Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

**КУРСОВАЯ РАБОТА**

**ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «ПОКУПКА БИЛЕТОВ В КИНОТЕАТРЕ»**

**ПМ.05 «ПРОЕКТИРОВАНИЕ И разработкА ИНФОРМАЦИОННЫХ СИСТЕМ»**

**МДК 05.03 «ТЕСТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ»**

**09.02.07 «ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ»**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  | **/** | Румянцев И.Н. |
|  | *подпись* |  | *И.О. Фамилия* |
| 02.09.2021 г. |  |  |  |
| Руководитель |  | **/** | *Н.К. Коровина* |
|  | *подпись* |  | *И.О. Фамилия* |
| 02.09.2021 г. |  |  |  |

Тольятти, 2022

Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

|  |  |  |
| --- | --- | --- |
|  |  | Утверждаю:  Заместитель директора по УР  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.С. Киронова  *« » 2022 г.* |

**ЗАДАНИЕ**

на курсовую работу

**по ПМ.05 Проектирование и разработка информационных систем модуля, выполняемой в рамках МДК.05.03 Тестирование информационных систем**

студента группы ИСП-31

*Румянцева Ивана Николаевича*

*Фамилия Имя отчество студента*

|  |  |
| --- | --- |
| Тема курсовой работы : | «Покупка билетов в кинотеатре» |

1. Содержание задания:

1.1 Тестирование информационной системы.

1. Исходные данные:

Исходные данные для практической реализации автоматизированной информационной системы (АИС) берутся из различных информационных источников (Интернет-ресурсы, печатные издания, периодика и др.).

1. Содержание курсовой работы

Введение

1 Анализ методов тестирования

1.1 Критерии и принципы тестирования

1.2 Методы тестирования

2 Тестирование информационной системы «Покупка билетов в кинотеатре»

2.1. Разработка тестовой документации (тест-дизайн)

2.2. Разработка тестовых сценариев

Заключение

Список использованных источников

Приложения

Дата выдачи задания: « » 202 г.

Дата сдачи работы на отделение: « » 202 г.

Руководитель курсового(ой) проекта(работы) \_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись расшифровка подписи

Государственное бюджетное профессиональное образовательное учреждение

Самарской области

«Тольяттинский социально-экономический колледж»

**КАЛЕНДАРНЫЙ ПЛАН**

выполнения курсовой работы

Студентом 3 курса ИСП-31 группы Румянцевым И.Н.

*Фамилия, И.О.*

По теме «Покупка билетов в кинотеатре»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  этапа  работы | Содержание этапов работы | Плановый срок выполнения этапа | Планируемый объем выполнения  этапа, % | Отметка  о  выполнении  этапа |
| 1 | Выбор, обоснование темы и объекта исследования | Январь 2022 | 5% |  |
| 2 | Утверждение темы, согласование плана. Введение, библиография | Январь 2022 | 10% |  |
| 3 | Изучение и анализ информационных материалов по теме | Февраль 2022 | 15% |  |
| 4 | Обоснование актуальности выбранной темы применительно к профессиональной деятельности (введение) | Февраль 2022 | 20% |  |
| 5 | Изложение материала основной части по теме курсовой работы | Февраль 2022 | 20% |  |
| 6 | Подведение итогов проведенного анализа, формулировка выводов УИР применительно к профессиональной деятельности (заключение) | Март 2022 | 20% |  |
| 7 | Оформление работы и сдача на проверку | Март 2022 | 10% |  |
| 8 | Защита работы |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  | **/** | **Румянцев И.Н.** |
|  | *подпись* |  | *И.О. Фамилия* |
| 02.09.2021 г. |  |  |  |
| Руководитель |  | **/** | *Н.К. Коровина* |
|  | *подпись* |  | *И.О. Фамилия* |
| 02.09.2021 г. |  |  |  |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc100908991)

[1 АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ 6](#_Toc100908992)

[1.1 Критерии и принципы тестирования 6](#_Toc100908993)

