МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ
Государственное бюджетное профессиональное образовательное учреждение Московской области
«Воскресенский колледж»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

ПМ.04 «Сопровождение и обслуживание программного обеспечения компьютерных систем»

Наименование специальности

09.02.07 «Информационные системы и программирование»

Квалификация выпускника

Программист

Методические рекомендации по выполнению лабораторных работ по профессиональному модулю разработаны на основе Федерального государственного образовательного стандарта (далее – ФГОС) по специальности среднего профессионального образования (далее – СПО) 09.02.07 «Информационные системы и программирование»

Организация-разработчик: Государственное бюджетное профессиональное образовательное учреждение Московской области «Воскресенский колледж»

Разработчик:

Комиссаров С.А., преподаватель компьютерных дисциплин

Рабочая программа профессионального модуля рассмотрена на заседании предметной (цикловой) комиссией компьютерных дисциплин

«___»_____2020г.

Председатель цикловой комиссии _____/Рязанцева О.В./

Утверждена зам директора по УР _____/Куприна Н.Л./

«___»_____2020 г.

ВВЕДЕНИЕ

Данные методические указания для проведения лабораторных работ по ПМ.04 «Сопровождение и обслуживание программного обеспечения компьютерных систем» предназначены для реализации ФГОС СПО по специальности 09.02.07 «Информационные системы и программирование» с целью закрепления теоретических знаний и практических умений.

В сборнике содержатся методические указания по выполнению лабораторных работ по МДК.04.01 и МДК.04.02.

При выполнении лабораторных работ студент должен

иметь практический опыт:

- Выполнять инсталляцию, настройку и обслуживание программного обеспечения компьютерных систем.
- настраивать отдельные компоненты программного обеспечения компьютерных систем.
- Измерять эксплуатационные характеристики программного обеспечения компьютерных систем на соответствие требованиям.
- Модифицировать отдельные компоненты программного обеспечения в соответствии с потребностями заказчика.
- выполнять отдельные виды работ на этапе поддержки программного обеспечения компьютерных систем.
- Обеспечивать защиту программного обеспечения компьютерных систем программными средствами;

уметь:

- Подбирать и настраивать конфигурацию программного обеспечения компьютерных систем.
- Проводить инсталляцию программного обеспечения компьютерных систем.
- Производить настройку отдельных компонент программного обеспечения компьютерных систем.
- Измерять и анализировать эксплуатационные характеристики качества программного обеспечения.
- Определять направления модификации программного продукта.
- Разрабатывать и настраивать программные модули программного продукта.
- Настраивать конфигурацию программного обеспечения компьютерных систем.
- Использовать методы защиты программного обеспечения компьютерных систем.
- Анализировать риски и характеристики качества программного обеспечения.
- Выбирать и использовать методы и средства защиты компьютерных систем программными и аппаратными средствами.
- Решать проблемы совместимости программного обеспечения
- Составлять сопроводительную документацию при внедрении и поддержке ПО
- Разрабатывать сценарии внедрения ПО
- Разрабатывать сценарии сопровождения ПО
- Оценивать эффективность внедрения ПО в компьютерную систему
- Составлять команду сотрудников по внедрению и поддержке ПО
- Определять задачи сопровождения ПО;

знать:

- Основные методы и средства эффективного анализа функционирования программного обеспечения.
- Основные виды работ на этапе сопровождения ПО.
- Основные принципы контроля конфигурации и поддержки целостности конфигурации ПО.
- Основные средства и методы защиты компьютерных систем программными и аппаратными средствами.
- Причины возникновения проблем совместимости программного обеспечения
- Основные виды документации при внедрении и поддержке ПО
- Основные типы сценариев внедрения и поддержки ПО
- Показатели эффективности внедрения и сопровождения ПО
- Виды ответственности между сотрудниками и состав команды сотрудников по внедрению и поддержке ПО
- Показатели качества поддержки и внедрения ПО
- Факторы угрозы надёжности ПО
- Стандарты качества ПО;

Каждая лабораторная работа имеет следующую структуру: тема, цели, краткие теоретические сведения, порядок проведения работы, требования к составлению отчета.

После выполнения лабораторной работы студент должен представить отчет о проделанной работе. Оценку по практической работе студент получает, если студентом работа выполнена в полном объеме, студент может пояснить выполнение любого этапа работы, отчет выполнен в соответствии с требованиями к выполнению работы, студент отвечает на контрольные вопросы на удовлетворительную оценку и выше.

Зачет по выполнению лабораторных работ студент получает при условии выполнения всех предусмотренных программой лабораторных работ с отчетами по всем работам.

Отчет к лабораторной работе должен содержать (в зависимости от задания):

- Тему работы
- Задание для выполнения, включая индивидуальное задание
- Описание ключевых решений
- Описание выбранного ПО или оборудования
- Описание диаграмм
- Построенные диаграммы и схемы
- Методы внедрения
- Методы сопровождения
- Методы обслуживания ПО
- Документацию для соответствующих действий
- Выводы

МДК.04.01 «Внедрение и поддержка программного обеспечения компьютерных систем»

Лабораторная работа №1. Постановка цели внедрения. Составление иерархии целей.

Теоретический материал.

Многие предприятия сегодня ставят перед собой задачу провести автоматизацию учетных, контрольных, управленческих и аналитических функций с целью повышения эффективности управления, инвестиционной привлекательности и устойчивости в высококонкурентных средах.

Для решения таких задач на предприятие внедряется программное обеспечение с набором необходимых функций.

Для внедрения программного обеспечения разрабатывается проект.

Проект – любое временное предприятие, предназначенное для создания уникальных продуктов или услуг. Понятие «проект» объединяет разнообразные виды деятельности, характеризуемые рядом признаков, наиболее общими из которых являются следующие:

- направленность на достижение конкретных целей, определенных результатов;

- координированное выполнение многочисленных, взаимосвязанных действий;

- ограниченная протяженность во времени, с определенным началом и концом.

Управление проектами – область деятельности, в ходе которой определяются и достигаются четкие цели проекта при балансировании между объёмом работ, ресурсами (такими как деньги, труд, материалы, энергия, пространство и др.), временем, качеством и рисками.

Описание предметной области

«1С: Предприятие 8.4.1» является современной технологической платформой, рассчитанной на высокие нагрузки и одновременную работу большого количества пользователей. Проводимые нагрузочные испытания и опыт реальных внедрений показывают, что платформа «1С: Предприятие 8.4.1» и типовые прикладные решения могут быть успешно использованы для создания информационных систем масштаба предприятия.

Успешное внедрение программных продуктов "1С" подразумевает запуск в эксплуатацию в максимально короткие сроки реально работающую систему, соответствующую потребностям и ожиданиям предприятия. В результате предприятие получает современную, мощную и быстро действующую автоматизированную систему учета, построенную на базе программных продуктов «1С: Предприятие».

Для нашего предприятия требуется автоматизировать финансовый и бухгалтерский учет, значит, определяем для него программные продукты «1С: Бухгалтерия» и «1С: Бюджетная отчетность».

Внедрением системы 1С на предприятие будет заниматься Технический отдел, в который входят Менеджер и Программист.

Документацией будет заниматься Бухгалтерия и Юридический отдел.

Введем некоторые обозначения: М – менеджер, П – программист, Б – бухгалтерия, ЮО – юридический отдел.

Ниже представлены таблицы с режимом рабочего дня для каждого сотрудника.

Таблица 1 – Режим работы менеджера

	Часы
Начало рабочего дня	9.00
Обед	13.00 – 14.00
Окончание рабочего дня	18.00

Таблица 2 – Режим работы программиста

	Часы
Начало рабочего дня	9.00
Обед	13.00 – 14.00
Окончание рабочего дня	18.00

Таблица 3 – Режим работы бухгалтерии

	Часы
Начало рабочего дня	9.00
Обед	13.00 – 14.00
Окончание рабочего дня	18.00

Таблица 4 – Режим работы юридического отдела

	Часы
Начало рабочего дня	9.00
Обед	13.00 – 14.00
Окончание рабочего дня	18.00

Ниже представлена таблица по оплате труда сотрудников за 1 месяц работы.

Таблица 5 – Оплата труда сотрудников за 1 месяц работы

Сотрудник	Руб./ месяц
М	33 000
П	27 000
Б	14 000
ЮО	17 000

Подготовительная часть проекта включает в себя:

· Изучение технической архитектуры и среды внедрения (М).

· Определение параметров внедрения (П).

· Проектирование этапов внедрения и интеграции пакета (М).

Техническая часть проекта включает в себя непосредственную установку программного обеспечения, проведение тестирования программ, разработка обучающих материалов (П).

Проект будет длиться 62 дня. Бюджет проекта не более 100 тыс. руб.

Примечание: использование любого работника в сверхурочное время оплачивается в двойне.

Планирование проекта

Определение целей проекта

Целью проекта является внедрение системы «1С: Предприятие 8.4.1» на предприятие, соблюдая сроки выполнения работ и не превышая установленный бюджет.

Построение иерархической структуры работ

Иерархическая структура работ - это разбиение проекта на более мелкие и измеримые части. **Иерархическая структура работ** описывает все результаты/работы, которые должны быть получены/выполнены для завершения проекта.

Иерархическая структура проекта представлена на рисунке 1.

Рисунок 1 - Иерархическая структура проекта

Структурная схема предприятия

Структурная схема предназначена для того, чтобы показать подчиненность участников. Структура проекта показана на рисунке 2

Рисунок 2 - Структурная схема предприятия

Участники проекта

Участники проекта – основной элемент структуры проекта, так как именно они обеспечивают реализацию его замысла.

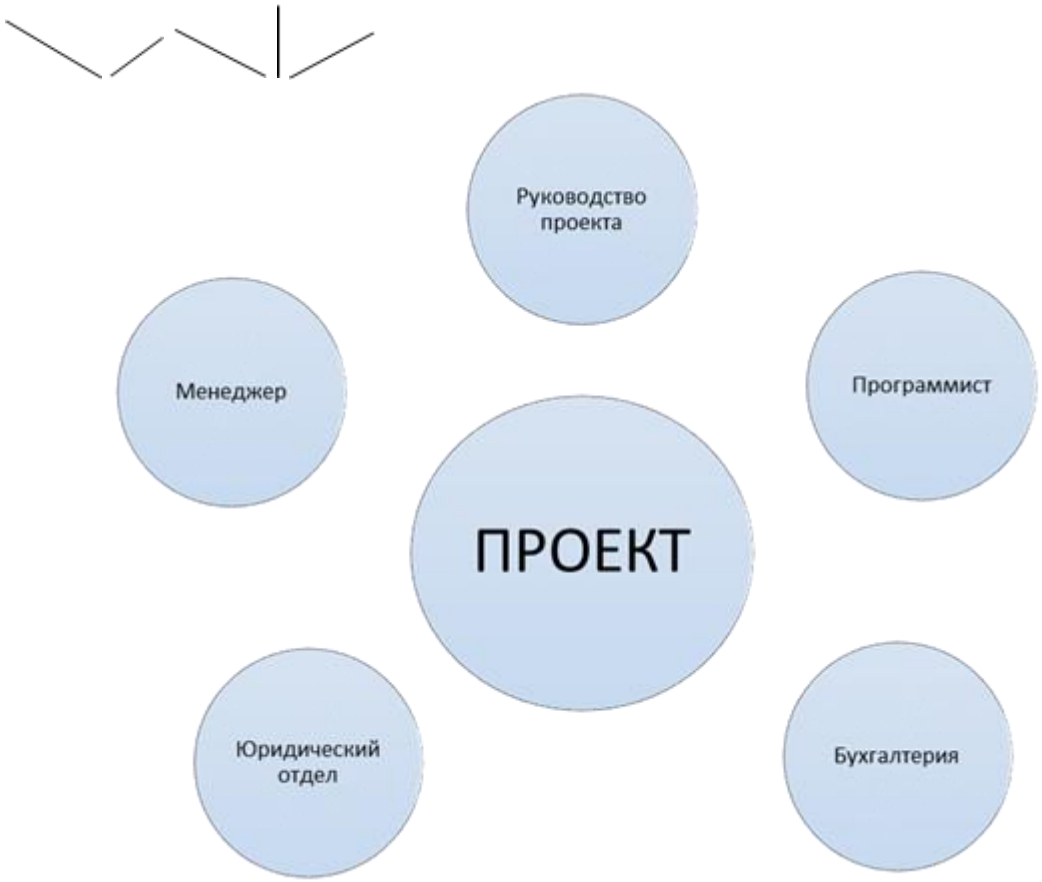


Рисунок 3 - Участники проекта

Матрица ответственности

Матрица ответственности решает задачу демонстрации межорганизационного или межгруппового взаимодействия и, как следствие, позволяет избежать недоразумений, которые время от времени возникают в проектах между подразделениями и организациями из-за неясности, к кому следует обращаться по тем или иным вопросам и кто должен принимать по ним решение, а кто – непосредственно реализовать принятую резолюцию.

Важно как можно раньше произвести размежевание всех формальных полномочий, прав и обязанностей, пока участники проекта еще не приступили к его реализации. В противном случае, когда у сотрудников сложится собственное представление о своем месте в проекте, расхождения во мнениях по этим вопросам может перерасти в затяжные конфликты и оказать негативное влияние на сроки выполнения проекта. Матрица ответственности данного проекта представлена в таблице 6.

Таблица 6 – Матрица ответственности

Выполняемая работа	Менеджер	Программист	Бухгалтерия	Юридический отдел
Обзор технической архитектуры	+			
Изучение пакета	+		+	
Изучение среды внедрения	+			
Определение этапов внедрения	+			
Разработка матрицы отслеживания требований	+			
Определение итераций		+		
Разработка сценариев	+			
Оптимизация рабочих характеристик		+		
Установка «1С: Предприятие 8.4.1» на компьютеры		+		
Тестирование		+		

Разработка обучающих материалов	+		
Подготовка документации проекта		+	+

Ресурсный план

Ресурсный план определяет способ распределения ресурсов среди различных пользователей. Ресурсный план проекта представлен в таблице 7.

Таблица 7 - Ресурсный план

Работа	Ресурсы	Длительность
Обзор технической архитектуры	Менеджер; Информационные системы	12 часов
Изучение пакета	Бухгалтерия; Информационные системы	12 часов
Изучение среды внедрения	Менеджер; Информационные системы	12 часов
Определение этапов внедрения	Менеджер; Информационные системы	12 часов
Разработка матрицы отслеживания требований	Менеджер; Информационные системы	12 часов
Определение итераций	Программист; Информационные системы	12 часов
Разработка сценариев	Менеджер; Информационные системы	12 часов
Оптимизация рабочих характеристик	Программист; Информационные системы	24 часа
Установка «1С: Предприятие 8.4.1» на компьютеры	Программист; Информационные системы	24 часа
Тестирование	Программист; Информационные	12 часов

	системы	
Разработка обучающих материалов	Программист	24 часа
Подготовка документации проекта	Бухгалтерия; Юридический отдел	48 часов

Описание проекта

Описание структуры проекта

Подготовительная часть данного проекта (рисунок 4) включает:

· Изучение технической архитектуры и среды внедрения

· Определение параметров внедрения

· Проектирование этапов внедрения

· Проектирование этапов интеграции пакета

Рисунок 4 - Подготовительная часть проекта

Техническая часть данного проекта (рисунок 5) включает:

· Установка пакета «1С: Предприятие 8.4.1» на компьютеры

· Тестирование

· Разработка обучающих материалов

Рисунок 5 - Техническая часть проекта

Характеристика проекта

Функционирование современного предприятия связано с осуществлением проектной деятельности, поскольку любое его действие, принятое решение или результат жизнедеятельности представляет собой либо самостоятельный проект, либо элемент более сложного проекта.

Характеристика проекта представлена в таблице 8.

Таблица 8 – Характеристика проекта

Классификационные признаки	Вид проекта
По главной цели реализации	Коммерческий
По характеру изменений	Оперативный
По масштабу (размеру)	Малый
По длительности (по срокам реализации)	Краткосрочный
По отраслевой принадлежности	Комплексный
По специфике конечного продукта (по типу проекта, по основной сфере деятельности)	Технический
По функциональному направлению	Производственный
По характеру привлеченных сторон	Проект одного предприятия
По степени сложности (по классу)	Монопроект

По составу и структуре привлеченных организаций	Однофункциональный
По требованиям к качеству проекта	Стандартный
По степени взаимного влияния	Независимый

Структура жизненного цикла проекта

Жизненный цикл проекта – промежуток времени между моментом появления проекта и моментом его ликвидации.

Стадии жизненного цикла – это состояния, которые проходит проект в своем развитии.

Жизненный цикл проекта делится на три стадии:

1. Прединвестиционная;

2. Инвестиционная;

3. Эксплуатационная.

Прединвестиционная стадия жизненного цикла – это промежуток времени между моментом появления первоначального замысла проекта и моментом принятия окончательного решения по его реализации.

1. Обзор технической архитектуры – 12 часов

2. Изучение пакета – 12 часов

3. Изучение среды внедрения – 12 часов

4. Определение этапов внедрения – 12 часов

5. Разработка матрицы отслеживания требований – 12 часов

6. Определение итераций – 12 часов

7. Разработка сценариев – 12 часов

Итого: 3, 5 дней (время выполнения прединвестиционной стадии жизненного цикла было рассчитано в программе MS Project) занимает прединвестиционная стадия жизненного цикла проекта.

Инвестиционная стадия жизненного цикла – это промежуток времени от момента начала проектно-изыскательных работ до выхода предприятия на проектную мощность.

1. Оптимизация рабочих характеристик – 24 часа

2. Покупка программного продукта «1С: Предприятие 8.4.1» – 24 часа

3. Установка «1С: Предприятие 8.4.1» на компьютеры – 24 часа

4. Тестирование – 24 часа

5. Разработка обучающих материалов – 24 часа

6. Подготовка документации проекта – 48 часов

Итого: 7 дней (время выполнения прединвестиционной стадии жизненного цикла было рассчитано в программе MS Project) занимает инвестиционная стадия жизненного цикла проекта.

Эксплуатационная стадия жизненного цикла – это промежуток времени между выходом предприятия на проектную мощность и завершением проекта, то есть ликвидацией предприятия.

При условии, что установка пакета «1С: Предприятие» выполнится качественно и при отсутствии внешних факторов, которые могут повлиять на работоспособность системы, она будет функционировать достаточно долгое время.

Влияние окружающей среды на проект (SWOT-анализ)

Состояние организации зависит от того, насколько она способна реагировать на различные воздействия извне. Анализируя внешнюю ситуацию, необходимо выделять наиболее существенные на конкретный период времени факторы. Взаимосвязанное рассмотрение этих факторов с возможностями организации позволяет решать возникающие проблемы. При решении разного уровня задач необходимо также четко представлять, поддаются ли критические факторы контролю со стороны организации. Являются ли они внутренними или внешними, поддающимися изменениям усилиями организации или это внешние события, на которые организация влиять не в состоянии. Одним из самых распространенных методов, оценивающих в комплексе внутренние и внешние факторы, влияющие на развитие организации можно назвать SWOT-анализ. SWOT-анализ является необходимым элементом исследований, обязательным предварительным этапом при составлении любого уровня стратегических и маркетинговых планов. Данные, полученные в результате ситуационного анализа, служат базисными элементами при разработке стратегических целей и задач организации.

Аббревиатура SWOT обозначает:

Strengths – сильные стороны компании

Weakness – слабые стороны компании

Opportunities – возможности компании для развития и повышения финансовой эффективности

Threats – угрозы для компании, которые могут представлять конкуренты, администрация, смена спроса потребителя, природные факторы.

SWOT-анализ – это анализ сильных и слабых сторон организации и возможностей и угроз со стороны внешней окружающей среды.

SWOT-анализ проекта по внедрению «1С: Предприятие 8.4.1» на предприятие представлен в таблице 9.

Таблица 9 - Матрица SWOT-анализа

Наименование	Описание

Сильные стороны	
Высокий уровень обслуживания	Создает устойчивое конкурентное преимущество
Высокая квалификация персонала	Выполнение работ будет качественным
Активная рекламная деятельность	Привлечение большого числа новых клиентов
Слабые стороны	
Узкий ассортимент услуг	Препятствует привлечению целевой группы с низшими средними доходами, которые могут стать ведущими по прибыли сегментом
Возможности развития компании	
Повышение уровня услуг	Диктует необходимость введения более дорогих услуг
Угрозы для развития компании	
Конкуренты	Возможность того что конкуренты предложат более выгодное предложение
Изменение потребностей и вкуса покупателей	Может привести к отказу от предоставляемых услуг

Установка параметров проекта

Календарь

Представление «Календарь» показывает деловой календарь, позволяющий показать работы, выполняемые в соответствующий рабочий день. Диаграмма удобна тем, что представляет план проекта в виде традиционного календаря.

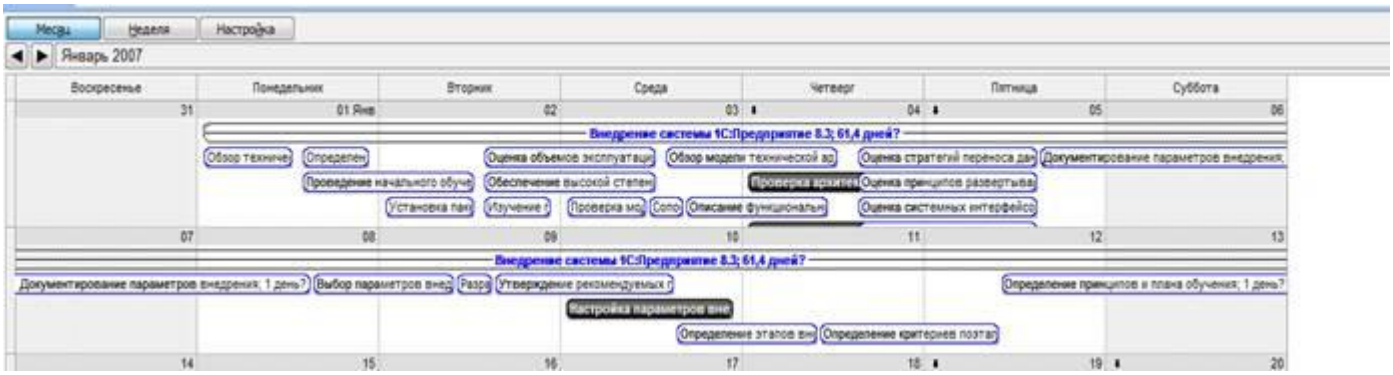
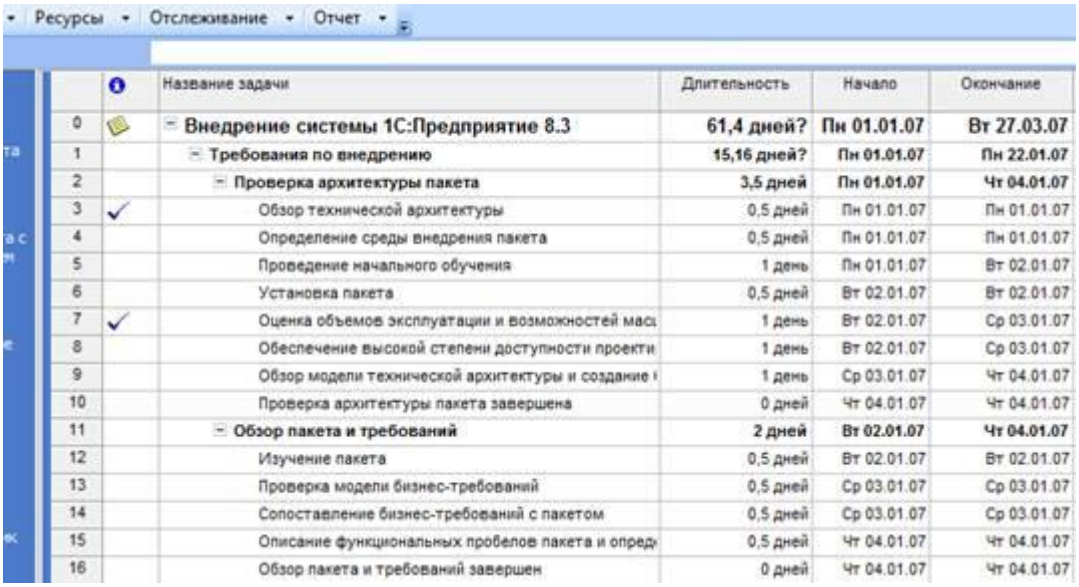


Рисунок 6 - Календарь

Длительность

Длительность задач определяется значением, введенным в колонке «Длительность». Длительность в колонку можно вводить в часах, днях или неделях. Длительность задач проекта представлена на рисунке – 8.



		Название задачи	Длительность	Начало	Окончание
0		**Внедрение системы 1С:Предприятие 8.3**	**61,4 дней?**	**Пн 01.01.07**	**Вт 27.03.07**
1		**Требования по внедрению**	**15,16 дней?**	**Пн 01.01.07**	**Пн 22.01.07**
2		**Проверка архитектуры пакета**	**3,5 дней**	**Пн 01.01.07**	**Чт 04.01.07**
3	✓	Обзор технической архитектуры	0,5 дней	Пн 01.01.07	Пн 01.01.07
4		Определение среды внедрения пакета	0,5 дней	Пн 01.01.07	Пн 01.01.07
5		Проведение начального обучения	1 день	Пн 01.01.07	Вт 02.01.07
6		Установка пакета	0,5 дней	Вт 02.01.07	Вт 02.01.07
7	✓	Оценка объемов эксплуатации и возможностей масш	1 день	Вт 02.01.07	Ср 03.01.07
8		Обеспечение высокой степени доступности проекти	1 день	Вт 02.01.07	Ср 03.01.07
9		Обзор модели технической архитектуры и создание	1 день	Ср 03.01.07	Чт 04.01.07
10		Проверка архитектуры пакета завершена	0 дней	Чт 04.01.07	Чт 04.01.07
11		**Обзор пакета и требований**	**2 дней**	**Вт 02.01.07**	**Чт 04.01.07**
12		Изучение пакета	0,5 дней	Вт 02.01.07	Вт 02.01.07
13		Проверка модели бизнес-требований	0,5 дней	Ср 03.01.07	Ср 03.01.07
14		Сопоставление бизнес-требований с пакетом	0,5 дней	Ср 03.01.07	Ср 03.01.07
15		Описание функциональных пробелов пакета и опред	0,5 дней	Чт 04.01.07	Чт 04.01.07
16		Обзор пакета и требований завершен	0 дней	Чт 04.01.07	Чт 04.01.07

Рисунок 8 – Длительность.

Способ планирования.

Когда требуется контролировать дату начала или конца, можно добавить ограничение. Гибкие ограничения учитывают связи между задачами, чтобы перенести задачу как можно раньше или как можно позже, насколько позволяет связь. Например, задача с ограничением «как можно раньше» и связью «окончание-начало» будет начинаться сразу по завершении предшествующей.

Ограничения со средней гибкостью запрещают задаче начаться или окончиться до или после выбранной даты. Например, задача с ограничением «начало не позднее» 1 января и связью типа «окончание-начало» с другой задачей может начаться в любое время, если ее

предшественница закончится, например, до 15 марта, но не может быть начата после 1 января.

Негибкие ограничения не подвергаются влиянию связей и «привязывают» задачу к выбранной вами дате. Например, задача с ограничением «фиксированное начало» на 1 января и связью типа «окончание-начало» с другой задачей всегда будет находиться в расписании на 1 января вне зависимости от того, закончится ее предшественница раньше или позже.

Для своего проекта я выбрал тип ограничения «как можно раньше», т.к. с этим ограничением MS Project размещает задачу в расписании как можно раньше с учетом других параметров плана. Никаких дополнительных ограничений на задачу не распространяется. Это ограничение по умолчанию накладывается на все задачи, если проект планируется от даты начала.

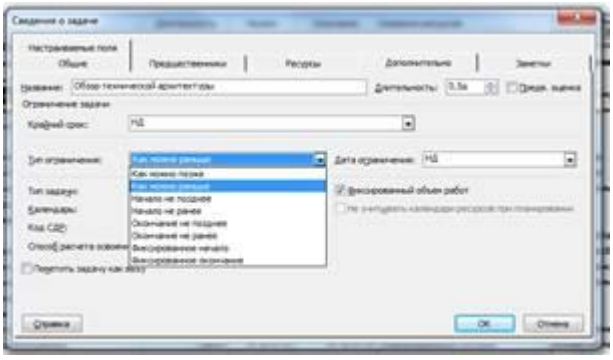


Рисунок 9 - Способ планирования

Вид связи с предшествующими работами

В своем проекте я использовал связь с предшествующими работами – «окончание-начало», т.к. это стандартная последовательность, при которой предшествующая работа должна завершиться до начала последующей.

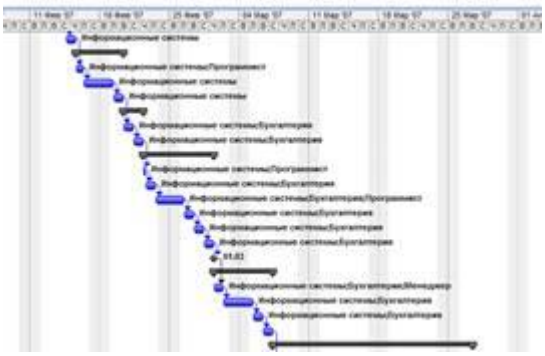


Рисунок 10 - Вид связи «окончание-начало»

Приоритет

Приоритет – это свойство задачи, отражающее важность ее исполнения для проекта, варьирующееся в диапазоне от 1 до 1000. Его можно изменить с помощью одноименного столбца в таблице или на вкладке «Общие» в диалоговом окне сведений о задаче.

Приоритет задач по умолчанию определяется программой и обычно равен 500.

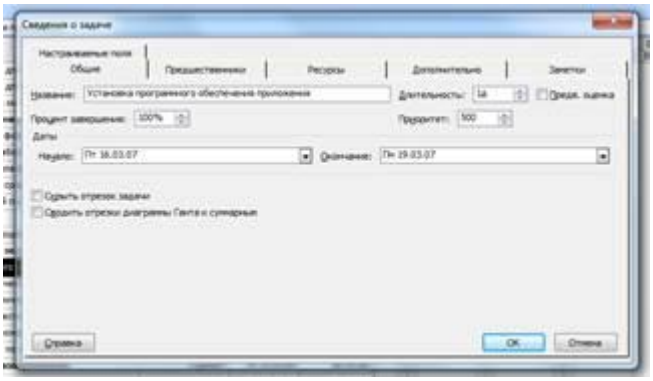


Рисунок 11 - Настройка приоритета

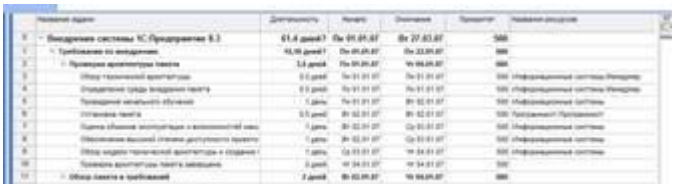


Рисунок 12 - Приоритет

Использовать это свойство имеет смысл тогда, когда проект содержит много дополнительных задач, исполнением которых можно пренебречь. Эти задачи можно выделить, присвоив им низкий приоритет, и в дальнейшем использовать эту информацию при планировании.

Просмотр и оценка состояния проекта

Оценка стоимости проекта

Общая стоимость проекта складывается из фиксированной стоимости ресурсов и задач и стоимости назначений.

Стоимость моего проекта складывается из стоимости самого программного продукта «1С: Предприятие 8.4.1», а также затрат на заработную плату сотрудников.

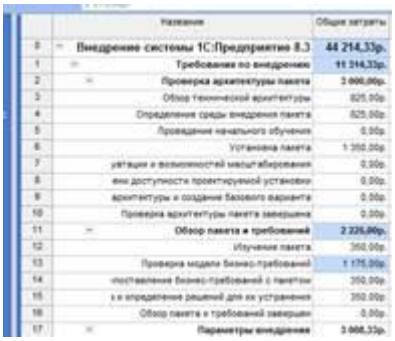


Рисунок 13 - Оценка стоимости проекта

Стоимость программного продукта «1С: Предприятие 8.4.1»:

«1С: Бухгалтерия 8» – 3300 рублей;

«1С: Бюджетная отчетность» - 33600 рублей.

Итого: бюджет моего проекта составляет 81 тысячу 174 рубля 33 копейки (44 214,33 + 3300 + 33660 =81 174,33)

Запаздывающие задачи

Любая задача на критическом пути является критической задачей. Нужно регулярно отслеживать выполнение этих задач, чтобы знать, не запаздывают ли некоторые из них. Если критическая задача запаздывает, то будет запаздывать и дата окончания проекта. Запаздывание – это время, на которое задерживается задача относительно исходного базового плана.

В моем проекте запаздывающие задачи отсутствуют.

	Название задачи	Начало	Окончание	Отклон. начала	Отклон. окончания
0	Внедрение системы 1С:Предприятие 8.3	Пн 01.01.07	Вт 27.03.07	0 дней	0 дней
1	Подготовительная часть	Пн 01.01.07	Пн 22.01.07	0 дней	0 дней
2	Проверка архитектуры пакета	Пн 01.01.07	Чт 04.01.07	0 дней	0 дней
3	Обзор технической архитектуры	Пн 01.01.07	Пн 01.01.07	0 дней	0 дней
4	Определение среды внедрения пакета	Пн 01.01.07	Пн 01.01.07	0 дней	0 дней
5	Проведение начального обучения	Пн 01.01.07	Вт 02.01.07	0 дней	0 дней
6	Установка пакета	Вт 02.01.07	Вт 02.01.07	0 дней	0 дней
7	Оценка объемов эксплуатации и возможност	Вт 02.01.07	Ср 03.01.07	0 дней	0 дней
8	Обеспечение высокой степени доступности п	Вт 02.01.07	Ср 03.01.07	0 дней	0 дней
9	Обзор модели технической архитектуры и соз	Ср 03.01.07	Чт 04.01.07	0 дней	0 дней
10	Проверка архитектуры пакета завершена	Чт 04.01.07	Чт 04.01.07	0 дней	0 дней
11	Обзор пакета и требований	Вт 02.01.07	Чт 04.01.07	0 дней	0 дней
17	Параметры внедрения	Чт 04.01.07	Ср 10.01.07	0 дней	0 дней
28	Проектирование полезного коммерческого	Ср 10.01.07	Ср 17.01.07	0 дней	0 дней
35	Заключительный этап определения требован	Ср 17.01.07	Пн 22.01.07	0 дней	0 дней

Рисунок 14 – Запаздывающие задачи

Затраты задач на различных уровнях

Из приведенного ниже рисунка видно, что затраты на подготовительную часть составляют 11 тысяч 314 рублей 33 копейки, а затраты на техническую часть составляют 32 тысячи 900 рублей. Также по приведенному рисунку можно проследить затраты на каждом уровне.

	Название задачи	Затраты	Общие затраты
0	Внедрение системы 1С:Предприятие 8.3	44 214,33р.	44 214,33р.
1	Подготовительная часть	11 314,33р.	11 314,33р.
2	Проверка архитектуры пакета	2 000,00р.	2 000,00р.
3	Обзор технической архитектуры	625,00р	625,00р
4	Определение среды внедрения пакета	625,00р	625,00р
5	Проведение начального обучения	0,00р	0,00р
6	Установка пакета	1 200,00р	1 200,00р
7	Оценка объемов эксплуатации и возможностей масштабирования	0,00р	0,00р
8	Обеспечение высокой степени доступности проектируемой установки	0,00р	0,00р
9	Обзор модели технической архитектуры и создание базового варианта	0,00р	0,00р
10	Проверка архитектуры пакета завершена	0,00р	0,00р
11	Обзор пакета и требований	2 225,00р	2 225,00р
17	Параметры внедрения	3 000,33р	3 000,33р
28	Проектирование полезного коммерческого внедрения пакета	1 525,00р	1 525,00р
35	Заключительный этап определения требований	1 550,00р	1 550,00р
40	Этап определения требований завершен	0,00р	0,00р
41	Техническая часть	32 900,00р	32 900,00р
42	Планирование итераций пакета	1 350,00р	1 350,00р
47	Разработка сценариев	6 875,00р	6 875,00р
72	Проектирование интеграции пакета завершено	0,00р	0,00р
73	Интеграция пакета	13 425,00р	13 425,00р
111	Развертывание пакета	11 250,00р	11 250,00р

Рисунок 15 – Затраты задач на различных уровнях

Затраты ресурсов и их отклонения

Для определения времени, которое предполагается затратить на выполнение задач, вводятся значения трудозатрат. Трудозатраты – это объем работ или число человеко-часов, необходимое для завершения задачи. Трудозатраты можно задавать непосредственно в поле Трудозатраты. Трудозатраты измеряются в рабочих днях.

Отклонения трудозатрат можно проследить по столбцу Отклонения по трудозатратам.

По приведенному ниже рисунку видно, что в моем проекте присутствуют отклонения по трудозатратам. Это значит, что ресурсы перегружены.

	Название ресурса	Тип	Трудозатраты	Отклонение по трудозатратам
1	Высшее руководство	Трудовой	6 ч	6 ч
2	Информационные систе	Трудовой	598,53 ч	598,53 ч
3	Бухгалтерия	Трудовой	178,9 ч	178,9 ч

Рисунок 16 – Затраты ресурсов и их отклонения.

Выравнивание ресурсов.

Если в проекте присутствуют перегрузки ресурсов, то необходимо выполнить выравнивание ресурсов. В моем проекте присутствуют три перегруженных ресурса: высшее руководство, информационные системы и бухгалтерия. Превышение доступности ресурса заключается в том, что для выполнения назначенной работы ресурсу требуется больше времени, чем у него реально имеется.

		Название ресурса	Тип	Краткое название	Макс. единиц	Стандартная ставка	Начисление	Базовый календарь
1		Высшее руководство	Трудовой	В	50%	40 000,00р./мес	Пропорциональн	Стандартный
2		Информационные системы	Трудовой	И	100%	0,00р./ч	Пропорциональн	Стандартный
3		Бухгалтерия	Трудовой	Б	50%	14 000,00р./мес	Пропорциональн	Стандартный
4		Юридический отдел	Трудовой	Ю	100%	17 000,00р./мес	Пропорциональное	Стандартный

Рисунок 17 – Перегруженные ресурсы

Выравнивание можно произвести двумя способами:

· Автоматическим

· Ручным

В своем проекте я использовал автоматическое выравнивание ресурсов. Результаты выравнивания отображаются в представлении «Диаграмма Ганта с выравниванием», которую можно вызвать с помощью кнопки «Другие представления».

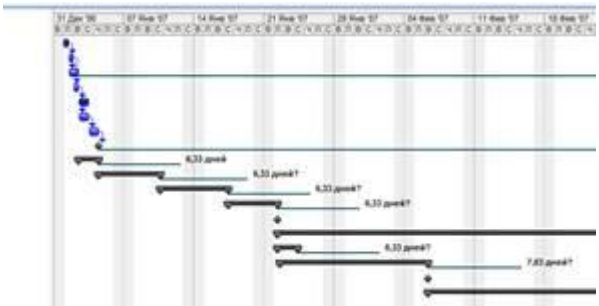


Рисунок 18 – Выравнивание

На разновидности диаграммы Ганта с выравниванием с помощью дополнительной графики Вы можете увидеть изменение загрузки при выполнении отдельных задач, а в таблице – новые значения загрузки по всем задачам. На диаграмме отображаются два набора отрезков задач: до выравнивания (выделены зелёным цветом) и после выравнивания (традиционные синий и черные цвета). Кроме того, в новом варианте плана могут появиться тонкие линии, обозначающие задержку задачи в результате выравнивания (отображаются зеленым цветом) и временной резерв (время, на которое задача может быть отложена – коричневым).

