Отчет состоит из страниц, таблиц, рисунков

**1) Цель работы –** Разработка системы сдачи электронной отчетности в региональное управление Росприроднадзора

**Ключевые слова:** требования, контракт, спецификация, ограничения, атрибут

**2) Цель работы –**  Технология использования порубочных остатков для рекультивации нефтезагрязненных земель.

**Ключевые слова:** Древесные отходы, порубочные остатки, неликвидная древесина, щепа, нефтеемкость, измельчение древесных отходов.

**3 Цель работы –** разработка нового метода определения количества микроорганизмов в микробной взвеси

**Ключевые слова:** бактерии, стафилококки, госпитальные и спорадические штаммы

Объектом исследования служили спорадические и госпитальные штаммы S. аureus.

**СПИСОК ИСПОЛНИТЕЛЕЙ**

1)

1) Разработка системы сдачи электронной отчетности в региональное управление Росприроднадзора

Этап 1. Реализация обозначенных в спецификации требований на прикладном программном уровне.

Исполнитель работы: Я. О. Аксенов

2

1) Технология использования порубочных остатков для рекультивации нефтезагрязненных земель.

Этап 1. Создание технологии переработки неликвидной древесины в щепу. Подбор оптимального размера щепы и глубины ее внесения в почву.

Исполнитель работы: Т. А. Бастракова.

3) **:**

1)разработка нового метода определения количества микроорганизмов в микробной взвеси

Этап 1. Выявление в популяции госпитального и спорадического штамма St. Aureus культивируемых и некультивируемых бактерий

Исполнитель работы: Бутков И.И.

СОДЕРЖАНИЕ

Цели работ, ключевые слова ……………………………………………………………………1

Список исполнителей…………………………………………………………………………... 3

Содержание …………………………………………………………………………………….. 5

Определения ……………………………………………………………………………………. 6

Обозначения и сокращения ………………………………………………………………….... 8

Введение ………………………………………………………………………………………..10

Основная часть …………………………………………………………………………………19

Заключение……………………………………………………………………………………...70

Список использованной литературы …………………………………………………………74

Приложения к работам…………………………………………………………………………79

**ОПРЕДЕЛЕНИЯ**

**Программные требования –** свойства программного обеспечения, которые должны быть надлежащим образом представлены в нём для решения конкретных практических задач.

**Контракт –** это спецификация компонентов системы.

**Контрактное программирование –** это метод проектирования программного обеспечения на основе котнрактов.

**Зеленая щепа** - щепа, содержащая примеси коры, хвои и/или листьев;

**Топливная щепа** - фракционный состав щепы, предназначенной для сжигания, определяется в основном конструкцией топки;

**Дробленка** - древесные частицы, полученные при измельчении древесины на дробилках и молотковых мельницах;

**Древесная стружка** - тонкие древесные частицы, образующиеся при резании древесины;

**Древесные опилки** - мелкие частицы древесины, образующиеся в процессе пиления;

**Технологические древесные опилки** - опилки, пригодные для производства целлюлозы, древесных плит и продукции лесохимических и гидролизных производств;

**Древесная мука** - древесные частицы заданного гранулометрического состава, полученные путем сухого механического размола древесины;

**Древесная пыль** - несортированные древесные частицы размером менее 1 мм.

**Деловая древесина -** круглые и колотые лесоматериалы, кроме дров, пневый осмол, а также технологическая щепа;

**Дровяная древесина -** древесина, используемая для топлива (дрова) и технологической переработки (технологические дрова);

**Ликвидная древесина -**древесина, которая может быть использована в хозяйственных целях; включает деловую древесину и дрова;

**Неликвидная древесина*****-***древесина, которая не может быть использована в хозяйственных целях вследствие утраты технических качеств из-за повреждений гнилью, а также в результате пожаров и других стихийных бедствий.

**ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

**CLR –** Common Language Runtime (общеязыковая исполняющая среда)

**Business Requirements –** Бизнес-требования

**User Requirements –** Пользовательские требования

**Functional Requirements –** Функциональные требования

**Business Rules –** Бизнес-правила

**External Interfaces –** Внешние интерфейсы

**Quality Attributes –** Атрибуты качества

**Constraints -** Ограничения

**System Requirements –** Системные требования

**ГС** — госпитальные штаммы

**СШ** – спорадические штаммы

**ВБИ** – внутрибольничные инфекции

**МПА** – мясопептонный агар

**МК** – микробные клетки

**СМ** – стандарт мутности

**ФР** — физиологический раствор хлорида натрия

**ЖСА** – желточно солевой мясопептонный агар

**ИСМП** – инфекции связанные с оказанием медицинской помощи

**ЛПУ** – лечебно-профилактические учреждения

**ВВЕДЕНИЕ**

**Введение по теме 1**

Программные требования – Software Requirements – свойства программного обеспечения, которые должны быть надлежащим образом представлены в нём для решения конкретных практических задач. Данная область знаний касается вопросов извлечения (сбора), анализа, специфицирования и утверждения требований.

Опыт индустрии информационных технологий однозначно показывает, что вопросы, связанные с управлением требованиями, оказывают критически-важное влияние на программные проекты, в определенной степени - на сам факт возможности успешного завершения проектов. Только систематичная работа с требованиями позволяет корректным образом обеспечить моделирование задач реального мира и формулирование необходимых приемочных тестов для того, чтобы убедиться в соответствии создаваемых программных систем критериям, заданным реальными практическими потребностями.

На практике часто применяется подход, используемый в различных методологиях разработки ПО и базирующийся на определении групп требований к продукту. Такой подход обычно включает группы (типы, категории) требований, например: системные, программные, функциональные, нефункциональные (в частности, атрибуты качества) и т.п. Классический пример высокоуровневого структурирования групп требований как требований к продукту описан в работах одного из классиков дисциплины управления требованиями – Карла Вигерса.

Область знаний управления требованиями включает 7 секций, каждая из которых представлена в виде ключевых тем. Кроме того, данная область знаний тесно связана со следующими областями:

1. Software Design
2. Software Testing
3. Software Maintenance
4. Software Configuration Management
5. Software Engineering Management
6. Software Engineering Process
7. Software Quality

**Цель этапа 1**

Реализоватьобозначенные в спецификации требования на прикладном программном уровне.

**Введение по теме 2**

Для получения щепы и ее применения на местах аварийных нефтяных разливов необходимо создание технологии ее получения из порубочных остатков и неликвидной древесины.

Перед началом разработки технологии, необходимо рассмотреть вопрос о целесообразности сбора отходов и переработки их в щепу. А так же вопрос технологической линии переработки отходов в продукт и всех составляющих технологии. По мере рассмотрения всех этих вопросов возникает еще один – возможность использования щепы не только как носителя нефтеокисляющей микрофлоры, но и как сорбента нефтепродуктов.