[1.2 Методы тестирования 7](#_Toc100908994)

[2 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «ПОКУПКА БИЛЕТОВ В КИНОТЕАТРЕ» 10](#_Toc100908995)

[2.1. Разработка тестовой документации (тест-дизайн) 10](#_Toc100908996)

[2.2. Разработка тестовых сценариев 13](#_Toc100908997)

[ЗАКЛЮЧЕНИЕ 16](#_Toc100908998)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc100908999)

[ПРИЛОЖЕНИЯ 17](#_Toc100909000)

# ВВЕДЕНИЕ

Процесс автоматизации начался намного раньше, чем нам могло бы казаться, автоматизация на самом деле появилась практически сразу же с возникновением производства, а само по себе производство существует уже так давно, что точно никто и не скажет. Высокая экономическая эффективность, технологическая целесообразность и часто эксплуатационная необходимость способствовали широкому распространению автоматизации.

Автоматизированная информационная система данных осуществляет хранение, редактирование, ведение, создание диаграмм и отчетов.

Актуальность темы: определяется тем, что в связи с моральным и физическим устареванием платформы, форматов невозможно использовать существующую автоматизированную информационную систему, которая замедляет работу, частыми техническими сбоями в системе и определяется актуальность разработки информационной системы книжного магазина.

Все выше изложенное в целом на теоретико-методологическом уровне определило проблему настоящего исследования: разработка программного модуля ИС «Покупка билетов в кинотеатре».

1. Анализ предметной области.
2. Разработка функциональной модели.
3. Разработка логической и физической структуры.
4. Разработка информационной системы «Покупка билетов в кинотеатре».
5. Разработка интерфейса ИС.
6. Разработка руководства пользователя.
7. Тестирование разработанного ПП.

Методы исследования: Для разработки информационной системы будут применяться методы анализ деятельности предприятия с выявлением его функции, а также функциональное моделирование системы.

# 1 АНАЛИЗ МЕТОДОВ ТЕСТИРОВАНИЯ

## 1.1 Критерии и принципы тестирования

Структурные критерии используют информацию о структуре программы, что предполагает знание исходного текста программы или спецификации программы в виде потокового графа управления. Структурные критерии базируются на основных элементах графа управления - операторах, ветвях и путях.

* Условие критерия тестирования команд (критерий С0) - набор тестов в совокупности должен обеспечить прохождение каждой команды не менее одного раза.
* Условие критерия тестирования ветвей (критерий С1) - набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза.
* Условие критерия тестирования путей (критерий С2) - набор тестов в совокупности должен обеспечить прохождение каждого пути не менее 1 раз.

Принцип 1 — Тестирование демонстрирует наличие дефектов (Testing shows presence of defects). Тестирование может показать, что дефекты присутствуют, но не может доказать, что их нет. Тестирование снижает вероятность наличия дефектов, находящихся в программном обеспечении, но не гарантирует их отсутствия.

Принцип 2 — Исчерпывающее тестирование недостижимо (Exhaustive testing is impossible). Полное тестирование с использованием всех комбинаций вводов и предусловий физически невыполнимо, за исключением тривиальных случаев. Вместо исчерпывающего тестирования должны использоваться анализ рисков и расстановка приоритетов, чтобы более точно сфокусировать усилия по тестированию.

Принцип 3 — Раннее тестирование (Early testing). Чтобы найти дефекты как можно раньше, активности по тестированию должны быть начаты как можно раньше в жизненном цикле разработки ПО или системы, и должны быть сфокусированы на определенных целях.

Принцип 4 — Скопление дефектов (Defects clustering). Разные модули системы могут содержать разное количество дефектов – то есть плотность скопления дефектов в разных частях кода может отличаться. Усилия по тестированию должны распределяться пропорционально фактической плотности дефектов. В основном, большую часть критических дефектов находят в ограниченном количестве модулей.