Практическое задание.

1. Произвести анализ соответствующей предметной области согласно выданному варианту
2. Выбрать программное обеспечение для внедрения в сферу соответствующей предметной области.
3. Сформулировать цели внедрения специализированного программного обеспечения.
4. Оформить сформулированные цели внедрения в виде иерархической структуры в порядке убывания важности.

Выполнить задания в рабочей тетради.

Варианты предметных областей:

1) небольшая сеть автомобильных заправочных станций; 2) автоматизация работы городской сети мини-ресторанов быстрого питания; 3) автоматизация работы городских библиотек; 4) автоматизация работы ЖКХ в пределах одного муниципального района; 5) автоматизация производства и учета мебельной фабрики; 6) создание интернет-провайдера; 7) автоматизация выездной розничной торговли; 8) спортивно-массовые мероприятия районного масштаба; 9) создание охранной системы промышленного предприятия; 10) создание АРМ сотрудников юридической консультации; 11) создание системы учета технологических операций автосборочного завода; 12) создание системы учета продаж и заказов автосалона.

Лабораторная работа №2. Описание и обоснование этапов внедрения ПО.

Теоретический материал.

Внедрение проекта включает в себя три этапа:

подготовка объекта к внедрению проекта;

опытное внедрение проекта

сдача его в промышленную эксплуатацию.

На этапе **Подготовка объекта к внедрению проекта** осуществляется комплекс работ по подготовке предприятия к внедрению разработанного проекта ИС.

На этапе **Опытное внедрение** осуществляют проверку правильности работы некоторых частей проекта и получают исправленную проектную документацию и составляют **Акт о проведении опытного внедрения**.

На этапе **Сдача проекта в промышленную эксплуатацию** осуществляют комплексную системную проверку всех частей проекта, в результате которой получают доработанный Техно-рабочий проект и **Акт приемки проекта в промышленную эксплуатацию**.

Эксплуатация и сопровождение проекта включает этапы:

эксплуатация проекта;

сопровождение и модернизация проекта.

На этапе **Эксплуатация проекта** получают информацию о работе всей системы в целом и отдельных ее компонентов и собирают статистику о сбоях системы в виде замечаний, которые накапливаются для выполнения следующего этапа.

На этапе **Сопровождение проекта** выполняются два вида работ:

ликвидируются последствия сбоев в работе системы и исправляются ошибки, не выявленные при внедрении проекта,

осуществляется модернизация проекта. В процессе модернизации проект либо дорабатывается, т.е. расширяется по составу подсистем и задач, либо производится перенос системы на другую программную или техническую платформу с целью адаптации ее к изменяющимся внешним и внутренним условиям функционирования, в результате чего получают документы модернизированного **Техно-рабочего проекта**.

На стадии ***внедрение проекта*** проводятся подготовка и постепенное освоение разработанной проектной документации ИС заказчиками системы и осуществляется выявление частных и системных принципиальных недоработок в предлагаемом проектном решении.

Внедрение может осуществляться с использованием следующих методов:

- **последовательный метод**, когда последовательно внедряется одна подсистема за другой и одна задача следует за другой задачей. Недостаток: увеличение длительности внедрения, что ведет за собой рост стоимости проекта.
- **параллельный метод**, при котором все задачи внедряются во всех подсистемах одновременно. Недостаток:возникает возможность пропуска ошибок в проектной документации (хотя сокращается время внедрения).
- **смешанный подход**, согласно которому проектировщики, внедрив несколько подсистем первым методом и накопив опыт, приступают к параллельному внедрению остальных. Используется чаще других.

Внедрение проекта осуществляется в течение трех этапов:

1. этап. Подготовка объекта к внедрению:

- изменяется организационная структура объекта;
- набираются кадры соответствующей;
- оборудуется здание под установку вычислительной техники;
- выполняются закупка и установка вычислительной техники;
- в цехах, отделах устанавливаются средства сбора, регистрации первичной информации и передачи по каналам связи;
- осуществляется создание файлов информационной базы с нормативно-справочной информацией.

В результате выполнения этапа составляется **Акт готовности объекта к внедрению** проекта ИС. Затем формируется состав приемной комиссии, разрабатывается **Программа проведения опытного внедрения** и издается **Приказ о начале опытного внедрения**.

2 этап. Опытное внедрение:

- подготовка исходных оперативных данных для задач, которые проходят опытную эксплуатацию;
- ввод исходных данных в ЭВМ и выполнение запланированного числа реализаций;
- анализ результатных данных на предмет наличия ошибок.

В случае обнаружения ошибок осуществляются поиск причин и источников ошибок, внесение коррективов в программы, в технологию обработки информации, в работу технических средств, в исходные оперативные данные и в файлы с условно-постоянной информацией. Кроме того, выявляется неквалифицированная работа операторов, что служит основанием для проведения комплекса мер по улучшению подготовки кадров.

После устранения ошибок получают **Акт о проведении опытного внедрения**.

3 этап. Сдача проекта в промышленную эксплуатацию.

- проверка соответствия выполненной работы договорной документации по времени выполнения, объему проделанной работы и затратам денежных средств;
- проверка соответствия проектных решений по ИС требованиям ТЗ;
- проверка соответствия проектной документации ГОСТам;
- проверка технологических процессов обработки данных по всем задачам и подсистемам;
- проверка качества функционирования информационной базы, оперативности и полноты ответов на запросы;
- выявление локальных и системных ошибок и их исправление.

В результате выполнения работ данном этапе составляется **Акт сдачи проекта в промышленную эксплуатацию**.

На стадии ***Эксплуатация и сопровождение проекта*** решается вопрос о том, чьими силами (персоналом объекта-заказчика или организации-разработчика) будут осуществляться эксплуатация и сопровождение проекта, и в случае выбора второго варианта заключается **Договор о сопровождении проекта**.

В процессе выполнения этапа ***эксплуатация проекта*** осуществляются исправления в работе всех частей системы при возникновении сбоев, регистрация этих случаев в журналах,

отслеживание технико-экономических характеристик работы системы и накопление статистики о качестве работы всех компонентов системы.

На этапе ***Сопровождение и модернизация проекта*** выполняется анализ собранного статистического материала, а также анализ соответствия параметров работы системы требованиям окружающей среды. Анализ осуществляет создаваемая для этих целей комиссия. По результатам анализа:

- дается заключение о необходимости модернизации всего проекта или его частей;
- определяется объем доработок, сроки и стоимость выполнения этих работ.

Метод классификации важности функций управления.

Метод MuSCoW.

Mu - Must have – необходимые функции, они обеспечивают критичные для успешной работы системы возможности.

S - Should have – желательные функции.

Co - Could have – возможные функции.

W – Won't have – отсутствующие функции. Используются для четкого представления границ проекта.

Кроме важности определяют трудоемкость и в первую очередь автоматизации подлежат важные и трудоемкие работы.

Для сложных ИС иногда на предпроектной стадии включают третий этап – разработка эскизного проекта. В эскизном проекте описываются алгоритмы обработки информации, описываются информационные потребности пользователей на уровне названий документов и показателей. Содержание эскизного проекта задается в ТЗ на систему.

Состав и содержание работ на стадии технорабочего (ТРП) проектирования. Работы на стадии технорабочего проектирования выполняются на основе утвержденного ТЗ. ТРП выполняется в 2 этапа: техническое и рабочее.

Техническое проектирование. Выполняются работы по логической разработке и выбору наилучших вариантов проектных решений. Результатом является технический проект.

Сначала уточняются цели создания ИС и выполняемые ею функции, устанавливается взаимосвязь с другими системами, уточняется и при необходимости изменяется организационная структура предприятия, затем разрабатываются локальные проектные решения, к числу которых относят следующие операции:

1. Разработка постановки задачи
2. Проектирование форм входных и выходных документов, системы введения документов и макетов экранных форм документов
3. Проектирование классификаторов экономической информации и системы ведения классификаторов
4. Проектирование состава и структур файлов информационной базы
5. Уточнение состава технических средств

Содержание документа, постановка задачи:

1. Цель, назначение решения конкретной задачи, периодичность решения задачи, описание связей с другими задачами.
2. Описание входной информации: перечень входных документов, периодичность возникновения и сроки получения информации, наименование и идентификаторы по каждой форме документа
3. Описание выходной информации
4. Описание алгоритма решения задачи: перечень формул расчета результатных показателей или описание математической модели и перечень последовательных шагов выполнения расчета.

Рабочее проектирование.

Этот этап связан с физической реализацией выбранного варианта проекта и получение документа рабочий проект. На этом этапе оформляется несколько видов документов.

1. Программная документация:

1. Описание программ
2. Спецификация программ
3. Тексты программ
4. Контрольные примеры
5. Инструкция для системного программиста, оператора и пользователя.

2. Технологическая документация. Предназначена для использования специалистами на каждом автоматизированном рабочем месте. В ее состав входят: технологические карты, которые разрабатываются на каждый процесс обработки информации при решении задач каждого класса, инструкционные карты, составляемые на каждую технологическую операцию.
3. Правовые инструкции. Определяют права и обязанности специалистов, работающих в условиях функционирования на предприятиях компонентов ИС.

Состав и содержание работ на стадиях внедрения, эксплуатации и сопровождения проекта.

Внедрение проекта включает в себя 3 этапа:

1. Подготовка объекта к внедрению проекта
2. Опытное внедрение проекта. На этом этапе проверка правильности работы некоторых частей проекта и получают исправленную проектную документацию и составляют акт о проведении опытного внедрения.
3. Сдача проекта в промышленную эксплуатацию. Осуществляют комплексную проверку всех частей проекта, в результате которой получают доработанный технорабочий проект и акт приемки проекта в промышленную эксплуатацию.

Эксплуатация и сопровождение включают 2 этапа.

Методология моделирования предметной области.

План:

1. Структурная модель предметной области

2. Функциональные методики моделирования предметной области
3. Объектные методики моделирования предметной области

Структурная модель предметной области.

Модель предметной области – это некоторая система имитирующая структуру или функционирование исследуемой предметной области и отвечающая основному требованию – быть адекватной этой области.

К моделям предметной области предъявляются следующие требования: формализация (обеспечивает однозначное описание структуры предметной области), понятность (должно быть понятно и заказчикам и разработчикам), реализуемость (наличие средств физической реализации модели), обеспечение оценки эффективности реализации модели на основе определенных методов и вычисляемых показателей.

Для реализации перечисленных требований строится система моделей, которая отражает структурный и оценочный аспекты функционирования предметной области.

Структурный аспект предполагает построение:

1. Объектной структуры, которая отражает состав взаимодействующих в процессах материальных и информационных объектах предметной области.
2. Функциональной структуры, отражающих взаимосвязь функций по преобразованию объектов в процессах.
3. Структура управления, отражающей события и бизнес-правила, которые воздействуют на выполнение процессов.
4. Организационной структуры, отражающей взаимодействие организационных единиц предприятия и персонала в процессах.
5. Технической структуры, описывающей топологию расположения и способы коммуникации комплекса технических средств.

Язык моделирования - это нотация (в основном графическая), которая используется для описания предметной области. Нотация является синтаксисом языка моделирования и представляет собой совокупность графических объектов, используемых в модели.

Оценочные аспекты. Связаны с разрабатываемыми показателями эффективности автоматизируемых процессов, к которым относятся:

1. Время решения задачи
2. Стоимостные затраты на обработку данных
3. Надежность процессов
4. Косвенные показатели (объемы производства, производительность труда, рентабельность и т.д.)

В основе различных методологий лежат принципы последовательной детализации. И обычно модели строятся на 3х уровнях:

1. Внешний (определение требований). На этом уровне определяется состав основных компонентов системы: объектов, функций, событий, технических средств.
2. Концептуальный (спецификация требований). Определяется характер взаимодействия компонентов системы.

3. Внутренний (реализация требования). Модель отвечает на вопрос: С помощью каких программно-технических средств реализуются требования к системе.

Практическое задание.

1. Перечислить виды работ по внедрению ПО согласно указанному в теоретическом материале и адаптировать их перечень согласно предметной области, выбранной вами ранее.
2. Подробно описать действия по внедрению на каждом этапе исходя из выбранной предметной области.
3. Дать краткие обоснования необходимости выполнения вышеперечисленных действий.

Лабораторная работа №3. Оценивание эффективности внедрения программного обеспечения.

Теоретический материал.

Критерии эффективности использования программных продуктов

При оценке эффективности использования информационной системы управления проектами необходимо рассматривать обширный набор аспектов-критериев. Существуют различные подходы к оценке эффективности использования ИСУП (Project Management Value), основывающиеся на методиках различных организаций (как коммерческих, так и независимых научных), оптимизированных для использования в разных областях хозяйственной деятельности.

Оценка эффективности основывается на определении, выборе критериев для рассмотрения и оценки системы по этим качествам. Набор критериев может зависеть от сферы деятельности организации, характеристики проектов и состава системы.

Критерии, показатели и оценки можно условно разделить на две группы: **качественные и количественные. Количественные** оценки дают легко осязаемый, наглядный показатель эффективности, однако не всегда дают полное представление о всех преимуществах использования ИСУП. При оценке эффективности необходимо рассматривать набор показателей по различным аспектам проектной деятельности, таким как финансовые, временные, методологические, организационные и др.

Одна из методологий **качественной** оценки эффективности основана на экспертной оценке Критических факторов успеха (КФУ), выполнение которых необходимо для успешной реализации проекта.

При формулировании целей проекта всегда следует помнить о конкретных критериях успеха, которые оказывают непосредственное влияние на эффективность проекта.

Система Критических факторов успеха проекта механизм для стратегической оценки проекта в целом, основанный на экспертной оценке.

Данный метод рекомендуется использовать неоднократно на этапе выполнения проекта. Его проводят циклически через определенные промежутки времени, например, каждый месяц или при закрытии этапа проекта.

На основе анализа успешных проектов, было выявлено несколько критических факторов, оказывающих наибольшее влияние на проект.

В соответствии с разработанной моделью успех проекта зависит от таких факторов, как:

· анализ со стороны высшего руководства - понимание высшим руководством организации важности проекта, готовность обеспечить проекту необходимую поддержку посредством личного участия или делегирования соответствующих полномочий членам команды;

· задачи проекта ? исходная ясность миссии проекта, понимание полезности результатов проекта;

· четкое планирование работ ? понимание путей достижения целей (за счет каких работ будут достигнуты цели проекта, в какие сроки, какие ресурсы для этого потребуются);

· взаимоотношения с Заказчиком предусматривают активную работу с Заказчиком при разработке проекта, информирование его о продвижении работ в рамках проекта;

· учет потребительских требований определяет удовлетворенность пользователей результатами проекта, обеспечивает успешную сдачу системы в эксплуатацию;

· наличие необходимых технологий (используемые в проекте технологические решения доступны, надежны, опробованы, осуществляется необходимый контроль их правильного использования);

· наличие подготовленного персонала (подготовленность сотрудников к осуществлению проекта конкретного профиля, готовность провести обучение сотрудников или набор соответствующих специалистов, иногда привлечение консультантов).

Количественная оценка эффективности проектной деятельности компании может проводиться методом сравнительного анализа тенденций изменения определённых характеристик:

· отклонения по стоимости проекта отклонения бюджета проекта, вызванные его перерасходом или недорасходом;

· отклонения в расписании сдвиги в расписании проекта, вызванные отставанием или опережением работ;

· устранение недостатков, найденных при проверке и оценке качества ? оценка эффективности работы команды проекта по устранению недостатков, выявленных в ходе выполнения проекта;

· количество неразрешенных проблем ? эффективность реагирования команды проекта на возникающие трудности;

· укомплектованность команды проекта ? определение всех участников проекта, а также команды управления проектом.

Потребностям в количественных оценочных механизмах деятельности компании, а также механизмах опережающего, повседневно-стратегического управления полностью соответствует методология стратегического управления Balanced Scorecard ? Система Сбалансированных Показателей (ССП).

В рамках ССП организация рассматривается и оценивается в четырех перспективах:

· в перспективе, связанной с финансовым состоянием (общепринятые финансовые показатели);

· в перспективе, связанной с позицией компании на рынке (число клиентов, доля рынка и т.д.);

· в перспективе, связанной с внутренними бизнес процессами (насколько они настроены и эффективны);

· в перспективе, связанной с развитием и обучением персонала.

Для каждой определённой цели компании вырабатываются ключевые показатели деятельности (КПД, Key Performance Indicator KPI). С помощью подбора ключевых показателей деятельности, которые являются, по сути, измерителями достижимости целей, компания получает хорошо сбалансированную картину кратко- и среднесрочных целей, финансовых и нефинансовых показателей деятельности.

Методика КПЭ инструмент, облегчающий процесс принятия управленческих решений за счет обеспечения руководства полноценной информацией.

Примером обобщённого КПЭ проектной деятельности компании может служить показатель ?Проектное отклонение?:

Главной особенностью процессов проектно-ориентированной организации является их стандартная структура и стандартные ограничения. Именно эти стандартные ограничения по времени, стоимости реализации проектов и по качеству результатов и могут быть использованы для построения обобщенного показателя, характеризующего деловые процессы проекта через оценку возникающих отклонений.

Показатели эффективности

Оценка эффективности является частью управления сферой ИТ. Оценка эффективности рассматривается в COBIT и включает в себя постановку и контроль целей (достижение которых поддается оценке), которые определяют результаты ИТ процессов и путь достижения этих результатов (потенциал процесса и эффективность).

Для оценки эффективности выделяют 5 ключевых областей управления ИТ: **с**оответствие стратегии, полезность, управление ресурсами, управление рисками, оценка эффективности

· *Соответствие стратегии* делает акцент на связи между планами бизнеса и ИТ; выявлении, поддержке и контроле за ценностным предложением ИТ; а также на соответствии ИТ и бизнес операций.

· *Полезность* представляет собой реализацию ценностного предложения, контроль за тем, чтобы ИТ обеспечивали определенные стратегией преимущества, сосредоточение на оптимизации затрат и подтверждение подлинной ценности ИТ.

· *Управление ресурсами* посвящено вопросам, связанным с управлением критичными ИТ ресурсами, а именно, оптимизацией инвестиций и должному руководству приложениями, информацией, инфраструктурой и персоналом. Ключевые вопросы касаются оптимизации знаний и инфраструктуры.

· *Управление рисками* требует осведомленности высшего руководства в области рисков, четкого понимания корпоративного подхода в их отношении, соответствия требованиям прозрачности в отношении существенных рисков, включения функции управления рисками в практику организации.

· *Оценка эффективности* представляет собой контроль за реализацией стратегии, результатами проектов, использованием ресурсов, эффективностью процессов и сервисным обслуживанием. Для этого применяются, в частности, системы сбалансированных показателей, которые преобразуют стратегию в последовательность

действий, результаты которых измеряются иными, по сравнению с бухгалтерским учетом, методами.

В приложении E выведен список соответствия процессов и ключевых областей управления. Для отдельно взятого процесса область может быть приоритетной, второстепенной или незначимой. Матрицу можно использовать для отбора значимых процессов при оптимизации работы той или иной области управления. С другой стороны, руководитель может ориентироваться на данную таблицу при оценки процессов соответствия моделям зрелость.

В приложении D, о котором говорилось ранее, приведена значимость критериев оценки информации в каждом процессе. Так, в процессе «Разработка стратегического плана развития ИТ» значимыми являются параметры «Результативность» и «Эффективность». При этом первый критерий является приоритетным, а второй второстепенным. Матрица «Критерии оценки информации в процессах» определяет критерии информации на которые нужно ориентироваться при оценке выбранного процесса.

Другим фактором для оценки модели зрелости являются показатели результативности (KGI) и показатели эффективности (KPI).

Показатель результативности говорит о том, достигнуты ли определенные цели. Эти показатели могут быть измерены только после совершения факта и, поэтому, называются «индикаторами задержки».

Показатели эффективности говорит о том, вероятно ли вообще достижение цели. Эти показатели могут быть измерены до получения результата и, поэтому, называются «индикаторами опережения».

Показатели KPI и KGI можно разделить на три уровня: уровень бизнес целей, уровень ИТ процесса, уровень действий. На первом уровне показатели эффективности ИТ определяют, что является вкладом ИТ в достижение бизнес целей и как это измерить. На втором уровне показатели эффективности ИТ процесса демонстрируют, что является вкладом ИТ процесса в достижение ИТ целей и как это измерить. На последнем уровне показатель эффективности отдельных видов деятельности (действий) определяет что должно произойти внутри ИТ процесса для достижения требуемой эффективности и как это измерить. Показатели для всех уровней и процессов представлены в таблице 1.

Для оценки выбранных процессов используются критерии эффективности и результативности. В приложении F заявлены все показатели оценки представленные в CobiT, а так же даны меры их измерения и желаемые результаты. По способам расчеты критерии можно сгруппировать в несколько типов

1) Тип «Доля» (%). Доля представляет собой отношение количества измеряемого атрибута, указанного в наименовании критерия, и общего количества данного свойства. Атрибутами могут выступать инвестиции, денеги, рабочие часы или дни, число сотрудников, заинтересованные стороны, должности, проекты, простои и др. Вычисленное значение умножается на сто, таким образом получается процентное выражение показателя.

2) Тип «Число» (шт, денежные единицы, часы/дни/недели). Количественный показатель, который определяет простым подсчетом необходимого атрибута. Например, число обращений, число подразделений. Иногда такие показатели так же могут быть измерены в процентах по принципу вычисления «Долей».

3) Тип «Оценка». Это показатели, для которых единицами измерения выступаю баллы от 0 до 5. Принцип подсчета таких показателей взят по аналогии с моделями зрелости. Каждому из баллов соответствует значения.

Таблица 1 – Значение оценки при измерении коэффициента в баллах от 0 до 5

нет данных	Несуществующий
очень низкая	Начальный
низкая	Интуитивный
средняя	Определенный
выше среднего	Управляемый и измеряемый
высокая	Оптимизированный

Желаемые результаты (KGI) представляют собой краткие рекомендации или максимальное значение показателя результативности. Данные рекомендации так же можно распределить по типами.

1) Тип «Оптимизация». Данный критерий означает, что не существует числового или идеального конечного результата для оценки показателя. Задачей управления является вывести значение показателя на оптимальный уровень, соответствующий стратегическим и тактическим целям организации.

2) Тип «Максимизация». Измеряемый показатель обладает конечным значением, но является недостижимым в силу субъективных и объективных причин, поэтому желаемым результатом в оценке такого показателя является максимизация (увеличение) значения показателя. Примером, такого показателя может служить «Доля членов Совета директоров, прошедших обучение управления ИТ». Объективно понятно, что добиться сто процентного результата при оценки данного показателя практически невозможно, а так же не является столько необходимым. Но пользу от увеличение числа грамотных управленцев ИТ в Совете директоров нельзя не учитывать.

3) Тип «Минимизация». Измеряемый показатель обладает конечным значением, но является недостижимым в силу субъективных и объективных причин, поэтому желаемым результатом в оценке такого показателя является минимизация(уменьшение) значения показателя. Например, показатель «Временная задержка между выявлением потребности в обучении и предоставлении обучения». Разумно предположить, что невозможно сделать данный показатель равный нуль, но стремление к уменьшению значения показателя является приоритетным в управлении ИТ.

4) Тип «100 %» и тип «0 %» . Максимально возможное значение и минимально возможное значение показателя.

Измерив, показатели мы можем соотнести их с максимальными значениями и определить к какому уровню зрелости возможно отнести процесс в целом.

Пример методики для оценки программных продуктов

Критерии оценки программных продуктов

№	Признак	Описание
1	Описание КИС	Критерии, с помощью которых осуществляется оценка непосредственно самой КИС
1.1	Структура	Критерии оценки структуры КИС
1.2	Функционал	Критерии оценки функциональных возможностей КИС
1.3	Принципы	Критерии оценки принципов построения КИС
1.4	Технические требования	Критерии оценки технических требований функционирования КИС
1.5	Архитектура	Критерии оценки особенностей архитектуры заложенной при создании КИС
1.6	Стоимость	Критерии оценки стоимостных параметров КИС
1.7	Интеграция	Критерии оценки внутренней и внешней интеграции КИС
1.8	Бизнес-логика	Критерии оценки реализации бизнес-логики, заложенной в КИС
1.9	Элементы КИС	Критерии оценки элементов КИС
2	Описание фирмы-разработчика и ее партнеров	Критерии оценки фирмы-разработчика КИС и фирм-партнеров по продвижению и внедрению данной КИС
2.1	Технологии	Критерии оценки технологий, подходов, методов, применяемых фирмами-внедренцами в процессе внедрения КИС
2.2	Опыт	Критерии оценки опыта успешных и неудачных проектов фирм-внедренцев
2.3	Специалисты	Критерии оценки квалификационного уровня специалистов фирм-внедренцев
3	Принципиальность	Отношение *критериев выбора* КИС к соответствующим этапам выбора КИС
3.1	Принципиальные	Критерии, которые должны быть оценены в первую очередь и, которые определяют основные принципы выбираемой КИС
3.2	Не принципиальные	Критерии, которые не являются сильно принципиальными при выборе КИС

4	По степени детализации	На сколько критерий конкретно описывает исследуемый объект
4.1	Общие	Критерии, описывающие объект исследования в общем виде
4.2	Конкретные	Критерии, конкретно описывающие объект исследования
5	По сложности оценки	Доступность и достоверность информации для самостоятельной оценки
5.1	Сложный	Возможность самостоятельного получения полной и достоверной информации по данному критерию очень мала
5.2	*Средней сложности*	Возможно самостоятельное получение полной и достоверной информации
5.3	Легкий	Возможно самостоятельное получение полной и достоверной информации. Информация в общедоступных источниках
6	По типу значения	В зависимости от типа значения критерия: возможность количественного измерения значения, либо качественный показатель
6.1	Количественный	Критерии, значения которых могут быть определены в виде конкретных числовых показателей
6.2	Качественный	Критерии, значения которых не могут быть определены в виде конкретных числовых показателей
7	По важности для потенциальных пользователей	Описывают степень важности того или иного *критерия выбор* а КИС для потенциальных пользователей
7.1	Высший приоритет	Критерии, имеющие наибольшую значимость для пользователя
7.2	Средний приоритет	Критерии, имеющие среднюю значимость для пользователя
7.3	Низший приоритет	Критерии, имеющие низшую значимость для пользователя

Группы критериев оценки программных продуктов:

1. назначение и возможности пакета (область использования, степень обеспечения функций, общего назначения или специализированный);
2. отличительные признаки и свойства пакета (входной язык, структура массивов данных, способы проверки данных);
3. требования к техническим и программным средствам (объем ОП, периферийные устройства, тип ОС);

4. документация пакета (наличие руководства по использованию, руководства программиста, руководства системного программиста);
5. факторы финансового порядка (затраты на приобретение, необходимость ежегодных платежей);
6. особенности установки пакета (объем работ, время установки, требования к квалификации программистов);
7. особенности эксплуатации пакета (надежность, защита данных, возможность эксплуатации силами предприятия);
8. помощь поставщика по внедрению и поддержанию пакета (обучение персонала, внесение модификаций, обновление версий);
9. оценка качества пакета и опыт его использования (число внедрений пакета, оценки пользователей, номер версии);
10. перспективы развития пакета (совместимость версий, дополнение функциональных возможностей, развитие методов).

Пример оценки программного продукта

Для выбора программного продукта, наилучшим образом удовлетворяющего потребности Предприятия, консультанты провели *анализ*программных продуктов в соответствии с системой требований.

При этом использовались методики оценки, приведенные ниже в настоящем разделе.

Оценка существующей функциональности программного продукта

При анализе тиражируемых программных продуктов, предполагаемых к внедрению как основы ИСУ Предприятия, функциональные возможности программных продуктов оценивались по степени их соответствия разработанным требованиям по десятибалльной шкале.

При оценке применяется следующая шкала баллов:

0 - функция отсутствует в имеющейся конфигурации;

2 - функция реализована частично, для ее реализации необходима серьезная доработка программного кода при настройке/внедрении;

4 - функция реализована частично, для ее реализации необходима незначительная доработка программного кода при настройке/внедрении;

6 - функция реализована удовлетворительно, требуется адаптация под нужды Предприятия в процессе настройки/внедрения средствами ИСУ;

8 - функция реализована хорошо, однако в перспективе могут понадобиться ее доработки;

10- функция реализована полностью, удовлетворяет требованиям (в том числе - на перспективу).

Оценки "по умолчанию" выстроены по шкале четности, при заполнении теста могут применяться нечетные оценки в случае, если ответ находится на грани двух смежных четных оценок.

Оценка прочих аспектов

Оценка соответствия прочих аспектов тиражируемых программных продуктов и Поставщиков разработанным требованиям производится также по десятибалльной шкале. Количество баллов определяет степень соответствия программного продукта рассматриваемому требованию.

Система весовых коэффициентов

Для получения интегральной оценки программных продуктов и поставщиков введены весовые коэффициенты для определения значимости тех или иных критериев для Предприятия.

Используется следующая система весовых коэффициентов:

1 - реализация функции в ИСУ имеет низкую важность[1];

2 - реализация функции важна в ИСУ;

3 - реализация функции в ИСУ критически важна для Предприятия.

Предпочтение должно отдаваться программным продуктам, имеющим наибольший рейтинг (суммарную оценку с учетом весовых коэффициентов).

Основные выводы по результатам анализа программных продуктов

В соответствии с Техническим заданием консультантами были проанализированы следующие программные продукты:

- "Рос-1" версия 5.8;
- "Рос-2" версия 2.3.3;
- ""Рос-3":Предприятие" версия 7.0;
- "Зап-1";
- "Зап-2" версия 2.6.

Основные результаты анализа приведены в следующей таблице.

Таблица Основные результаты анализа программных продуктов

№	Наименование критерия	"Рос-1"	"Рос-2"	"Рос-3"	"Зап-1"	"Зап-2"	Макс. балл
Общесистемные функциональные требования		**1 105**	**1 056**	**1 128**	**951**	**1 126**	**1 680**
Функциональные требования по подсистемам управления		**5 452**	**3 431**	**3 334**	**3 486**	**3 385**	**8 310**
1.	Маркетинг	400	146	262	112	110	**610**

2.	Сбыт	328	284	262	300	300	**420**
3.	Производство	1 410	684	420	810	918	**2 070**
4.	Снабжение	327	183	210	372	381	**720**
5.	Управление складами	222	174	72	222	186	**300**
6.	*Управление персоналом*	555	402	552	90	24	**900**
7.	Управление транспортом	168	18	114	24	0	**210**
8.	Управление строительством	36	36	24	12	18	**120**
9.	Взаиморасчеты	490	340	278	406	342	**660**
10.	Финансы	524	438	286	420	396	**940**
11.	Управление себестоимостью	138	102	96	78	84	**210**
12.	Бухгалтерский и *налоговый учет*	854	624	758	640	626	**1 150**
Прочие требования		**564**	**460**	**525**	**362**	**434**	**730**
Итого:		**7 121**	**4 947**	**4 987**	**4 799**	**4 945**	**10 720**

Анализ программных продуктов проведен в соответствии с описанной выше методикой.

В ходе анализа консультанты исходили из того, что наиболее важными для Предприятия являются те функции программных продуктов, которые дают Предприятию следующие возможности:

- планирование и контроль фактического выполнения работ (входит в подсистему управления "Производство" в табл.
- управление движением товарно-материальных ценностей (подсистемы "Снабжение" и "Управление складами");
- планирование и контроль финансовых потоков, контроль задолженностей и взаиморасчетов (подсистемы управления "Финансы" и "Взаиморасчеты");
- бухгалтерский и *налоговый учет* (подсистема управления "Бухгалтерский и *налоговый учет*");
- *управление персоналом*, включая кадровый учет и расчет заработной платы (подсистема "*Управление персоналом*").

Анализ указанных выше программных продуктов показал следующее.

Из рассмотренных консультантами программных продуктов наилучшими функциональными характеристиками обладает система "Рос-1". Основные причины этого следующие:

- только два программных продукта ("Рос-1" и "Зап-2") обладают возможностями планирования и контроля фактического исполнения работ (подсистема "Производство");

- по подсистеме "Снабжение" наилучшей функциональностью обладают программные продукты "Рос-1", "Зап-1" и "Зап-2";
- по подсистеме "Управление складами" лучше всего удовлетворяют выдвинутым требованиям программные продукты "Рос-1" и "Зап-1";
- в подсистеме "*Управление персоналом*" наиболее развитую функциональность имеют программные продукты "Рос-1", "Рос-2" и "Рос-3";
- в подсистемах "Финансы", "Взаиморасчеты" и "Бухгалтерский и *налоговый учет*" функциональные возможности "Рос-1" заметно шире, чем у других программных продуктов.

Таким образом, **с точки зрения функциональности наиболее приемлемым программным продуктом для Предприятия является система "Рос-1".**

Практическое задание.

1. Выбрать и записать качественные показатели эффективности внедрения ПО, исходя из выбранной предметной области, и аргументировать выбор показателей.
2. Выбрать и записать количественные показатели эффективности внедрения ПО, исходя из выбранной предметной области, и аргументировать выбор показателей.
3. Сделать подробный вывод о том, является ли внедренное вами ПО целесообразным и эффективным на фоне аналогичных решений.

Лабораторная работа №4. Оценивание эффективности внедрения программного обеспечения.

Теоретический материал.

1.Расчет экономической эффективности внедрения программного продукта

В данной работе рассматриваются направления инвестирования на компьютеризацию управленческого труда, на внедрение компьютерных технологий. Целью мероприятия является разработка и внедрение программного продукта «Разработка АИС в подразделении отдела кадров ГУ ОВО г.Кумертау».

Целью работы является расчет экономической эффективности от внедрения программного продукта «Разработка АИС в подразделении отдела кадров ГУ ОВО г.Кумертау».

Таблица 1

Общая характеристика мероприятия

Мероприятие	Сроки работы, ч.	Размер инвестиций	Участие инвестора

Разработка программного продукта	160	40249,26	Собственные средства
Обучение персонала	80	600	Собственные средства

1. Общее направление инвестирования (автоматизация расчета и составления сметы затрат на строительно-монтажные работы);
2. Период осуществления проекта -1 месяц, в том числе:

- разработка алгоритма программы – 40 ч.;
- разработка программного продукта – 72 ч.;
- отладка программы – 8 ч.;
- создание интерфейса – 40 ч.;

3. Размер инвестиций составляет 40249,26 тыс. руб.;
4. Источниками инвестирования являются собственные средства.

2.Формирование общих затрат проекта

Необходимо определить размер инвестиций, требуемые для реализации данного проекта и направленные на компьютеризацию управленческого труда.

Расчет инвестиций производятся по форме, приведенной в таблице 2.

Таблица 2

Расчет инвестиций

Инвестиции	Общая стоимость, руб..
Оборудование:	
1 Компьютер	20700
2 Принтер	9371
3 Стол	1556,74
4 Стул	1187,88
Программные средства:	
1 MS Office	4833
2 Windows’2007	2000,64
Переподготовка персонала	600
Итого:	40249,26

3.Расчет полного фонда заработной платы до внедрения

Расчеты основной заработной платы персонала осуществляются с помощью определения категории работающих. Расчеты производятся по форме, приведенной в таблице 3.

Таблица 3

Расчет основной заработной платы персонала

Профессия	Оклад ($Т_с$), руб	$N_ф$, час	Числ., чел	Сумма, руб		Доплаты 12%, руб	Премия, 30%, руб	ФЗП с учетом районного коэффициента
				мес	год			
Инженер	5500	-	2	11000	132000	15840	44352	221020,8
Итого:	-	-		-	-	-	-	221020,8

Фонд заработной платы - это сумма заработной платы персонала с доплатами и премиями и с учетом районного коэффициента.

Дополнительная заработная плата персонала на данном предприятии составляет 12% от годового фонда заработной платы, премии соответственно– 30%, уральский районный коэффициент – 15%.

Отчисления на социальные нужды составляют 30% от суммы ФЗП

$$О_{СН} = ФЗП \cdot 30\%, \ руб., \quad (1)$$

где $О_{сн}$ - отчисления на социальные нужды, руб.;

ФЗП – фонд заработной платы персонала с учетом районного коэффициента (таблица 3), руб.

$$О_{СН} = 221020{,}8 \cdot 30\% = 66306{,}24 \ руб.$$

Затраты на оплату труда до модернизации составят 287327,04 рублей (221020,8 + 66306,24)

Расчеты затрат на вспомогательные материалы производятся по форме, приведенной в таблице 4.

Таблица 4

Расчет стоимости материалов

Наименование материалов	Расход на год, ед.	Стоимость, руб.	
		единица	общая
1 Диск	10	13	130
2 Картридж	3	2500	7500

3 Бумага	5	125	760
Итого:	-	-	8390

Далее с помощью технических характеристик необходимо определить расход электроэнергии одного комплекта оборудования (в комплект входят: компьютер, принтер) за час работы.

$$P_{.час} = \sum_{1}^{n} P_{уд.час.} \quad (2)$$

где n- количество оборудования, шт.

$P_{уд.час.}$ - удельный расход электроэнергии одного комплекта оборудования за час работы, кВт.

Удельный расход электроэнергии за год определяется как произведение $P_{уд.час.}$ на общий годовой объем работ.

$$P_{уд.год.} = P_{уд.час.} \cdot V, \ кВт \quad (3)$$

где $P_{уд.год.}$ - удельный расход электроэнергии одного комплекта оборудования за год работы, кВт;

V - количество часов работы в месяц, V= 176 час.

$$P_{уд.год.} = 0{,}288 \cdot 176 = 50{,}688 \ кВт$$

По данным практики расход электроэнергии в месяц на освещение рабочего помещения равен 42 кВт. Стоимость одного кВт/час электроэнергии по тарифу на сентябрь 2013 года составляет 4,5 рублей.

Стоимость электроэнергии за месяц рассчитывается по формуле:

$$C_{эл} = к \cdot (P_{уд.год.} + P_{осв.}) \cdot a, \ руб., \quad (4)$$

где а - стоимость кВт/час, руб.

где $P_{осв.}$ - годовой расход электроэнергии на освещение рабочего помещения, кВт.

$$C_{эл} = (50{,}688 + 42) \cdot 4{,}5 = 417{,}096 \ руб.$$

Т.к. разработка программы будет длиться 2 месяца затраты на создание программного продукта составят:

$$C = З + М + C_{эл} \ руб., \quad (5)$$

где З- зарплата инженера, занимающегося разработкой программного продукта;

Эл – электроэнергия;

М – затраты на материалы.

$$C = (2 * 10000) + (2 * 417{,}096) + 8390 = 29224{,}192 \, рублей$$

Практическое задание.

1. Перечислить виды работ по внедрению ПО согласно указанному в теоретическом материале и адаптировать их перечень согласно предметной области, выбранной вами ранее.
2. Подробно описать действия по внедрению на каждом этапе исходя из выбранной предметной области.
3. Дать краткие обоснования необходимости выполнения вышеперечисленных действий.

Лабораторная работа №5. Оценка рисков при внедрении

Теоретический материал.

К факторам, которые гарантировано приводят к неудачным внедрениям, следует отнести слабое руководство проектом со стороны заказчика и отсутствие у него методологии для автоматизируемых участков. Рассмотрим их подробнее.