**Цель этапа** Создание технологии переработки неликвидной древесины в щепу. Подбор оптимального размера щепы и глубины ее внесения в почву.

**Введение по теме 3**

В последние годы отмечается рост заболеваемости внутрибольничными инфекциями (ВБИ), которые представляют собой одну из наиболее актуальных проблем здравоохранения во всех странах мира и наносят огромный социально-экономический ущерб. В Национнальной Концепции (утв. Руководителем Федеральной службы по надзору в сфере защиты прав потребителей и благополучия человека, Главный государственный санитарный врач Российской Федерации, 2011г.) отмечено, что ИСМП являются стратегической задачей здравоохранения в силу широкого распространения, негативных последствий для здоровья пациентов, персонала и экономики государства.

Одной из основных причин роста заболеваемости ВБИ является селекция и формирование госпитальных штаммов, обладающих высокой вирулентностью и множественной лекарственной устойчивостью. Известно значительное количество факторов, влияющих на генетическую изменчивость микроорганизмов. Меняющиеся условия окружающей среды вызывает у бактерий состояние стресса, в результате которого меняется способность бактерий размножаться на питательных средах. Под воздействием стрессовых факторов окружающей среды, влияющих на культуральные свойства бактерий, иногда даже значительное количество бактерий могут переходить в жизнеспособное некультурабельное состояние. Установлено, что значительное количество бактериальных клеток, обладающих жизнеспособными патогенными свойствами, не размножаются в лабораторных условиях. Так, при использовании стандартных микробиологических методов исследований только 0,01-0,1% бактерий культурабельны при исследовании проб морской воды. Таким образом, в микробных популяциях формируется определенное количество бактерий не способных размножаться на питательных средах, используемых для накопления и идентификации бактерий, вызывающих инфекционные заболевания у людей.

Цель этапа 1: Выявление в популяции госпитального и спорадического штамма St. Aureus культивируемых и некультивируемых бактерий:

**ОСНОВНАЯ ЧАСТЬ**

**Полученные научные и научно-технические результаты по теме 1**

Определение требований (Definition of a Software Requirement) – в данном случае подразумевается не процесс определения (извлечения, сбора, формирования, формулирования) требований, а дается само понятие “требования”, как такового, и отмечаются его основные характеристики, например, верифицируемость (проверяемость) требования.

Необходимо обратить внимание на следующие определения понятия “требование” (на основе работ Вигерса и стандарта IEEE Standard Glossary of Software Engineering Terminology, 1990):

Условие или возможность, требуемая пользователем для решения задач или достижения целей.

Условие или возможность, которые должны удовлетворяться системой/компонентом системы или которыми система/компонент системы должна обладать для обеспечения условий контракта, стандартов, спецификаций или др. регулирующими документами.

Документальная репрезентация (зафиксированное представление, определение, описание) условий или возможностей, перечисленных в предыдущих пунктах.

Требования к продукту и процессу (Product and Process Requirements) – проводится разграничение соответствующих требований как свойств продукта, который необходимо получить, и процесса, с помощью которого продукт будет создаваться; отмечается, что ряд требований может быть заложен неявно и программные требования могут порождать требования к процессу, например: работа в режиме 24x7 (как бизнес-требование) наверняка приведет к ограничению выбора тех или иных программных средств, платформ развертывания и архитектурных решений; в свою очередь, выбор платформы J2EE (Java 2 Enterprise Edition) и ее реализации в виде конкретного сервера приложений практически наверняка потребует применения модульного тестирования (Unit testing) как практики процесса разработки и JUnit, как инструмента реализации этой практики.

Функциональные и нефункциональные требования (Functional and Non-functional Requirements) – функциональные требования задают “что” система должна делать; нефункциональные – с соблюдением “каких условий” (например, скорость отклика при выполнении заданной операции); часто функциональные требования представляют в виде сценариев (вариантов использования) Use Сase.

Систематизируя работы Вигерса, Лефингвелла и Видрига, Коберна, а также других экспертов, Возможно и необходимо привести классификацию различных категорий (видов) требований и связанных с ними понятий, важнейших с точки зрения их понимания и дальнейшего практического применения:

Потребности (needs) – отражают проблемы бизнеса, персоналии или процесса, которые должны быть соотнесены с использованием или приобретением системы.

**Группа функциональных требований**

Бизнес-требования (Business Requirements) – определяют высокоуровневые цели организации или клиента (потребителя) – заказчика разрабатываемого программного обеспечения.

Пользовательские требования (User Requirements) – описывают цели/задачи пользователей системы, которые должны достигаться/выполняться пользователями при помощи создаваемой программной системы. Эти требования часто представляют в виде вариантов использования (Use Cases).

Функциональные требования (Functional Requirements) – определяют функциональность (поведение) программной системы, которая должна быть создана разработчиками для предоставления возможности выполнения пользователями своих обязанностей в рамках бизнес-требований и в контексте пользовательских требований.

**Группа нефункциональных требований (Non-Functional Requirements)**

Бизнес-правила (Business Rules) – включают или связаны с корпоративными регламентами, политиками, стандартами, законодательными актами, внутрикорпоративными инициативами, учетными практиками, алгоритмами вычислений и т.д. На самом деле, достаточно часто можно видеть недостаточное внимание такого рода требованиям со стороны сотрудников ИТ-департаментов и, в частности, технических специалистов, вовлеченных в проект. Business Rules Group дает понимание бизнес-правила, как “положения, которые определяют или ограничивают некоторые аспекты бизнеса. Они подразумевают организацию структуры бизнеса, контролируют или влияют на поведение бизнеса”. Бизнес-правила часто определяют распределение ответственности в системе, отвечая на вопрос “кто будет осуществлять конкретный вариант, сценарий использования” или диктуют появление некоторых функциональных требований.

В контексте дисциплины управления проектами (уже вне проекта разработки программного обеспечения, но выполнения бизнес-проектов и бизнес-процессов) такие правила обладают высокой значимостью и, именно они, часто определяют ограничения бизнес-проектов, для автоматизации которых создается соответствующее программное обеспечение.

Внешние интерфейсы (External Interfaces) – часто подменяются “пользовательским интерфейсом”. На самом деле вопросы организации пользовательского интерфейса безусловно важны в данной категории требований, однако, конкретизация аспектов взаимодействия с другими системами, операционной средой (например, запись в журнал событий операционной системы), возможностями мониторинга при эксплуатации – все это не столько функциональные требования (к которым ошибочно приписывают иногда такие характеристики), сколько вопросы интерфейсов, так как функциональные требования связаны непосредственно с функциональностью системы, направленной на решение бизнес-потребностей.