Принцип 5 — Парадокс пестицида (Pesticide paradox). Если одни и те же тесты будут прогоняться много раз, в конечном счете этот набор тестовых сценариев больше не будет находить новых дефектов. Чтобы преодолеть этот «парадокс пестицида», тестовые сценарии должны регулярно рецензироваться и корректироваться, новые тесты должны быть разносторонними, чтобы охватить все компоненты программного обеспечения, или системы, и найти как можно больше дефектов.

Принцип 6 — Тестирование зависит от контекста (Testing is concept depending). Тестирование выполняется по-разному в зависимости от контекста. Например, программное обеспечение, в котором критически важна безопасность, тестируется иначе, чем новостной портал.

Принцип 7 — Заблуждение об отсутствии ошибок (Absence-of-errors fallacy). Отсутствие найденных дефектов при тестировании не всегда означает готовность продукта к релизу. Система должна быть удобна пользователю в использовании и удовлетворять его ожиданиям и потребностям.

## 1.2 Методы тестирования

Широко используемыми методами тестирования являются модульное тестирование, интеграционное тестирование, приемочное тестирование, и тестирование системы. Программное обеспечение подвергается этим испытаниям в определенном порядке.

* модульное тестирование;
* интеграционное тестирование;
* системное тестирование;
* приемочные испытания;

**Модульное тестирование.**

В первую очередь проводится модульный тест. Как подсказывает название, это метод испытания на объектном уровне. Отдельные программные компоненты тестируются на наличие ошибок. Для этого теста требуется точное знание программы и каждого установленного модуля. Таким образом, эта проверка осуществляется программистами, а не тестерами. Для этого создаются тест-коды, которые проверяют, ведет ли программное обеспечение себя так, как задумывалось.

**Интеграционное тестирование.**

Отдельные модули, которые уже были подвергнуты модульному тестированию, интегрируются друг с другом, и проверяются на наличие неисправностей. Такой тип тестирования в первую очередь выявляет ошибки интерфейса. Интеграционное тестирование можно осуществлять с помощью подхода "сверху вниз", следуя архитектурному сооружению системы. Другим подходом является подход «снизу вверх», который осуществляется из нижней части потока управления.

**Системное тестирование.**

В этом тестировании, вся система проверяется на наличие ошибок и багов. Этот тест осуществляется путем сопряжения аппаратных и программных компонентов всей системы, и затем выполняется ее проверка. Это тестирование числится под методом тестирования "черного ящика", где проверяются ожидаемые для пользователя условия работы программного обеспечения.

**Приемочные испытания.**

Это последний тест, который проводится перед передачей программного обеспечения клиенту. Он проводится, чтобы гарантировать, что программное обеспечение, которое было разработано отвечает всем требованиям заказчика. Существует два типа приемо-сдаточных испытаний - то, которое осуществляется членами команды разработчиков, известно, как внутреннее приемочное тестирования (Альфа-тестирование), а другое, которое проводится заказчиком, известно, как внешнее приемочное тестирования.

Если тестирование проводится с помощью предполагаемых клиентов, оно называется приемочными испытаниями клиента. В случае если тестирование проводится конечным пользователем программного обеспечения, оно известно, как приемочное тестирование (бета-тестирование).

**Основные тесты.**

Есть несколько основных методов тестирования, которые формируют часть режима тестирования программного обеспечения. Эти тесты обычно считаются самодостаточными в поиске ошибок и багов во всей системе.

**Тестирование методом черного ящика.**

Тестирование методом черного ящика осуществляется без каких-либо знаний внутренней работы системы. Тестер будет стимулировать программное обеспечение для пользовательской среды, предоставляя различные входы и тестируя сгенерированные выходы. Этот тест также известен как Black-box.

**Тестирование методом белого ящика.**

Тестирование методом "Белого ящика", в отличие от "черного ящика", учитывает внутреннее функционирование и логику работы кода. Для выполнения этого теста, тестер должен иметь знания кода, чтобы узнать точную часть кода, имеющую ошибки. Этот тест также известен как White-box, Open-Box или Glass box тестирование.