Слабое руководство проектом со стороны заказчика

Вопрос назначения руководителя проекта внедрения на многих предприятиях решается непросто. С одной стороны есть ключевые менеджеры, которым проект на самом деле нужен, но с другой – они сильно перегружены текущей работой, и не в состоянии выполнять функции руководителя проекта.

Часто бывают случаи, что руководителем проекта назначают нового сотрудника, недавно принятого на работу. Такому руководителю проекта очень трудно добиться оперативного принятия решений на этапе проектирования системы, а также заставить пользователей работать с системой на этапе внедрения. Несмотря на то, что формально рычаги воздействия у него есть (может, например, подготовить приказ, согласовать его с вышестоящим руководством и довести до сотрудников) – на самом деле складывается ситуация, что любое, даже самое мелкое решение принимается несколько дней. Это, конечно же, влечет за собой срыв сроков проекта.

Похожая ситуация наблюдается и в случае, когда руководителем проекта назначают уже работающего сотрудника, но не очень заинтересованного в самом проекте. Например, представителя ИТ службы заказчика.

Задача компании, внедряемой решение, состоит в том, чтобы до начала проекта снять с заказчиком все вопросы, связанные с руководством проекта.

Бывают случаи, когда несмотря на наши предостережения, заказчик назначает руководителем проекта человека не наделенного особыми полномочиями по проекту. В этом случае внедряющая компания участвует в проекте до окончания этапа предпроектных работ, по окончанию этапа, если риски оправдываются – приостанавливает проект до решения проблемы с руководством со стороны заказчика.

Рассмотрим это на примере проекта автоматизации крупной промышленной компании.

Переговоры о начале работ давали основание полагать, что проект будет успешным. Острую заинтересованность в успешности работ показал генеральный директор, а также основной заказчик системы – финансовый директор. Перед началом работ были оговорены с заказчиком возможные риски.

В день начала предпроектных работ был представлен руководитель проекта – бывший сотрудник большой известной консалтинговой компании с опытом выполнения подобных проектов. Следует отметить, что этот человек принят на работу в компанию всего несколько дней назад и не интегрирован в функциональную структуру заказчика.

Первые дни выполнения предпроектных работ прошли без срыва сроков, но как только приблизился этап согласования частей технической документации – ответственные сотрудники заказчика, как обычно, начали затягивать сроки, мотивируя это тем, что у них нет времени (заняты текущей деятельностью). Руководитель проекта со стороны заказчика пытался эту проблему решить исключительно бюрократичными методами – подписывал у руководства распоряжения о сроках завершения согласования. Сроки, конечно же, срывались, на что всегда находились аргументированные, с точки зрения исполнителя, пояснения.

Работа выполнялась только под давлением финансового директора, который добивался от исполнителей результата к концу текущего дня.

Результатом такой предпроектной работы стало превышение в 2 раза срока сдачи этапа и как следствие финансовые потери для внедряющей компании.

Продолжение проекта было возможно только при условии назначения руководителем проекта финансового директора, который способен оказывать реальное влияние на менеджеров среднего звена.

Отсутствие у Заказчика методологии для автоматизируемых участков

Чтобы проект был успешным заказчик должен понимать, зачем этот проект нужен, какие цели и каким образом он будет их достигать с точки зрения методологии учета (программное обеспечение вторично).

Часто задачу построения методологии, которая приводит к достижению целей, берут на себя консалтинговые компании – методология в результате предельно формализирована. Можно участвовать в проектах и когда нет формального документа, например, «методология управленческого учета», но главное – чтобы были люди в команде заказчика, которые понимают, как они должны работать (с точки зрения методологии).

На этапе переговоров выявить степень готовности методологической части (если она не формализирована) у заказчика очень сложно (редко какой руководитель признается, что не имеет понятия, как ведется управленческий учет). В этом случае, консультанты могут участвовать в проекте до окончания этапа предпроектных работ, а по окончанию этапа, если риски оправдываются – проект следует приостановить до решения проблемы методологии учета.

Рассмотрим это на примере проекта автоматизации сети торговых предприятий.

По результатам первичных переговоров было выяснено, что все торговые предприятия сети ведут учет по почти единой методологии бухгалтерского и налогового учета, лишь с небольшими отклонениями. Главный бухгалтер всего холдинга – профессиональный специалист и понимает, что ему нужно получить в результате внедрения проекта автоматизации.

Первые же дни работы специалистов внедряющей компании в разных торговых центрах показали, что учетная политика у них сильно отличается: одни работают по упрощенной схеме налогообложения, другие ведут лизинговую деятельность, существуют разные политики по учету НДС, учету ТМЦ на складах.

Опаснее всего для реализации проекта было то, что при демонстрации всех различий главный бухгалтер холдинга оперативно не реагировал на просьбы специалистов внедряющей компании привести методологии предприятий в соответствии с требованиями проекта, что в свою очередь удлиняло сроки. Главный бухгалтер на самом деле даже и не знал о таких отличиях учета на разных предприятиях холдинга, и нужно было время, чтобы принять решение, удовлетворяющее все торговые центры.

В результате у заказчика была создана методологическая группа, в которую вошли представители торговых центров и самого холдинга, ответственная за принятие решений по различию методологий.

После создания группы работа пошла быстрее, но эта ситуация не могла не отразиться на сроках выполнения предпроектных работ (и, на финансовых результатах проекта).

Работа по проекту в целом была доведена до конца, проект был выполнен, но с существенным (на несколько месяцев) сдвигом сроков.

По результатам этого проекта был сделан вывод, что на предпроектном этапе заказчик должен представить документ, подтверждающий единую учетную политику. Если же его нет –то следует порекомендовать заказчику составить такой документ самостоятельно, или обратиться в одну из консалтинговых компаний для получения помощи в его составлении.

Существуют и другие риски, не менее важные при выполнении проектов внедрения автоматизированных решений. К ним отнесем:

- согласование проектной документации;
- наличие работающей схемы управления изменениями;
- ответственность пользователей за обучение;

- ответственность пользователей за результаты опытной эксплуатации.
- Кратко остановимся на каждом из них.
  Согласование проектной документации

  Разработка системы выполняется в рамках согласованной обеими сторонами проектной документации. Нередко бывают случаи, что исполнители от заказчика не очень ответственно подходят к вопросу согласования проектной документации, полагая, что все документы можно будет согласовать уже на этапах внедрения системы. Такое отношение к согласованию проектной документации приводит при внедрении системы к большому количеству отклонений (дополнительных требований, не зафиксированных в проектной документации). Чтобы этого избежать, проводятся следующие предупредительные мероприятия.

  На первом и последнем совещании управляющего комитета по этапу «предпроектные работы» руководитель проекта с внедряющей стороны объясняет, что означает согласование заказчиком проектной документации, и какой порядок действий будет в дальнейшем, если будут проходить отклонения от проектной документации.

  В разделе подписей проектной документации вставляется формулировка «Подтверждаю корректность и полноту …» – для того, чтобы исполнители еще раз обратили на это внимание.

  Подобного рода предупредительные мероприятия привели к тому, что эта проблема на проектах почти исчезла.

  Наличие работающей схемы управления изменениями

  Очень важным аспектом организации работы на проекте является управление потоком информации, т.е. вопросами и предложениями, возникающими как со стороны исполнителя, так и со стороны заказчика.

  Вопросы пользователей системы не должны остаться без внимания и ответа, чтобы них не накапливался негатив против самого проекта внедрения. Для обеспечения такой возможности необходимо регистрировать каждое такое обращение.

  Для предупреждения такого рода проблем, на этапе тестирования системы исполнители устанавливают у Заказчика конфигурацию (разработанную по технологии ITIL), в которой любой пользователь может задать вопрос, зафиксировать ошибку, или высказать пожелание. В уставе проекта фиксируется порядок разбора такого рода информации от пользователей, а также регламентируются сроки реакции на эту информацию. Почти всегда в цепочке обработки этой информации присутствуют ответственные лица заказчика (спонсор проекта – согласовывает доработки в части финансов, ответственные за методологию – согласовывают функциональную часть доработок, группа тестирования – если такая есть – проверяет наличие ошибки). Конфигурация дает возможность в автоматическом режиме отслеживать сроки реакции на сообщения, и выносить вопросы на управляющий комитет проекта, в случае, если есть отклонения от оговоренных сроков.

Работа такого механизма, и в том числе задача руководителя проекта со стороны заказчика заключается в обеспечении формальной регистрации вопросов пользователей.

К тому же такой механизм регистрации обращений позволяет анализировать текущую ситуацию и избежать ошибок на следующих этапах.

Ответственность пользователей за обучение

Часто бывает так, что пользователи рассматривают обучение новой программе не как работу, а как ничем не обремененный отдых. Это приводит к очень низкой усвояемости материала.

Чтобы повысить результативность обучения проводятся следующие действия:

- после каждого занятия подписывается у пользователей отзыв о прохождении обучения;
- по результатам обучения проводится тестирование пользователей.
  Задача руководителя проекта со стороны заказчика на этом этапе – обеспечить учет результатов тестирования в схеме мотивации сотрудников.

Ответственность пользователей за результаты опытной эксплуатации

Одним из важнейших факторов успешной реализации проекта является этап опытной эксплуатации системы. На этом этапе пользователи должны ввести в систему тестовые или реальные данные и получить прогнозируемый результат.

Часто бывают, что пользователи ограничиваются небольшим количеством простых операций, на основании чего делают вывод о работе системы. Но проблемы в работе системы проявляются как раз на сложных операциях и их взаимосвязях. Поэтому часто такое отношение к опытной эксплуатации приводит к тому, что на этапе начала эксплуатации системы в рабочем режиме возникает огромное количество вопросов, которые можно было бы закрыть при опытной эксплуатации. В такой ситуации возникает вопрос о переносе сроков начала работы пользователей с системой, что влечет за собой срыв сроков всего проекта.

Для предотвращения возникновения подобных проблем желательно предпринять следующие меры:

- руководителю проекта со стороны Заказчика и всему управляющему комитету детально разъясняется роль опытной эксплуатации в успехе проекта;
- разрабатывается программа опытной эксплуатации, в которой эксперты со стороны исполнителя обязательно должны увидеть отражение самых сложных операций заказчика.

Практическое задание к работе №5.

1. Рассчитать экономическую эффективность от внедрения программного обеспечения в соответствующую предметную область.
2. Данные для расчёта берутся на основании анализа усреднённых показателей из открытых источников сети Интернет.

Лабораторная работа №6. Составление команды по внедрению

Теоретический материал.

Структура команды внедрения

Внедрение системы осуществляет команда, в которую входят координационный комитет, руководитель группы внедрения (руководитель проекта) и непосредственно группа внедрения (группа проекта). Все они должны согласовывать друг с другом свои действия, что позволит получить в итоге хорошо сбалансированный проект внедрения ERP-системы. При этом необходимо помнить, что проекты внедрения ERP-систем имеют свои особенности, что определяет структуру команды проекта. Эти особенности обусловлены прежде всего тем, что ERP-системы охватывают практически всю деятельность предприятия. Поэтому при их внедрении большое значение приобретают задачи коммуникаций и совместной работы представителей различных и часто конкурирующих подразделений предприятия. Ниже эти элементы рассматриваются подробнее.

Координационный комитет

Координационный комитет должен состоять из людей, жизненно заинтересованных в успехе проекта и имеющих значительное влияние на предприятии, но в силу объективных причин не располагающих достаточным временем для непосредственного участия в работе над ним. Как правило, это руководители или заместители руководителей основных операционных подразделений, затрагиваемых внедрением снабжение, сбыт, производство, планирование, финансы, бухгалтерия и т. п. Однако не стоит включать человека в координационный комитет только потому, что он является руководителем, или из политических соображений. В лучшем случае он просто не будет работать, в худшем — станет тормозом проекта.

Роль координационного комитета заключается в том, чтобы обеспечивать поддержку проекта и быть внешней контролирующей инстанцией. Часто говорят, что три самых важных фактора, влияющих на успех проекта, — это *«поддержка высшего руководства, поддержка высшего руководства и еще раз поддержка высшего руководства»*.

Координационный комитет и его члены должны следовать четкой стратегии взаимодействия с остальными участниками проекта. В их задачи не входит решение конкретных рабочих вопросов, однако они должны постоянно проявлять внимание к проекту и оказывать всяческую поддержку группе внедрения. Они не должны перераспределять усилия группы внедрения, игнорировать высказываемые группой предложения и ставить под сомнение компетенцию руководителя группы. Несоблюдение этих требований вызовет потерю мотивации в рядах группы внедрения.

Координационный комитет должен собираться *по крайней мере* раз в месяц для обсуждения хода проекта, будущих этапов и результатов выполненных работ.

В случае внедрения системы на малом предприятии (менее 200 человек) создание координационного комитета не всегда целесообразно. В данной ситуации функции

координационного комитета может выполнять один человек — кто-либо из руководителей предприятия или даже руководитель группы внедрения. Однако необходимо учитывать связанные с таким решением риски.

По своей сути ERP-системы являются интегрированными, и при их использовании ранее четко определенные функции отделов предприятия часто переплетаются. В силу этого при внедрении системы происходит перераспределение бизнес-функций между представителями различных подразделений. В случае конкурирующих отделов* такое перераспределение ролей часто вызывает конфликтные ситуации, разрешить которые могут либо руководители этих подразделений, либо человек, одинаково хорошо понимающий их функции и имеющий значительный вес в организации. Если руководитель проекта или один из руководителей предприятия, возглавляющие проект, не обладают полной картиной функционирования компании или, хуже того, оказывают предпочтение какому-либо подразделению, это может привести к некорректному описанию бизнес-процессов в ERP-системе, а в дальнейшем и к малоэффективному ее использованию. Например, если предприятие ставит целью внедрения ERP-системы повышение эффективности работы подразделений сбыта/маркетинга и уровня обслуживания клиентов, назначение на роль руководителя проекта директора по сбыту/маркетингу не будет правильным решением. В данном случае, с большой долей вероятности, не будет уделено значительного внимания функциям планирования и производства, функциям, от которых также зависит уровень обслуживания клиента.

Руководитель группы внедрения

Руководитель группы внедрения — это основное лицо, обеспечивающее «физическое» продвижение проекта. Кроме того, он является основным посредником между предприятием и представителями поставщика системы или консалтинговой фирмы, если таковые привлекаются в проект. Он должен обеспечить четкое видение проекта у группы внедрения, контролировать ход работ и вести диалог с членами группы в областях их ответственности. Этот человек должен уметь предвидеть потребности, которые могут возникнуть на этапах внедрения, а также обеспечить обучение членов группы внедрения на отдельных специальных курсах в рамках проекта.

Руководитель проекта может являться представителем одного из подразделений предприятия, но он не должен оказывать с этой позиции давление на членов группы внедрения, представляющих другие подразделения. Это может оказать общее негативное воздействие на группу внедрения и установить систему жесткого командно-административного управления, что недопустимо.

Взаимодействие между группой внедрения и остальными подразделениями на предприятии должно происходить исключительно через руководителя группы и, следовательно, он должен владеть всей информацией по проекту. Руководитель группы должен координировать разрешение внутренних и внешних конфликтов. Его задача — направлять работы в русло, согласующееся с мнением координационного комитета и своими собственными представлениями о возможных улучшениях бизнеса предприятия.

Руководитель проекта должен 100% своего рабочего времени посвящать проекту. Совмещение этой деятельности с выполнением других обязанностей в значительной степени повышает риск неудачи проекта.

Современная философия управления промышленным предприятием заключается в том, что в качестве руководителя проекта должен выступать всесторонне развитый человек. И именно этот принцип рекомендуют ведущие западные промышленные консультанты [2].

При первых попытках внедрения ERP-систем на Западе, а сейчас и в России, руководителем проекта часто назначался руководитель ИТ-отдела (отдела АСУ), так как интегрированные системы управления расценивались только как программное обеспечение. Такой подход вызывает определенные проблемы. Например, руководитель какого-либо подразделения всегда может сказать: «Она (система) не работает, она не подходит для наших методов ведения бизнеса. Возьми эту компьютерную программу и настрой так, чтобы она работала нормально». Такой подход ведет к многочисленным задержкам внедрения, полной переделке интегрированных систем управления и потере чувства собственности у пользователей.

Кроме того, отдел АСУ (так же, как и его руководитель) ориентирован на клиентов, которыми для него являются конечные пользователи других подразделений предприятия. Это, в свою очередь, ставит отдел АСУ в двусмысленную ситуацию: задача отдела АСУ — облегчить жизнь конечным пользователям, что не всегда обязательно при внедрении ERP-систем. Наилучшая роль отдела АСУ в процессе внедрения — осуществление технической поддержки, но никак не руководство внедрением.

Необходимо отметить, что ERP-система — прежде всего *бизнес-система*, а не компьютерная программа. Поэтому выбирать и внедрять ее должны люди, прежде всего хорошо ориентирующиеся в бизнесе предприятия. В силу этого руководитель проекта должен обладать всесторонними знаниями о методах ведения бизнеса, способностями обучать людей и преподносить им целостное видение функционирования предприятия.

Чтобы выбрать руководителя проекта правильно, вообразите следующий разговор между генеральным директором предприятия и консультантом по внедрению.

ГД: Мы не можем позволить себе освободить одну из ключевых и наиболее способных фигур в управлении предприятием и на 100% сделать его руководителем проекта. У нас нет для него замены. Мы должны взять на работу руководителя проекта со стороны.

К: В самом деле? Предположим, что один из ваших ключевых руководителей завтра (не дай Бог) попадет под трамвай. Вы говорите, что ваша компания перестанет работать из-за этого?

ГД: Я думаю, нет.

К: Что бы вы сделали в этом случае?

ГД: Как я уже сказал, у нас нет для него замены. Нам пришлось бы нанять другого человека со стороны.

К: Отлично. Предположите, что именно эта беда случилась с одной из ваших ключевых фигур. То есть сделайте его руководителем проекта со 100-процентной загрузкой. И затем, если это абсолютно необходимо, наймите нового руководителя ему на смену [4].

Вывод. Если на вашей организации никак не отразилось то, что один из ее руководителей стал руководителем проекта и перестал выполнять свои прямые обязанности, значит, ваш руководитель проекта, по-видимому, не тот человек. И наоборот, если вы выбрали человека, которого вы можете позволить себе освободить в наименьшей мере — ваш выбор верный.

То есть выбор руководителя проекта может рассматриваться как первый тест, который показывает, насколько руководство заинтересовано в успехе внедрения.

Необходимо отметить, что руководитель группы внедрения должен одновременно обладать как способностями видеть общую картину функционирования предприятия, так и качествами лидера. Как уже говорилось, он преподносит общее видение хода внедрения каждому члену группы и осуществляет контроль над ходом выполнения этапов проекта. Особенно это актуально для малых предприятий, где руководитель проекта может одновременно выполнять и функции координационного комитета. В больших компаниях или при одновременном или последовательном внедрении на группе предприятий (например, холдинг) внимание руководителя группы внедрения больше смещается в сторону управления ходом работ и устранения препятствий на пути развития проекта. Он поддерживает связи с внешними объединениями, задействованными в проекте, занимается подготовкой подразделений предприятия к возможным изменениям, которые могут возникнуть в будущем, разрешает конфликтные ситуации внутри группы внедрения. В этом случае, однако, для каждого из предприятий холдинга должна быть создана своя группа внедрения вместе со своим лидером, отвечающим за локальное внедрение.

Группа внедрения

На группе внедрения лежит основная нагрузка по выполнению работ, связанных с внедрением ERP-системы на всем предприятии. При формировании группы внедрения необходимо обеспечить вхождение в ее состав представителей всех служб предприятия, затрагиваемых внедрением.

Оптимальное количество членов в группе внедрения для среднего предприятия — это шесть-восемь человек. Если группа внедрения много меньше, то будет достаточно сложно внедрить систему на всем предприятии сразу. Кроме того, это может сказаться на качестве принятых решений. Если же группа слишком большая, то вклад членов группы, способность принятия решения и эффективность выполнения задач внедрения могут значительно упасть.

Нецелесообразно включать в группу внедрения свыше десяти человек, так как в этом случае группа становится трудноуправляемой. При необходимости увеличения количества человеческих ресурсов, задействованных во внедрении**, предпочтительнее создавать несколько отдельных групп - по одной на каждое предприятие или сферу бизнеса. Но руководство ими осуществлять из одного центра (одним человеком). Примеры внедрений ERP-систем показывают, что такая организация ресурсов часто способствует развитию

соревновательного духа, что при умелом управлении положительно сказывается на развитии проекта.

Правило выбора членов группы внедрения заключается в следующем: «Из каждого подразделения по одному представителю, которого организация может позволить выделить под проект и которого вы хотите видеть в группе внедрения». Это очень точное выражение, отражающее требования к навыкам каждого члена группы. Члены группы внедрения должны видеть всю ситуацию целиком, для того чтобы перевести бизнес-процессы предприятия на качественно более высокий уровень. Они должны уметь мыслить шире, нежели в рамках их профессиональной деятельности. Каждый член группы внедрения должен взять на себя полную ответственность за понимание своей функциональной области и областей, с которыми им приходится сталкиваться в процессе непосредственной работы на предприятии.

Члены группы внедрения должны нести ответственность за качество и сроки выполнения возложенных на них задач. Поэтому при планировании проекта необходимо оценить, сколько времени и ресурсов понадобится членам группы внедрения для того, чтобы выполнять работу в рамках проекта, а также определить сферы их ответственности. Это может потребовать передачи им дополнительных полномочий, выделения ресурсов, временного изменения их рабочих позиций.

Основное требование к членам группы внедрения: те, кто будет систему использовать, должны ее и внедрять.

Возможные проблемы и конфликты, а также способы их предотвращения

Даже идеальный подбор персонала не избавляет от появления проблем и возникновения конфликтов в команде внедрения. Эти проблемы лежат прежде всего в области снижения мотивации и личной заинтересованности и приводят к потере эффективности и качества работ.

Высшее руководство предприятия должно декларировать высочайшую приоритетность проекта и заявить о полной поддержке команды проекта. Цели и задачи проекта в контексте решения критически важных для предприятия вопросов должны быть поставлены перед группой внедрения достаточно четко.

Обсуждение текущих дел, анализ состояния проекта и обмен мнениями между представителями группы внедрения и координационного комитета должны проводиться на постоянной основе. Спорные вопросы и способы решения конфликтных ситуаций должны обсуждаться полным составом группы внедрения, при необходимости — на координационном совете. Каждый член группы представляет свое подразделение и, следовательно, имеет собственное видение проблем и методов их решения. Единый порядок и нормы взаимодействия должны вырабатываться с учетом этих различий в подходах.

Другой потенциальный источник конфликтных ситуаций — это приоритеты управления и оценка производительности. Объем нагрузки членов группы внедрения должен быть правильно оценен, и для выполняемых ими работ необходимо расставить приоритеты. Руководство и координационный комитет должны осознавать, что без выделения достаточных человеческих ресурсов, проведения полноценного обучения и организации стимулирования невозможно ожидать от членов группы внедрения адекватной отдачи.

Внедрение системы само по себе является чрезвычайно интересным и захватывающим процессом, так как это привносит новые позитивные изменения в работу подразделений. Однако нельзя сбрасывать со счетов тот факт, что процесс внедрения является достаточно длительным и сопровождается повышенной эмоциональностью и стрессовыми ситуациями. На каждом предприятии есть энтузиасты, но вряд ли найдется много людей, которые будут на протяжении года-двух постоянно перерабатывать.

Как правило, члены группы внедрения выполняют параллельно свои непосредственные обязанности на предприятии и обязанности члена группы внедрения в рамках проекта. В этой ситуации необходимо: 1) не поощрять в финансовом плане членов группы внедрения за совмещение двух работ; 2) адекватно оценивать их вклад в общее дело.

Быть членом группы внедрения очень почетно, но отсутствие дополнительной компенсации, как правило, приводит к потере мотивации. Исторически сложилось, что оценка действий сотрудника в основном зависит от того, насколько успешно он работает на своем непосредственном рабочем месте. Если же будет построена система учета вклада члена группы внедрения в проект, то это гарантирует, что он будет лучше осознавать собственную значимость, а также предоставит ему дополнительные стимулы к повышению эффективности выполняемых работ.

Вместе с тем моральное и материальное поощрение группы внедрения может стать источником других конфликтных ситуаций, как в пределах группы, так и при взаимодействии с другими подразделениями предприятия. Существует множество систем поощрений, которые можно использовать в целях повышения мотивации. Любая из них должна быть тщательно взвешена с позиции ее влияния на работу группы.

---

* Под конкурирующими отделами подразумеваются отделы, которые стремятся выполнить основную цель предприятия, а именно: увеличение прибыли за счет уменьшения издержек и увеличения объема продаж, в ущерб другим целям. Например, минимизация издержек для производства достигается за счет постоянной загрузки производственных мощностей и создания постоянных запасов сырья. Это противоречит целям отдела продаж, для которых увеличение прибыли достигается через увеличение запасов готовой продукции и принятия любых заказов клиентов, без оглядки на производственные возможности.

** Такая необходимость может возникнуть в случае внедрения системы на предприятии, имеющем принципиально различные методы организации производства, сбыта, снабжения. Кроме того, увеличение персонала, занятого во внедрении, может понадобиться, если предприятие состоит из нескольких территориально удаленных друг от друга производств или участков.

В функции координационного комитета входит:

- определение целей и путей развития проекта;
- сопоставление этапов развития проекта с запланированными сроками;
- утверждение необходимости в дополнительном выделении ресурсов (внутренних или внешних) под проект;

- использование своего влияния в целях устранения различных препятствий на пути развития проекта, созданных интересами отдельных подразделений, практикой деловых отношений, внутриполитическими отношениями предприятия

Основные требования к руководителю проекта

Руководитель проекта должен:

- 100% времени уделять руководству проектом;
- быть сотрудником компании, а не нанятым специально под проект;
- являться представителем операционного подразделения, глубоко вовлеченным в повседневную деятельность компании (обслуживание клиентов, планирование, управление производством и т. д.);
- быть самым лучшим из всех возможных кандидатов;
- являться сотрудником, работающим на предприятии длительное время;
- быть хорошим руководителем и уважаемым человеком.

Обязанности руководителя группы внедрения

- Ведение проекта
- Создание сплоченной, работоспособной группы внедрения
- Устранение препятствий на пути к повышению эффективности функционирования группы
- Создание и корректировка графика внедрения
- Составление и дальнейшая корректировка бюджета проекта
- Представление корректировок бюджета и графика внедрения перед координационным комитетом и группой внедрения
- Помощь в идентификации требований и проблем процесса внедрения

Члены группы внедрения должны:

- решать, что наиболее приемлемо для данного предприятия, и предлагать соответствующие корректировки бизнес-процессов предприятия;
- понимать слабые и сильные стороны их подразделений с точки зрения персонала и процессов при разработке процедур в своей функциональной области;
- иметь полномочия для изменения текущих процедур, установки или изменения сферы ответственности подразделений, а также для создания или ликвидации структурных единиц или должностей в организации, связанных с рекомендуемыми операционными изменениями.

Обязанности членов группы внедрения

- Брать полную ответственность за свою функциональную область.
- Посещать все собрания, связанные с внедрением, и принимать в них активное участие.

- Понимать цели и способы их достижения в рамках своей функциональной области.
- Разрабатывать детальные процедуры на основе знаний, полученных в процессе обучения и на этапе тестирования.
- Определять программные продукты, которые не поставляются вместе с системой, но применение которых необходимо в их функциональной области, а также все возможные изменения, которые могут понадобиться вносить в способы управления предприятием или в систему (что нежелательно).
- Участвовать в переносе данных в систему, а также быть ответственным за их достоверность в рамках своей функциональной области.
- После окончания внедрения составить обзорный документ по своим функциям с целью выявления открытых вопросов.

Практическое задание к работе №6.

1. Определить тактику внедрения, исходя из рисков, выявленных в предыдущей работе. Ответ сформулировать в произвольной форме
2. Определите степень (глубину) внедрения программного продукта в вашей предметной области. Ответ записать в развернутом виде: что меняется в конкретном наборе функций каждого пользователя внедряемого программного продукта, как это затрагивает существующее рабочее место пользваотеля (требуется ли переустановка ОС и т.д.), требуется ли новое оборудование в организации и если да, то какое, и т.п.

Для выполнения задания используйте ранее выбранные предметные области.

Лабораторная работа №7. Составление матрицы ответственности

Теоретический материал.

О таком умном словосочетании, как «разделение полномочий» говорят часто. Но все ли знают, как его применять на практике, и кому удается этим реально воспользоваться? Приглядевшись внимательно, делаем вывод, что такое явление происходит по большому счету, в компаниях частного сектора, в особенности тех, кто работает с иностранным клиентом.

Именно из-за границы до нас дошла любопытная аббревиатура под названием RACI. При этом, зачастую перед ней можно наблюдать разного рода умности а-ля «матрица» или «модель». Что это и с чем его едят, попытаюсь объяснить читателю далее. Возможно, кому-то уже повезло работать в коллективах, где каждый знает свои обязанности и область ответственности – за таких людей можно только порадоваться. При этом лично я верю, что далеко не у всех всё идеально в сфере разделения полномочий. Для таких людей данная статья может оказаться полезной.

Итак, что же представляет собой RACI матрица? Эта аббревиатура разбивается на четыре конкретных роли:

1. Responsible (на матрице отмечается буквой **R**) – ответственный непосредственно за выполнение работы
2. Accountable (**A**) – подотчетный, такую роль может занимать только один человек на одной задаче
3. Consulted (**C**) – один сотрудник или группа, с которыми проводятся консультации касательно задачи и мнения которых должно учитываться
4. Informed (**I**) – сотрудники, уведомляемые о выполнении конкретной задачи.

Также, существует два расширенных варианта модели. **RACI-VS**, здесь добавляются 2 роли:

- Verifies(**V**) – один сотрудник или группа, проверяющие соответствие результата выполнения задачи согласованным заранее допустимым критериям
- Signs off (**S**) – утверждает сдачу продукта заказчику (выполнения задачи). Данную роль можно совместить с подотчетной ролью.

При использовании варианта модели **RASCI**, появляется одна новая роль: Supportive (**S**) – предоставляет дополнительные ресурсы для выполнения работы, такой как поддержка внедрения продукта, к примеру.

С одной стороны – «много букв» и ничего не ясно. С другой – дабы стало понятней, хочу на примере показать, как выглядит сама RACI матрица.

	Director Service Management	Service Level Manager	Problem Manager	Security Manager	Procurement Manager
Activity 1	AR	C	I	I	C
Activity 2	A	R	C	C	C
Activity 3	I	A	R	I	C
Activity 4	I	A	R	I	
Activity 5	I	I	A	C	I

Не надо быть «Кэпом», чтобы понять, что шапка таблицы отображает список функциональных ролей, ответственных за ту или иную задачу или же участвующих непосредственно в принятии решения. Пункты «Activity 1-5» являют собой собственно функции, опускающиеся на плечи вышеуказанных ролей.

Для того, чтобы понимать, по какому принципу такая табличка должна рисоваться, а также как ее использовать на практике (в реальных проектах), рекомендуется уделить должное внимание следующему порядку действий при построении матрицы:

- Определяем список необходимых активностей/процессов в поставленной задаче (проекте)
- Определяем и указываем функциональные роли (людей, которые заинтересованы или которых тем или иным образом касается данная задача)
- Собираем митинг и назначаем RACI коды (собственно — буквы) конкретным ролям, непосредственно разграничиваем ответственности
- Определяем несоответствия (например, слишком много ответственных либо отсутствие таковых)
- Описываем таблицу и собираем отзывы
- Контролируем выполнение назначенных ролей

Допускаю, что кому-то будет нелегко ассоциировать латинские буквы с теми функциями, которые под ними подразумеваются. Тем не менее, не советовал бы использовать их русские

варианты – вполне вероятно, что возникнет путаница, станет еще сложнее разобраться, что к чему.

Ну и напоследок, после того, как мы закончили разработку RACI модели, не помешало бы и проанализировать результаты.

По функциональным ролям, анализировать можем, отвечая на такие вопросы:

- *Много «A»* — правильно ли распределены обязанности? Есть ли в наличии «узкие места»?
- *Много «R»* — не многовато ли мы ответственности повесили на одну роль?
- *Отсутствие пустых ячеек в таблице* – действительно ли эта роль должна быть вовлечена в такое количество задач?

Также, не забудьте провести анализ по выполняемым активностям:

- *Более одного «A»* — только одна роль должна быть подотчетной
- *Отсутствие «A»* — необходимо найти подотчетного
- *Более одного «R» или отсутствие такового* – кто-то должен быть ответственный, однако нам не нужно, чтобы ответственность была широко распределена – есть риск того, что задача не будет выполнена
- *Много «C»* — стоит ли нам консультироваться с многими ролями и будет ли это эффективно?
- *Отсутствие «C» и «I»* — правильно ли у нас установлены коммуникации?

На этом и хотелось бы остановиться, дабы не навевать читателю скуку. Насколько адекватным является такое распределения функций, ровно, как и пользоваться ли такой моделью – судить Вам. Заработает ли оно в постсоветских реалиях – вопрос спорный. Однако, работая с иностранным заказчиком, RACI матрица станет не только не лишней, но и даст четкое понимание всем о происходящих в проекте активностях и людях, выполняющих их. А заказчик, как известно, в своем большинстве, хочет четко знать, кто и за что несет ответственность в поставленной им задаче.
Также хотелось бы упомянуть о том, что RACI матрица ни в коем случае не является инструментом для определения козла отпущения, хотя бы потому, что люди, согласившиеся потратить время на ее составление, априори должны понимать ее цель и назначение. Допустим, у нас есть авиакомпания, которая на своем сайте собирается внедрить систему online check-in. Глобальные активности, необходимы к выполнению в контексте задачи, будут

приблизительно следующие:сбор требований к системе; дизайн решения; непосредственная разработка решения (development); внедрение; собственно – стадия “production”; оптимизация решения.
Далее — определяем список функциональных ролей, в данной задачи возможны такие варианты:внутренний сервис провайдер (IT отдел авиакомпании) или же внешний сервис провайдер в случае отсутствия первого; ISP – компания предоставляющая хостинг для сайта авиакомпании; бизнес подразделение авиакомпании (представляющее интересы заказчика); финансовое подразделение (бухгалтерия); сервис менеджер (в зависимости от размеров организации, может входить во внутренний IT отдел); команда разработчиков (в зависимости от размеров организации, может входить во внутренний IT отдел).

Попробуем расставить RACI коды соответственно ролям и выполняемым ими активностям (ясно, что данный процесс проходит при участии всех сторон).

	IT Отдел	ISP	Бизнес подразделение	Бухгалтерия	Сервис менеджер	Разработчики
Требования	R	I	C	C	A	I
Дизайн	C	C	I	I	AR	C
Разработка	C	I	I	I	C	AR
Внедрение	AR	R	C	I	C	C
Production	AR	R	I	I	I	I
Оптимизация	R	C	C	C	A	R

Таким образом будут распределены роли и функции в данной задаче.

Практическое задание к работе.

1. Определить тактику внедрения, исходя из рисков, выявленных в предыдущей работе. Ответ сформулировать в произвольной форме
2. Определите степень (глубину) внедрения программного продукта в вашей предметной области. Ответ записать в развернутом виде: что меняется в конкретном наборе функций каждого пользователя внедряемого программного продукта, как это затрагивает существующее рабочее место пользваотеля (требуется ли переустановка ОС и т.д.), требуется ли новое оборудование в организации и если да, то какое, и т.п.

Лабораторная работа №8. Разработка сценария внедрения программного продукта для рабочего места

Теоретический материал.

Сценарии внедрения - это сводка задач внедрения продукта.
Информация по установке продукта предполагает следующие сценарии внедрения:

- Сценарий 1: Внедрение с автоматическим конфигурированием промежуточного ПО
- Сценарий 2: Автоматическое внедрение с использованием существующего промежуточного ПО
- Сценарий 3: Внедрение вручную с использованием существующего промежуточного ПО
- Сценарий 4: Автоматическое внедрение в кластеризованной среде.

В сценариях 2 и 3 существующее установленное промежуточное ПО переиспользуется в качестве компонентов Control Desk. Например, у вас может быть существующий экземпляр базы данных. Этот экземпляр может располагаться на существующем сервере баз данных. Установленные политики доступа, принятые меры по избыточности и планы резервного копирования могут влиять на порядок внедрения программного обеспечения в вашей организации.

Если вы планируете переиспользовать существующее промежуточное ПО, убедитесь, что оно соответствует уровню, который поддерживается Control Desk. Программа установки не обеспечивает механизм обновления серверов с неподдерживаемыми версиями промежуточного ПО. Программа установки не производит дистанционную проверку предварительных требований, которые позволили бы гарантировать их необходимый уровень. Следует использовать поставляемый с продуктом инструмент проверки предварительных требований.

Сценарий 1: Внедрение с автоматическим конфигурированием промежуточного ПО

При таком сценарии выполняется внедрение продукта в новой среде. Используются программы установки Control Desk и инструменты для установки и автоматического конфигурирования новой установки промежуточного ПО и продукта.

Oracle WebLogic Server нужно по-прежнему конфигурировать вручную.

Например, можно использовать программу установки Control Desk для установки DB2 и использовать программу конфигурирования Control Desk для автоматического конфигурирования.

Такой сценарий подходит для настройки демонстрационной среды.

Сценарий 2: Автоматическое внедрение с использованием существующего промежуточного ПО

При таком сценарии выполняется внедрение продукта с помощью промежуточного ПО, существующего на предприятии. Используются программы установки продукта и инструменты для автоматического конфигурирования промежуточного ПО. Такой сценарий подходит для ситуаций, когда в организации уже имеется промежуточное ПО.

Oracle WebLogic Server необходимо конфигурировать вручную, но можно использовать программу установки Control Desk для автоматического конфигурирования, например, существующей базы данных.

Этот сценарий предназначен только для опытных администраторов базы данных и серверов приложений.

Сценарий 3: Внедрение вручную с использованием существующего промежуточного ПО

При таком сценарии выполняется внедрение Control Desk с помощью промежуточного ПО, существующего на предприятии; промежуточное ПО конфигурируется вручную. Такой сценарий подходит для ситуаций, когда уже имеется промежуточное ПО. В конкретной компании могут быть установлены особые правила, ограничивающие использование автоматических инструментов конфигурирования при внедрении новых прикладных программ. Данный сценарий содержит всю информацию по конфигурированию промежуточного ПО вручную.

Этот сценарий предназначен только для опытных администраторов базы данных и серверов приложений.

Сценарий 4: Автоматическое внедрение в кластеризованной среде

В этом сценарии вы внедряете Control Desk в сконфигурированной вами кластеризованной среде. Используются программы установки продукта и инструменты для автоматического конфигурирования промежуточного ПО.

Этот сценарий предназначен для знающих администраторов серверов приложений, имеющих опыт по планированию и созданию кластерных сред в IBM® WebSphere Application Server.

Часто на этапе приобретения системы Заказчик видит полезные и актуальные функции системы, но не совсем понимает, как использовать их совокупность, чтобы получить конкретный результат. В то же время у информационных систем есть типовой сценарий работы и последовательность ввода данных для получения результата.

По результатам обследования деятельности Заказчика мы понимаем, насколько формализованы и реальны бизнес-процессы у Заказчика и насколько типовой функционал нашей системы подходит Заказчику.

