ящика»
В) Метод «серого ящика»

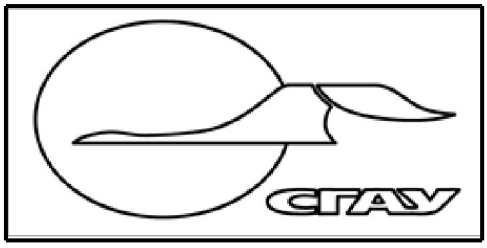
10. Тестирование при известном алгороитме это
А) Метод «черного ящика»
Б) Метод «белого ящика»
В) Метод «серого ящика»

11. Каковы основные принципы при тестировании методом «черного ящика»
А) Эквивалентное разбиение;
Б) Анализ граничных значений;
В) Функциональные диаграммы;
Г) Предположение об ошибке.
Д) Покрытие операторов (команд)
Е) Покрытие решений
Ж) Покрытие условий
З) Покрытие маршрутов (решений+условий).

12. Каковы основные принципы при тестировании методом «белого ящика»
А) Эквивалентное разбиение;
Б) Анализ граничных значений;
В) Функциональные диаграммы;
Г) Предположение об ошибке.
Д) Покрытие операторов (команд)
Е) Покрытие решений

Ж) Покрытие условий
З) Покрытие маршрутов (решений+условий).

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(СГАУ)



СОГЛАСОВАНО

Управление образовательных программ

_____/ А.В. Дорошин /

"_____"_____ 20___ г.

УТВЕРЖДАЮ

Проректор по учебной работе

_____/ В.Н. Матвеев /

"____" _____ 20___ г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Наименование модуля (дисциплины)

М1.В.ДВ.2.2. - Совр. методы и технологии тестирования ПО

Цикл, в рамках которого происходит освоение модуля (дисциплины)	М2 - профессиональный цикл
Часть цикла	М2.В.ДВ - дисциплины по выбору

Код учебного плана

230100.68.2-13-56-3022

Факультет	Информатики
Кафедра	Инф. систем и технологий

Курс	5		
Семестр	10		
Лекции (СЛ)	36		
Семинарские и практические занятия (СП)	18		
Лабораторные занятия (СЛР)	18	Экзамен	
Контроль самостоятельной работы / Индивидуальные занятия (КСР / ИЗ)	0	Зачет	10
Самостоятельная работа (СРС)	72		
Всего (Всего с экзаменами)	180		

Наименование стандарта, на основании которого составлена рабочая программа:

230100.68 "Информатика и выч. техника"

Соответствие содержания рабочей программы, условий ее реализации, материально-технической и учебно-методической обеспеченности учебного процесса по дисциплине всем требованиям государственных стандартов подтверждаем.

Составители:

К.Е. Климентьев _____
(подпись)

Заведующий кафедрой:

С.А. Прохоров _____
(подпись)

Рабочая программа обсуждена на заседании кафедры

Инф. систем и технологий

Протокол №___ от "____" _____ 20___ г.

Наличие основной литературы в фондах научно-технической библиотеки (НТБ) подтверждаем:

Директор НТБ _____ /_____/
(подпись) (расшифровка подписи)

Согласовано:

Декан _____ /_____/
(подпись) (расшифровка подписи)

1 Цели и задачи модуля (дисциплины), требования к уровню освоения содержания

1.1 Перечень развиваемых компетенций

ОК-11, ОК-14, ОК-15, ПК-2, ПК-3, ПК-9, ПК-10

1.2 Цели и задачи изучения модуля (дисциплины)

Целями курса являются освоение:
- методик тестирования программного обеспечения,
- методик составления тест-планов,
- способов реализации их на практике,
- способов использование результатов в целях улучшения качества программного обеспечения.

1.3 Требования к уровню подготовки студента, завершившего изучение данного модуля (дисциплины)

По окончанию изучения курса студент должен иметь представление: о концепциях, понятиях и методах тестирования программного обеспечения; о применяемых подходах к тестированию.
Знать: основные понятия теории тестирования программного обеспечения (ПО), основные её концепции и методы.
Уметь: самостоятельно проектировать и проводить тесты для разработанного ими приложения; составлять тестовые таблицы и планы тестирования.