**Тестирование методом серого ящика.**

Тестирование методом серого ящика или Gray box тестирование, это что-то среднее между White Box и Black Box тестированием, где тестер обладает лишь общими знаниями данного продукта, необходимыми для выполнения теста. Эта проверка осуществляется посредством документации и схемы информационных потоков. Тестирование проводится конечным пользователем, или пользователям, которые представляются как конечные.

# 2 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ УЧЁТ КОНТИНГЕНТА УЧАЩИХСЯ УЧЕБНОГО ЗАВЕДЕНИЯ

## 2.1. Разработка тестовой документации (тест-дизайн)

Документ описывает методы и подходы к тестированию, которые будут использоваться тестировщиками отдела тестирования, для тестирования приложения. План тестирования может использоваться как тестировщиками, так и менеджерами, разработчиками. Объект тестирования — это деятельность, направленная на проверку работоспособности функций приложения покупки билетов кинотеатра «CoolMovie».

Целью тестирования приложения покупки билетов кинотеатра «CoolMovie» является проверка корректной работы.

Итогом процесса тестирования будут следующие материалы:

1. Определить существующую информацию о проекте и программных компонентах, подлежащих тестированию.
2. Описать стратегии тестирования, которые будут использоваться.
3. Определить необходимые ресурсы для проведения работ по тестированию.
4. Привести результаты тестирования.

Результаты будут отправлены заказчику в виде отчетов.

Условия для тестирования. Приложение должно удовлетворять потребность пользователя в активностях, связанных с вводом, редактированием и удалением данных о сотрудниках, о жанрах, о залах, о местах, о сеансах, о фильмах; формированием данных к продаже. Стратегия процесса тестирования. Приведенный ниже план тестирования является формальным, так как для построения развернутого плана необходимо понимание текущего состояния проекта.

Основными задачами тестирования являются:

* проведение функционального тестирования каждого модуля и компонента системы для обеспечения его соответствия функциональным требованиям;
* тестирование данных и целостности базы данных.

Виды тестирования. Для решения указанных выше задач тестирования будут использоваться следующие виды тестирования.

1. Тестирование данных и целостности базы данных

Базы данных должны тестироваться как отдельные системы внутри покупки билетов кинотеатра «CoolMovie». Эти системы должны тестироваться отдельно от приложений (таких как интерфейс доступа к данным).

Необходимо провести дополнительное исследование СУБД на тему того, какие инструменты/техники существуют для выполнения нижеописанного тестирования.

1.1 Цель тестирования

Убедится в том, что методы доступа к данным работают правильно и без нарушения целостности БД.

1.2 Способы

**Авторизация**

1. Авторизация пользователя.

**Просмотр информации**

1. Просмотр информации о фильмах.
2. Просмотр информации о сеансах.
3. Просмотр информации о сотрудниках.
4. Просмотр информации о залах.
5. Просмотр информации о местах.
6. Выход

1.3 Критерий завершенности

Все методы и процедуры БД функционируют так, как им положено и без нарушения целостности самой БД.

1.4 Особые замечания

* При тестировании может понадобиться среда разработки СУБД или драйвера для корректного подключения к базам данных.
* Процедуры должны вызываться вручную.
* Для повышения видимости неприемлемых событий БД необходимо использовать небольшие БД или БД с ограниченным количеством записей.

2. Функциональное тестирование

1.1 Цель тестирования

Функциональное тестирование состоит в том, чтобы убедиться, что весь программный продукт работает в соответствии с требованиями, и в приложении не появляется существенных ошибок.

1.2 Способы

Авторизация пользователя

Отправка и доставка сообщений

1.3 Критерий завершенности

Программный продукт должен пройти все запланированные тесты.

1.4 Особые замечания

Ожидаемые результаты возникают при использовании достоверных данных.

Соответствующие сообщения об ошибках или предупреждения отображаются, когда используются неверные данные.