На данном этапе мы можем отнести проект к одной из 4-х приведенных ниже групп и принять решение о дальнейшем подходе к внедрению:

Функционал / Бизнес-процессы	Функционал системы подходит	Функционала системы недостаточно
Бизнес-процессы Заказчика формализованы	Адаптация бизнес-процессов под функционал системы → СТАНДАРТНЫЙ ПРОЕКТ	Бизнес-процессы требуют доработки системы → ЗАКАЗНАЯ СИСТЕМА
Бизнес-процессы Заказчика не формализованы	Предлагается стандартная модель бизнес-процессов → СТАНДАРТНЫЙ ПРОЕКТ	Описание бизнес-процессов → ЗАКАЗНАЯ СИСТЕМА

Если типовой функционал нашей системы в большей степени устраивает Заказчика и интегрируется в существующие бизнес-процессы, то мы предлагаем в первую очередь запустить его. В дальнейшем систему можно развивать, дорабатывать функционал, автоматизировать новые функциональные области, но начинать мы рекомендуем с запуска «ядра» системы — основного функционала для автоматизации проектного контура со стандартным сценарием использования типовой конфигурации. Такой проект мы называем Стандартным.

Его целью является достижение желаемого эффекта за счет корректной интеграции сценария работы системы и бизнес-процессов Заказчика. То есть в процессе обследования и интервьюирования Заказчика мы делаем микс: в основном формируем процессы как есть и тут же описываем и предлагаем, как их нужно видоизменить с точки зрения внедрения типового функционала системы.

«Стандартный проект» представляет собой ядро, необходимое для всей дальнейшей комплексной системы, и может использоваться как в качестве самостоятельного продукта, так и являться первой, стартовой очередью внедрения большого комплексного проекта автоматизации.

Если функционал не до конца подходит Заказчику, и мы видим необходимость в существенной доработке системы или отклонение от стандартного сценария ее использования, то мы предлагаем Заказчику индивидуальную разработку под его бизнес-процессы.

В случае если у Заказчика бизнес-процессы не формализованы, то до внедрения системы мы описываем бизнес-процессы. Для этого мы используем программное обеспечение

Business Studio. Чаще всего мы описываем бизнес-процессы верхнего уровня, декомпозируя их на сценарии работы пользователей.

Если есть необходимость, то на этапе описания бизнес-процессов мы предлагаем видоизменить и оптимизировать процессы с точки зрения нашей системы, которая включает в себя накопленный отраслевой опыт работы и лучшие практики, которые мы как раз и внедряем Заказчику. Нередки случаи, когда предлагаемые нами сценарии формируются в виде бизнес-процессов.

Сценарии работы пользователей — это своего рода инструкции, описывающие, как Заказчик будет использовать нашу систему в рамках автоматизируемых бизнес-процессов, кто будет задействован в том или ином процессе, какие объекты системы будут связаны с выполняемым процессом и так далее. В процессе создания сценариев мы также используем программное обеспечение Business Studio.

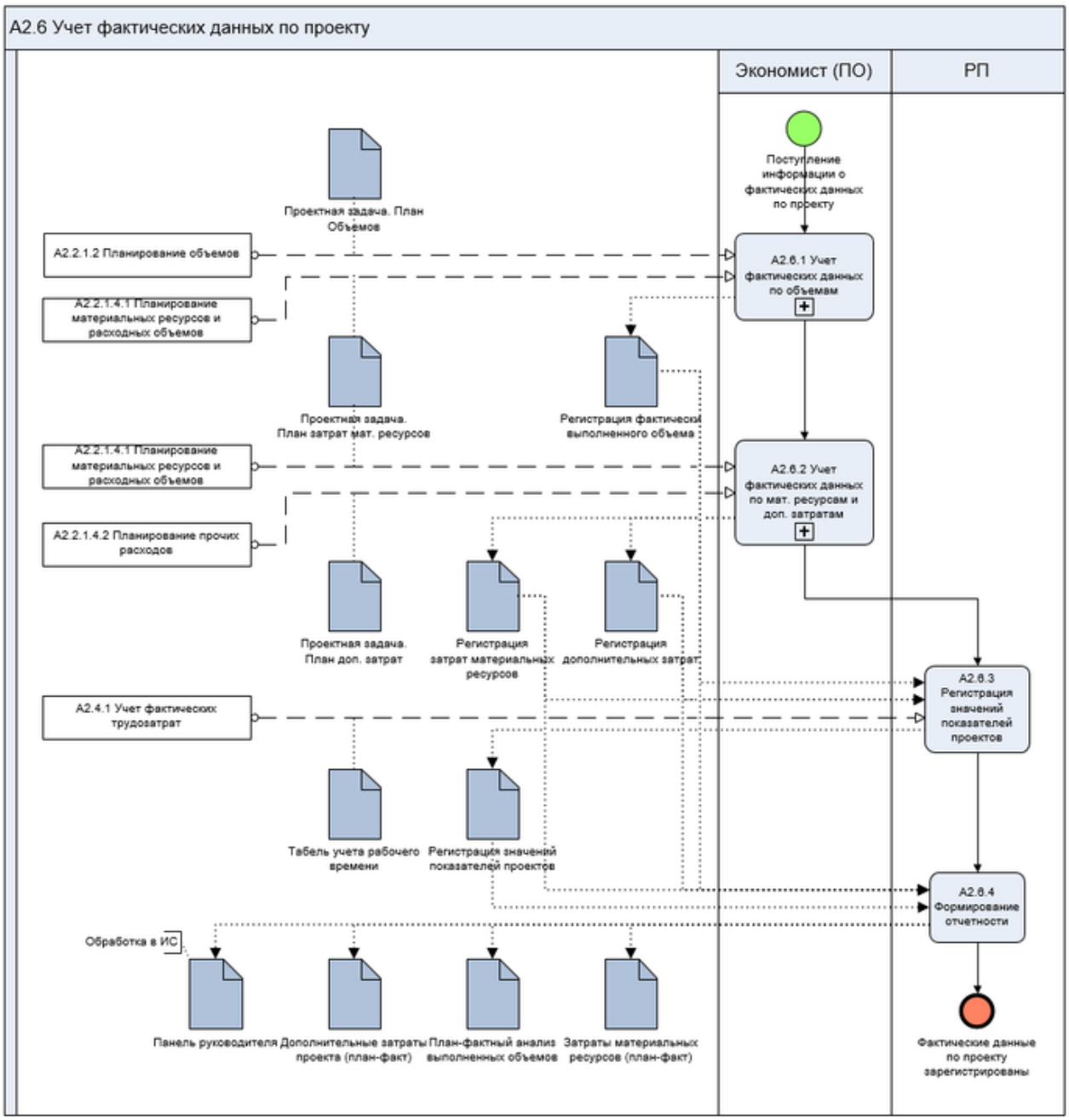


Рис. 1. Пример сценария работы пользователей

Сценарий представляет собой кликабельную HTML-модель сценариев и описывает функции нашей системы, а также последовательность шагов для их выполнения.

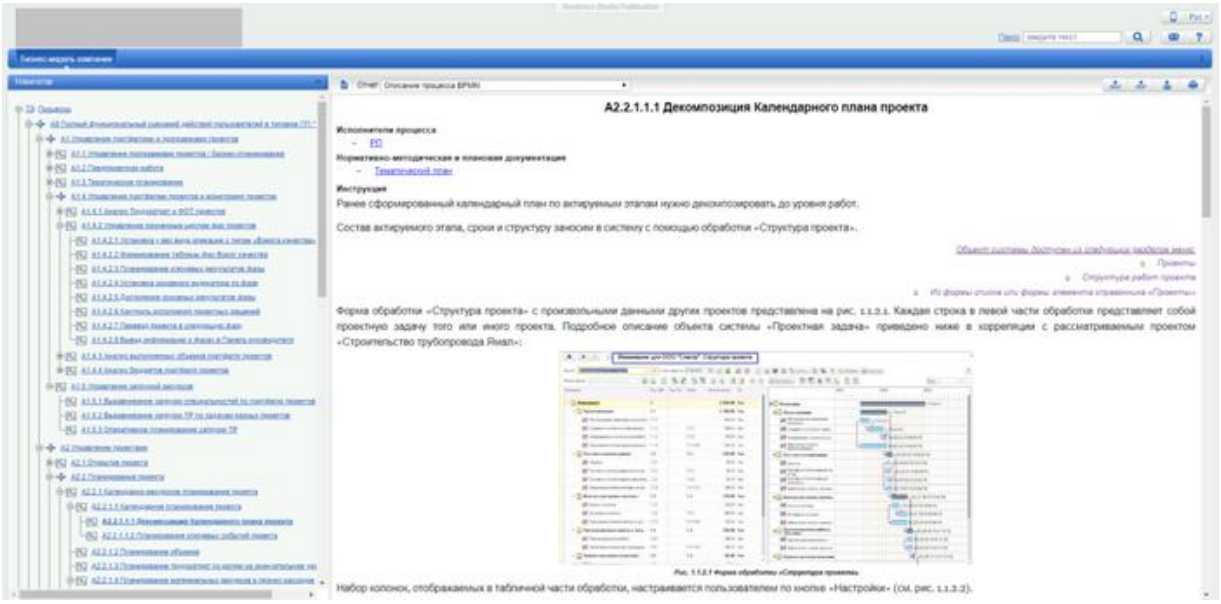


Рис. 2.HTML-модель сценариев

На основе сценария и матрицы ответственности процесса по ролям создается матрица для настройки прав доступа пользователей к объектам системы, которая также в дальнейшем трансформируется в матрицу прав доступа и ролевую модель информационной системы.

Перед запуском системы в опытную эксплуатацию Заказчику передается HTML-модель, содержащая сценарии использования, матрицу ответственности процесса по ролям (группам доступа) и инструкции пользователей.

Заказчик может использовать HTML-модель сценариев на каждом рабочем месте, что повышает удовлетворенность пользователей от использования нашей системы за счет наличия наглядной и доступной справочной информации.

В заказных проектах сценарии, сформированные в Business Studio, перекладываются сначала в логическую, а потом в физическую модель данных в системе проектирования прикладных решений «1С: СППР» для дальнейшего управления разработкой заказной системы.

Наибольший эффект система Business Studio приносит при внедрении «Стандартного проекта». С ее помощью мы разработали шаблон бизнес-процесса и сценариев действий пользователей в системе, который в рамках проекта по внедрению адаптируется под бизнес-процессы конкретного Заказчика.

За счет стандартизации сценариев использования, процессов и некоторых заготовленных инструментов (например, обработки по загрузке нормативно-справочной информации), мы достигаем оптимизации трудозатрат на моделировании работы пользователей в системе, а также на подготовке и оформлении документации на систему.

Все это позволило нам оптимизировать сроки и стоимость стандартного внедрения системы. Приведу пример: раньше, по итогам моделирования, оформление сценариев в виде отчетного

документа могло длиться 3 недели (120 часов работы системного аналитика), сейчас эта работа выполняется в пределах одной недели.

Так же мы разрабатываем расширенный стандартный проект, содержащий, в частности, подсистемы бюджетирования и казначейства. Таким образом, наша модель растет, её функциональное покрытие становится больше.

Одним из преимуществ, получаемых Заказчиком от внедрения «Стандартного проекта», является быстрый запуск системы в эксплуатацию. Как это работает: при реализации масштабного проекта автоматизации полгода требуется на проектирование системы, еще полгода на разработку, после этого начинается внедрение. Большинство Заказчиков не хотят ждать год, они хотят получить результаты от внедрения системы здесь и сейчас. За счет использования «Стандартных проектов» можно оптимизировать и стандартизировать первый шаг и уже через два-три месяца начать использовать систему, увидеть первые результаты ее работы.

Практическое задание.

1. Исходя из вашей предметной области, выберите тот сценарий внедрения программного обеспечения, который кажется вам наиболее подходящим.
2. Исходя из выбранного сценария, распишите его техническую часть: последовательность обследования технических параметров системы, мероприятия по адаптации информационной среды заказчика под устанавливаемое программное обеспечение, мероприятия по настройке установленного ПО.

Лабораторная работа №9. Анализ ошибок и технических сложностей при внедрении

Теоретический материал.

Анализ типичных ошибок в процессе внедрения программного обеспечения на предприятии или в организации

В процессе внедрения ИСУП на предприятия были выявлены ошибки. Некоторые из них не были устранены на ранней стадии и в какой-то момент сказывались на эффективности работы с системой. Указанные в первой главе категории типовых ошибок при внедрении можно применить для данного примера российской строительной компании.

Проблема сопротивляемости сотрудников нововведениям

Сопротивляемость сотрудников внедрению новой ИСУП практически отсутствовала. Это обуславливается тем, что компанией было проведено совещание непосредственно с планировщиками и были указаны проблемы, которые система была призвана решить.

Однако возникла проблема с тем, что расширение уже существующей системы и уже существующих бизнес-процессов повлекло за собой загруженность планировщиков из-за необходимости работать в новой системе, параллельно эксплуатируя текущую. Проблема так же не решилась путем освобождения планировщика от других процессов для сосредоточения именно на разработки графика пилотного проекта в новой ИСУП, поскольку его задачи перешли к сотрудникам, объем работы которых увеличился.
Вследствие быстрой передачи квалификации и обязательств другим сотрудником это отразилось не только на производительности работы, но и на качестве разрабатываемых в уже существующей ИСУП графиках.

Текущая квалификация сотрудников

Некоторые проблемы, касающихся текущей квалификации сотрудников были учтены еще при выборе решения о внедрении конкретной информационной системы. Так было определено, что обязательным требованием является русскоязычный интерфейс программного обеспечения, изначальный или локализированный, поскольку не все обладали достаточными навыками для работы с англоязычной версией продукта.
Так же текущие сотрудники были обучены силами сторонней консалтинговой компании, специализирующейся на консалтинге в области управления проектами. Обучение группы сотрудников, которые будут непосредственными пользователями ПО, на базе которого построена ИСУП, было построено таким образом, чтобы покрыть максимальное количество теоретических и практических знаний по работе с линейными проектами посредством данного ПО за короткий двухдневный срок. Был проведен стандартный курс подготовки пользователей по работе с системой с рассмотрением блоком вопросов наиболее часто встречающихся на практике. Также была создана инструкция для пользователей, содержащая в себе основные темы по работе в ИСУП.
Однако проблемой стал временной разрыв. Хронологически, система начала тиражироваться только после успешной реализации двух пилотных проектов. Для их реализации потребовалось определенное время, за которое другие планировщики потеряли часть своей квалификации работы в новой системе из-за отсутствия отработки практических навыков. Для их восстановления пришлось заключить дополнительный договор с консалтинговой компанией на выделение сотрудника для уже личного, а не группового обучения, что сказалось на фактическом бюджете внедрения ИСУП.
Другая проблемой, которая не относится напрямую к проблеме внедрения, но является последствие ее непроработки – передача компетенций. В случае когда планировщик решает покинуть свое место работы он должен передать компетенции другому сотруднику, который в свою очередь уже передаст ее новому лицу по факту его прихода на открытую вакансию в компании. По факту временного периода передачи недостаточно и необходимо привлекать консалтинговую компанию для обучения сотрудника навыкам работы с ИСУП.

Проблема интеграции систем

Данная проблема была решена с самого начала, так как одним из главных критериев поиска была система, которая обладала возможностью двухстороннего обмена информацией с уже существующим набором ПО. Однако в планах компании имеется дальнейшее расширение ИСУП, как было выяснено в анкете и интервью, и возможно потребуется интеграция с будущими системами. Данный фактор требуется принимать во внимание при выборе ИСУП для избегания будущих ошибок.

Проблемы в разработанных документах проекта

В рамках проекта внедрения ИСУП в компанию были разработаны следующие документы:

5. Методика планирования, актуализации и контроля графиков проектов
6. Регламент планирования, актуализации и контроля графиков проектов с использованием программного обеспечения

Отдельно стоит отметить, что кроме методики и регламента были написаны регламенты интеграции ИСУП и ролевые пользовательские инструкции при работе с ИСУП, однако они освещают в основном технические вопросы и в данной работе не рассматриваются на предмет наличия в них проблем.

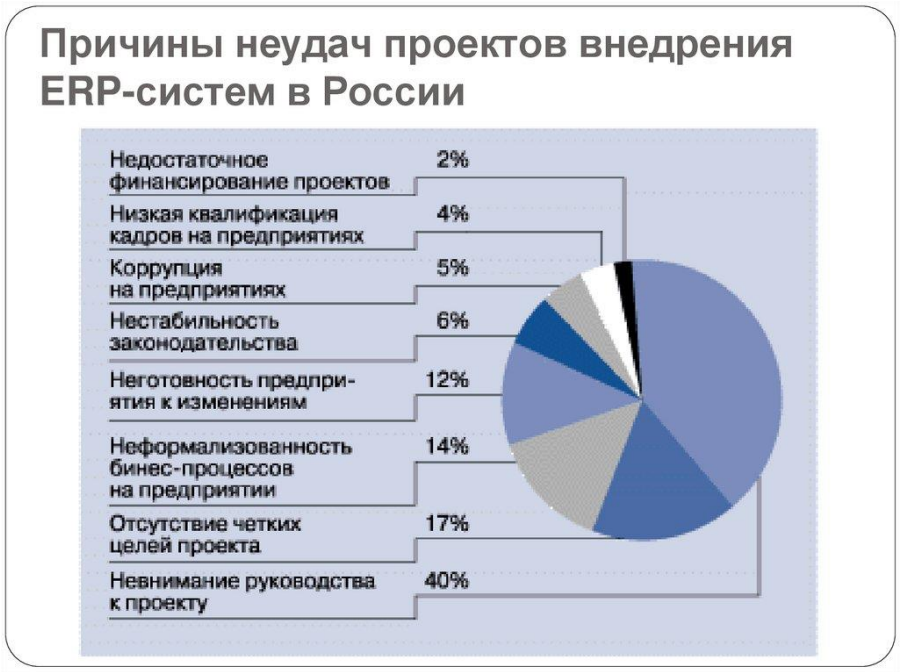
Ошибки подходов к изучению методики

Методика несет в себе методически рекомендации, которых следует придерживаться при разработке графика. Знакомство с методикой изначально проходила в рамках подготовки сотрудников работе с ИСУП, однако не все новоприбывшие сотрудники, не успевшие пройти обучение, знакомятся с методикой самостоятельно. При сравнении графиков на этапе согласования выявляются различия в подходах к формированию, что делает процесс сравнения затруднительным. Саму проблему можно обозначить как отношение к методике именно как к методическим рекомендациям, нежели как к обязательному документу и отсутствие людей и процессов, которые могли бы выявлять в графиках несоответствие методики, а так же проконтролировать процесс ее соблюдения.

Ошибки в регламенте

Регламент планирования, актуализации и контроля графиков проектов с использованием программного обеспечения и регламент интеграции описывает бизнес-процессы, в соответствии с которыми происходит процесс разработки графика, предоставления информации, его согласования и другое

При изучении регламента планирования, актуализации и контроля графиков проектов с использованием программного обеспечения также были выявлены определенные ошибки и недочеты, которые не позволяют устранить ошибки.



Технические проблемы

Здесь стоит говорить о нехватке системных ресурсов в автоматизируемой организации (или о несоответствии их требованиям программы):

- компьютеров,

- офисной сети,

- средств связи.

Нередко в распоряжении компании-заказчика просто нет тех технических средств, которые позволили ли бы программе работать корректно. Банальная нехватка места на сервере – уже причина того, что IT-решение просто не получится использовать.

Справедливости ради стоит сказать, что сложность решения проблем технического характера достаточно низкая. Решается она путём модернизации оборудования. Здесь же нужно отметить, что мешает внедрению компьютерной программы данная проблема сильно. Ведь если речь идёт о работе в условиях ограниченного бюджета или если обосновать необходимость покупки нового оборудования не удаётся (перед высшим руководством компании, например), то ситуация приобретает сложный характер. Пока согласовываются все моменты покупки нового оборудования, его замены и тому подобное, теряется время, которое является, пожалуй, самым ценным фактором применительно к понятию «автоматизация». В итоге компания не может приступить к процедуре налаживания работы на некоторых участках, созданию порядка в ведении бизнеса или учёте.

Как ни странно, но следующую сложность при реализации IT-проектов также стоит относить к категории технических проблем. Речь идёт о наличии и уровне подготовки IT-персонала компании-заказчика. Например, довольно часто системный администратор не знает специфики внедряемой программы и потому затрудняется её поддерживать. Вообще понятие «системный администратор» в последние 10-15 лет стали сильно упрощать. Сотрудник, который может сделать дефрагментацию диска, переустановить систему – уже гордо зовётся «системным администратором». На самом деле, роль этого специалиста в жизни компании, которая, к тому же, приняла решение об автоматизации деятельности, должна быть гораздо важнее. Это должен быть сильный специалист, с широким спектром знаний в области IT и способностью довольно быстро понимать специфику ПО. Применительно к отраслевым рынкам это очень важно.

Также нужно отметить, что сами пользователи и операторы программы бывают недостаточно подготовлены для работы с системными функциями (импортом данных, например). То есть они теряются при виде нового ПО. Конечно, путём обучения можно эту проблему решить, но здесь виной всему и нежелание учиться чему-то новому (об этом поговорим чуть позже). Впрочем, компьютерная грамотность сотрудников отечественных компаний порой является крайне низкой. Сложность решения проблемы, связанной с уровнем подготовки IT-персонала, можно охарактеризовать как среднюю. Решается она путём обучения системного администратора или другого IT-специалиста. Здесь, безусловно, руководству нужно идти на встречные шаги, выделять бюджет и время на обучение сотрудников, не пускать этот процесс на самотёк. Внедрению компьютерной программы упомянутая выше категория проблем мешает заметно (средне). В случае выделения бюджета компания-заказчик программного решения может обратиться за помощью к компании-разработчику. Как правило, солидные IT-компании, наряду с возможностью разработки специализированного ПО, обладают и достаточным опытом, знаниями, навыками и даже некими педагогическими способностями для обучения персонала.

Проблемы «Человеческого фактора»

Выше мы затронули проблему, связанную с IT-персоналом, но исключительно с точки зрения профессионализма и компьютерной грамотности. Лишь вскользь было упомянуто о нежелании сотрудников обучаться новому. Остановимся подробнее на этом. Итак, пожалуй, самые сложные проблемы, о которых стоит сказать, следующие:

- нежелание перемен со стороны сотрудников автоматизируемой компании,
- необходимость выполнять двойную работу сотрудниками автоматизируемой организации первое время,

- непонимание того, что впоследствии программа будет реально помогать в работе,
- страх незаменимых сотрудников стать ненужными,

- невозможность обучения работе в программе, т.к. сотрудники компании могут быть людьми «старой закалки» (к примеру, пенсионного возраста).

Перемены пугают почти всегда. Хорошие ли они, плохие ли они – неважно. Новое, неизведанное, то, чего раньше никогда не было, обычно сотрудников вводит в какой-то ступор, хотя объективно поводов для беспокойства может и не быть. Как правило, распространённая точка зрения такова: «Мы и так много работали, хорошо себя чувствовали, получали заработную плату, а тут какая-то программа». Сюда же можно отнести и непонимание того, что программа далее будет реально помогать в работе, что она не бремя и не груз (во всяком случае, если это качественная отраслевая разработка), а, наоборот, помощник. В общем-то, здесь есть как мотивы просто несерьёзного отношения к делу или поверхностного понимания роли ПО в жизни компании, так и более серьёзные предпосылки для нежелания осваивать программу. Например, очень часто специализированное решение помогает выявить недобросовестного сотрудника, который работает так, как удобно ему, а не компании. Конечно, для него внедрение программы, которая призвана навести порядок, нежелательно.

Что касается выполнения двойной работы: здесь имеется в виду, что на первых порах внедрения программы сотрудникам необходимо организовывать свою работу по двум сценариям. А именно: по старому сценарию, который был принят в компании (например, продолжать заполнять электронные таблицы), и по новому, который заложен в программу.

Ещё один момент: состав пилотной группы, сотрудники которой вводят данные в программу, может меняться. Допустим, сотрудники уже ввели данные в программу, были готовы начать работать с этими данными дальше, но вот кто-то из них уволился, а это значит, что новый сотрудник должен вновь осваивать этот участок работы, учиться вводить данные. Не исключено, что сначала новый сотрудник не будет понимать, как работать с программой, тогда им нужно будет несколько раз поработать над определённой задачей, но всё это этапы так называемой притирки. Ничего страшного в них нет.

Незаменимые сотрудники тоже компьютерные программы не жалуют. Им довольно лихо удаётся записывать всё в ежедневник, только они знают того самого Ивана Иваныча, который им сможет поставить материалы или оборудование со скидкой, только им известен телефон той самой фирмы, которая вовремя выполнит заказ. Иными словами, они работают так, как привыкли, у них есть доля мнимой, на самом деле, власти. В программу же нужно будет внести и контакты Ивана Иваныча, и все записи из ежедневника и всё остальное. И, по большому счёту, работать нужно будет так, как предусмотрено программой. Ведь опытные разработчики закладывают в программы логику, методики, на которых и базируются бизнес-процессы компании.

Сотрудники же «старой закалки» - кадры, как правило, ценные. Часто на уровне начального понимания компьютера у них возникают сложности. Теоретически обучать работе в программе таких сотрудников возможно, но говорить о том, что это просто, не приходится. Конечно, это не значит, что сотрудника, который реально полезен для компании и посвятил ей много лет трудовой деятельности, нужно увольнять. Обычно программы, которые разработаны на основании потребностей пользователей, понимании специфики работы компании, имеют интуитивный интерфейс. Поэтому хотя бы минимальное понимание работы в программе возможно практически для любой категории пользователей.

Сложность решения проблем, связанных с человеческим фактором, можно отнести к категории средних. Решается данная группа проблем путём проведения разъяснительной работы со стороны руководства и разработчика. При этом наличие таких проблем внедрению программы мешает сильно. Поскольку общее настроение коллектива, халатность и понимание того, что руководство «ничего не сделает», очень расслабляет и на работу в программе, конечно, не настраивает.

Отдельно хотелось бы выделить проблему, которую можно классифицировать как саботаж. Выражается эта проблема в явном нежелании сотрудников работать в программе, пропаганда негативной точки зрения на автоматизацию. То есть, если выше названные проблемы как бы вялотекущие и при должном купировании могут решиться быстро, то проблема саботажа не так безобидна. Сложность решения этой проблемы очень высока. Как правило, требуется принятие ряда мер по мотивации сотрудников, принятию других административных решений, порой поиск нового персонала. В такой ситуации дело может дойти до полной кадровой перестановки в компании, поэтому принимать меры нужно заблаговременно, чтобы не допустить этого. Внедрению программы саботаж мешает сильнее, чем что бы то ни было.

Проблема «плохого» IT-решения

Компьютерные программы, так же, как и еда, одежда, обувь, техника бывают плохими – это правда. Здесь важно понимать, что универсального решения не бывает, решения должны быть отраслевыми. То есть, если отрасль строительная – значит программа должна быть ориентирована на нужды строительных компаний. Многие разработчики же хотят охватить рынок по максимуму, разработав единое программное решение, которое и строительной компании и медицинской, будет, что называется, впору. Так не бывает. Во всяком случае, если речь идёт действительно о средстве автоматизации, ERP-системе, а не о текстовом редакторе, например. Решается такая проблема просто – путём внедрения специализированного отраслевого решения, а не программы «для всех типов компаний». Впрочем, внедрению какой бы то ни было программы проблема данного свойства сильно не мешает. Со временем доработать программу под нужды организации возможно, если заказчик согласен ждать месяцы и даже годы и платить за это.

Заметим, что существуют программы, которые не имеют отраслевой специфики. К примеру, это программы для ведения бюджетирования. Разработчики таких программ открыто говорят о том, что у них есть некоторый «каркас» программы, который далее можно наполнить новым функционалом. Однако в этом случае клиент должен быть готов к тому, что его индивидуальное решение будет создаваться некоторое время, нужно будет совместно с разработчиком согласовывать техническое задание на него, продумывать все детали.

Стоит сказать, что отраслевая направленность программы тоже не даёт стопроцентной гарантии удачного внедрения. Это лишь один из факторов, который должен добросовестно учитывать разработчик. Плюс ко всему, разработчик должен быть готов к быстрому, динамическому развитию программы. Например, в условиях меняющегося рынка, кризисных ситуациях, да и вообще в целом это очень важно. Та же самая строительная отрасль меняется

настолько стремительно, что программа, которая три года не менялась вообще, не было выпущено никаких релизов, уже не может называться актуальным инструментом для ведения бизнеса. Сложность решения этой проблемы высока. Ведь медленное развитие программы делает её заведомо неактуальной в современной рыночной ситуации, ненужной. Скорее всего, просто методом естественного отбора такая программа из продажи исчезнет, став своего рода гужевой повозкой в мире автомобилей. Парадоксально, но внедрению данная проблема совершенно не мешает, потому что является отложенной и выявится лишь в дальнейшем. Но заказчик должен понимать, что если программу не собираются развивать, то смысл её приобретения и внедрения практически отсутствует.

Проблема взаимодействия компании-заказчика и разработчика

Применительно к отраслевым программным решениям, независимо от того, заказные ли это разработки или тиражные программы, заказчик и разработчик должны находиться в постоянном контакте и профессиональном диалоге. Часто IT-проект обречён на неудачу, потому что заказчик не понимает, какой результат должен получиться, не может поставить задачу разработчику.

Связано это с тем, что заказчик и разработчик говорят на разных языках. Также изменение рыночной ситуации может повлиять на актуальность IT-проекта. Сложность решения данной проблемы средняя. Со стороны заказчика нужно сделать следующее: назначить ответственных за внедрение сотрудников, освободив их от части их прямых обязанностей для лучшего освоения программы. Со стороны разработчика требуется привлечение специалистов, знающих специфику отрасли. Возможно, следует привлечь экспертов извне, проконсультироваться с представителями выбранной отрасли и только затем предпринимать шаги по созданию IT-проекта. Внедрению, конечно, названная проблема мешает в значительной степени.

Расплывчатая стоимость системы тоже является камнем преткновения между компанией-разработчиком IT-проекта и компанией-заказчиком. Дело в том, что немногие компании-разработчики могут обеспечить пакетность услуг, обозначить конкретную стоимость системы и услуг по её внедрению (в процессе внедрения могут быть выявлены различные нюансы в построении бизнес-процессов компании-заказчика). На отраслевом рынке B2B – это явление, по сути, стандартное. Ответить на вопрос: «Сколько это стоит?» не могут многие разработчики. Дело в том, что иногда даже в готовое решение заказчик хочет добавить новый функционал, который будет ориентирован на специфику его компании. Соответственно, если такая возможность у исполнителя есть, то это будет расцениваться как дополнительная работа. Сложность решения этой проблемы довольно низкая. Достаточно прийти к компромиссному решению между заказчиком и исполнителем (поговорить о скидках, рассрочках платежей и так далее). Внедрению же она мешает сильно, т.к. из-за минимальных недоговорённостей процесс внедрения может быть прерван на полпути.

Вот каким образом видятся нам проблемы при реализации IT-проектов и способы их решения. Важно то, что хороший разработчик должен стремиться к тому, чтобы клиент был понят. Не стоит реагировать на каждую просьбу клиента положительным ответом. Очень часто клиент только думает, что ему нужно «это», а ему, на самом деле, нужно «то»

Практическое задание к работе.

1. Определите, какие главные технические сложности могут возникнуть при внедрении программного обеспечения. Оформите их в виде таблицы: предполагаемая проблема,

уровень влияния на процесс внедрения, предполагаемый путь устранения проблемы.

2. Определите другие проблемы, которые возможны в процессе внедрения, степень их влияния и возможные способы устранения.

Лабораторная работа №11. Проверка приложений на совместимость в среде виртуальной машины

Теоретический материал.

VMware Workstation создает полностью изолированные безопасные *виртуальные машины*, инкапсулирующие операционные системы и приложения. Уровень виртуализации *VMware* сопоставляет ресурсы физического оборудования с ресурсами *виртуальной машины*. Таким образом, каждая *виртуальная машина* получает собственные ЦП, *память*, диски и устройства ввода-вывода и является полным эквивалентом стандартного компьютера *x86*. *VMware Workstation* устанавливается в операционной системе узла и предоставляет широкую поддержку оборудования за счет наследования поддерживаемого оборудования операционной системы узла.

Любое *приложение*, работающее на стандартном ПК, запустится и в *виртуальной машине VMware Workstation*. *VMware Workstation* **– это эквивалент полного ПК с возможностью работы в сети и с поддержкой различных устройств**. У каждой виртуальной машины есть свой центральный *процессор*, *память*, диски, устройства ввода-вывода и т. д. В ней запускают любое *приложение*, которое работает в поддерживаемых гостевых ОС, включая Microsoft Office, Adobe Photoshop, Apache *Web Server*, Microsoft *Visual Studio*, отладчики ядра, брандмауэры, *ПО* для виртуальных частных сетей и др.

VMware Workstation использует файловую систему компьютера и создает файлы, которые привязываются к дискам виртуальной машины, поэтому создавать *разделы* для каждой операционной системы не нужно. Если на компьютере установлена другая *операционная система* и настроена *загрузка* обеих ОС, то *VMware Workstation* может запустить вторую операционную систему на виртуальной машине в операционной системе узла. Вместо выбора загружаемой ОС можно запустить обе операционные системы одновременно и переключаться между ними щелчком мыши.

После установки *VMware Workstation* выполняется настройка виртуальных машин – назначение памяти и дисков, портов и сетевых устройств. Затем *виртуальная машина* включается с подключенным установочным компакт-диском или ISO-образом ОС. После загрузки виртуальной машины начинается стандартная процедура установки операционной системы.

С точки зрения операционной системы узла *VMware Workstation* является обычным приложением. При ее установке на виртуальной машине изменение гостевой ОС не требуется. Приложения в гостевой операционной системе продолжают работать так же, как на узловом компьютере.

Виртуальные диски **– это дисковые разделы виртуальных машин**. Они хранятся в виде файлов в файловой системе операционной системы узла. Одной из главных возможностей *VMware Workstation* является *инкапсуляция*. Это означает, что среда

полностью инкапсулируется в набор файлов, которые можно копировать, перемещать и использовать. Поскольку *целый раздел диска* сохраняется как *файл*, виртуальные диски можно копировать и перемещать. Процедура резервного копирования существенно упрощается. Поддерживается создание виртуальных дисков с интерфейсами *SCSI, IDE* и *SATA* размером до 8 Тбайт.

Виртуальная машина может использовать подключение через сетевой *мост*, чтобы получить собственный IP-*адрес* (от сервера *DHCP*, если он доступен), или трансляцию адресов (*NAT*) и работать с IP-адресом узла. Можно также настроить внутреннюю *сеть* между узлом и виртуальной машиной, чтобы создать изолированную виртуальную *сеть*. Кроме этого, можно полностью отключить сетевые подключения, создав совершенно изолированную виртуальную машину.