Атрибуты качества (Quality Attributes) – описывают дополнительные характеристики продукта в различных “измерениях”, важных для пользователей и/или разработчиков. Атрибуты касаются вопросов портируемости, интероперабельности (прозрачности взаимодействия с другими системами), целостности, устойчивости и т.п.

Ограничения (Constraints) – формулировки условий, модифицирующих требования или наборы требований, сужая выбор возможных решений по их реализации. В частности, к ним могут относиться параметры производительности, влияющие на выбор платформы реализации и/или развертывания, которые, в свою очередь, могут относиться, например, к внешним интерфейсам.

Системные требования (System Requirements) – иногда классифицируются как составная часть группы функциональных требований (не путайте с как таковыми “функциональными требованиями”). Описывают высокоуровневые требования к программному обеспечению, содержащему несколько или много взаимосвязанных подсистем и приложений. При этом, система может быть как целиком программной, так и состоять из программной и аппаратной частей. В общем случае, частью системы может быть персонал, выполняющий определенные функции системы, например, авторизация выполнения определенных операций с использованием программно-аппаратных подсистем.

Необходимо сделать несколько важных замечаний по бизнес-правилам. Бизнес правила, как таковые, являются предметом пристального изучения различных специалистов в области как бизнес-моделирования, так и программной инженерии в целом. Практика разработки программных требований включает идентификацию и описание бизнес-правил как самостоятельных артефактов. Например, методология RUP выделяет отдельный артефакт Business Rule в рамках дисциплины Business Modeling. Все бизнес-правила, в рамках данной дисциплины, идентифицируются и описываются в документе Business Rules Document. При разработке требований, в сценариях Use Cases обычно включается ссылка на уже описанное бизнес-правило. Скотт Амблер так же выделяет бизнес-правило как один из артефактов, который используют в семействе Agile методологий.

В настоящее время разработаны методы и подходы формального представления бизнес-правил, вплоть до формальных языков описания (использование OCL – Object Constraint Language, BRML – Business Rules Markup Language).

Бизнес-правила могут быть не только использованы при определении требований к разрабатываемому ПО, но и могут отдельно оформляться в дизайне ПО (класс или группа классов), отражаясь в конечном итоге в программном коде на определенном языке программирования. Существуют специализированные инструментальные средства и библиотеки, позволяющие разрабатывать и поддерживать приложения, интенсивно использующие бизнес-правила.

Рассматривая бизнес-правила, как артефакты относящиеся к области программных требований можно отметить, вернее дать одно из пояснений, почему БП относят к нефункциональным требованиям: Например, при написании определенного шага в сценарии use case, используется ссылка на бизнес правило: «… система производит расчет стоимости в соответствии с бизнес-правилом BP41 …». В свою очередь данное бизнес-правило может определять алгоритм расчета стоимости. Т.е. налицо «с соблюдением каких условий система делает расчет».

Наравне с представленной классификацией требований, могут использоваться и другие подходы.

Даже в рамках этой классификации, существуют и различные взгляды на ее интерпретацию и детализацию. Например, как результат определения целевой аудитории и в рамках маркетинговой стратегии продвижения тиражируемого решения, возможно определять высокоуровневые возможности (ключевые характеристики, особенности) – “фичи” (features) разрабатываемого продукта. Иногда, такие возможности детализируются в смысле функциональных требований в некоторых agile-техниках, например, FDD – Feature-Driven Development (как вы видите вплоть до самого названия целого комплекса практик, метода разработки).

Вигерс, описывает feature как “множество логически связанных функциональных требований, которые предоставляют определенные возможности для пользователя и удовлетворяют бизнес-целям <организации>”. С точки зрения маркетинга программного обеспечения, как отмечает Вигерс, feature это «группа требований, узнаваемая заинтересованными лицами, которые вовлечены в процесс принятия решения по приобретению ПО – это список отличительных особенностей или возможностей, характеристик, присутствующий в описании продукта». В то же время, определяют feature как “сервисы, которые оказывает система для удовлетворения одной или более потребностей заинтересованных лиц (stakeholders needs)”. Ими же отмечается, что в реальных проектах могут быть не определены stakeholders needs (а их часто выделяют, особенно если у проекта/продукта есть много заинтересованных лиц со своими потребностями, и эти потребности могут носить взаимоисключающий характер), но существовать features могут и самостоятельно (как и stakeholder needs), и конечно же возможно существование и stakeholder needs и features со взаимной трассировкой.

Анализируя различные источники на предмет работы с features, следует отметить следующее:

С точки зрения инженерии требований, features являются самостоятельным артефактом, который может быть соотнесен как с функциональными требованиями, так и с нефункциональными (в т.ч. с ограничениями проектирования или атрибутами качества).

Необходимо также отметить, что Features обладают определенным дуализмом в своей интерпретации, зависимым от контекста конкретного продукта – с одной стороны это может быть «тот самый список характеристик, указанный на коробке продукта» в случае создания «коробочного ПО», с другой стороны это может список высокоуровневых возможностей системы, например при заказной разработке ПО автоматизации бизнес-процессов конкретной организации.

Features могут быть разного уровня детализации – от выражения высокоуровневых возможностей системы (например, «Расчет заработной платы …»), до достаточно конкретных требований (например, «Автоматическое уведомление Клиента по e-mail о резервировании товара на складе»)

Независимые или общие свойства (Emergent Properties) – эти свойства обозначают требования, которые адресованы к системе в целом, и не могут быть соотнесены с отдельными ее элементами. Т.е. данные требования относятся к тому синергетическому эффекту, которым может обладать такая система («целое больше, чем сумма его частей»). Примером может служить требования к «пропускной способности» колл-центра, которая будут зависеть от того, каким образом будут взаимодействовать коммуникационное оборудование, оператор и программное обеспечение в конкретных условиях.

Требования с количественной оценкой (Quantifiable Requirements) – требования, поддающиеся количественному определению/измерению, например, система должна обеспечить пропускную способность “столько-то запросов в секунду”; в то же самое время, крайне важно понимать, что постановка вопроса (то есть формулирование требования) в форме “система должна обеспечить рост пропускной способности” без указания конкретных количественных характеристик является просто некорректно определенным требованием.

При этом, например, требование “система должна вести журнал подключений пользователей” может и должно детализироваться с точки зрения описания информации, которую необходимо сохранять в журнале, однако, такое требование уже не будет являться количественным требованием. А требование с формулировкой “система должна обладать интуитивно-понятным пользовательским интерфейсом” – непроверяем. В определенных случаях, по мнению автора книги, это может выглядеть просто издевкой, даже не являясь изначально таковой – все зависит от точки зрения: например, в устах “целевого” пользователя специализированного программного обеспечения – системного администратора, привыкшего работать в kshell, объясняющего свои потребности аналитику, фиксирующему запросы пользователя и привыкшего оперировать Microsoft Office. Может ли такое требование быть переформулировано и/или детализировано для адекватности интерпретации? Да. Например, так – средний показатель ошибок оператора не должен превышать 2% от объема вводимой информации и 85% пользователей должны дать положительную оценку прототипу пользовательского интерфейса на этапе опытной эксплуатации.