Подготовлено тестовое окружение, приложение готово к тестированию на тестовой площадке.

Не будет проведено нагрузочное и тестирование безопасности в виду отсутствия необходимых ресурсов.

Отчеты об ошибках создаются для того, чтобы предоставить команде разработчиков и руководителю проекта исчерпывающую информацию об обнаруженных ошибках. Они должны быть полезны при определении причин ошибок и их исправлении.

Продукт должен работать в соответствии с требованиями и техническим заданием. Продукт не должен содержать критических и блокирующих дефектов в окончательной версии проекта.

## 2.2. Разработка тестовых сценариев

После установки базы данных можно приступить к ее использованию в тестах. В большинстве простых случаев в наборе тестов есть отдельная база данных, которая совместно используется несколькими тестами в нескольких классах тестирования, поэтому нам нужна некоторая логика, чтобы убедиться, что база данных создана и заполняется ровно один раз в течение всего времени существования тестового запуска.

При использовании Xunit это можно сделать с помощью [средства](https://xunit.net/docs/shared-context#class-fixture) класса, представляющего базу данных и совместно используемой для нескольких тестовых запусков:

public class TestDatabaseFixture

{

private const string ConnectionString = @"Server=(localdb)\mssqllocaldb;Database=EFTestSample;Trusted\_Connection=True";

private static readonly object \_lock = new();

private static bool \_databaseInitialized;

public TestDatabaseFixture()

{

lock (\_lock)

{

if (!\_databaseInitialized)

{

using (var context = CreateContext())

{

context.Database.EnsureDeleted();

context.Database.EnsureCreated();

context.AddRange(

new Blog { Name = "Blog1", Url = "http://blog1.com" },

new Blog { Name = "Blog2", Url = "http://blog2.com" });

context.SaveChanges();

}

\_databaseInitialized = true;

}

}

}

public BloggingContext CreateContext()

=> new BloggingContext(

new DbContextOptionsBuilder<BloggingContext>()

.UseSqlServer(ConnectionString)

.Options);

}

При создании экземпляра приведенного выше средства используется [EnsureDeleted()](https://docs.microsoft.com/ru-RU/dotnet/api/microsoft.entityframeworkcore.storage.idatabasecreator.ensuredeleted#microsoft-entityframeworkcore-storage-idatabasecreator-ensuredeleted) для удаления базы данных (если она существует из предыдущего запуска), а затем [EnsureCreated()](https://docs.microsoft.com/ru-RU/dotnet/api/microsoft.entityframeworkcore.storage.idatabasecreator.ensurecreated#microsoft-entityframeworkcore-storage-idatabasecreator-ensurecreated) для ее создания с помощью последней конфигурации модели ([см. документацию по этим API](https://docs.microsoft.com/ru-ru/ef/core/managing-schemas/ensure-created)). После создания базы данных средство заменит его некоторыми данными, которые могут использовать наши тесты. Стоит потратить некоторое время на размышления о начальных данных.

Чтобы использовать приспособление в тестовом классе, просто реализуйте IClassFixture его над типом светильника, а xUnit внедряет его в конструктор:

public class BloggingControllerTest : IClassFixture<TestDatabaseFixture>

{

public BloggingControllerTest(TestDatabaseFixture fixture)

=> Fixture = fixture;

public TestDatabaseFixture Fixture { get; }

Теперь класс тестирования имеет Fixture свойство, которое можно использовать тестами для создания полнофункциональный экземпляр контекста:

[Fact]

public void GetBlog()

{

using var context = Fixture.CreateContext();

var controller = new BloggingController(context);

var blog = controller.GetBlog("Blog2").Value;

Assert.Equal("http://blog2.com", blog.Url);

}

Вот здесь можно заметить некоторые блокировки в логике создания светильника выше. Если приспособление используется только в одном тестовом классе, оно гарантированно будет создано ровно один раз с помощью xUnit; Но обычно один и тот же компонент базы данных используется в нескольких тестовых классах.