Установка VMWare Workstation

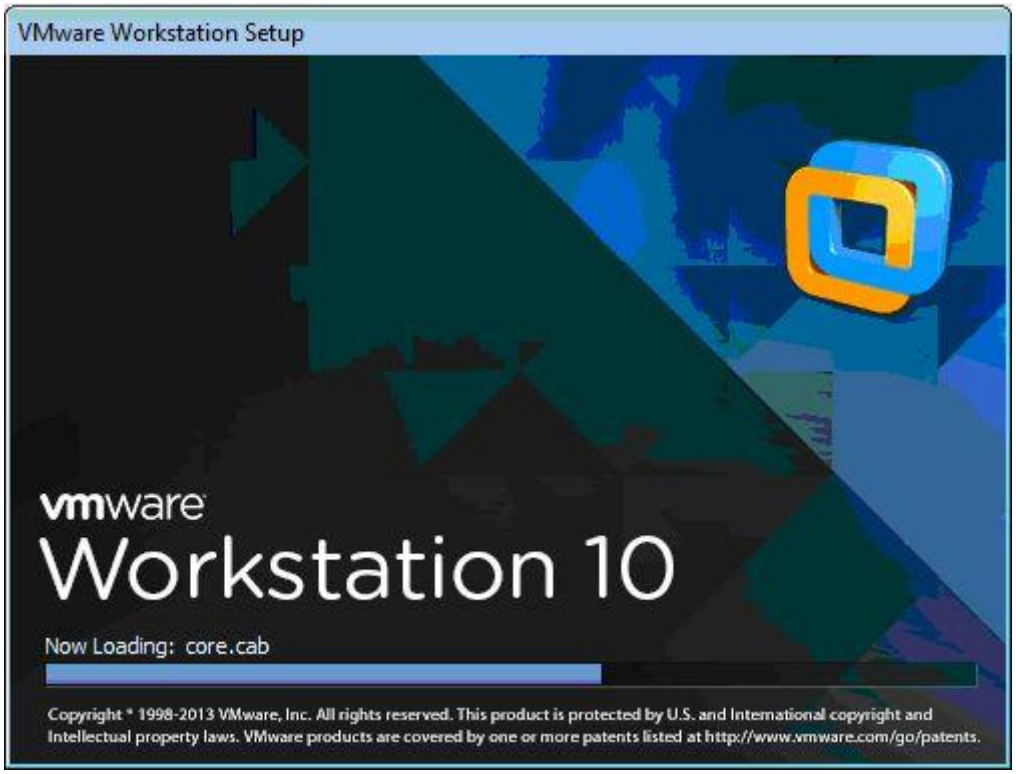


Рис. 1. Окно установщика VMware Workstation

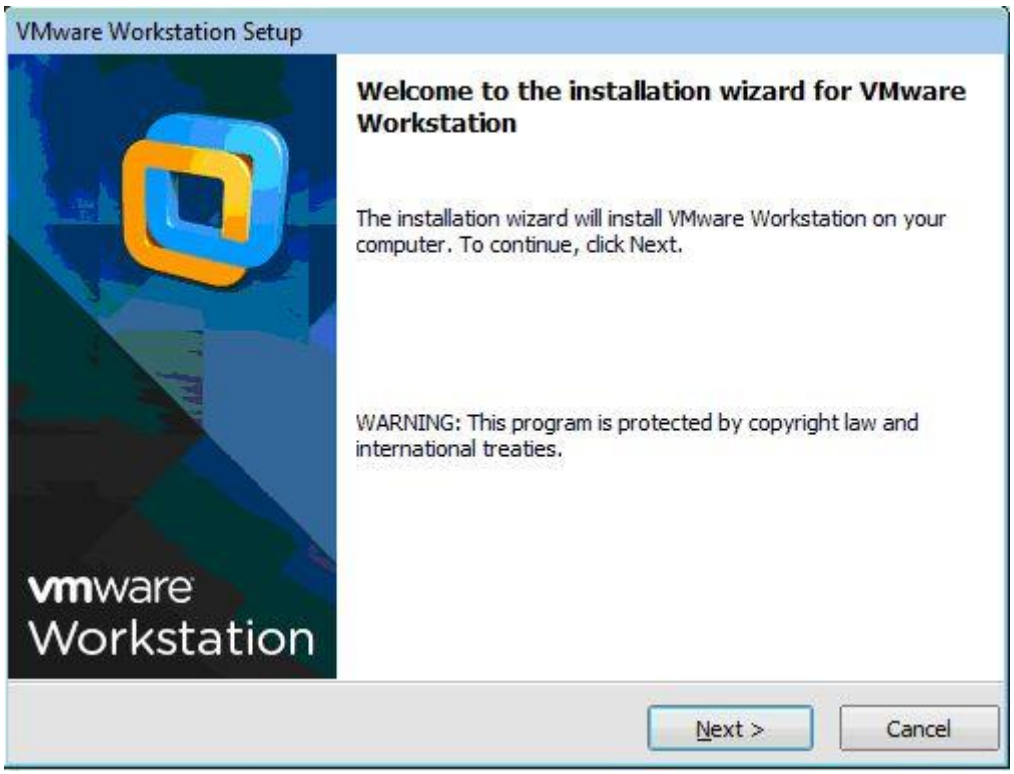


Рис. 2. Окно установщика VMware Workstation

Рис. 3. Лицензионное соглашение

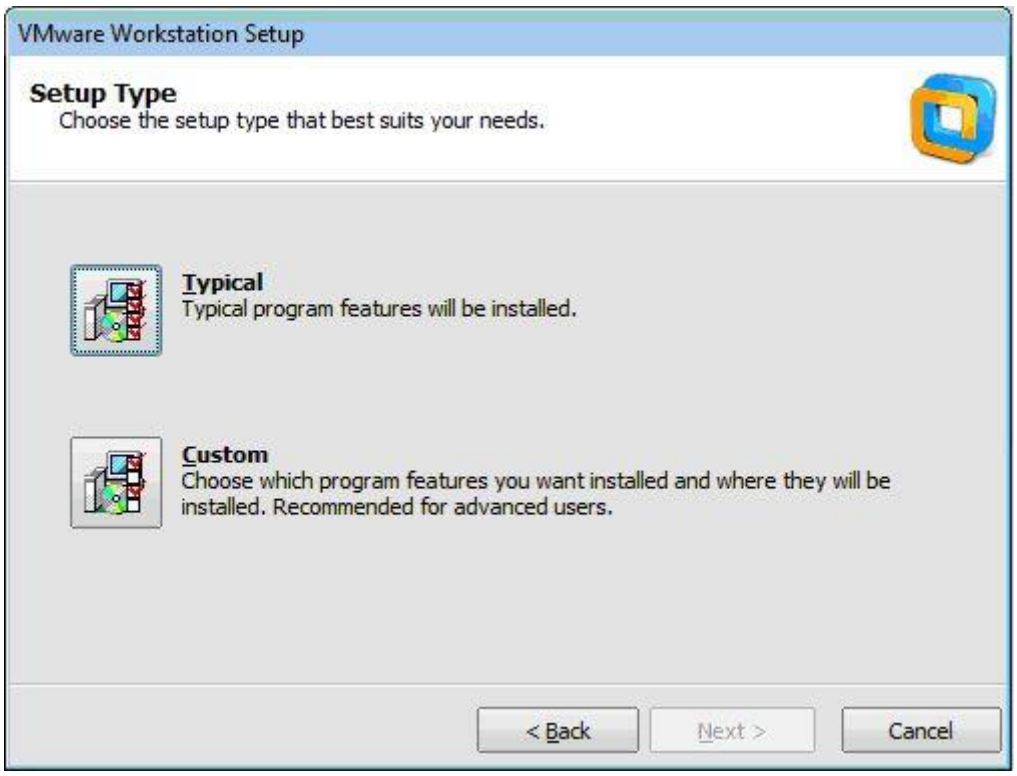


Рис. 4. Выбор типа установки

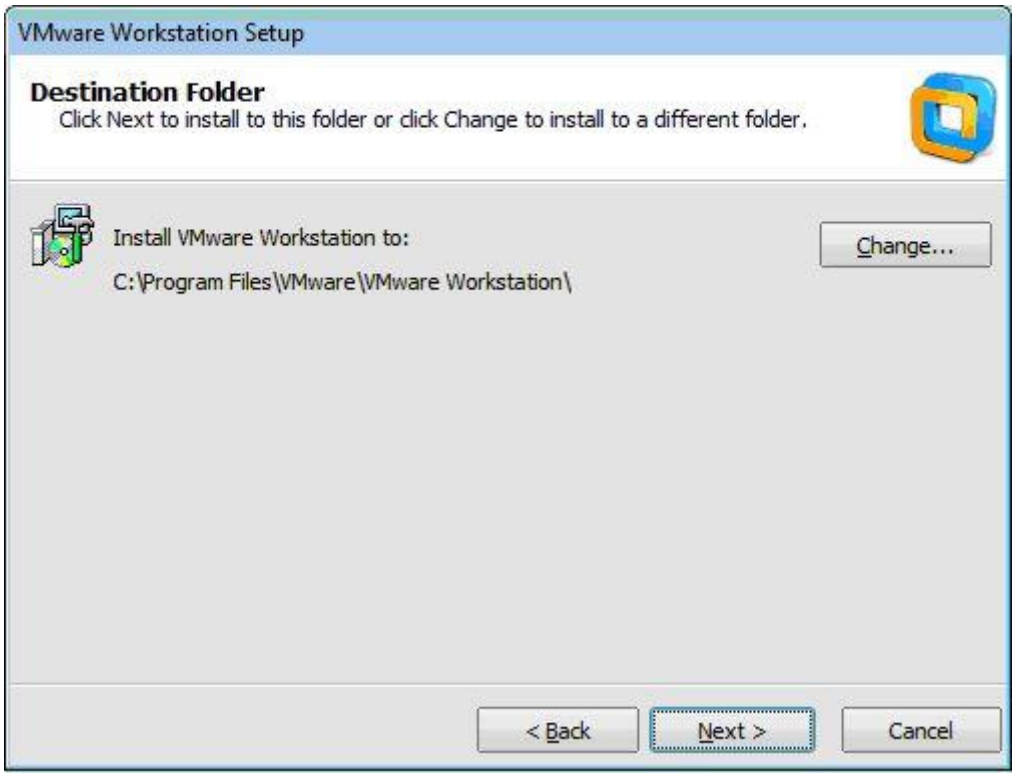


Рис. 5. Определение папки назначения

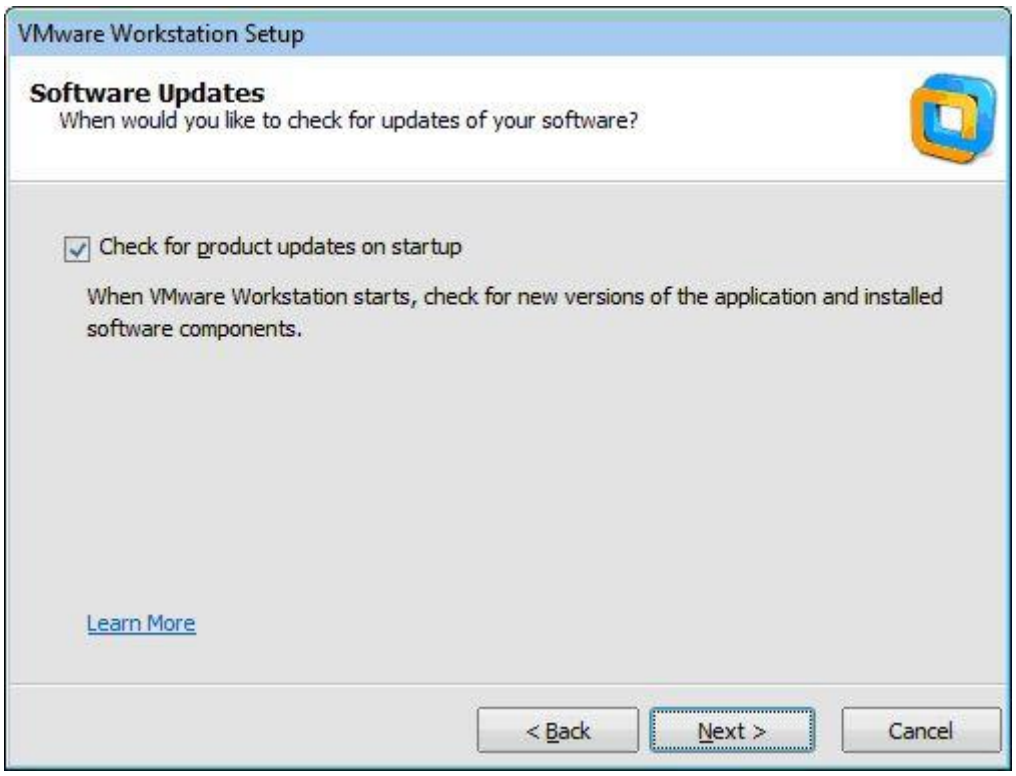


Рис. 6. Обновление программного обеспечения

Рис. 7. Улучшение качества программы

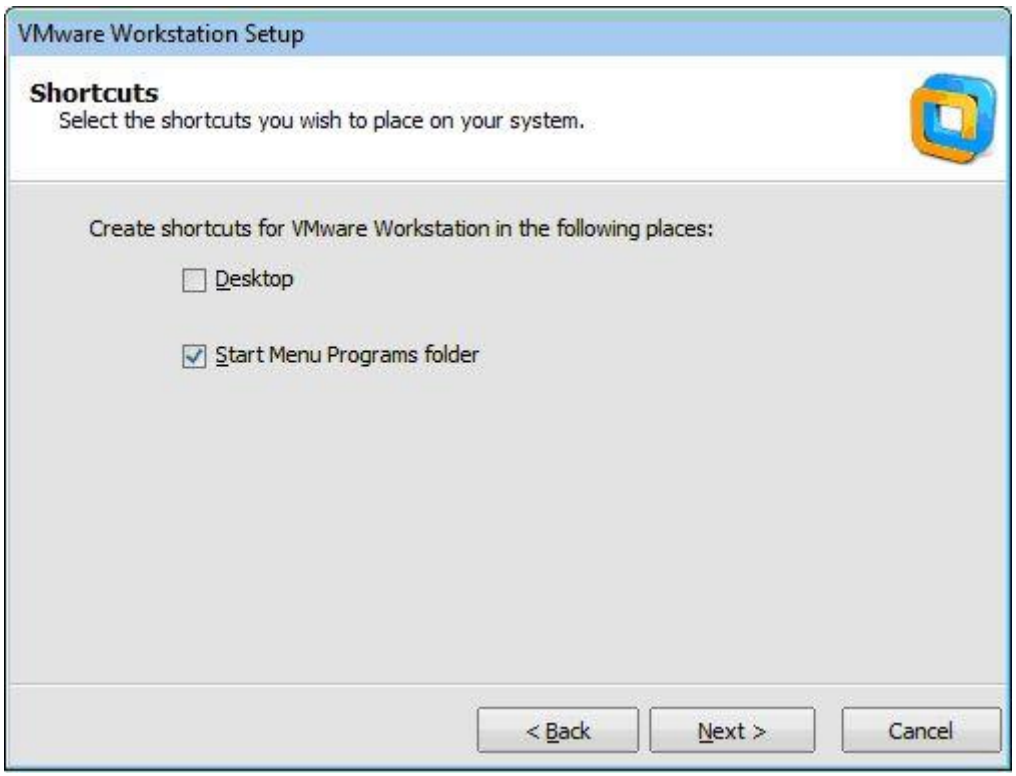


Рис. 8. Создание ярлыков

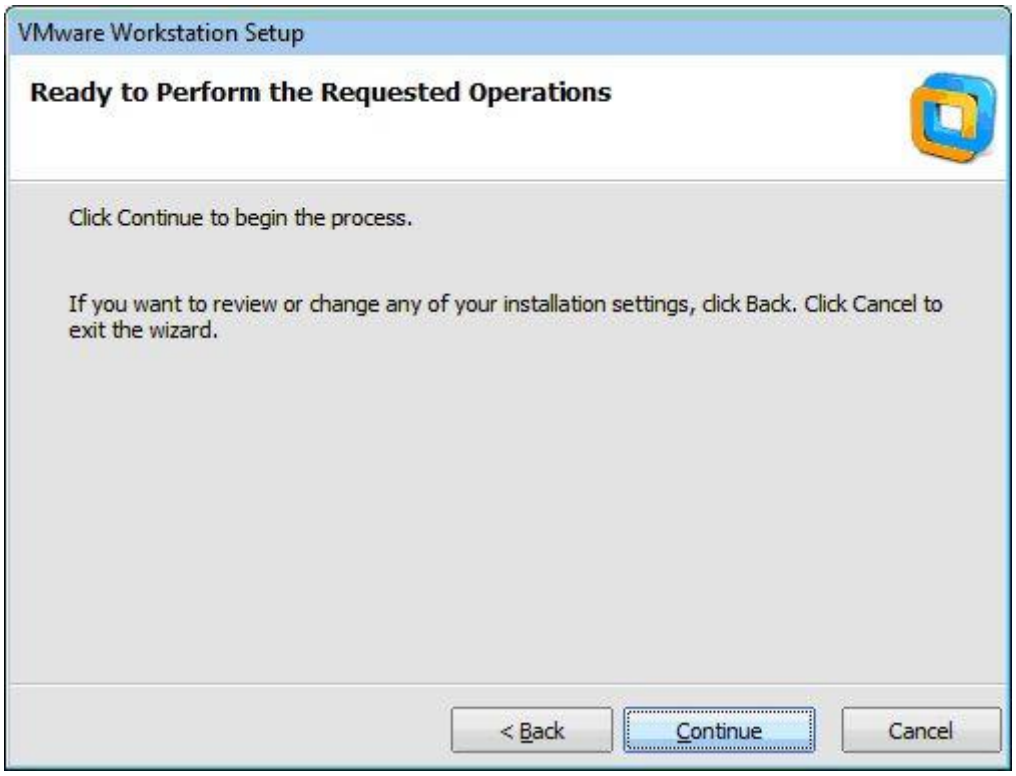


Рис. 9. Выполнение необходимых операций

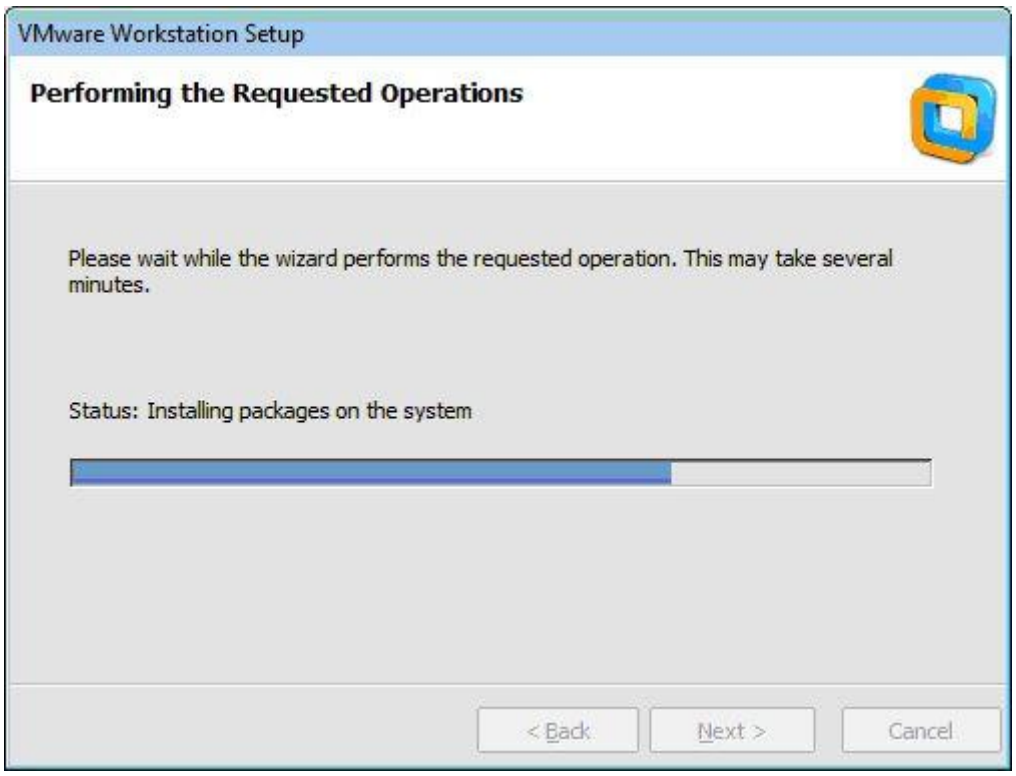


Рис. 10. Выполнение запрошенной операции

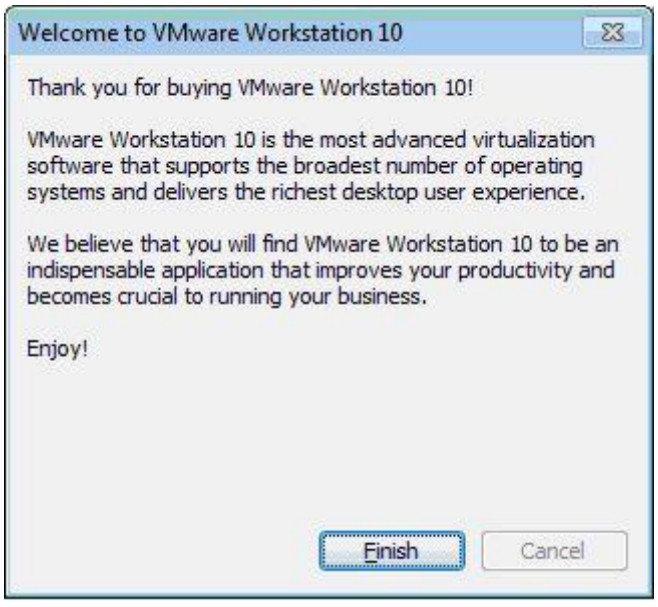


Рис. 11. Окончание установки

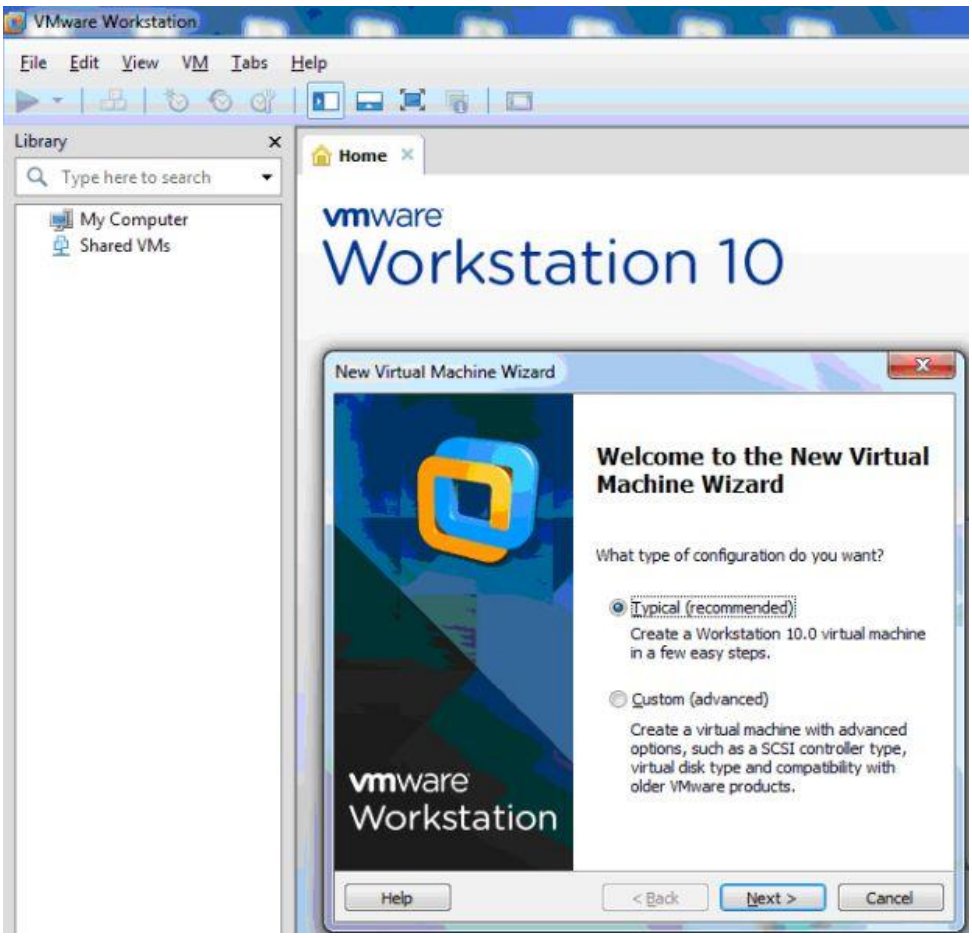


Рис. 12. Выбор конфигурации виртуальной машины

Рис. 13. Shared VMs

Сетевые настройки

Необходимо открыть *Пуск – Все программы – VMWare – Virual Network Editor* и провести конфигурацию *виртуальной сети*.