Такие требования должны однозначно отвечать на вопросы, предполагающие ответы с численными величинами – как часто? насколько быстро? в каком количестве? и т.п.

Большинство требований с количественной оценкой относится к атрибутам качества.

В качестве примера можно привести реальное требование, присутствующее в реальном проекте по электронному документообороту: “Система должна производить поиск документов <определенного вида> за время, не превышающее 5 секунд”. Это типичное требование с количественной оценкой, в котором определена верхняя граница диапазона времени, за которое должен быть осуществлен поиск документов. Несомненно, этот атрибут качества системы существует в контексте определенного функционального требования о возможности поиска документов по определенным критериям. И этот контекст или связь должна быть определена либо явно, в рамках иерархии требований, либо посредством трассировки, между требованиями разных видов (функционального и атрибута качества). Примечательно, что Вигерс в своей книге выделяет требования по производительности системы в отдельный вид требований, тем не менее входящих в понятие нефункциональных требований или атрибутов качества.

Системные требования и программные требования (System Requirements and Software Requirements) – данное разделение базируется на определении “системы”, данном INCOSE (International Council on Systems Engineering) “комбинация взаимодействующих элементов <созданная> для достижения определенных целей; может включать аппаратные средства, программное обеспечение, встроенное ПО, другие средства, людей, информацию, техники (подходы), службы и другие поддерживающие элементы”; таким образом, подразумевается, что система является более ёмким понятием, чем программное обеспечения и включает окружение, в котором функционирует ПО, как таковое; отсюда, естественным образом, вытекают требования к системе в целом и программному обеспечению (или программной системе), в частности. Часто в литературе по управлению требованиями встречается описание системных требований как “пользовательских требований” (user requirements), SWEBOK ограничивает применение понятия “пользовательское требование” требованиями к системе конечных пользователей/заказчиков. Системные требования по SWEBOK, в свою очередь, окружают пользовательские требования (или требования других заинтересованных лиц – stakeholders, например, регулирование полномочий) без указания идентифицируемого источника-человека.

**Code Contracts в .NET 4.0**

В .NET 4.0 в рамках CLR появилась такая новинка как Code Contracts. Code Contracts это развитие идеи программирования по контракту (Design by Contract), которая была введена Бертраном Мейером, создателем языка Эйфель.

Контракт – это спецификация компонентов системы. Контрактное программирование — это метод проектирования программного обеспечения. Он предполагает, что проектировщик должен определить формальные, точные и верифицируемые спецификации интерфейсов для компонентов системы.

Основная идея контрактного программирования — это модель взаимодействия элементов программной системы, основывающаяся на идее взаимных обязательств и преимуществ. Как и в бизнесе, клиент и поставщик действуют в соответствии с определённым контрактом. Контракт некоторого метода или функции может включать в себя:

1. конкретные обязательства, которые любой клиентский модуль должен выполнить перед вызовом метода — предусловия, которые дают преимущество для поставщика — он может не проверять выполнение предусловий;
2. конкретные свойства, которые должны присутствовать после выполнения метода — постусловия, которые входят в обязательства поставщика;
3. обязательства по выполнению конкретных свойств — инвариантов, которые должны выполняться при получении поставщиком сообщения, а также при выходе из метода.

Многие языки программирования позволяют учитывать такие обязательства. Контрактное программирование подразумевает эти требования критическими для корректности программ, поэтому они должны быть утверждены при проектировании. Таким образом, контрактное программирование предписывает начинать писать код с написания формальных утверждений корректности (assertions).

В объектно-ориентированном программировании контракт метода обычно включает следующую информацию:

1. возможные типы входных данных и их значение;
2. типы возвращаемых данных и их значение;
3. условия возникновения исключений, их типы и значения;
4. присутствие побочного эффекта метода;
5. предусловия, которые могут быть ослаблены (но не усилены) в подклассах;
6. постусловия, которые могут быть усилены (но не ослаблены) в подклассах;
7. инварианты, которые могут быть усилены (но не ослаблены) в подклассах;
8. (иногда) гарантии производительности, например, временная сложность или сложность по памяти.

При использовании контрактов сам код не обязан проверять их выполнение. Обычно в таких случаях в коде делают жёсткое падение («fail hard»), таким образом облегчая отладку выполнения контрактов. Во многих языках, таких как C, C++, Delphi, PHP, такое поведение реализуется оператором assert. В релизовом варианте кода это поведение может быть сохранено, либо проверки могут быть убраны чтобы повысить производительность.

Юнит-тесты проверяют модуль изолированно от других, проверяя, что модуль удовлетворяет предположениям контракта, а также свои контракты выполняют используемые им модули. Интеграционные тесты проверяют, что модули работают корректно вместе.

Контрактное программирование может повысить уровень повторного использования кода, поскольку обязательства модуля чётко документированы. Контракт модуля можно рассматривать также как способ документации программного обеспечения.

Следует заметить, что Code contracts достаточно давно существуют как проект в рамках Microsoft Research, но с приходом .NET 4.0 для него наступил звездный час и его внесли в CLR.

CLR без сode contracts располагает похожим механизмом. Казалось бы, все что привносят сode contracts можно было реализовать до этого с помощью Debug.Assert, Trace.Assert, или throw. Но тем не менее код с сode contracts выглядит намного чище, и прежние ассерты не давали возможности проводить статическую проверку на этапе компиляции, не было возможности создавать автоматически сгенерированы юнит-тесты, не было возможности сгенерировать документацию для объекта с учетом тех спецификаций, которые они накладывают на объект. Теперь же с появлением code contracts все это возможно.

Преимуществ перед старыми средствами достаточно много.

Code contracts есть 3 видов:

— Preconditions – используется для валидации аргументов

— Postconditions – для проверки состояния по завершению метода, независимо от того нормально он завершился или с исключением

— Object invariants – для проверки, что данные объекта находятся в хорошем состоянии на протяжении жизни объекта

Если одно из этих условий нарушается, то при runtime проверке мы получим ContractException. Существует также возможность подписаться на событие Contract.ContractFailed чтобы либо продолжить/прервать выполнение, либо отреагировать на нарушение контракта. Например подписку на ContractFailed можно использовать, например для того чтобы юнит-тесты не остановили процесс сборки тестов на билд машине.