1.4 Связь с предшествующими модулями (дисциплинами)

Изложение материала базируется на знании студентами следующих дисциплин:
- Программирование на ЭВМ;
- Математическая логика и теория алгоритмов.

1.5 Связь с последующими модулями (дисциплинами)

Дисциплина является предшествующей для выполнения квалификационной работы магистра.

2 Содержание рабочей программы (модуля)

Семестр 1		
СЛ 0,2 36 часов 1 ЗЕТ	Активные 0	
	Интерактивные 0	

	Традиционные 0	Постановка задачи тестирования. Различие понятий «тестирование» и «отладка». Классификации видов тестирования: по цели тестирования (функциональное, тестирование быстродействия и производительности, тестирование безопасности), по времени проведения, - 16
		Теоретические основы тестирования: логическая верификация алгоритмов; тестирование «черного ящика» (в т.ч. случайными наборами данных); тестирование «белого ящика»; методики покрытия всех команд, всех путей, всех условий; комбинаторные методики - 6 ч
		Особенности тестирования объектно-ориентированных и WEB-приложений. Автоматизация тестирования: поиска областей сходимости результатов; построения наборов тестовых даных; трассировки и фиксирования промежуточных результатов (средства Junit и HP QuickTest)
		4. Комплексный подход к тестированию. Этап тестирования как часть проекта. Организация коллектива. Подготовка к тестированию. Выполнение тестирования. Документирование и использование результатов. – 4 часа
СП 0,1 18 часов 0,5 ЗЕТ	Активные 0	
	Интерактивные 0	логическая верификация алгоритмов – 6 часов
		методики «покрытия» - 6 часов
		ручное построение функциональных блок-диаграмм – 6 часов
	Традиционные 0	
СЛР 0,1 18 часов 0,5 ЗЕТ	Активные 0	
	Интерактивные 0	тестирование проекта без средств автоматизации – 6 часов.
		использование Junit – 6 часов;
		использование QuickTest Pro - 6 часов.
	Традиционные 0	

КСР 0 0 часов 0 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	
СРС 0,4 72 часов 2 ЗЕТ	Активные 0	
	Интерактивные 0	
	Традиционные 0	

3 Инновационные методы обучения

Лабораторные работы выполняются с использованием вычислительной техники

4 Технические средства и материальное обеспечение учебного процесса

Лабораторные работы выполняются средствами вычислительной техники.
Требования к аппаратному обеспечению: наличие локальной сети.
Требования к программному обеспечению:
- операционная система Windows версий NT, 2000 или XP;
- UNIX-совместимая операционная система (например, FreeBSD или на основе ядра Linux) ;
- демонстрационные версии библиотек и пакетов Junit, TestNG, TestComplete и HP QuickTest Pro.

5 Учебно-методическое обеспечение

5.1 Основная литература

1. Орлов, Сергей Александрович. Технологии разработки программного обеспечения [Текст] : разраб. слож. програм. систем : [учеб. для вузов по специальности "Програм. обеспечение вычисл. техники и автоматизир.систем" направления подгот. дипломир. специалистов "Информатика и вычисл. техника"] / С. А. Орлов. - СПб. и др. : Питер : Питер принт, 2004.Экземпляров всего:14
2. Ауэр, Кен. Экстремальное программирование [Текст] : Постановка процесса. С первых шагов и до победного конца / Кен Ауэр, Рой Миллер. - СПб. [и др.] : Питер : Питер принт, 2004. - 367 с.Экземпляров всего:3

5.2 Дополнительная литература

1. Брауде, Эрик Дж. Технология разработки программного обеспечения [Текст] : [перевод] / Эрик Дж. Брауде. - СПб. [и др.] : Питер : Питер принт, 2004. - 654 с. Экземпляров всего:7
2. Макгрегор, Джон. Тестирование объектно-ориентированного программного обеспечения [Текст] : практ. пособие : [пер. с англ.] / Джон Макгрегор, Девид Сайкс. - Киев : ТИД "ДС" ; СПб. [и др.] : ДиаСофт, 2002. - 416 с.Экземпляров всего:3

5.3 Электронные источники и интернет ресурсы

Материалы сайтов http://www.software-testing.ru и http://www.protesting.ru

5.4 Методические указания и рекомендации

Лабораторная работа №1 выполняется после освоения на практических занятиях соответствующих тем