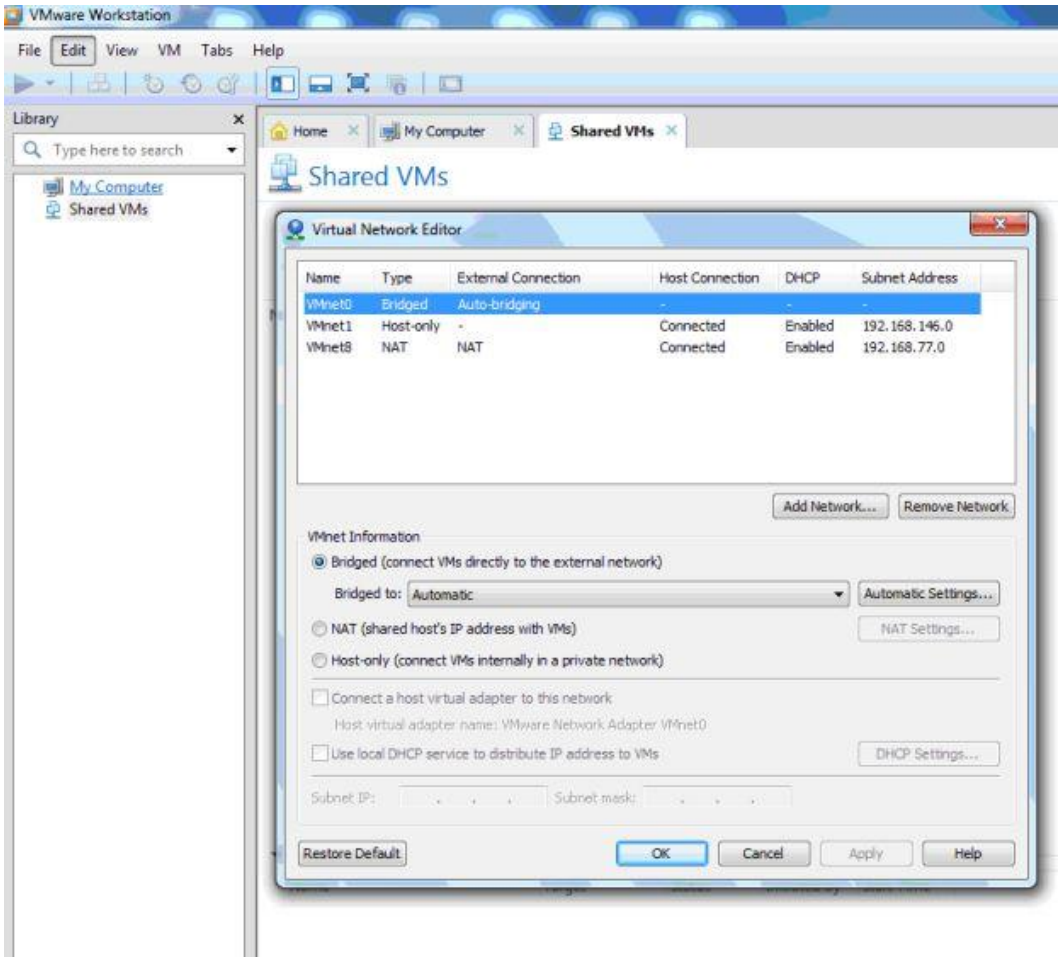


Рис. 14. Virual Network Editor

Создание виртуальной машины для гостевой операционной системы Windows 8

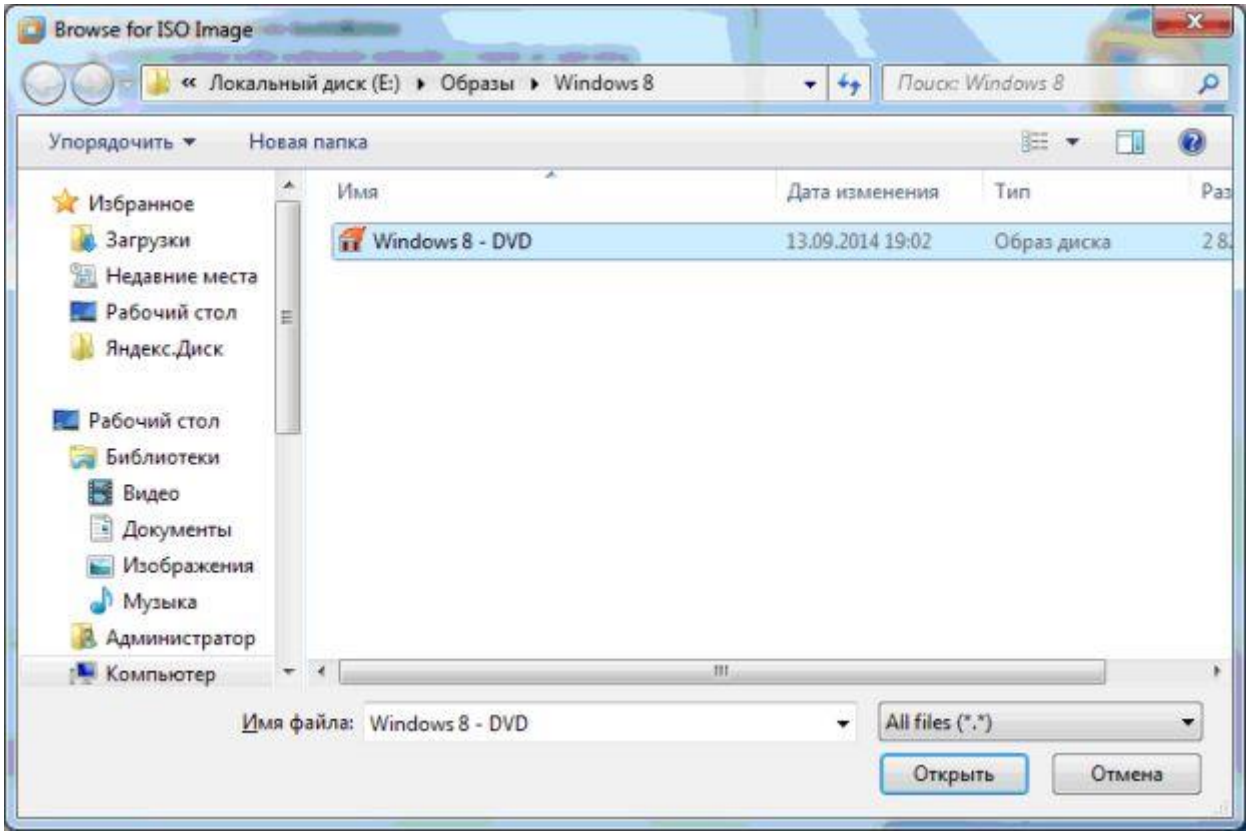


Рис. 15. Выбор операционной системы

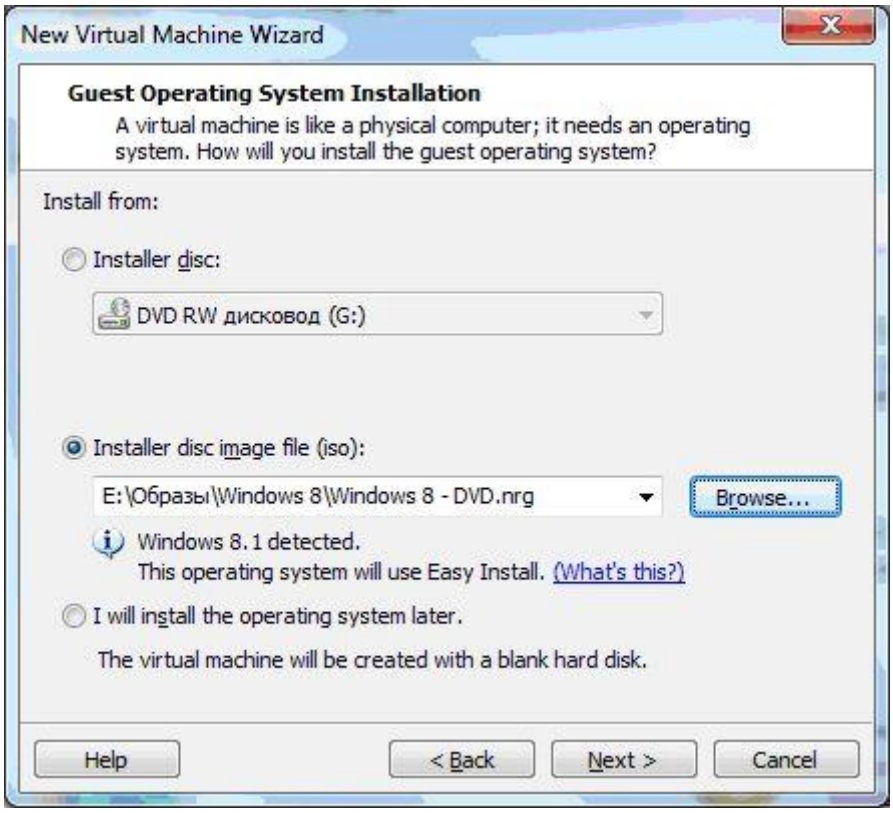


Рис. 16. Создание новой виртуальной машины

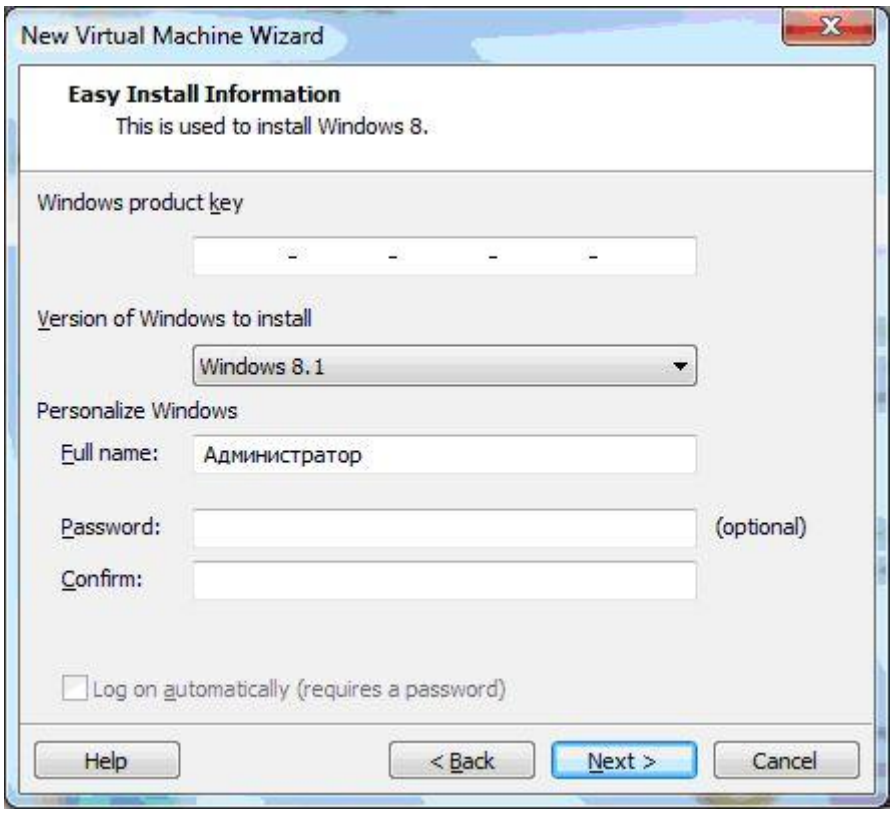


Рис. 17. Введение ключа продукта

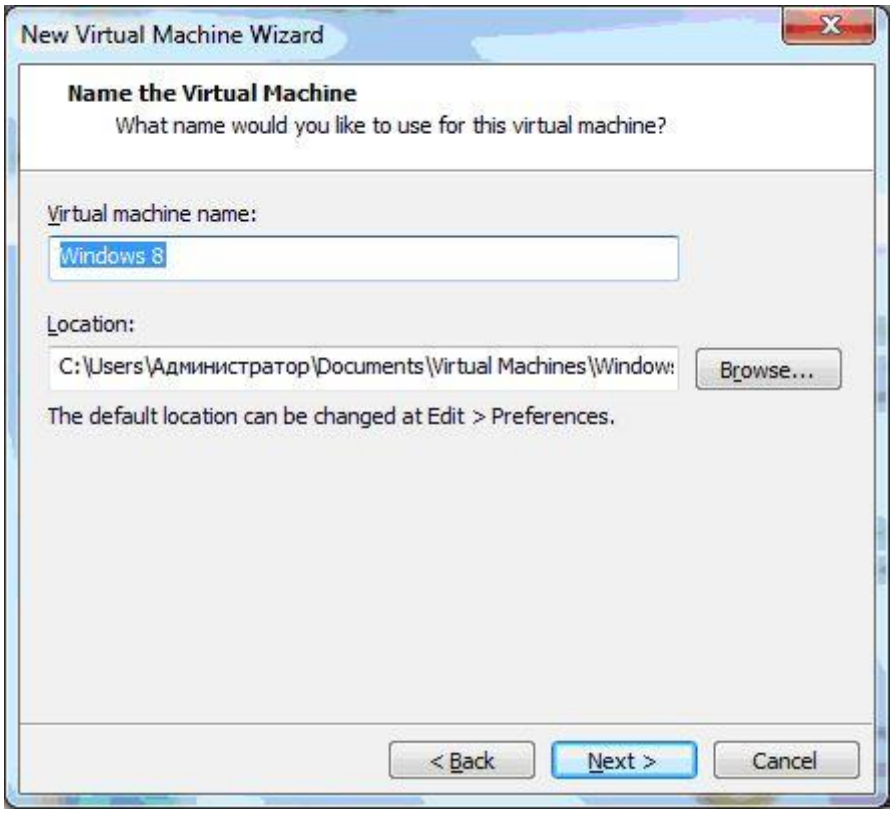


Рис. 18. Название виртуальной машины

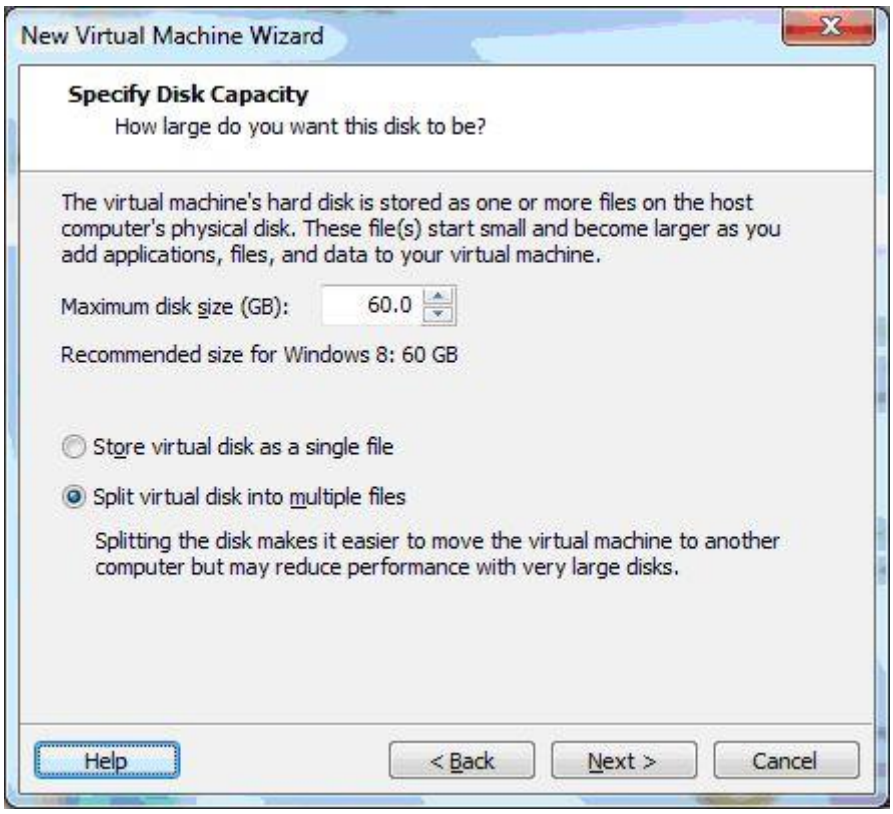


Рис. 19. Указание размера диска, разделение диска на несколько файлов

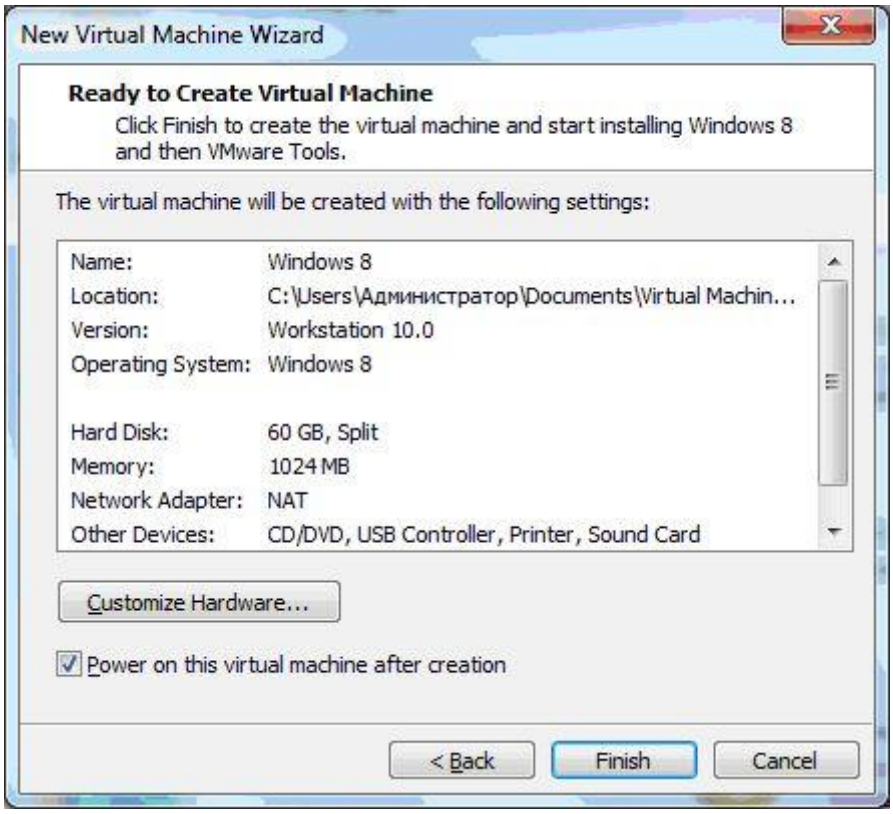


Рис. 20. Созданная виртуальная машина

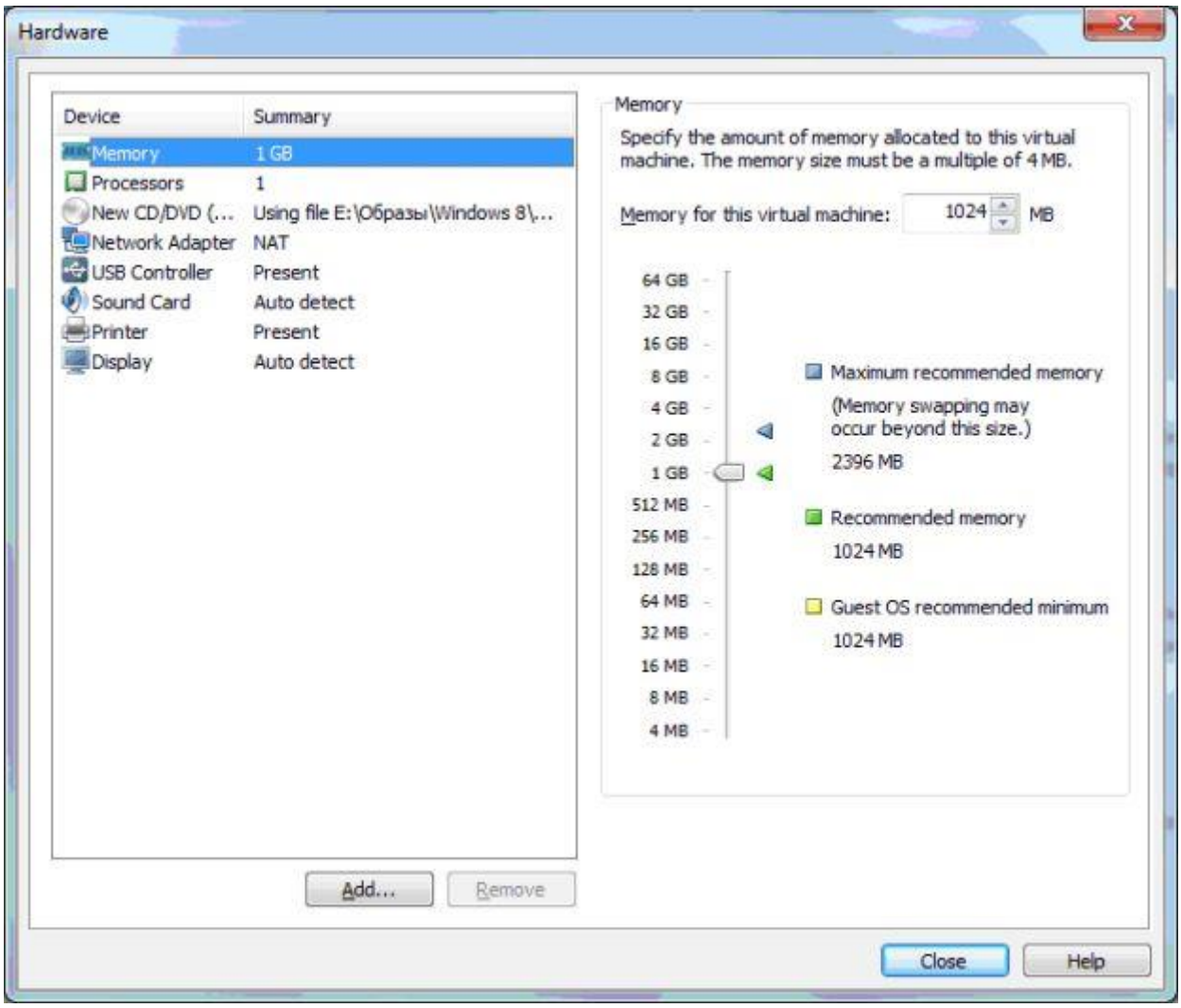


Рис. 21. Аппаратные средства

Рис. 22. Создание диска

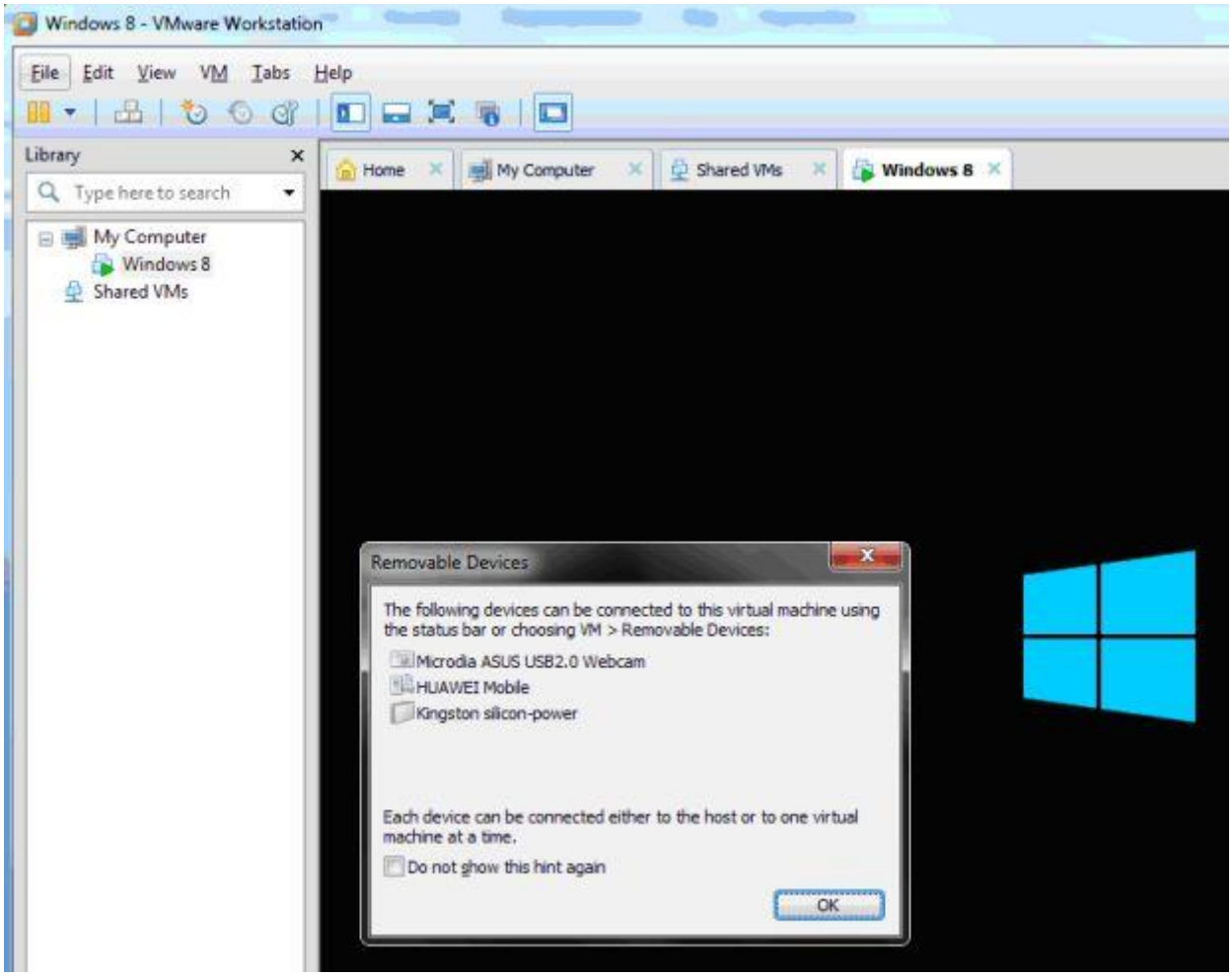


Рис. 23. Установка Windows 8

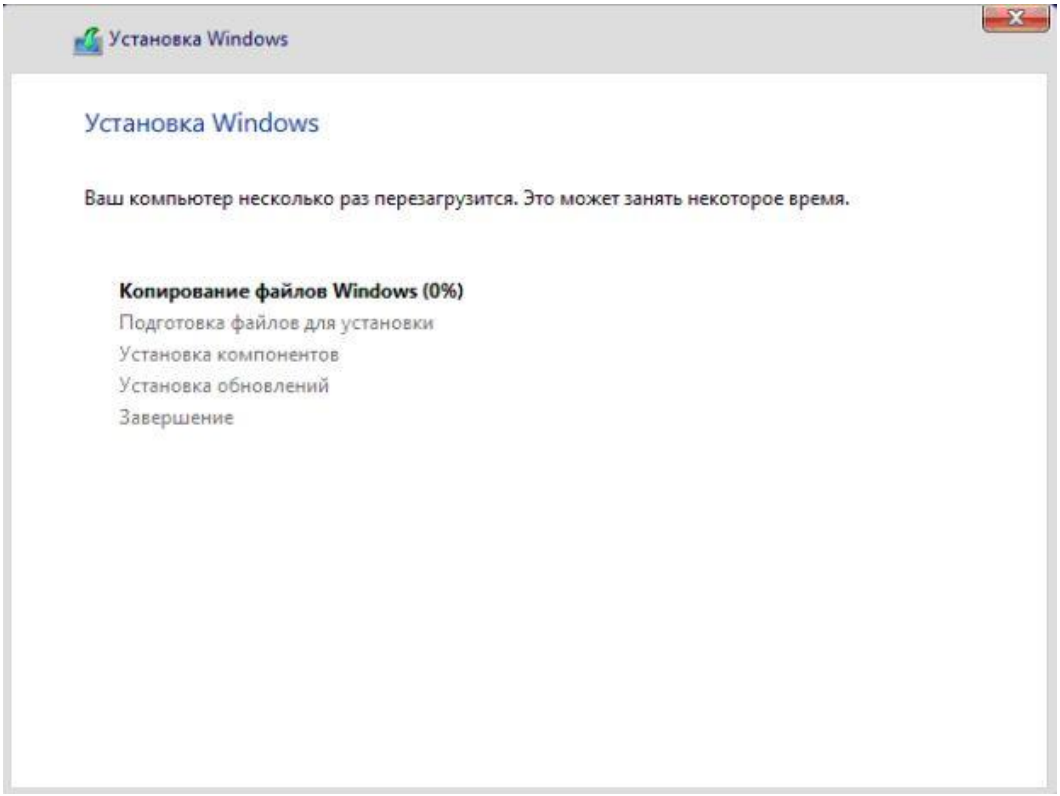


Рис. 24. Установка Windows 8

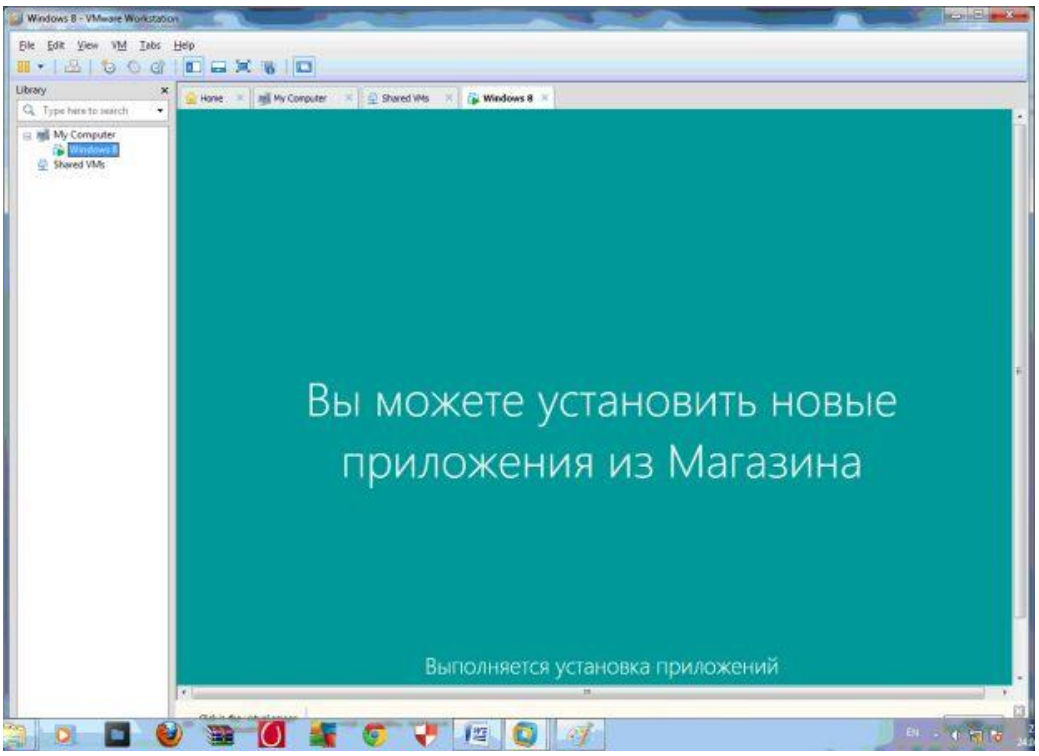


Рис. 25. Выполнение установки приложений

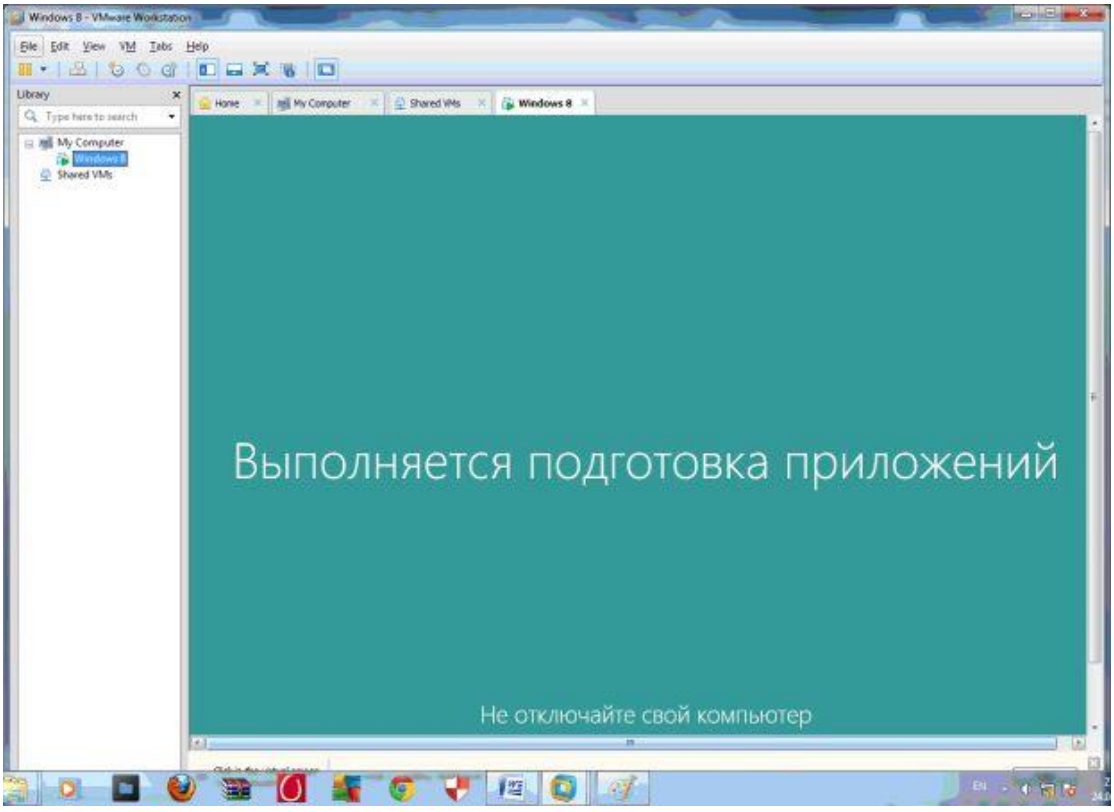


Рис. 26. Подготовка приложений

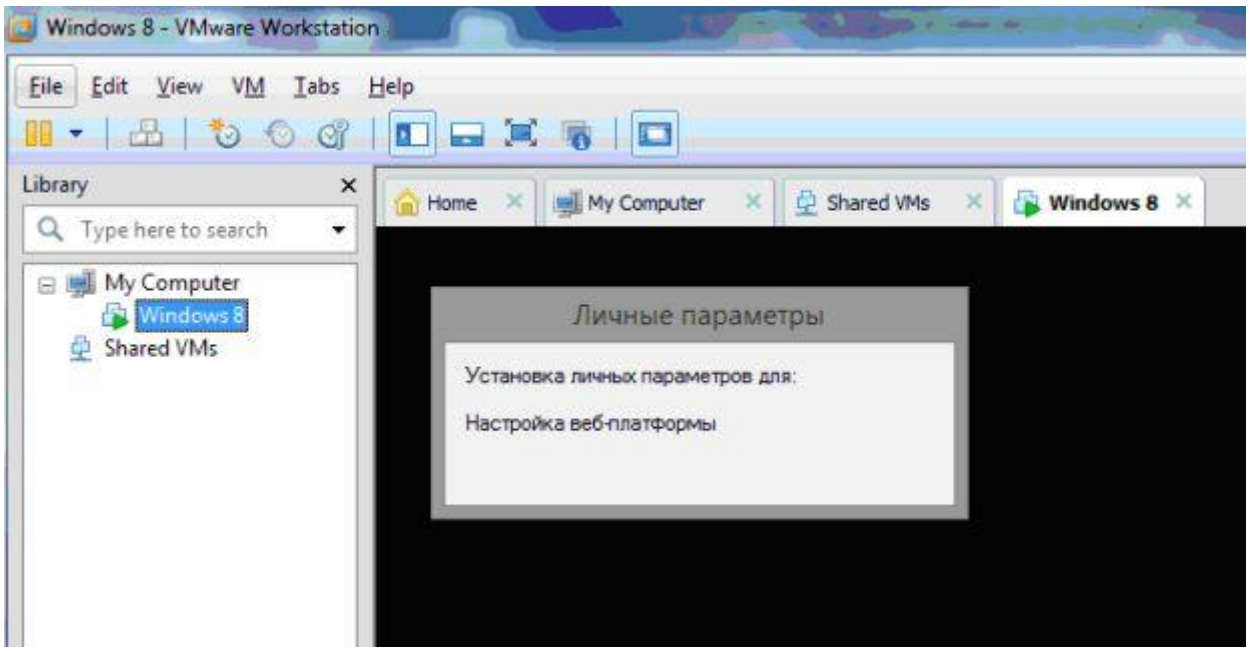


Рис. 27. Установка личных параметров

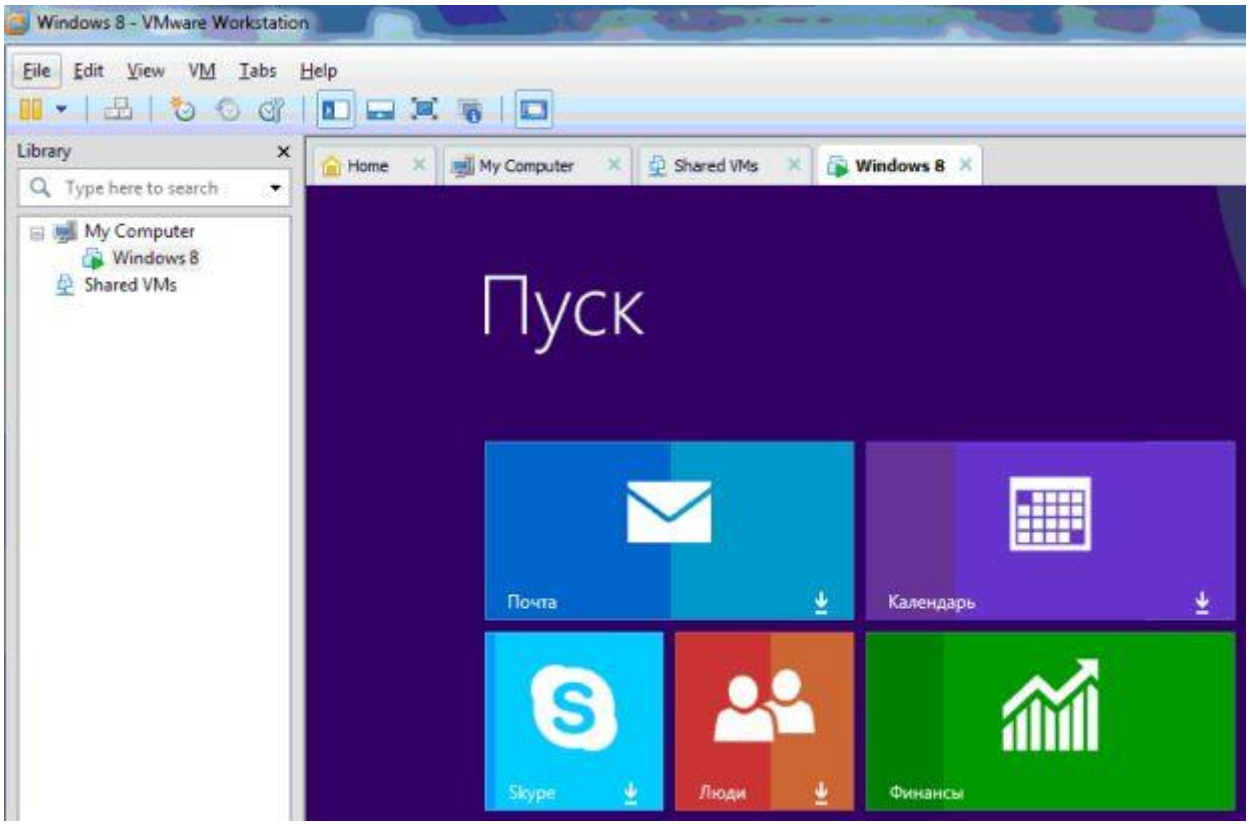


Рис. 28. Установленная операционная система

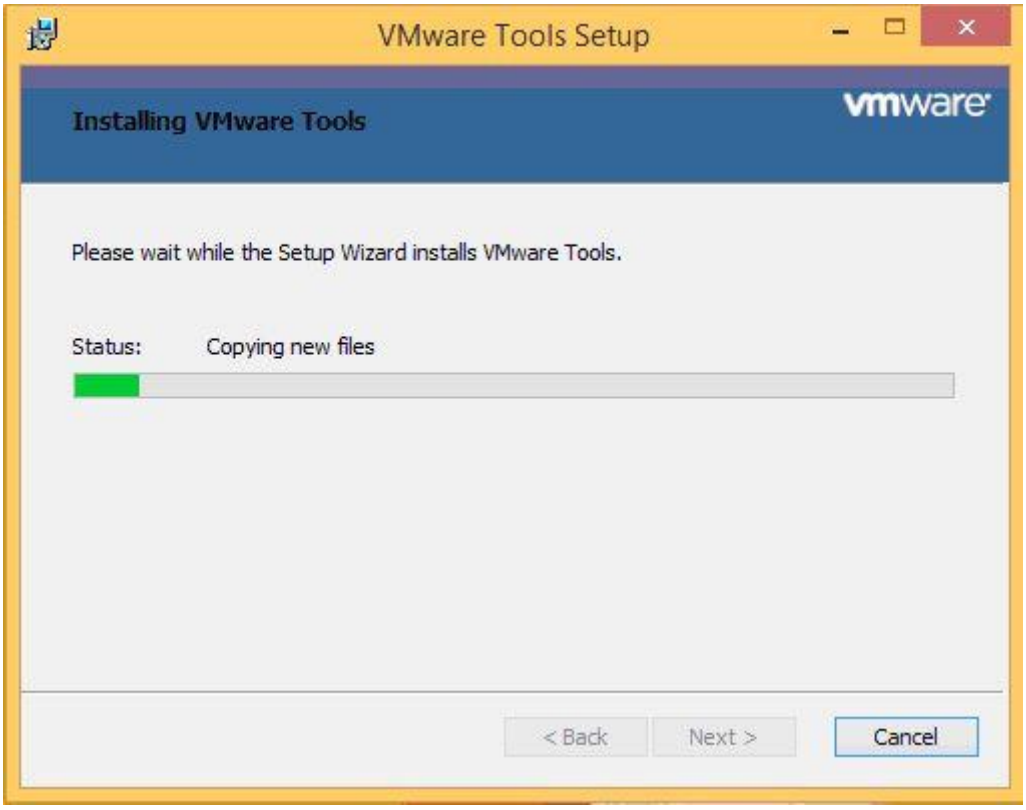


Рис. 29. Установка инструментов VMware

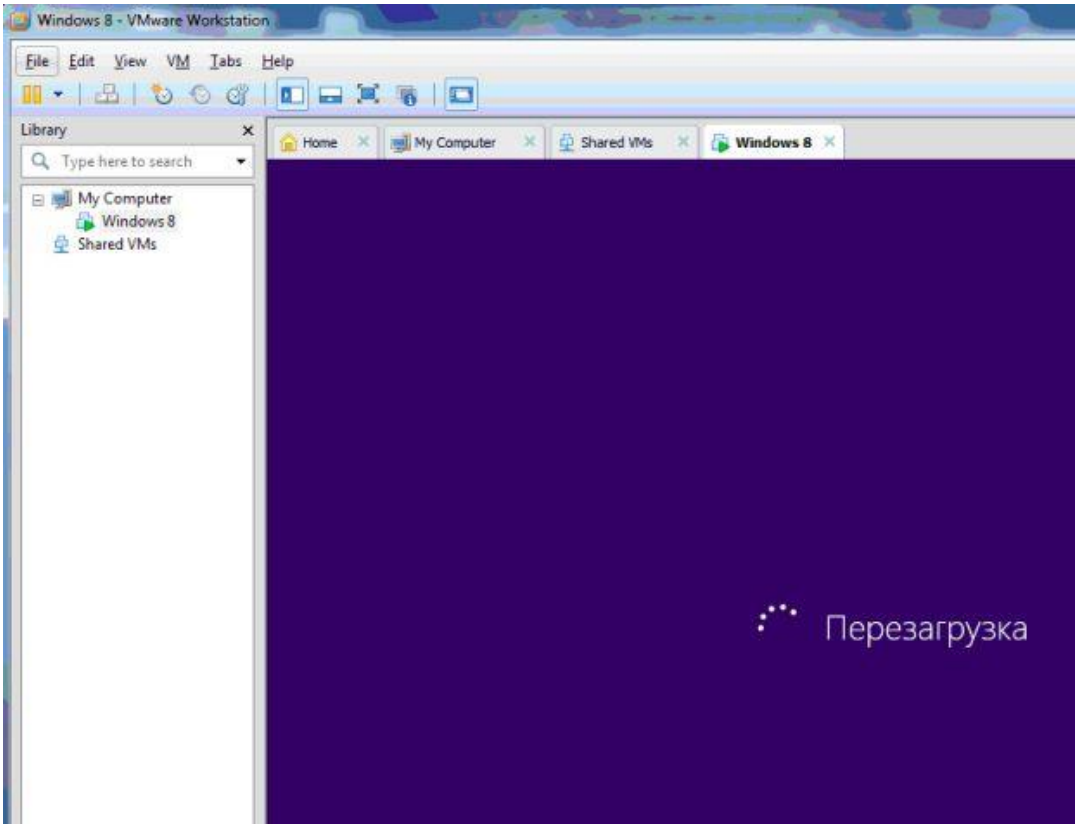


Рис. 30. Перезагрузка ОС

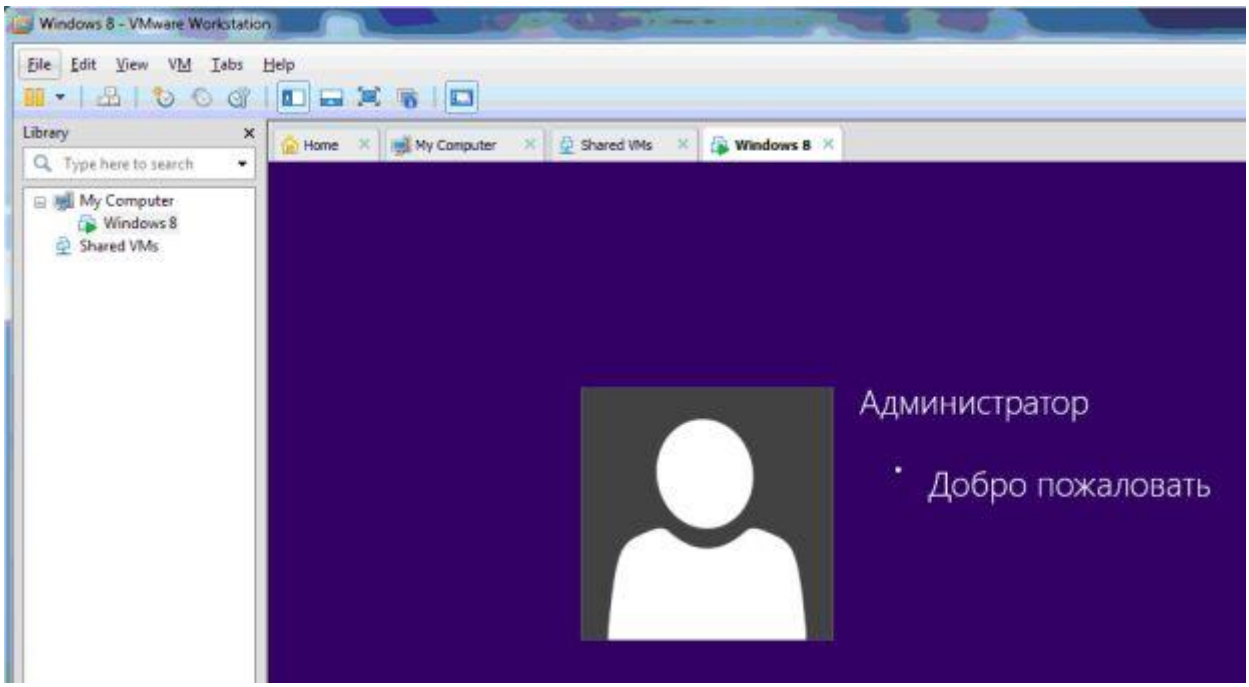


Рис. 31. Окно приветствия ОС

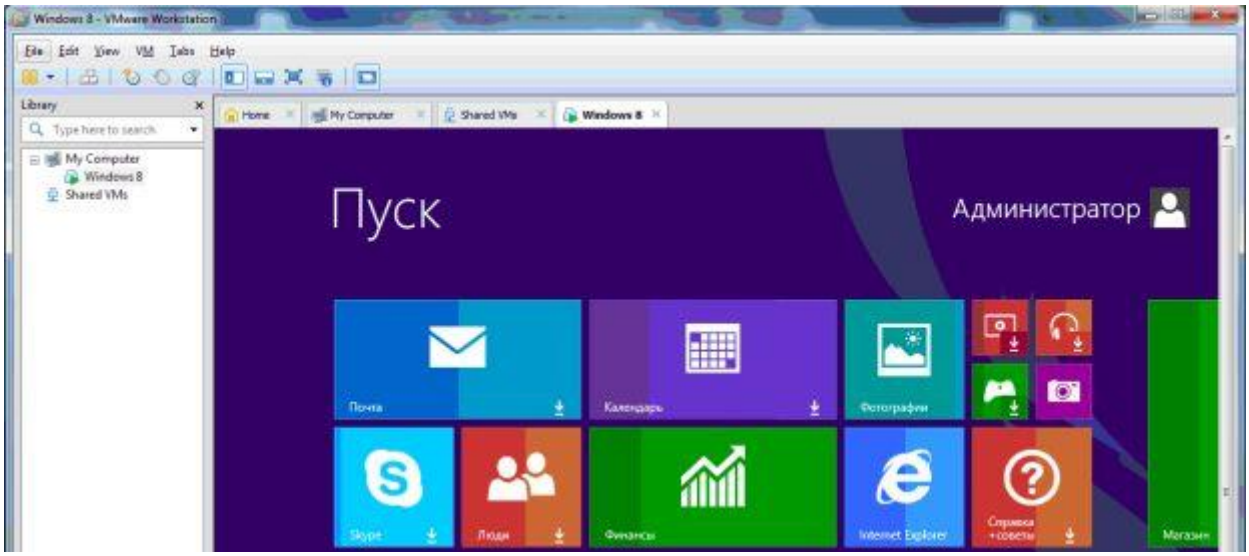


Рис. 32. Запуск ОС

Создание снимка виртуальной машины

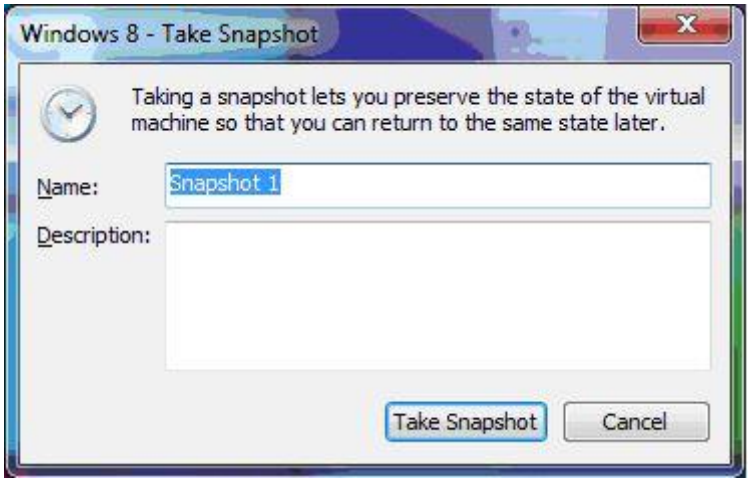


Рис. 33. Создание снимка виртуальной машины Snapshot 1

Рис. 34. Создание снимка виртуальной машины

Изменение в гостевой операционной системе

Рис. 35. Выполнение произвольных изменений в виртуальной машине

Используя возврат к предыдущему снимку, можно отменить изменения

Рис. 36. Возврат к предыдущему снимку виртуальной машины

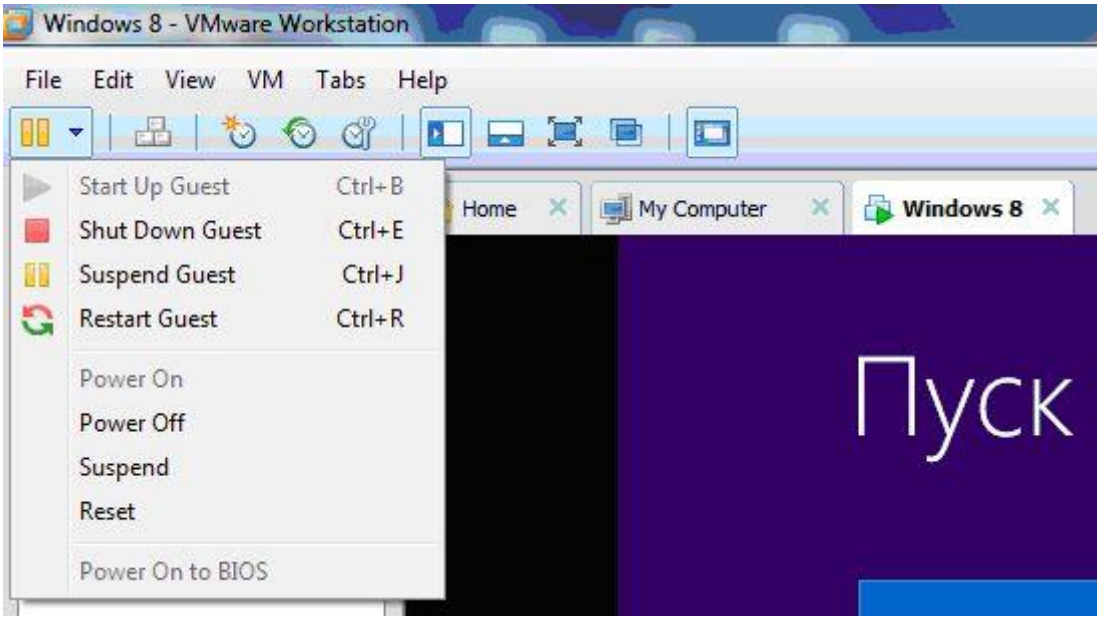


Рис. 37. Выключение виртуальной машины

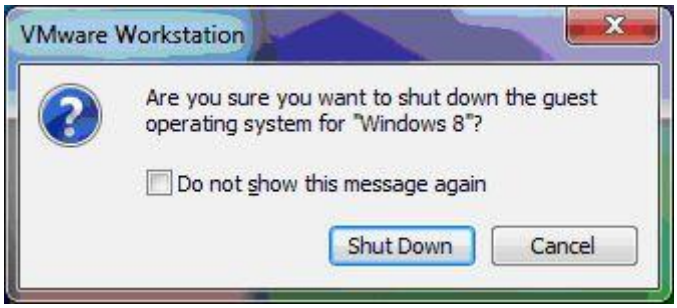


Рис. 38. Завершение работы виртульной машины

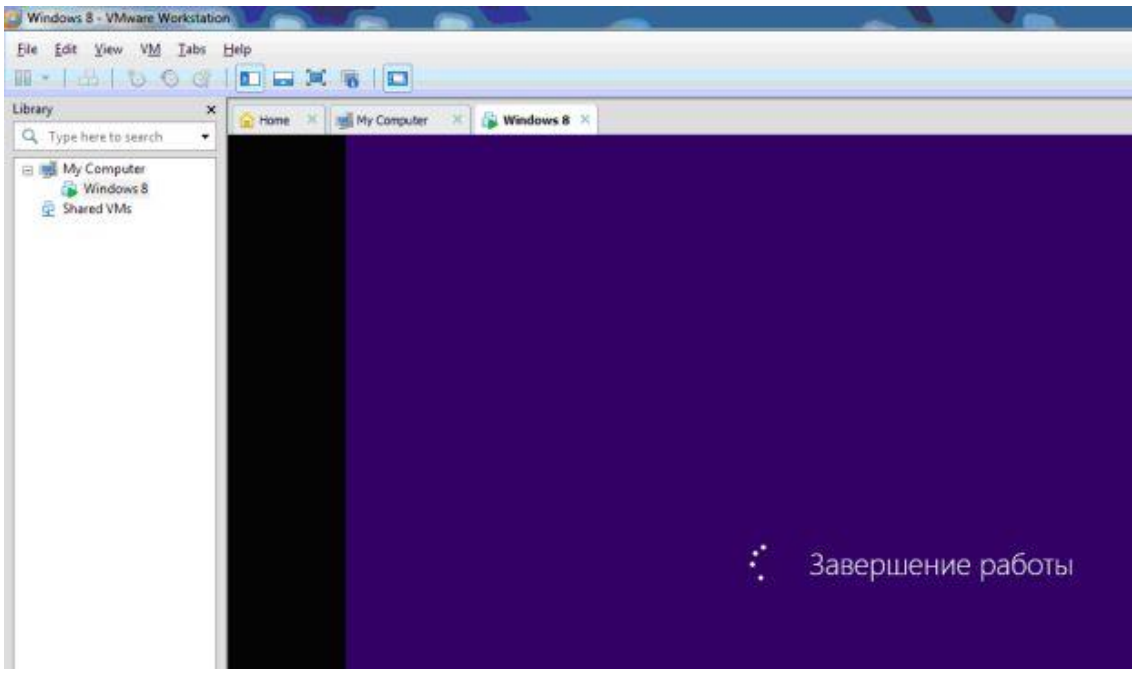


Рис. 39. Завершение работы виртуальной машины

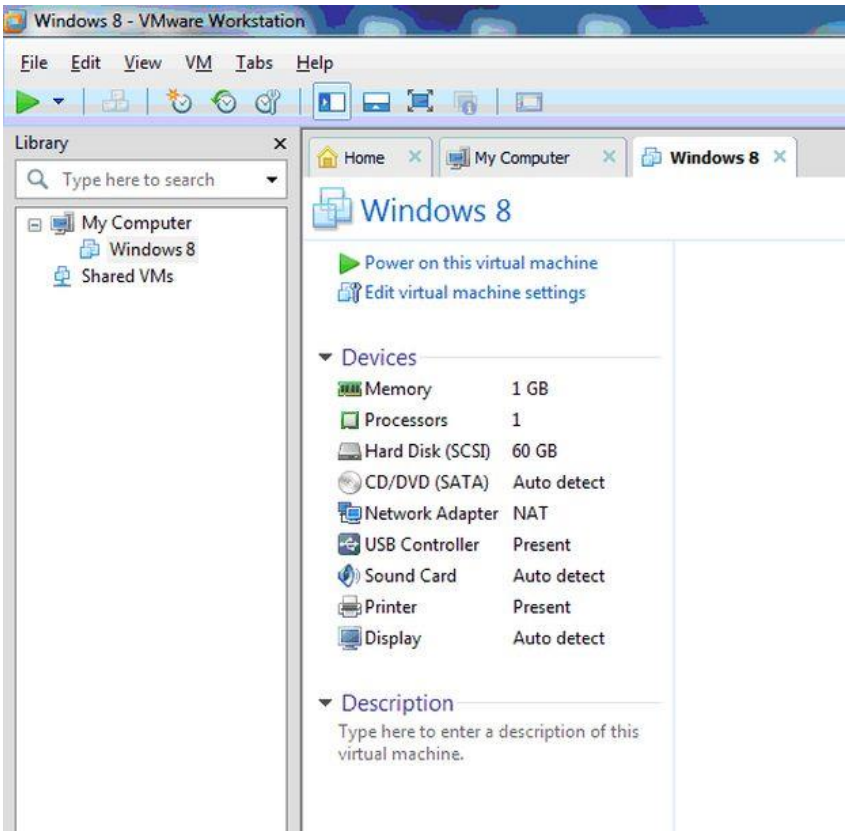


Рис. 40. Изменение конфигурации виртуальной машины

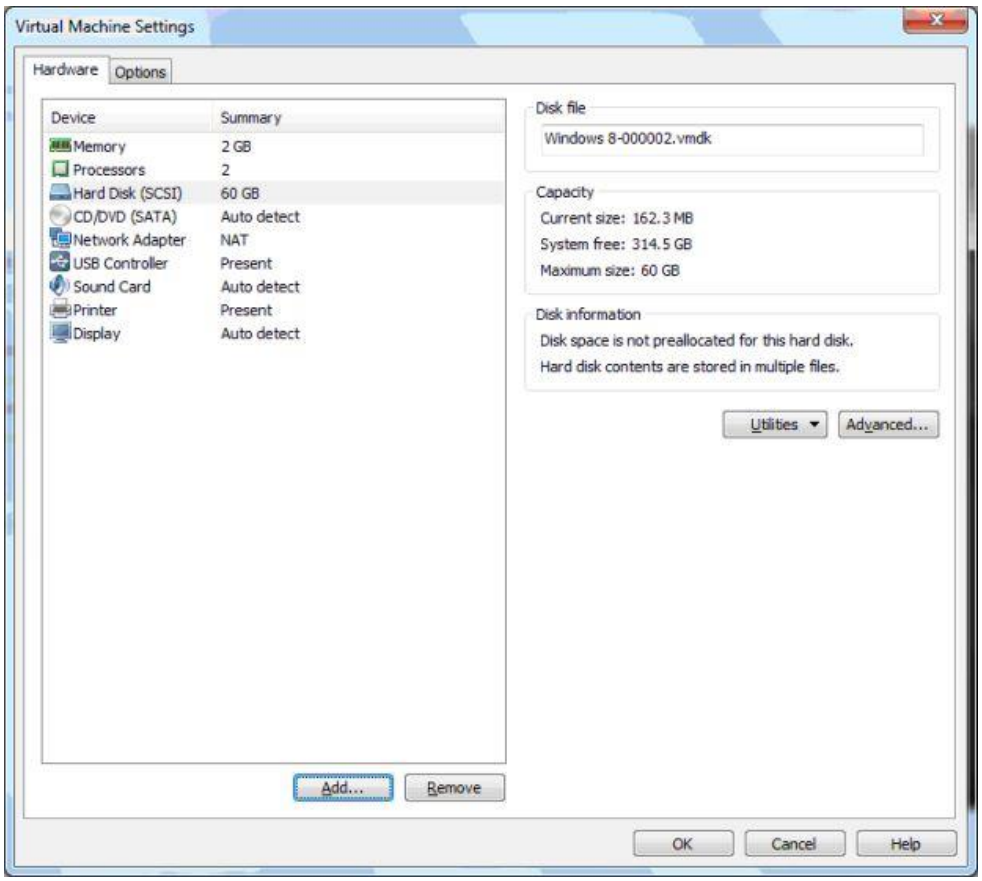


Рис. 41. Увеличение количества оперативной памяти и процессоров

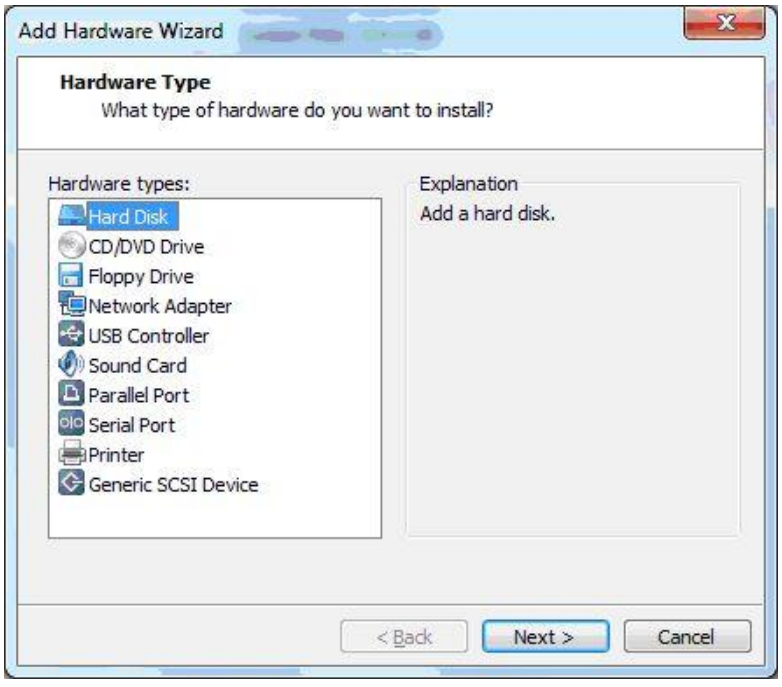


Рис. 42. Создание дополнительного жесткого диска

Рис. 43. Создание дополнительного жесткого диска SCSI

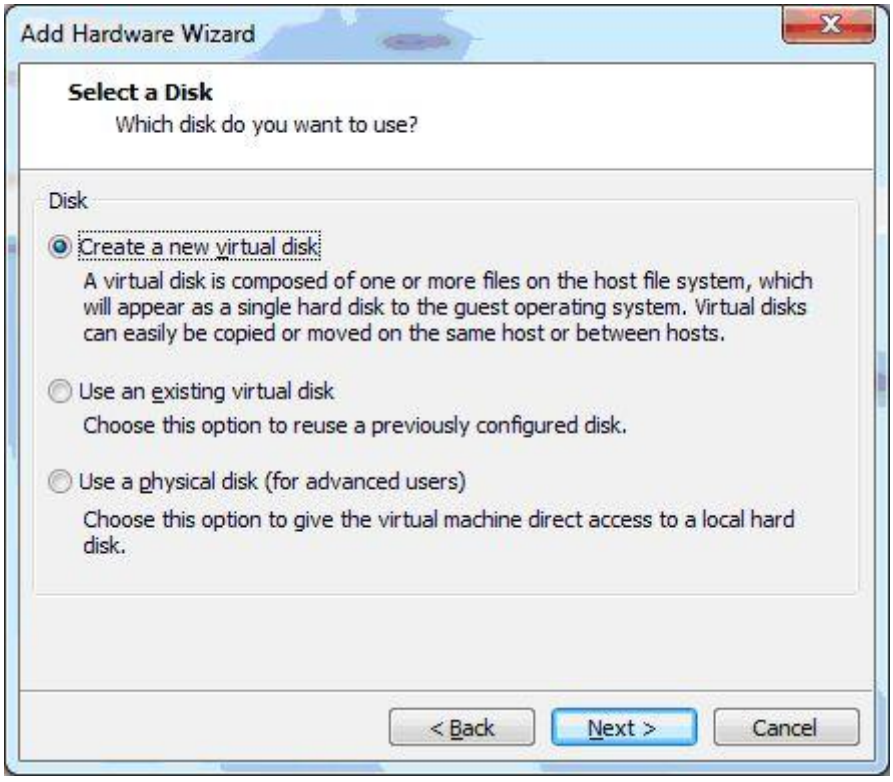


Рис. 44. Создание нового виртуального жесткого диска

Рис. 45. Размер нового виртуального жесткого диска

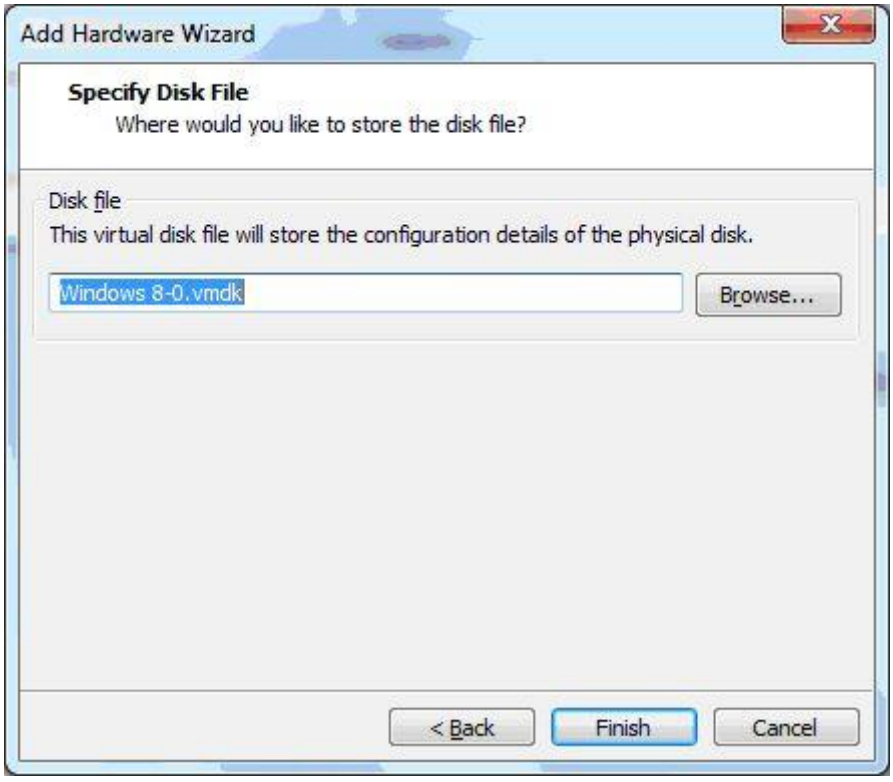


Рис. 46. Windows 8-0.vmdk

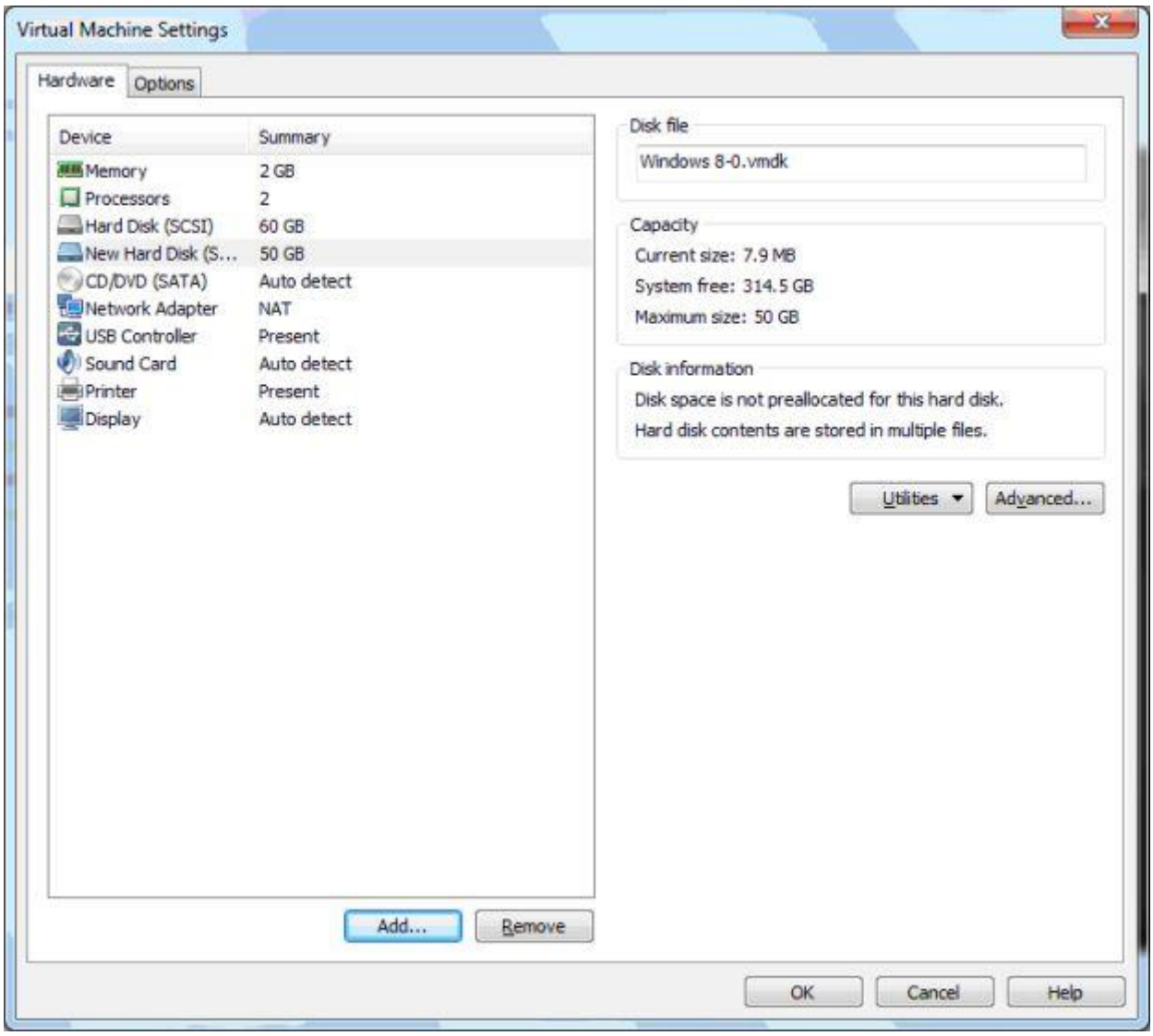


Рис. 47. Созданный дополнительный жесткий диск

Практическое задание к работе.

1. Установить под виртуальной машиной в каждой из установленных операционных систем одинаковый бесплатный антивирус на ваш выбор. Зафиксируйте полученные результаты.
2. Установить под виртуальной машиной в каждой из установленных операционных систем одинаковый пакет Open Office (кроме Ubuntu). Зафиксируйте полученные результаты.
3. Опробуйте в среде нескольких виртуальных ОС работу популярного интернет-магазина (зайдите под своим аккаунтом) и зафиксируйте полученную разницу.
4. Зафиксируйте, какие современные интернет-технологии НЕ поддерживаются устаревшими версиями операционных систем.
5. Запишите на флэш-накопитель в среде хост-машины несколько файлов большого объёма и попробуйте открыть его в среде виртуальных ОС. Результаты зафиксируйте.
6. Полученные результаты свести в таблицу следующего вида (для отчётности):

Вид испытания	Хостовая ОС (название)	Windows 10 (версия, номер сборки)	Windows 8.1 (версия, номер сборки)	Windows 7 (версия, номер сборки)	Windows XP (версия, номер сборки)	Ubuntu (версия, номер сборки)
Антивирус (название, версия)	Результат апробации (полностью работоспособен, не запускается с ошибкой, частично работоспособен)					
Open Office (версия)						
Интернет-магазин (адрес)						
Встроенный медиа-плеер в браузере (название, используемая технология)						
Чтение флэш-накопителя						

Лабораторная работа №12. Работа в среде виртуальной машины: изменение настроек по умолчанию, настройка обновлений, установка драйверов

Теоретический материал.

В IObit Software Updater, встроена система бэкапов. Когда вы что-то обновляете - делается резервная копия. Если вас что-то не устроило - достаточно зайти в меню "Восстановление", затем выбрать нужный бэкап (ориентируясь по дате) и запустить процедуру восстановления...

Если что-то пошло не так — восстановить.

Обновление драйверов.

Что такое драйверы и для чего они нужны

В нашей статье рассматривается только установка драйверов под ОС семейства Windows, но и в других операционных системах для работы устройств требуются специальные программы-драйверы. Существует несколько способов, как обновить драйверы на вашем ПК. Это можно реализовать как при помощи стандартных инструментов ОС Windows, так и с помощью сторонних приложений.

Для чего нужно обновлять драйверы на компьютере

Свежие обновления драйверов зачастую содержат новые и улучшенные функции аппаратных компонентов. А также в них регулярно исправляют недоработки прошлых версий. В значительной части случаев сбои и нестабильность работы компьютера объясняются той или иной проблемой с драйверами:

- установлен устаревший драйвер, некорректно взаимодействующий с другими аппаратными и программными компонентами;