Предусловия(preconditions) реализованы используя Contract.Requires и специфицируют, что OrderItem не должен быть null или уже заказан. Пост условия — Contract.Ensures, что OrderItem должен быть заказан и цена заказа должна возрасти. И есть еще инвариант. Инвариант реализуются с помощью методов с атрибутом ContractInvariantMethod при чем их может несколько но в итоге они будут объединены. Так вот, инварианты будут вызваны в конце вызова каждого public метода экземпляра класса. Условия используемые в preconditions, preconditions и invariants не должны изменять состояние объекта. Метод Contract.Requires<ArgumentNullException> например в случае неверного контракта выбросит соотвествущее исключение.

Паттерн «Спецификация» является паттерном поведения приложения и был представлен Эриком Эвансом и Мартином Фаулером. Данный паттерн отвечает за отделение логики фильтрации (валидации) сущности от самой сущности с целью сделать код более читабельным, облегчить внесение изменений и уменьшить дублирование. Эта ответственность инкапсулирована внутри отдельных классов — спецификаций.

Паттерн «Спецификация» применяется при наличии однотипных проверок условий в разных частях приложения.

Объекты-спецификации, как правило, содержат метод IsMacth, принимающий доменный объект и возвращающий boolean. Спецификация инкапсулирует предикат, используемый для тестирования объекта на предмет соответствия определенному критерию.

**Полученные научные и научно-технические результаты по теме 2**

При решении вопроса о целесообразности сбора отходов и переработки их в щепу, необходимо учитывать возможность и расстояние доставки щепы до места использования, технические характеристики техники для сбора порубочных остатков и переработки их в щепу.

При разработке и строительстве месторождений на Северных территориях лесные территории вырубались в огромных количествах. Вся вырубка складировалась вдоль дорог или на специализированных полигонах и находилась там длительное время. Помимо вырубки, Северные территории понесли большие ущербы в плане лесных насаждений, когда начались аварии при разработках и транспортировках нефти. Так что материала для изготовления щепы из порубочных остатков и неликвидной древесины на Севере в большом количестве. Для изготовления самой щепы необходимо оборудование, широкий спектр предложений на рынке обеспечит оптимальное сочетание «цена-качество» (при доступной цене установки, ее транспортировки к месту использования – оптимальную производительность в час). При этом предлагаемая технология подразумевает под собой исключение транспортировке на большие расстояния, а применение ее в местах изготовления щепы или в ближайших районах.

Виды измельченной древесины определяются ГОСТ 23246-78 «Древесина измельченная. Термины и определения».

Измельченная древесина – это древесные частицы различной формы и величины, получаемые в результате механической обработки. Указанный ГОСТ определяет следующие **виды измельченной древесины**:

**- технологическая щепа**;

**- зеленая щепа**;

**- топливная**;

**- дробленка**;

**- древесная стружка**;

**- древесные опилки**;

**- технологические древесные опилки**;

**- древесная мука**;

**- древесная пыль**.

Различают несколько уровней или степеней дробления древесины:

**первичное измельчение древесины**, при котором формируется в основном длина частиц. Наиболее распространенный способ – получение щепы в рубительных машинах различного типа. Первичное измельчение – наиболее энергозатратная операция, поскольку на этом этапе разрыв волокон древесины происходит поперек них – в направлении максимальной прочности древесины;

**доизмельчение древесных частиц**, то есть дополнительное измельчение щепы, стружки в стружечную или волокнистую массу, частицы которой имеют вполне определенные размеры.

Продуктом первичного измельчения с нормируемыми параметрами является технологическая щепа – сыпучий материал, частицы которого в среднем имеют примерно 25 мм в длину, 20-30 мм в ширину и 3-6 мм в толщину. Весьма универсальный по применению.

Для получения кондиционного продукта с определенным содержанием фракций щепу сортируют. Для сортировки обычно используют виброустановки с наклонными ситами, имеющими определенные размеры ячеек. Полученную после сортирования щепу добавляют к основной массе технологической щепы.

**Технология производства щепы**

В основе технологии получения щепы лежит измельчение порубочных остатков и неликвидной древесины на специальной технике.

Весь процесс производства условно можно разделить на несколько этапов:

1) Измельчение

Различные рубительные машины (Дробилки) измельчают древесное сырьё до фракции с размерами не более 25х25х2 мм для дальнейшей сушки. Лучше всего для снижения энергозатрат на сушку измельчать до более мелкой фракции.

2) Сушка

Сушка щепы необходима, так как длительное кучевое хранение сыпучей древесины чревато ее загниванием. Загнивание всегда сопровождается повышением температуры внутри кучи, вплоть до самовозгорания древесины Сухая щепа более емкая для следующего этапа - нанесения бактериального препарата.

3) Нанесение бактериального препарата

Нанесение бактериального препарата проводится специальным оборудованием, путем равномерного разбрызгивания на щепу. Бактериальный препарат наносится в виде суспензии в комплексе с минеральными удобрениями.

**Увеличение нефтеемкости древесных продуктов**

Нефтеемкость продуктов переработки непосредственно самой древесины, например опила, зависит от его размера и гидрофобности. Методика определения нефтеемкости заключается в следующем. В стакан (чашку Петри) наливают исследуемую водную фазу в объеме, достаточном для закрытия дна и установления водного зеркала. На поверхность воды приливают заданный объем нефти, после чего небольшими порциями на поверхность нефти насыпают щепу, которую стеклянной палочкой перемешивают с нефтью. Добавление щепы производят до тех пор, пока вся нефть не поглотится. Исходя из веса насыпанной щепы и объема поглощенной нефти рассчитывают ее нефтеемкость. Степень гидрофобности щепы определяют как отношение количества неутонувшего сорбента к общему количеству и выражают в %. Результаты исследования влияния размера на нефтеемкость приведены в табл. 1.1.

Естественная нефтеемкость щепы и ее гидрофобность могут быть изменены в процессе обработки сырья различными веществами. Характер влияния минеральных химических соединений на нефтеемкость и гидрофобность древесных продуктов отражена в табл. 1.2. Выбор номенклатуры кислот обусловлен их высокой реакционной способностью и широкой распространенностью. В качестве исходного сырья взята щепа с размером частиц 0,5...2,5 мм. Способ воздействия минеральных кислот на щепу заключался в добавлении к ней исследуемых растворов, выдержке на время реагирования и последующей сушке.