В приведенном выше примере показан тест только для чтения, который является простым случаем с точки зрения изоляции теста: так как ничего не изменяется, вмешательство теста невозможно. В отличие от этого, тесты, которые изменяют данные, являются более проблематичными, так как они могут вмешиваться друг в друга. Одним из распространенных способов изоляции тестов является перенос теста в транзакцию и откат этой транзакции в конце теста. Так как в базе данных ничего не зафиксировано, другие тесты не видят никаких изменений и интерференции не будут избегать.

Ниже приведен метод контроллера, который добавляет блог в нашу базу данных:

[HttpPost]

public ActionResult AddBlog(string name, string url)

{

\_context.Blogs.Add(new Blog { Name = name, Url = url });

\_context.SaveChanges();

return Ok();

}

Мы можем протестировать этот метод следующим образом:

[Fact]

public void AddBlog()

{

using var context = Fixture.CreateContext();

context.Database.BeginTransaction();

var controller = new BloggingController(context);

controller.AddBlog("Blog3", "http://blog3.com");

context.ChangeTracker.Clear();

var blog = context.Blogs.Single(b => b.Name == "Blog3");

Assert.Equal("http://blog3.com", blog.Url);

}

Некоторые примечания к приведенному выше коду теста:

* Запускаем транзакцию, чтобы убедиться, что приведенные ниже изменения не зафиксированы в базе данных и не вмешиваются в другие тесты. Так как транзакция никогда не фиксируется, она неявно откатывается в конце теста при удалении экземпляра контекста.
* После внесения нужных обновлений очистим средство отслеживания [ChangeTracker.Clear](https://docs.microsoft.com/ru-RU/dotnet/api/microsoft.entityframeworkcore.changetracking.changetracker.clear)изменений экземпляра контекста, чтобы убедиться, что мы фактически загружаем блог из базы данных ниже. Вместо этого можно использовать два экземпляра контекста, но затем необходимо убедиться, что одна и та же транзакция используется обоими экземплярами.

# ЗАКЛЮЧЕНИЕ

# Цель тестирования — проверка соответствия ПО предъявляемым требованиям, обеспечение уверенности в качестве ПО, поиск очевидных ошибок в программном обеспечении, которые должны быть выявлены до того, как их обнаружат пользователи программы. И цель данной курсовой работы — это тестирование разработанного приложения для ИС кинотеатра «CoolMovie». В ходе работы был проведен анализ методов тестирования:

* критериев и принципов тестирования;
* методов тестирования.

А так же было осуществлено само тестирование информационной системы книжного магазина, была разработана тестовая документация (тест-дизайн) и тестовые сценарии.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 34.201-89 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Виды, комплектность и обозначение документов при создании автоматизированных систем.
2. ГОСТ 34.601-90 ЕСС АСУ. Автоматизированные системы. Стадии создания.
3. ГОСТ 34.602-89 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
4. РД 50-34.698-90 МЕТОДИЧЕСКИЕ УКАЗАНИЯ. ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Автоматизированные системы. Требования к содержанию документов.
5. ГОСТ 21.404-85 Автоматизация технологических процессов. Обозначения условные приборов и средств автоматизации в схемах.
6. ГОСТ 34.603-92 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Виды испытаний автоматизированных систем.
7. ГОСТ 24.701-86 ЕСС АСУ. Надёжность автоматизированных систем управления. Основные положения.
8. ГОСТ 25861-83 Машины вычислительные и системы обработки данных.
9. СНиП 3.05.07-85 Системы автоматизации.
10. СТП 7.3-03-2008 СТАНДАРТ ПРЕДПРИЯТИЯ. Порядок разработки, внедрения, сопровождения и эксплуатации автоматизированных систем управления технологическими процессами.
11. ГОСТ 34.003-90 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. Автоматизированные системы. Термины и определения.
12. ГОСТ 24.104-85 ЕСС АСУ. Автоматизированные системы управления. Общие требования.