- файл(ы) драйвера повреждён(ы);
- установлен стандартный системный драйвер, а не программа от производителя оборудования;
- отсутствуют драйверы на те или иные периферийные устройства.

Во всех приведённых выше случаях проблема решится обновлением драйверов.

ВАЖНО: иногда, напротив, источником проблем становится более новый драйвер (например, обновлённая версия некорректно работает со старым ПО). Поэтому настоятельно рекомендуется перед каждым обновлением драйверов делать контрольную точку восстановления системы.

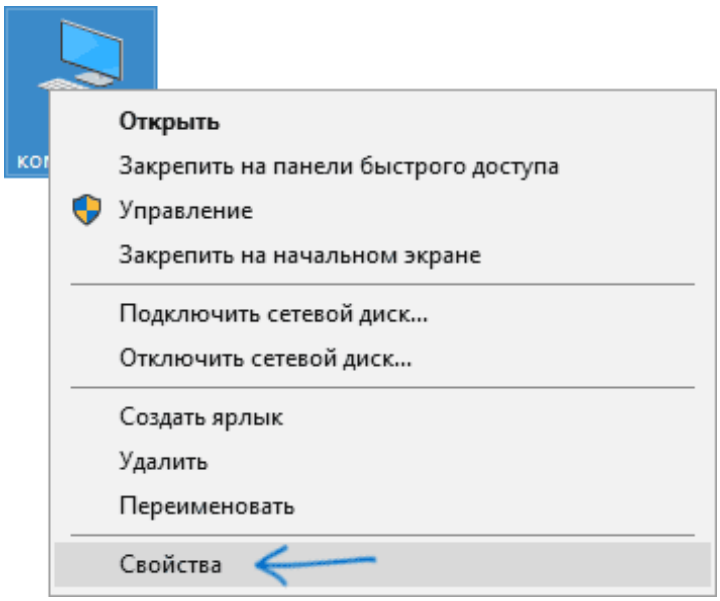
Как обновить драйверы на компьютере: пошаговая инструкция со скриншотами

Принципы установки драйверов сходны для всех ОС семейства Windows. В любой версии операционной системы от Microsoft драйверы можно ставить либо вручную (через Диспетчер устройств), либо автоматически (с помощью специального программного обеспечения)

Обновление драйверов вручную (через Диспетчер устройств)

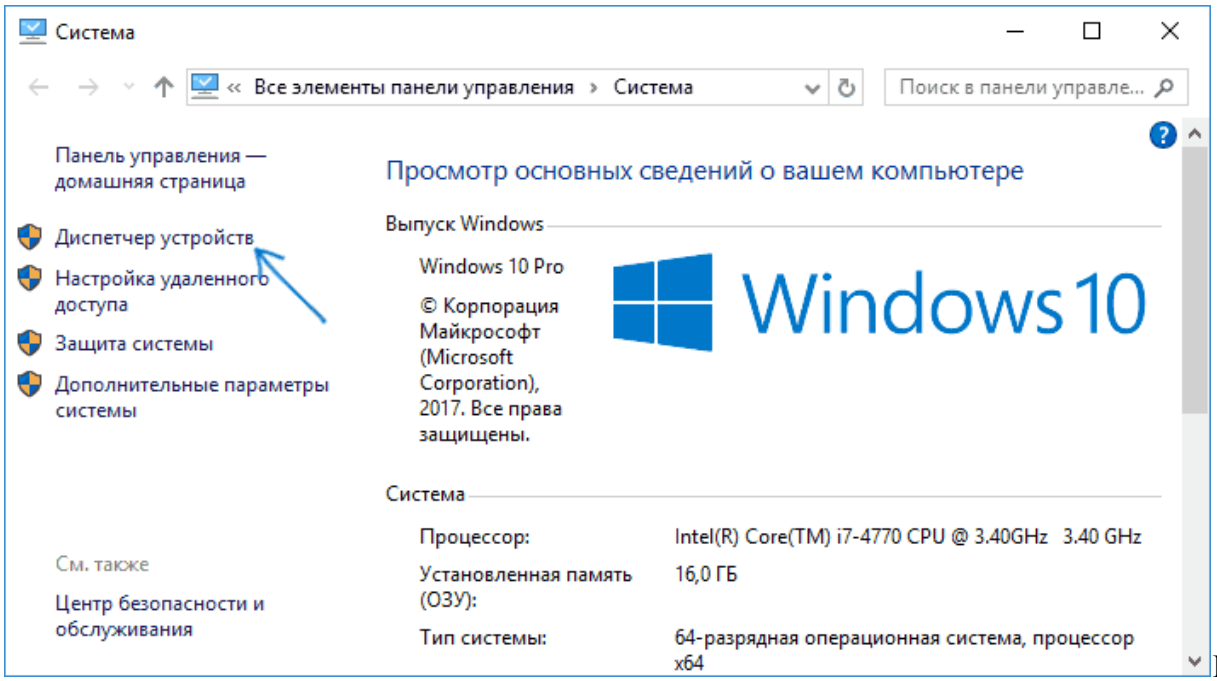
Во всех Windows, от XP до 10, ручное обновление драйверов производится через Диспетчер устройств (Device Manager). Диспетчер устройств можно вызвать либо напрямую запустив devicemgr.msc с помощью окна «Выполнить» (Win+R или «Пуск» → «Выполнить»), либо через пункт «Свойства» контекстного меню значка «Этот компьютер» («Мой компьютер» в более ранних версиях Windows), либо через Панель управления.

Доступ к Диспетчеру устройств через контекстное меню иконки «Этот компьютер»:



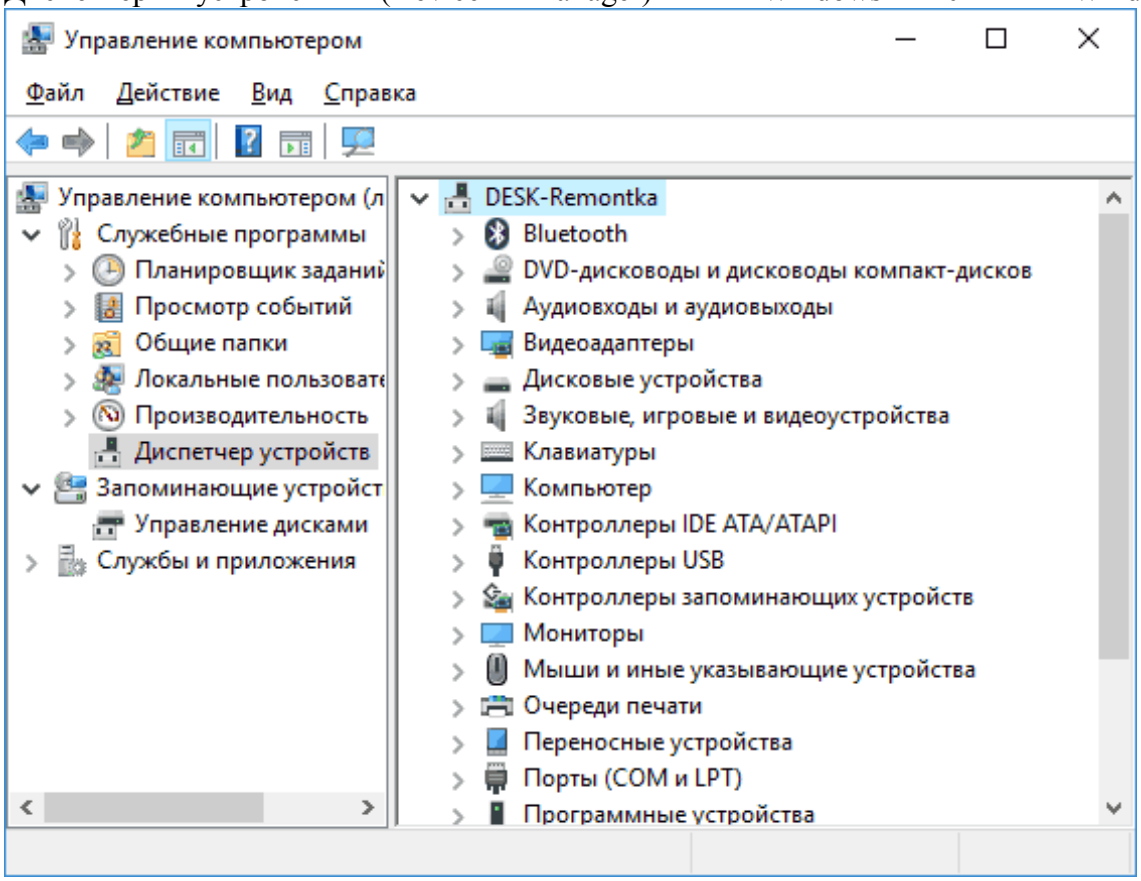
Меню «Свойства» иконки «Мой компьютер» — самый быстрый способ попасть в Диспетчер устройств

Доступ к Диспетчеру устройств через «Панель управления» → «Система».



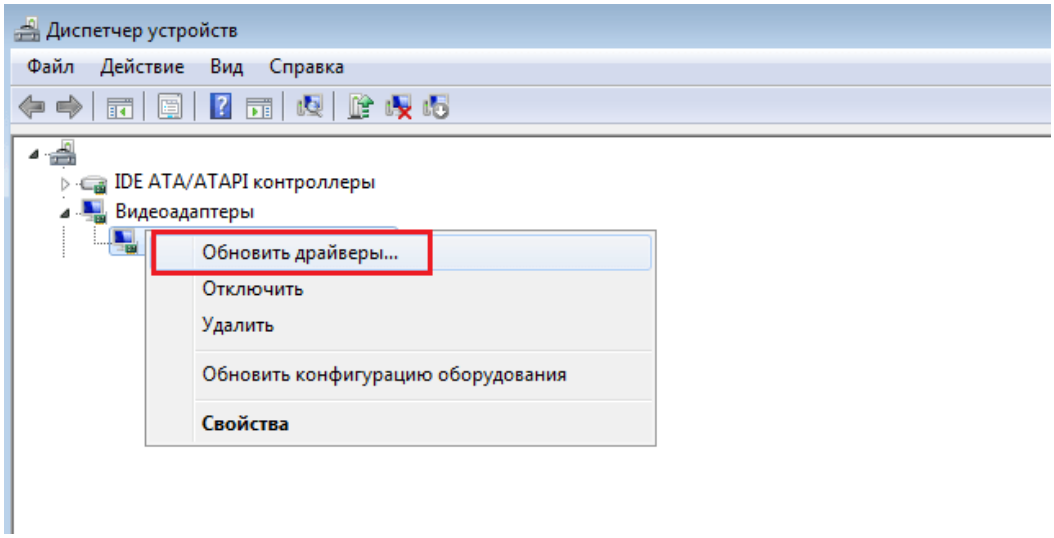
В Диспетчер устройств можно попасть через «Панель управления, вкладка «Система»

Диспетчер устройств (Device Manager) в Windows 10 и Windows XP:



Так выглядит диспетчер устройств в последней версии ОС Windows

Как вы можете видеть, внешний вид Диспетчера устройств в разных версиях ОС Windows несколько отличается.



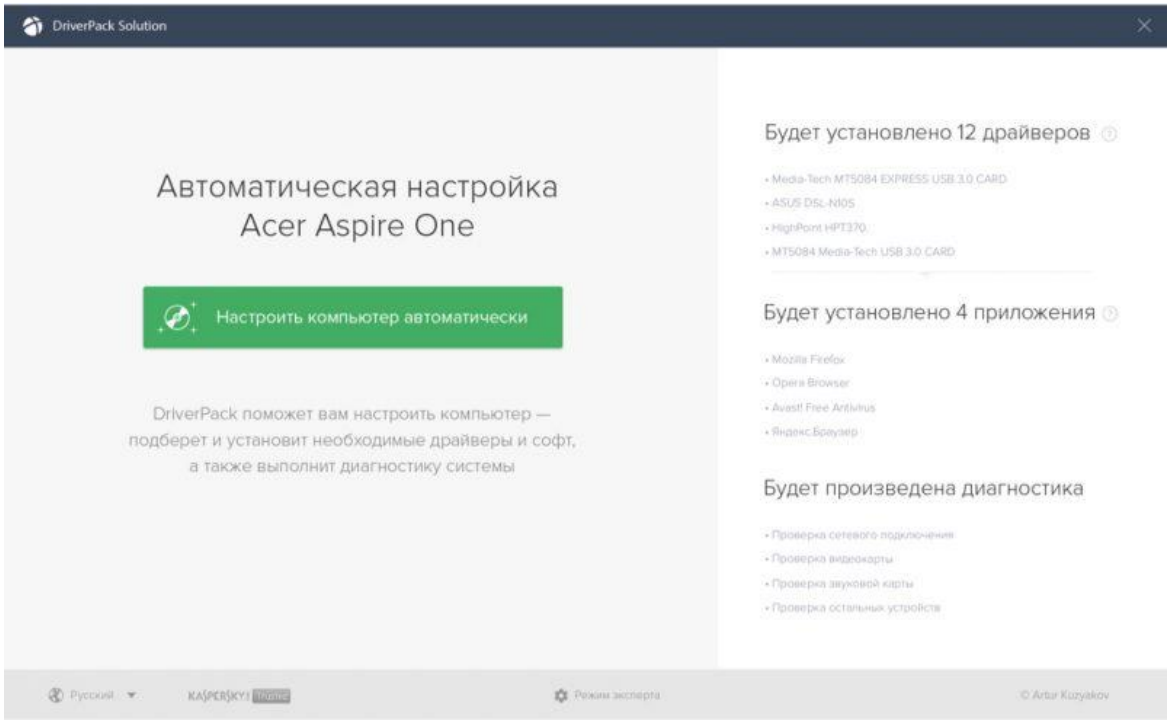
Так выглядит окно Диспетчера устройств в Windows XP

На последнем скриншоте виден жёлтый знак вопроса на пункте «Другие устройства» и знак вопроса с восклицательным знаком на жёлтом фоне на подпункте «Мультимедийное оборудование». Это значит, что в системе не установлены драйверы звуковой карты, и звук не воспроизводится ни в фильмах, ни в играх, ни при попытке воспроизведения музыкальных файлов. Для установки драйвера вручную необходимо раскрыть ветку и для каждого устройства выбрать «Обновить драйвер», после чего указать месторасположение нужного драйвера. Аналогично и с другими устройствами.

ВАЖНО: всегда предпочтительней скачивать «родные» драйверы с официального сайта производителя оборудования. В сети могут попадаться драйверы с вирусами или «битые».

Автоматическое обновление драйверов с помощью специальных программ

DriverPack Solution



DriverPack Solution Самое простое и интуитивно понятное приложение для установки/обновления драйверов, к тому же бесплатное

Менеджер установки драйверов DRP Su, DriverPack Solution, — лучшее программное обеспечение автоматической установки драйверов устройств под Windows. Программа разрабатывается с 2008 года российским программистом Кузяковым и распространяется бесплатно с открытым кодом по лицензии GNU GPL. Возможна установка драйверов без выхода в интернет.

ПО представлено в трёх сборках:

- DriverPack Online: на локальный носитель устанавливается клиент (около 0,47 Мб), драйвера скачиваются из сети.
- DriverPack Offline Network: программа + база данных драйверов LAN/Wi-Fi (около 470 Мб), оптимально при отсутствии доступа к интернету, после установки соединения недостающие драйвера скачиваются из сети.
- DriverPack Offline Full: автономный доступ к полной базе данных драйверов офлайн (порядка 16 Гб).

Возможности программы:

- полностью автоматические распознавание оборудования и установка/переустановка/обновление драйверов;
- автоматический поиск в интернете недостающих драйверов по ID устройства;
- самая полная база актуальных драйверов на все типы устройств;

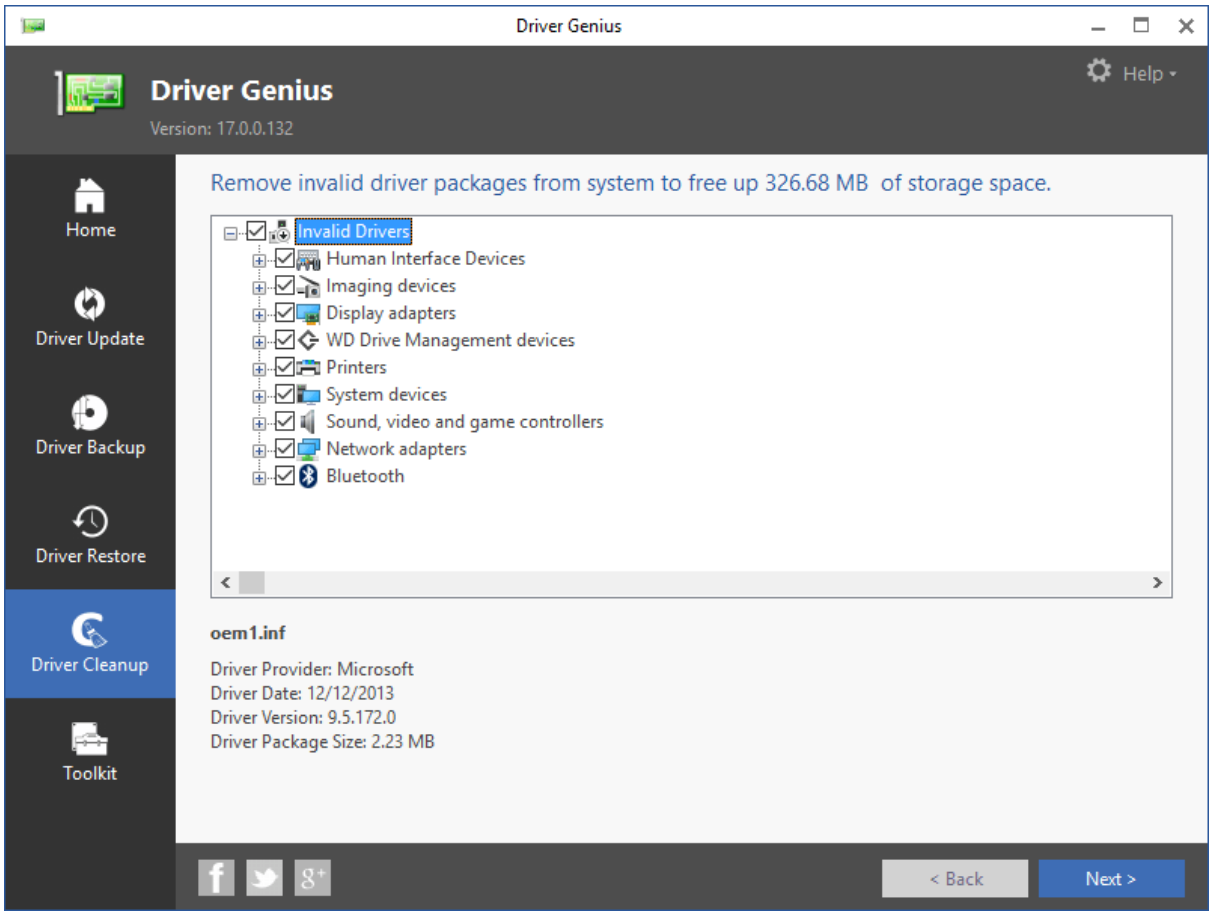
- регулярное обновление базы данных;
- простая интеграция альтернативных пакетов драйверов в базу данных

Недостатки: не обнаружены.

Основные сведения:

- разработчик: DriverPack Solution, Россия;
- языки интерфейса: 36 языковых пакетов, включая русский и английский;
- стоимость: бесплатно;
- количество пользователей: 40+ миллионов;
- операционная система: Windows XP, 7, 8/8.1, 10;

Driver Genius



Приложение Driver Genius — многофункциональный инструмент для работы с драйверами, с возможностью создания точки восстановления, но является платным

Универсальная программа по управлению драйверами. В бесплатной версии отсутствует опция автоматического скачивания найденных драйверов и функция резервного копирования/восстановления.

Возможности программы:

3. поиск устаревших, недостающих и неработающих драйверов;
4. обновление драйверов в один клик (только версия Pro);
5. бэкап/восстановление из резервной копии (только версия Pro);
6. регулярно обновляемая база данных драйверов более 30 тысяч устройств;

7. сканирование по расписанию.

Driver Genius создаёт резервные копии части или всех драйверов, что удобно при восстановлении системы.

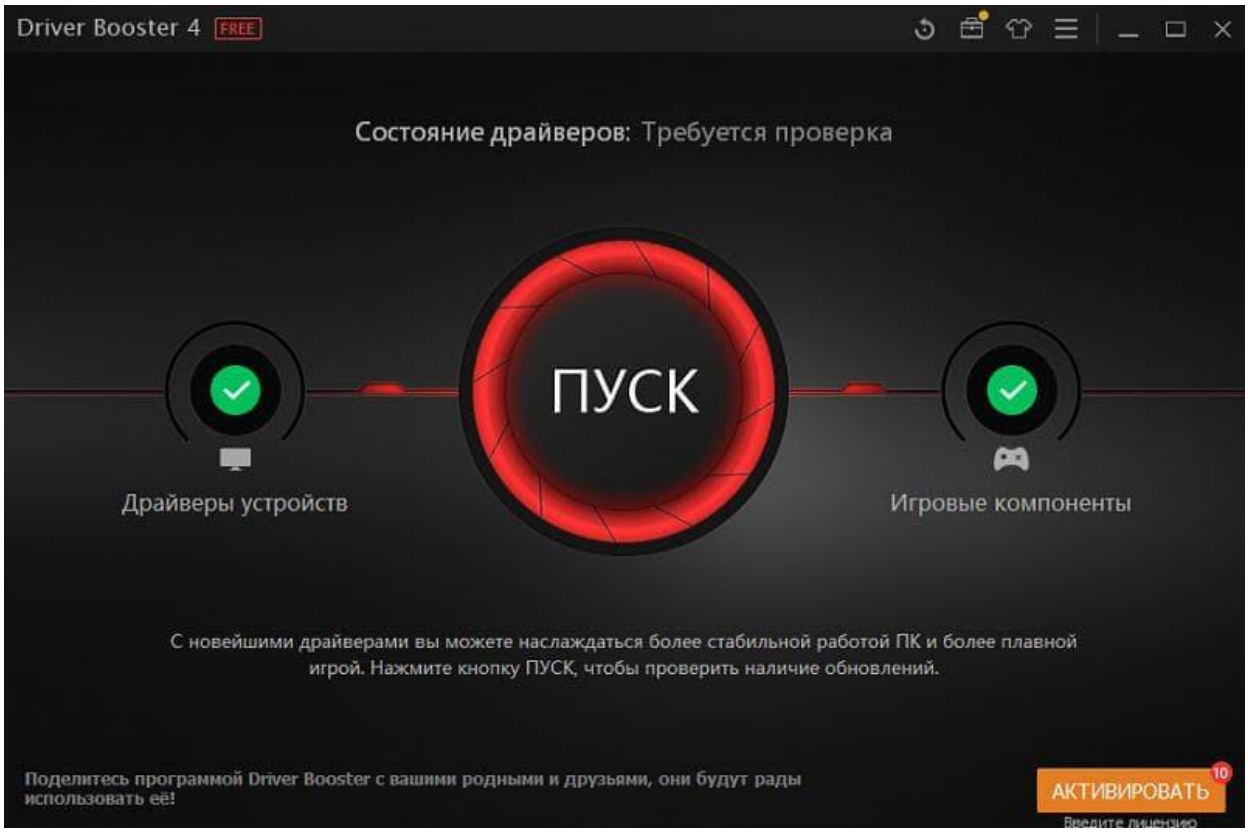
Недостатки:

1. полнофункциональная версия платная, с ежегодным продлением лицензии;
2. программа иногда пропускает устаревшие драйвера.