Таблица 1.1

Нефтеемкость древесных продуктов

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика материала | Размер частиц, мм | Гидрофобность, % | Нефтеемкость, кг/кг |
| Древесная пыль | менее 0,5 | 80 | 7,7 |
| Опил мелкий | менее 1 | 40 | 5,2 |
| Опил средний | 1...3 | 60 | 4,8 |
| Опил крупный | 3...7 | 80 | 4,6 |
| Стружка | 7...15 | 100 | 4,0 |

Таблица 1.2

Нефтеемкостъ и гидрофобностъ опила после воздействия на него минеральными кислотами

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Реагент воздействия на опил | Концентрация реагента, % | Гидрофобность, % | Нефтеем­костъ, кг/кг | Увеличение нефтеемко­сти, % |
| Исходный опил |  | 50 | 4,3 |  |
| Соляная кислота | 10 | 50 | 5,0 | 16 |
| Фосфорная кислота | 10 | 70 | 5,3 | 23 |
| Азотная кислота | 10 | 60 | 4,4 | 2 |
| Серная кислота | 1 | 70 | 5,2 | 21 |
| Серная кислота | 5 | 80 | 6,2 | 44 |
| Серная кислота | 10 | 80 | 6,6 | 53 |
| Серная кислота | 25 | 70 | 5,0 | 16 |
| Серная кислота | 70 | 60 | 4,0 | -7 |
| Гидроксид натрия | 10 | 30 | 5,0 | 16 |

Установлено, что путем воздействия на щепу минеральными химическими соединениями можно достигнуть некоторого увеличения нефтеемкости и гидрофобности исходного сырья.

Характер влияния на нефтеемкостъ и гидрофобность щепы органических соединений отражен в табл. 1.3.

Как видно из приведенных результатов, характер воздействия органических соединений на исходное сырье многообразен, а диапазон достигаемых за счет этого результатов довольно широк. В связи с этим открываются новые возможности и перспективы использования органических соединений для повышения сорбционной способности щепы как исходного материала для получения нефтяных сорбентов.

Получение абсорбента из шлиф-пыли заключается в ее гидрофобизации. Для этого смолы, содержащиеся в шлиф-пыли, переводят в раствор, нагревая со щелочью при интенсивном перемешивании, после чего осаждают, добавляя эквимолекулярное по отношению к ним количество алюмокалиевых квасцов [A1K(SО4)2\*12H2О]. Для осаждения смол можно использовать любые растворимые соли алюминия. Алюмокалиевые квасцы выбраны как наиболее дешевые из доступных солей. Гидрофобизированную шлиф-пыль отделяют от раствора на фильтре, затем сушат.

Поскольку древесина содержит в своем составе до 6...8 % смол, переходящих в водный раствор при щелочной экстракции, то для удешевления процесса гидрофобизации щелочную экстракцию проводят белым щелоком - модифицированным отходом целлюлозно-бумажного производства, содержащим 10...15 % активной щелочи. Так, например, белый щелок варочного производства Сыктывкарского лесопромышленного комбината содержит 10... 11 % активной щелочи.

Таблица 1.3

Нефтеемкость и гидрофобность опила после воздействия на него органическими соединениями

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Реагент воздействия на опил | Количество реагента, % | Гидрофобность, % | Нефтеемкость, кг/кг | Увеличение  Нефтеемкости, % |
| Исходный опил |  | 50 | 4,3 |  |
| Сульфаниловая | 10 | 50 | 5,0 | 16 |
| кислота |  |  |  |  |
| Сульфосапициловая | 10 | 50 | 6,3 | 46 |
| кислота |  |  |  |  |
| Триэтанол- | 10 | 0 | 2,0 | -115 |
| гидрохлорид |  |  |  |  |
| Машинное | 10 | 80 | 3,6 | -19 |
| масло |  |  |  |  |
| Бензольная | 10 | 80 | 5,0 | 16 |
| вытяжка торфа |  |  |  |  |
| Глицерин | 10 | 70 | 4,5 | 5 |
| Парафин | 10 | 100 | 8,3 | 93 |

Абсорбент имеет структуру рыхлого порошка с насыпной массой 100...200 кг/м3. Для придания абсорбенту формы гранул в раствор щелочи вместе со шлиф-пылью добавляют 2...3 % от массы шлиф-пыли целлюлозы. Форма гранул в некоторых случаях предпочтительна, так как облегчает сбор сорбента с поверхности акваторий.

Полученные по вышеописанной технологии сорбенты имеют нефтеемкость в пределах 4,6...5,6 кг/кг, а водопоглощение — 1,2...2,1 кг/кг сорбента.

Технология низкотемпературного процесса пиролиза лигнинсодержащего сырья в среде химических реагентов позволяет получить макропористый инертный углеродный поглотитель нефтепродуктов. Для его производства древесные опилки вначале подвергают сушке при температуре 70...80 °С *в* среде азота, затем для получения гидрофобной и олеофильной поверхности поглотителя осуществляют стадию карбонизации древесных опилок при температуре 230...280 °С. В процессе карбонизации древесные опилки взаимодействуют со смесью галогенсодержащими, серосодержащими и кислородсодержащими веществами. В качестве галогенсодержащего вещества используют фтористый калий или фтористый аммоний в количестве 0,3...1 масс. % от исходной массы опилок, серосодержащего вещества - персульфат калия или элементарную серу в количестве 3,5...6 масс. %, а в качестве кислородсодержащего вещества - перманганат калия в количестве 0,01...0,5 масс. %.

Образующийся на стадии карбонизации макропористый инертный углеродный поглотитель охлаждают в среде инертного газа и выгружают.

Полученный поглотитель имеет следующие параметры:

1. размер частиц поглотителя - 0,3...6 мм;
2. удельная поверхность - 0,1... 1 *м2/г;*
3. свободный объем макропор - 3,7...4,2 см3/г;
4. средний радиус пор около 1000 нм, при следующей функции их распределения по размерам, %:
5. поры радиусом менее 10 нм - 1...5;
6. поры радиусом 10...1000 нм - 4...10;
7. поры радиусом 100...5000 нм - 75...80;
8. поры радиусом более 5000 нм - 10...15.

Нефтеемкостъ полученного сорбента составляет 4,4 кг/кг.

При низкой величине водопоглощения 0,005 кг/кг сорбент способен долгое время находиться на поверхности воды и сорбировать даже радужную пленку. Низкая степень нефтеотдачи (через 24 часа всего 0,0007 %) не позволяет нефти и нефтепродуктам вытесняться впоследствии водой и вновь всплывать на поверхность.

**Полученные научные и научно-технические результаты по теме 3**

Общие сведения о культивировании золотистого стафилококка в лабораторных условиях

Для диагностики стафилококковых заболеваний используют бактериологический метод исследования. В случае, если исследуемый материал содержит постороннюю микрофлору для первичного накопления возбудителя используют элективные питательные среды: желточно-солевой агар Чистовича или молочно-солевой агар Петровича. Желточно-солевой агар позволяет более четко дифференцировать патогенные и непатогенные штаммы стафилококка по сравнению с кровяным агаром. Исследуемый материал втирают в поверхность питательной среды и инкубируют 1-2 суток при температуре 37ºС. В зависимости от исследуемого материала (гной открытых ран, отделяемое ран или свищей мокроты, рвотных масс, промывных вод желудка, пищевых продуктов, изъятых при пищевых отравлениях, кровь больного) используют различные питательные среды, сроки и условия культивирования (аэробные и анаэробные). Накопленную чистую культуру бактерий на скошенном агаре идентифицируют. Для внутривидовой дифференциации стафилококков использовали коагулазный тест, способность ферментировать маннит в анаэробных условиях, способность синтезировать термостабильную ДНК-азу, способность агглютинировать сенсибилизированные эритроциты барана. Для экспресс-идентификации при меняли тест латекс-агглютинации с использованием коммерческих наборов частиц латекса, нагруженных антителами. Для выявления источника стафилококковой инфекции использовали один из методов эпидемиологического маркирования – фаготипирование. Выделение бактерий одного фаговара от разных больных указывало на общий источник заражения. Для дифференциации спорадических и госпитальных штаммов стафилококка использовали патент Л.Б.Козлова (2005г.).

Разработка способа выделения некультивируемых бактерий S.aureus

Как было отмечено выше в периодической научной литературе имеются сообщения о наличии в популяциях бактерий жизнеспособных, но потерявших способность размножаться на питательных средах, используемых в лабораторных условиях и для перехода этих бактерий в культивируемое состояние необходимо воздействовать на бактерии стрессовыми факторами, обеспечивающими переход бактерий из некультурабельного состояния в культурабельное. Таким образом, имеются теоретические обоснования возможности перехода некультивируемых бактерий в лабораторных условиях в культивируемое состояние. В частности, установлено, что холодовое воздействие на некультивируемые бактерии вызывает переход их в культивируемое состояние. Проведены следующие исследования: на культуру стафилококка воздействовали температурой 18-20ºС и +4ºС в течение 24 и 48 часов. Наибольшее количество бактерий перешедших из некультивируемого состояния в культивируемое отмечалось при стрессовом воздействии температуры +4ºС в течение 48часов и последующее культивирование микробной взвеси при 37ºС позволило выделить чистую культуру некультивируемых бактерий.

На основании проведенных экспериментальных исследований предложен способ выделения некультивируемых бактерий в лабораторных условиях. Предложенный способ заключается в следующем: проводят посев исследуемого материала, выделенного от больных и из различных объектов ЛПУ, на скошенный мясопептонный агар (МПА) и культивируют бактерии при температуре 37ºС в течение 24 часов. Для определения количества бактерий выросших на МПА проводят посев серийных разведений бактерий на чашки Петри с желточно-солевым агаром и определяют титр микробной взвеси через 24 часа по количеству выделенных изолированных колоний стафилококка. Из культуры бактерий, выросших на МПА, по стандарту мутности в 5 единиц, соответствующей 500 000 мк в 1 мл, получают взвесь бактерий, из которой готовят серийные разведения до 107. Серийные разведения бактерий выдерживают 48 часов при +4ºС и определяют концентрацию бактерий по результатам посева бактерий на чашки Петри с желточно-солевым агаром. Наличие роста бактерий из разведений 106- 107 степени свидетельствует о наличии в популяции некультвируемых в лабораторных условиях бактерий. По результатам проведенных исследований подана заявка на изобретение «Способ выделения некультивируемых бактерий стафилококка». Авторы: Л.Б.Козлов, …И.И.Бутков. Заявка № 2011146052 от14.11.2011 г.

Исследования по выделению и идентификации выделенных чистых культур стафилококка проведены на базе бактериологической лаборатории Тюменского филиала ФГУЗ «Центр гигиены и эпидемиологии по железнодорожному транспорту» под руководством врача-микробиолога высшей категории Диц Е.В. Идентификацию бактерий проводили до вида, используя метод эпидемиологического маркирования – фаготипирование.

Результаты опытов по определению некультивируемых бактерий S.aureus

Примеры практического использования предложенного способа

Пример 1.

В декабре 2010 - январе 2011 гг. возникла вспышка, вызванная госпитальным штаммом стафилококка в 3-м роддоме г. Тюмени. От больных выделен возбудитель инфекции. После идентификации возбудителя инфекции провели посев выделенной культуры стафилококка на скошенный МПА. Культивировали бактерии при температуре 37ºС в течение 24 часов. Для определения количества бактерий выросших на МПА провели посев серийных разведений бактерий на чашки Петри с желточно-солевым агаром. Через 24 часа роста бактерий титр микробной взвеси составил 9,0.105. Из культуры бактерий, выросших на МПА, по стандарту мутности в 5 единиц, соответствующей 500000 мк в 1 мл, приготовили серийные разведения до 107. Серийные разведения бактерий выдержали 48 часов при +4ºС и определили концентрацию бактерий по результатам посева бактерий на чашки Петри с желточно-солевым агаром. Концентрация бактерий составила 47.105. Следовательно, около 42 микробных клеток из состояния некультивируемого перешло в культивируемое состояние в лабораторных условиях.

Пример 2.

От больного с гнойно-воспалительным процессом после идентификации возбудителя выделен спорадический штамм стафилококка. Бактериологической петлей провели посев выделенных бактерий на скошенный МПА и культивировали при температуре 37ºС в течение 24 часов. Для определения количества бактерий, выросших на МПА, провели посев серийных разведений бактерий на чашки Петри с желточно-солевым агаром. Через 24 часа титр микробной взвеси составил 10,4.105. Из культуры бактерий, выросших на МПА, по стандарту мутности в 5 единиц, соответствующей 500 000 мк в 1 мл, получили взвесь бактерий, из которой приготовили серийные разведения до 107. Серийные разведения бактерий выдержали 48 часов при +4ºС и определили концентрацию бактерий по результатам подсчета выросших колоний на чашки Петри с желточно-солевым агаром. Концентрация бактерий составила 60.105. Следовательно, около 50 микробных клеток из состояния некультивируемого перешло в культивируемое состояние в лабораторных условиях. Предложенный способ позволил выделять из микробных популяций одного вида бактерий некультивируемые в лабораторных условиях бактерии.

Всего проведено 10 исследований. Количество некультивируемых бактерий в популяции госпитального штамма стафилококка восстановивших способность размножаться на питательных средах составило 39±4,3 мк, а спорадического штамма стафилококка – 53±2,5 мк.

Анализ проведенных исследований показал, что в чистых культурах госпитальных и спорадических штаммах стафилококка содержится значительное количество некультивируемых бактерий. В микробных популяциях стафилококка количество некультивируемых бактерий превышает число культивируемых бактерий на 1-2 логарифма.

**ЗАКЛЮЧЕНИЕ**

**Заключение по теме 1**

Выбранный в качестве реализации требований к ПО механизм Code Contracts, реализующий парадигму контрактного программирования в среде .NET обеспечит требуемый уровень качества исполнения требований и контроля над исключительными ситуациями.

**Заключение по теме 2**

В ходе рассмотрения создания технологии получения щепы из порубочных остатков и неликвидной древесины, можно сказать, что разработка данной технологии является целесообразной, так как Северные территории в периоды разработки и строительства месторождений были завалены вырубками. Вырубка деревьев проводилась на огромных территориях будущих месторождений, в связи с тем, что целью вырубки были огромные пустые территории для строительства месторождений, ни у кого не возникало идеи реализовывать рубленную древесину. По мере разработки новых месторождений, а так же последующей тренировки нефтепродуктов, происходят аварийные разливы. Для ликвидации которых необходимо отсутствие любой растительности на данной территории. Соответственно вырубка этих деревьев тоже складировалась вдоль дорог или транспортировалась на полигон.

В процессе рассмотрения создания технологии, возникла мысль использования щепы не только как носителя субстрата микроорганизмов, но так же, как и дополнительного нефтяного сорбента, в данном отчете рассмотрено, как увеличить нефтеемкость щепы. Оптимальным размером щепы является щепа размером не более 25мм Х 25 мм, а по оптимальной глубине внесения щепы, необходимо провести ряд лабораторных испытаний, и выбрать наиболее оптимальный вариант.

**Заключение по теме 3**

Предложенный способ позволяет при одинаковых затратах на проведение микробиологических лабораторных исследований по сравнению со стандартными методами повысить эффективность выделения возбудителей стафилококковой инфекции, уменьшить количество ложноотрицательных результатов исследований. В результате перехода некультивируемых бактерий в культивируемое состояние снижается количество ложноотрицательных результатов исследований. Выделение некультивируемых бактерий в лабораторных условиях позволит более качественно проводить специфическое лечение пациентов учитывая антибиотикорезистентность некультивируемых в лабораторных условиях бактерий. В эпидемиологическом плане предложенный метод позволяет более эффективно расследовать эпидемические вспышки внутрибольничных инфекций, вызываемых госпитальными штаммами, что дает основание для проведения в полном объеме противоэпидемических мероприятий в лечебно-профилактических учреждениях (ЛПУ).

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

**Список использованной литературы по теме 1**

1. Арлоу Д., Нейштадт А. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание. – Пер с англ. – СПб.: Символ-Плюс, 2007. – 624с, ил.

2. Гудлиф П. Ремесло программиста. Практика написания хорошего кода. – Пер. с англ. – СПб.: Символ-Плюс, 2009 – 704 с., ил.

3. Купер А., Рейман Р., Кронин Д. Алан Купер об интерфейсе. Основы проектирования взаимодействия – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 688 с., ил.

4. Мацяшек Лешек А. Анализ и проектриование информационных систем с помощью UML 2.0, 3-е изд. – Пер. с англ. – М ООО «И.Д. Вильямс», 2008 – 816 с., ил. – Парал. тит. англ.

5. Лайза Криспин, Джанет Грегори Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical Guide for Testers and Agile Teams — М.: «Вильямс», 2010. — 464 с. — (Addison-Wesley Signature Series). — 1000 экз. — ISBN 978-5-8459-1625-9.

6. Канер Кем, Фолк Джек, Нгуен Енг Кек Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений — Киев: ДиаСофт, 2001. — 544 с. — ISBN 9667393879.

7. Калбертсон Роберт, Браун Крис, Кобб Гэри Быстрое тестирование — М.: «Вильямс», 2002. — 374 с. — ISBN 5-8459-0336-X.

8. Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения — М.: БИНОМ, 2008. — 368 с. — ISBN 978-5-94774-825-3.

9. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем — СПб.: Питер, 2004. — 320 с. — ISBN 5-94723-69

**Список использованной литературы по теме 2**

<#0902 0902#>

**Список использованной литературы по теме 3**

1. Ахмедова Р.Р. с соавт. /Микробиологический мониторинг объектов окружающей среды в обеспечении эпидемиологического надзора за внутрибольничными инфекциями // Тез. докл. II Российск. научно–практ. конф. с межд. участием. – М., 1999. – С. 28.

2. Волянський. Ю. Л. Некультурабельний стан аспорогенних бактерій: теоретичні аспекти проблеми та її практична значущість // Інфекційні хвороби. - 2004. - № 1. - С. 5-9.

3. Головлев Е.Л. Другое состояние неспорулирующих бактерий // Микробиология. - 1998. - № 6. - С. 725-735

4. Лабинская А.С. Микробиология с техникой микробиологических исследований. М., Медицина, 1978, 394 с.

5. Медицинская микробиология, вирусология и иммунология /ред. А.А.Воробьева. – М., 2004. – 691 с.

6. Митрофанова Н.Н., Мельников В.Л., Слетов А.М. Результаты мониторинга видового состава и основных биологических характеристик микроценозов многопрофильного стационара // Известия высших учебных заведений. Поволжский регион, 2009. - № 4. С. 90-97.

7. Национальная концепция профилактики инфекций, связанных с оказанием медицинской помощи. – М.: Минздрав России, 2011.– 22 с.

8. Патент RU №2245922. Опубл. 10.02.2005. Бюл. № 4. «Способ выявления госпитальных штаммов». Правообладатель - Л.Б.Козлов.

9. Патент RU № 2285257. Опубл. 10.10.2006. Бюл. №28. «Способ определения концентрации микробных клеток по стандарту мутности».

10. Colwell R. R. Bacterial death revisited // Nonculturable microorganisms in the environment / - Ed. D. J. Grimes. - Washington, D.C.. ASM Press, 2000. - p. 325– 342.

11. Kogure K., Simidu U., Taga N. Atentative direct microscopic method for counting living marine bacteria // Can. J. Microbiol. – 1979. – V. 25. P.415-420.

12. Rollins D., Colwell R. Viable but nonculturable stage of Campylobacter jejuni and its role in survival in the natural aquatic environment // Appl. Environ. Microbiol. – 1986. – Vol. 52. – P. 531-538.

13. Staley J., Konopka A. Measurement of in situ activities of nonphotosynthetic microorganisms in aquatic and terrestrial habitats // Annu. Rev. Microbiol. – 1985. –Vol. 39 – P. 321-346.

**ПРИЛОЖЕНИЯ**

**Приложения к работе по теме № 1**

<#1001 1001#>

**Приложения к работе по теме № 2**

<#1002 1002#>

**Приложения к работе по теме № 3**

|  |  |  |  |
| --- | --- | --- | --- |
| **1** |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  | **1** |