Основные сведения:

1. разработчик: Driver Soft;
2. стоимость: годовая лицензия версии Pro на 3 ПК — около 23$, на 2 года — +9,9$;
3. операционная система: Windows 2000, XP, Vista, 7, 8/8.1, 10;

Driver Booster



Платное приложение Driver Booster может похвастать рядом интересных технических возможностей

Программа автоматического комплексного обновления драйверов Windows.

Возможности программы:

1. автоматическое определение устаревших, отсутствующих, неработающих драйверов;
2. автоматическое комплексное обновление нуждающихся в апдейте драйверов в один клик (только Pro);
3. тюнинг драйверов для оптимизации производительности в играх (только Pro);

4. создание резервной копии драйверов (только Pro);
5. автоматическое обновление Driver Booster (только Pro).

Недостатки:

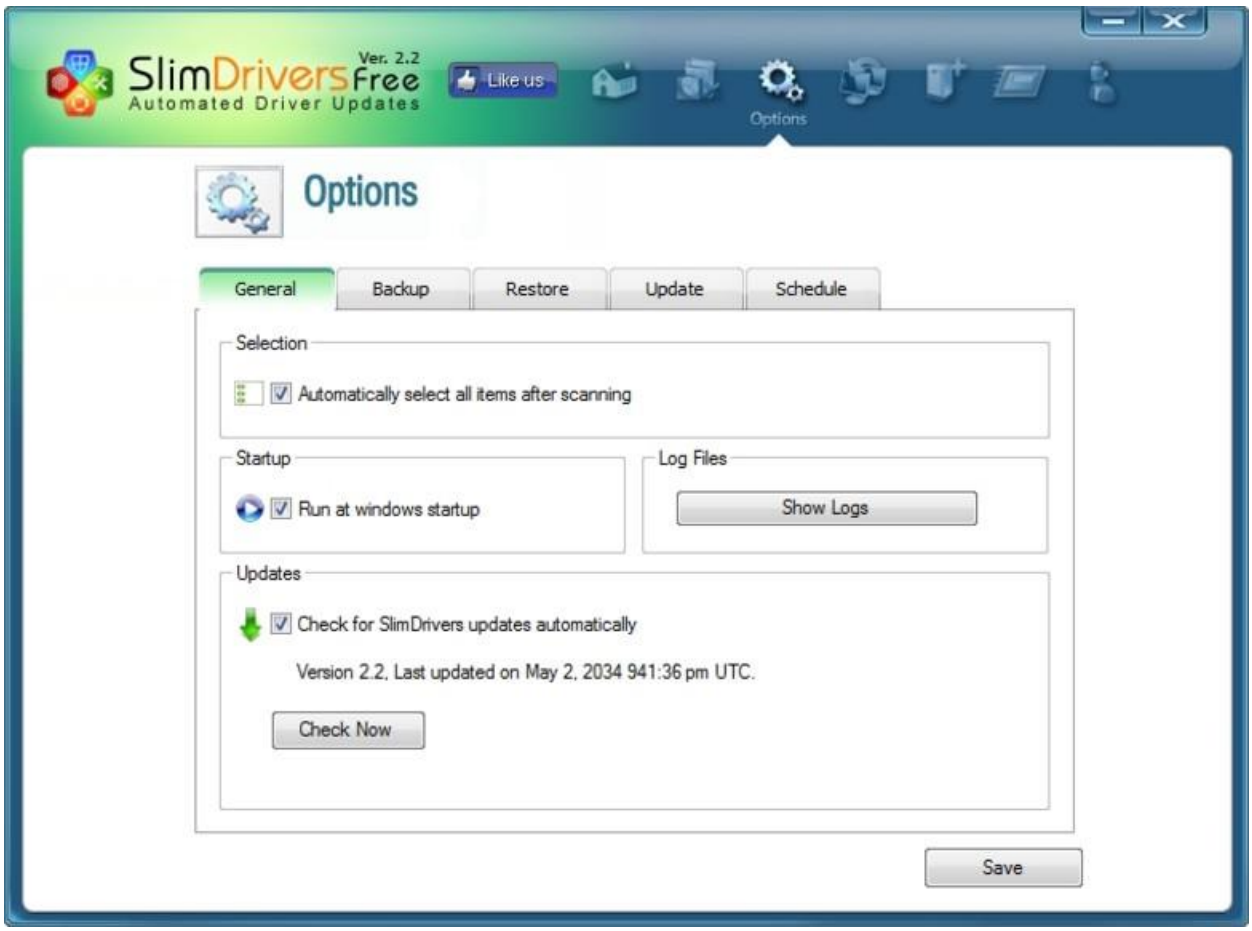
1. полнофункциональная версия платная. Ограниченные возможности бесплатной версии;
2. в процессе инсталляции предлагает дополнительную подписку и устанавливает дополнительное ПО, не связанное с обновлением драйверов (снимайте галочки в соответствующих чекбоксах).

Основные сведения:

1. разработчик: IObit;
2. стоимость: 1500 руб. (1 ПК/1 год, 2018);
3. размер: около 20 Мб;
4. операционная система: Windows XP, Vista, 7, 8/8.1, 10;

Видео: обновление драйверов при помощи Driver Booster

SlimDrivers



Автоматический установщик драйверов SlimDrivers является платным, и стоимость достаточно высока

SlimDrivers — автоматический установщик драйверов. Несколько ограничены возможности базовой платной версии, довольно дорого стоит версия Premium (в 5 раз дороже базовой). На

ПК скачивается и устанавливается клиентская часть программы, поиск нужных драйверов ведётся онлайн.

Возможности программы:

1. автоматические поиск/обновление драйверов;
2. поиск/удаление вредоносного ПО;
3. возможность восстановления последней рабочей конфигурации драйверов;
4. резервное копирование/восстановление из резервной копии.

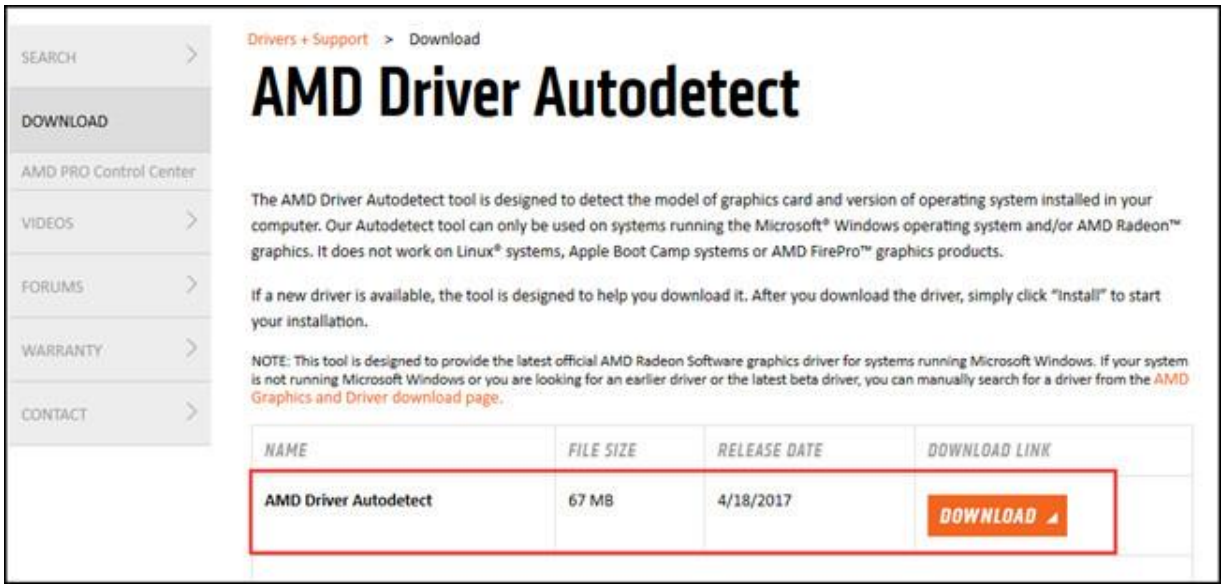
Недостатки:

1. недешёвая полнофункциональная версия Premium.

Основные сведения:

1. разработчик: SlimWare;
2. стоимость: 19,97$ 1 ПК/год или 59,97$ за постоянную лицензию (версия базовая);
3. операционная система: Windows XP и последующие;

AMD Driver Autodetect



Узкоспециализированная утилита AMD Driver Autodetect обновляет только графические драйверы AMD

От описанных выше утилит AMD Driver Autodetect отличается узкой специализацией. Этот установщик автоматически обновляет только драйвера графической подсистемы AMD (AMD Radeon).

Возможности программы:

- обновление драйверов AMD Radeon под Windows до самой последней версии.

Недостатки:

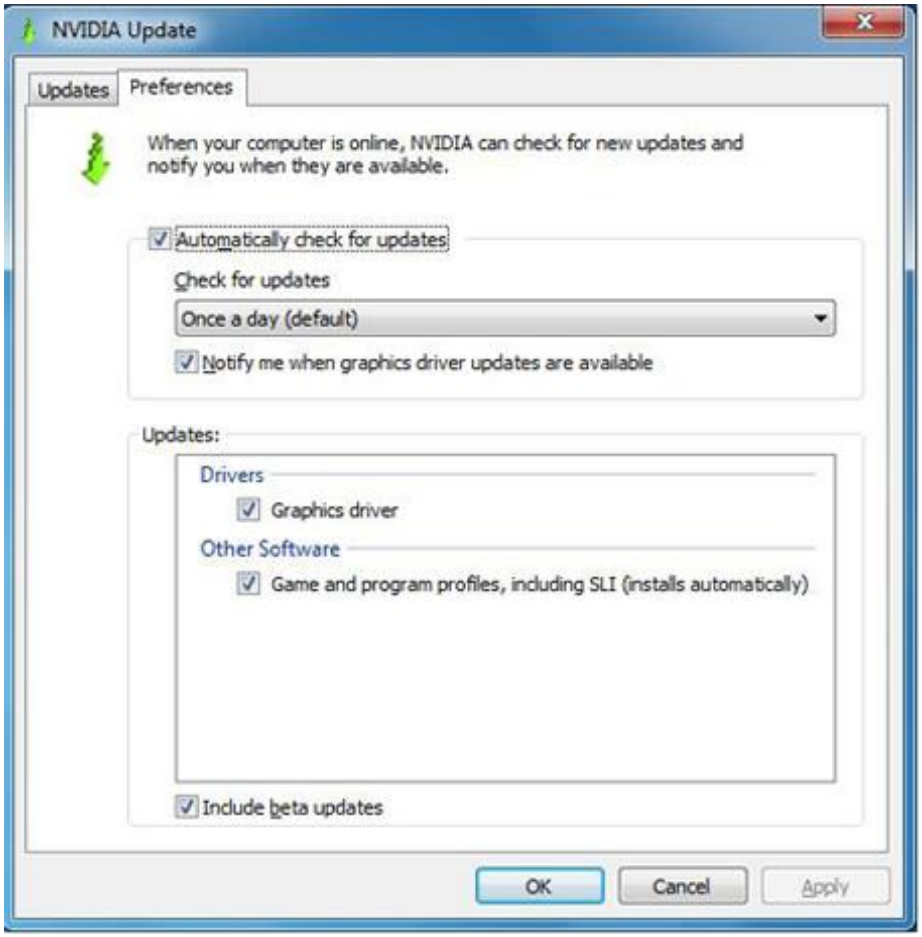
- узкая специализация (только комплектующие AMD);

- автоматическая установка ТОЛЬКО последних драйверов. Если корректно работает только один из старых драйверов, ставить его придётся вручную.

Основные сведения:

1. разработчик: AMD;
2. стоимость: бесплатно;
3. размер: 39 Мб;
4. операционная система: Windows XP и последующие;

NVIDIA Update



NVIDIA Update тоже узкоспециализированная программа, обновляющая графические драйверы устройств семейств GeForce и ION производства NVIDIA

NVIDIA Update — «родная» утилита обновления драйверов от NVIDIA.

Внимание: по состоянию на весну 2018, поддерживаются только GPU семейств GeForce и ION. Остальные видеокарты NVIDIA пока не поддерживаются.

Возможности программы:

- автоматическое обновление драйверов видеокарт NVIDIA GeForce и NVIDIA ION;
- устанавливается частота проверки обновлений;
- автоматическое обновление профилей игр и программ, в том числе SLI-профилей.

Недостатки:

•узкая специализация (только NVIDIA GeForce и ION).

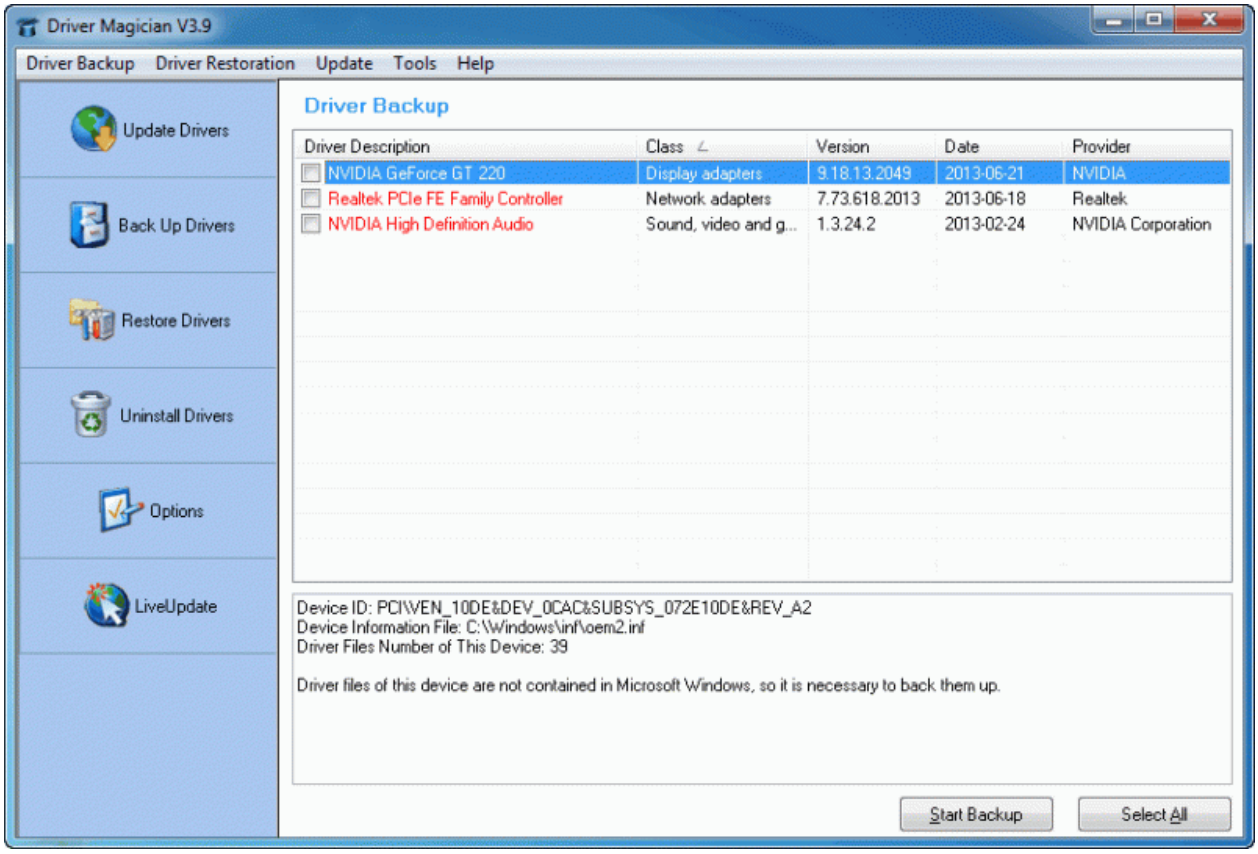
Основные сведения:

•разработчик: NVIDIA;

•стоимость: бесплатно;

•операционная система: Windows XP и более новые ОС семейства Windows;

Driver Magician



Driver Magician — условно-бесплатное приложение, позволяющее, при необходимости, восстановить обновлённые драйверы в один клик

Условно-бесплатный менеджер установки драйверов. Имеющаяся база данных драйверов обновляется вручную или автоматически.

Возможности программы:

•4 режима резервного копирования (драйвера + реестр, содержимое Рабочего стола и папок Мои документы и Избранное);

•автоматическое удаление и обновление драйверов;

•восстановление из резервной копии в один клик;

•упаковка существующих драйверов в установочный исполняемый файл (*.exe).

Основные сведения:

•разработчик: GoldSolution Software;

•тип распространения: условно-бесплатная;

•стоимость: бесплатный триальный период (15 дней), цена версии Pro — 22$;

•размер файла: около 4,5 МБ дистрибутив, около 30 Мб на диске;

•операционная система: Windows XP, 7, Vista, 2003, 8/8.1, 10;

Другие программы

Многочисленные программы автоматического обновления драйверов можно условно разделить на три группы:

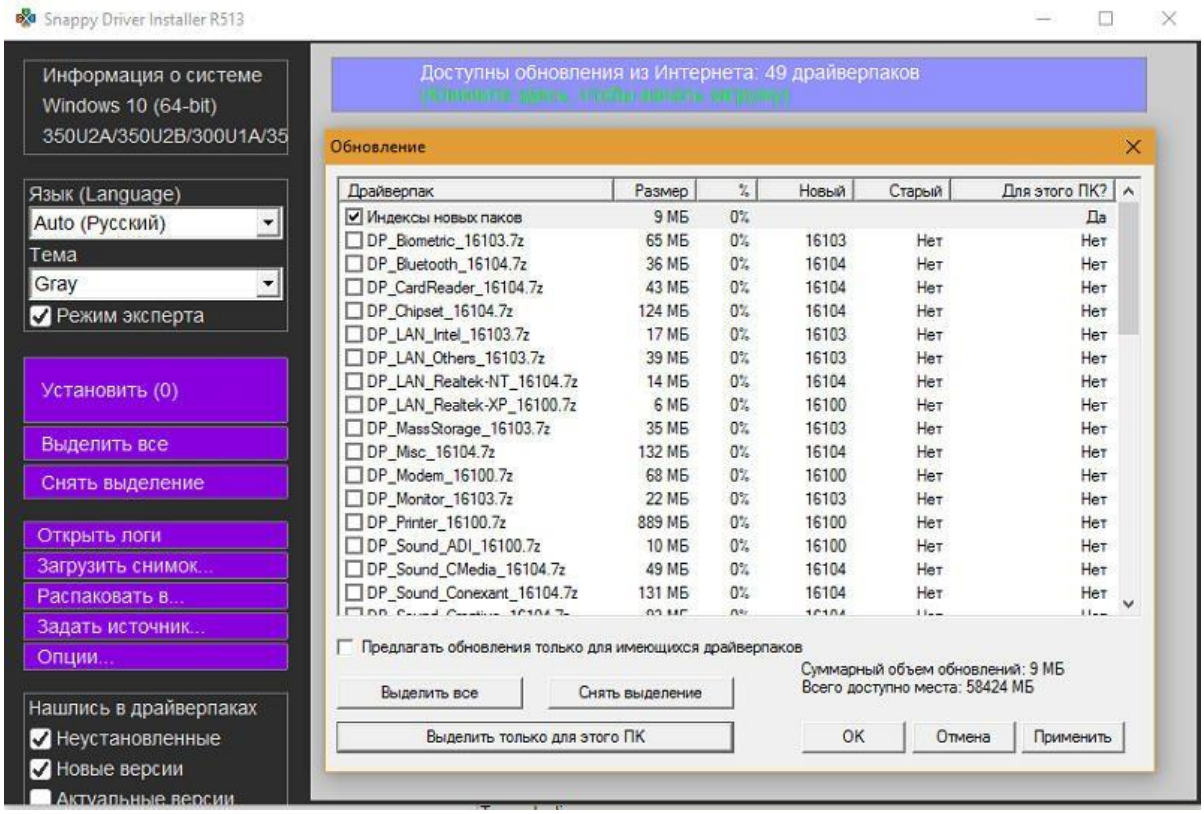
1.Утилиты от производителей компьютерных комплектующих. Отличаются узкой специализацией и рассчитаны на работу с ПО для оборудования конкретного производителя и упоминавшиеся выше AMD Driver Autodetect и NVIDIA Update).

2.Специализированные установщики драйверов, не привязанные к конкретному производителю (большая часть нашего списка).

3.Более универсальные системные утилиты, в том числе умеющие обновлять драйвера.

Утилиты первого типа стоит искать на официальных сайтах конкретных производителей оборудования. Из не вошедших в наш список, но достойных упоминания приложений стоит отметить такие, как Snappy Driver Installer, Driver Checker, Driver Max Free.

Snappy Driver Installer



Snappy Driver Installer можно установить в двух вариантах: для скачивания драйверов из сети, или с встроенной базой данных, содержащей драйверы на несколько сотен тысяч устройств

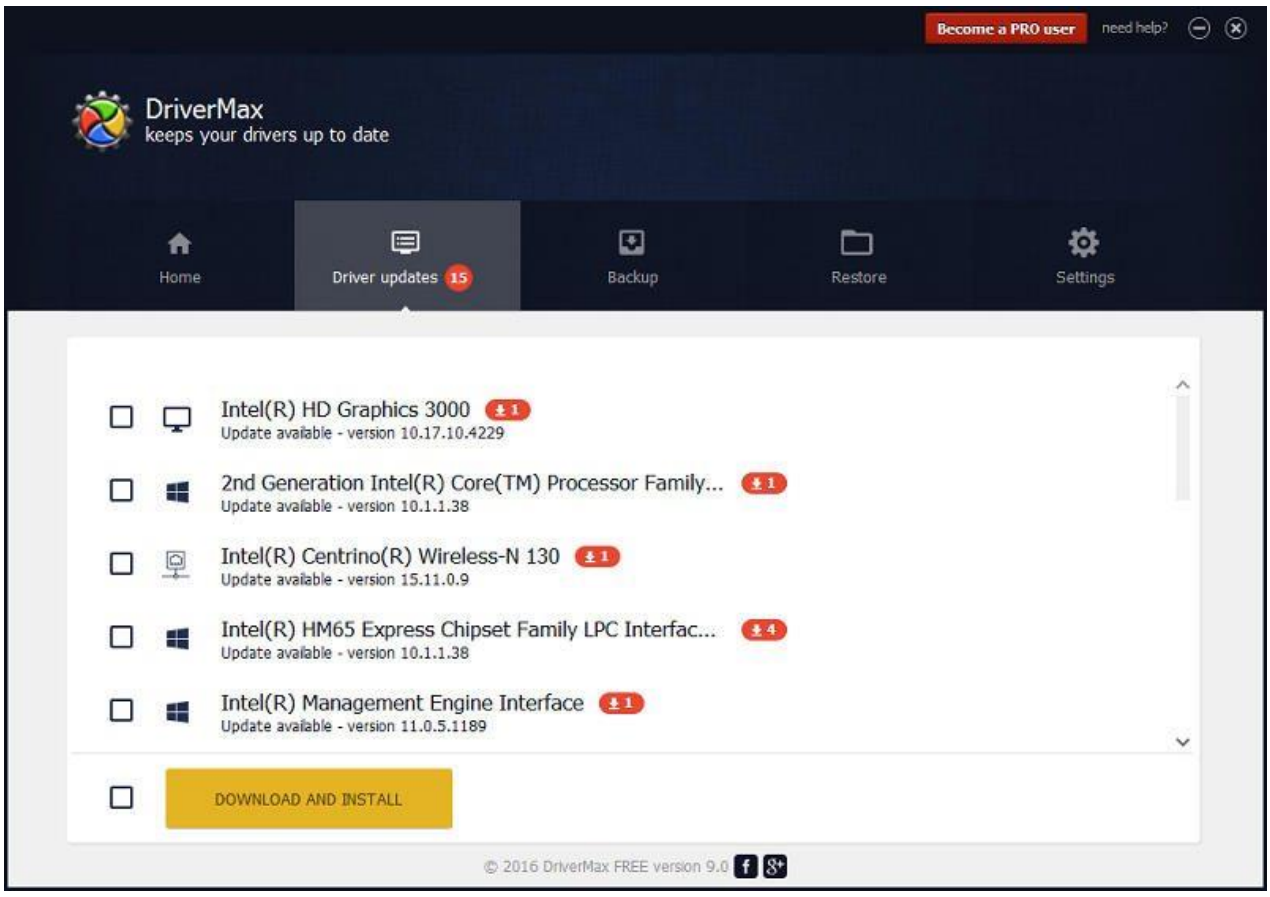
Утилита написана одним из создателей DriverPack Solution и во многом повторяет особенности этого пакета. Предлагается в двух видах: SDI Lite и SDI Full. Первый вариант: клиентский модуль на 3,6 Мб, драйверы берутся из сети. Второй — полнофункциональный пакет для автономного обновления драйверов. Включает в себя БД на 36,1 Гб (драйверы на несколько сотен тысяч устройств). Не нуждается в инсталляции, работает с USB, DVD/CD.

Driver Checker

Driver Checker — недешёвая утилита без поддержки русского языка

Платная утилита (от порядка 36$ за копию). Помимо прочего, умеет экспортировать драйверы по локальной сети. Недостатки: отсутствие русской локализации и работоспособной триальной версии.

DriverMax Free



DriverMax Free — платная утилита, с возможностью создания точки восстановления

Бесплатная утилита не поражает функциональными возможностями, однако с заявленной задачей (обновлением и резервным копированием/восстановлением драйверов) справляется отлично. Производит все манипуляции с драйверами после автоматического создания точки восстановления системы. Полностью автоматическая версия — платная, от 10,35$ за копию.

Практическое задание к работе.

3. Установить под виртуальной машиной в любой из установленных операционных систем обновления на ранее установленную ПО внутри виртуальной ОС.
4. Установите автообновления установленных ранее программ внутри виртуальной ОС.
5. Измените тип обновления виртуальных операционных систем на «Предупреждать пользователя, но не загружать автоматически».
6. Измените следующие параметры для каждой ОС: уменьшите дисковое пространство на 1 Гб, увеличьте использование памяти на 25% сверх первоначально установленной для каждой ОС, измените файловые системы установленных в виртуальной машине ОС. Опишите, как повлияли изменения на поведение компьютера в целом и на каждую виртуальную ОС.
7. Подключите в хостовой ОС выданное преподавателем оборудование и продублируйте попытки в виртуальных ОС.

В случае невозможности присутствия студента на занятиях обучающийся выполняет пункт №5 с использованием собственного оборудования.

Лабораторная работа №14. Настройка управления питанием. Оптимизация использования процессора.

Теоретический материал.

Пункт меню BIOS	Описание
ACPI Function	Активация режима расширенного управления питанием ACPI (Advanced Configuration and Power Interface), который: автоматически выключает компьютер после завершения работы ОС; переводит компьютер в спящий режим; выводит компьютер из спящего режима; экономит заряд батарей у ноутбуков. Возможные значения:

	Enabled - режим ACPI включен; Disabled - режим ACPI отключен. При включенном режиме ACPI компьютер может находиться в следующих состояниях: **Normal** - обычное состояние компьютера; **Doze** - пониженное энергопотребление; **Standby** - ждущий режим, при котором отключены некоторые устройства; **Suspend** - спящий режим, при котором отключаются все устройства за исключением тех, которые выводят компьютер из спящего режима.
ACPI Suspend Type	Определение спящего режима компьютера. Возможные значения: S1 - Power of Suspend - стандартный режим; S3 - Suspend to RAM - состояние компьютера перед "засыпанием" сохраняется в оперативной памяти компьютера, из которой потом восстанавливается при "просыпании".
Automatic Power Up	Время автоматического включения компьютера. Возможные значения: Everyday - в поле Time Alarm указывается время ежедневного автоматического включения компьютера; By Date - указывается время и дата автоматического включения компьютера; Disabled - не производится автоматическое включение компьютера.
CPUFAN Off In Suspend	Активация выключения кулера (вентилятора) процессора во время режима энергосбережения Suspend. Возможные значения: Enabled - напряжение во время Suspend на кулер не подается; Disabled - напряжение подается всегда.
Date (of Month)	Дата автоматического включения компьютера при Automatic Power Up = By Date. Если установить значение 0, то будет производиться ежедневное автоматическое включение компьютера.
FDC/LPT/COM Ports	Активация режима слежения за состоянием дисковода, параллельных и последовательных портов с целью недопущения перехода компьютера в спящий режим, если эти устройства задействованы. Режим - Monitor.
Hard Disk Timeout	Время, по прошествии которого при отсутствии обращений к жесткому диску тот будет отключен. Рекомендуется не задействовать эту опцию.
IRQs Activity Monitoring	Указываются номера аппаратных прерываний состояние которых следует мониторить на предмет недопущения перехода компьютера в режим пониженного

	энергосбережения, если эти прерывания задействованы. Возможные значения: Enabled - мониторинг производится; Disabled - мониторинг не производится.
Keyboard Power On	Активация режма включения компьютера из спящего состояния нажатием любой клавиши. Возможные значения: Enabled - режим включен; Disabled - отключен.
PM Control By APM	Активация режима автоматического управления питанием со стороны операционной системы: Возможные значения: Enabled - режим включен; Disabled - отключен.
Power Button Function	Реакция компьютера на нажитие кнопки Power. Возможные значения: On/Off - обычный режим кнопки Power, отключающий компьютер; Suspend - режим перевода компьютера в спящий режим по нажатию кнопки Power.
Power Button Over Ride	Реакция компьютера на нажитие кнопки Power. Возможные значения: Enabled - завершение работы компьютера возможно только программными средствами (меню Пуск - Выключить компьютер); Disabled - обычный режим кнопки Power, отключающий компьютер.
Power Management	Активация режима слежения за действиями пользователя на предмет перехода компьютера в энергосберегающий режим, если никаких действий не производится. Возможные значения: Min Saving - компьютер переходит в режим энергосбережения через 0,5..2 часа; Max Saving - компьютер переходит в режим энергосбережения через 0,5..1 мин; User Defined - время задается вручную; Disabled - автоматический переход в энергосберегающий режим запрещен.
Power On By Alarm	Активация возможности автоматического включения компьютера в заданное время. Возможные значения: Enabled - режим включен; Disabled - отключен.
Power On By Modem/Lan	Активация режима включения компьютера от поступления

	сигнала на модем или локальную сеть. Возможные значения: Enabled - режим включен; Disabled - отключен (рекомендуется).
Power On By PCI Card	Активация режима включения компьютера от поступления сигнала на шину PCI. Возможные значения: Enabled - режим включен; Disabled - отключен (рекомендуется).
Primary INTR	Аналогично IRQs Activity Monitoring.
Primary Master IDE Primary Slave IDE	Активация режима слежения за состоянием первого канала контроллера IDE с целью недопущения перехода компьютера в спящий или ждущий режим, если этот канал задействован. Режим - Monitor.
PS/2 Mouse Power On	Активация возможности включения компьютера при помощи мыши, подключенной к PS/2. Возможные значения: Enabled - режим включен; Disabled - отключен.
Mouse PowerOn Function	Активация возможности включения компьютера при помощи мыши. Возможные значения: Enabled - режим включен; Disabled - отключен.
PWR Button < 4 Secs	Настройка реакции кнопки Power на кратковременное нажатие (менее 4 секунд). Возможные значения: Soft Off - программное завершение работы компьютера; Suspend - переход компьютера в спящий режим; Suspend - нет никакой реакции.
PWR Up PS2 KB/Mouse	Объединяет опции Keyboard Power On и PS/2 Mouse Power On.
RTC Alarm Hour	Час автоматического включения компьютера при Automatic Power Up = By Date.
RTC Alarm Minute	Минута автоматического включения компьютера при Automatic Power Up = By Date.
RTC Alarm Second	Секунда автоматического включения компьютера при Automatic Power Up = By Date.
Secondary Master IDE Secondary Slave IDE	Активация режима слежения за состоянием второго канала контроллера IDE с целью недопущения перехода компьютера в спящий или ждущий режим, если этот канал задействован. Режим - Monitor.
Sleep State LED	Поведение индикатора LED (индикатор питания на блоке

	компьютера). Возможные значения: Blinking - режим мигания во время "сна" компьютера; On - режим свечения во время "сна" компьютера; Off/Dual - во время "сна" компьютера индикатор не светится;
Soft Off By PWR BTTN	Аналогично PWR Button < 4 Secs.
Specific Key for Power On	Задание клавиши по которой будет происходить включение компьютера при выходе из энергосберегающего режима.
Standby Mode	Время, по истечении которого (при отстутствии каких-либо действий пользователя) компьютер переходит в режим Standby.
State After Power Failure	Реакция компьютера на внезапное отключение питания после восстановления питания. Возможные значения: Off - компьютер остается в выключенном состоянии; On - компьютер включается; Auto - компьютер возвращается в то состояние, в котором он был до отключения питания.
Suspend Mode	Аналогично Standby Mode.
Suspend Time Out (Minute)	Аналогично.
Suspend to RAM Capability	Разрешение сохранения в оперативной памяти текущего состояния операционной системы перед переходом компьютера в режим энергосбережения. Возможные значения: Enabled - сохранение разрешено; Disabled - запрещено.
Wake on Lan	Аналогично Power On By Modem/Lan, но только для сети.
Wake Up By USB device	Аналогично, но для шины USB.
Wake Up By Keyboard	Клавиша, выводящая компьютер из спящего режима. Возможные значения: Enabled - компьютер "просыпается" по нажатию любой клавиши; Какая-либо клавиша - вывод компьютера из спящего режима по нажатию указанной клавиши; Disabled - вывод компьютера из спящего режима запрещен.
Make Up By Mouse	Выход компьютера из спящего режима при манипуляции мышью. Возможные значения: Enabled - режим включен; Disabled - отключен.
Wake Up Events PCI	Выход компьютера из спящего режима по активности PCI-

Master	устройств. Возможные значения: Enabled - режим включен; Disabled - отключен.
Wake Up Events VGA	Выход компьютера из спящего режима при активности видеокарты. Возможные значения: Enabled - режим включен; Disabled - отключен.

Оптимизация использования процессора

Способы оптимизации и ускорения работы процессора

Все манипуляции по улучшению качества работы ЦП можно поделить на две группы:

•Оптимизация. Основной акцент делается на грамотное распределение уже доступных ресурсов ядер и системы, дабы добиться максимальной производительности. В ходе оптимизации трудно нанести серьёзный вред ЦП, но и прирост производительности, как правило, не очень высокий.

•Разгон. Манипуляции непосредственно с самим процессором через специальное ПО или BIOS для повышения его тактовой частоты. Прирост производительности в этом случае получается весьма ощутимым, но и возрастает риск повредить процессор и другие компоненты компьютера в ходе неудачного разгона.

Узнаём, пригоден ли процессор для разгона

Перед разгоном обязательно просмотрите характеристики своего процессора при помощи специальной программы (например AIDA64). Последняя носит условно-бесплатный характер, с её помощью можно узнать подробную информацию обо всех компонентах компьютера, а в платной версии даже проводить с ними некоторые манипуляции. Инструкция по использованию:

1.Чтобы узнать температуру ядер процессора (это один из главных факторов при разгоне), в левой части выберите пункт “**Компьютер”**, затем перейдите в пункт “**Датчики”** из главного окна или меню пунктов.

2.Здесь вы сможете просмотреть температуру каждого ядра процессора и общую температуру. На ноутбуке, при работе без особых нагрузок она не должна превышать 60 градусов, если она равна или даже немного превышает этот показатель, то от разгона лучше отказаться. На стационарных ПК оптимальная температура может колебаться в районе 65-70 градусов.

Если всё нормально, то перейдите в пункт “**Разгон”**. В поле “**Частота ЦП”** будет указано оптимальное число МГц при разгоне, а также процент, на который рекомендуется увеличить мощность (обычно колеблется в районе 15-25%).

Способ 1: оптимизация при помощи CPU Control

Чтобы безопасно оптимизировать работу процессора, потребуется скачать CPU Control. Данная программа имеет простой интерфейс для обычных пользователей ПК, поддерживает русский язык и распространяется бесплатно. Суть данного способа заключается в равномерном распределении нагрузки на ядра процессора, т.к. на современных многоядерных процессорах, некоторые ядра могут не участвовать в работе, что влечёт потерю производительности.

Скачать CPU Control

Инструкция по использованию данной программы:

1.После установки откроется главная страница. Изначально всё может быть на английском. Чтобы это исправить, перейдите в настройки (кнопка “**Options”** в правой нижней части окошка) и там в разделе“**Language”** отметьте русский язык.

2.На главной странице программы, в правой части, выберите режим “**Ручной”**.

В окне с процессорами выберите один или несколько процессов. Чтобы сделать выбор нескольких процессов, зажмите клавишу **Ctrl** и щёлкайте мышкой по нужным элементам.

3.Затем нажмите правую кнопку мыши и в выпавшем меню выберите ядро, которое вы бы хотели назначить для поддержания той или иной задачи. Ядра носят названия по следующему типу CPU 1, CPU 2 и т.д. Таким образом можно “поиграться” с производительностью, при этом шанс что-либо сильно испортить в системе минимален.

Если вы не хотите назначать процессы вручную, то можно оставить режим “**Авто”**, который стоит по умолчанию.

4.После закрытия программа автоматически сохранит настройки, которые будут применятся при каждом запуске ОС.

Способ 2: разгон при помощи ClockGen

ClockGen — это бесплатная программа, подходящая для ускорения работы процессоров любой марки и серии (за исключением некоторых процессоров Intel, где разгон невозможен сам по себе). Перед разгоном убедитесь, что все температурные показатели ЦП в норме. Как пользоваться ClockGen:

1.В главном окне перейдите во вкладку **«PLL Control»**, где при помощи ползунков можно изменить частоту процессора и работы оперативной памяти. Не рекомендуется за раз слишком сильно передвигать ползунки, лучше небольшими шагами, т.к. слишком резкие изменения могут сильно нарушить работу ЦП и ОЗУ.

2.Когда получите необходимый результат, нажмите на **«Apply Selection»**.

3.Чтобы при перезапуске системы настройки не сбивались, в главном окне программы перейдите в пункт**«Options»**. Там, в разделе **«Profiles Management»**, поставьте флажок напротив **«Apply current settings at startup»**.

Способ 3: разгон процессора в BIOS

Довольно сложный и “опасный” способ, особенно для неопытных пользователей ПК. Перед разгоном процессора рекомендуется изучить его характеристики, в первую очередь, температуру при работе в штатном режиме (без серьёзных нагрузок). Для этого воспользуйтесь специальными утилитами или программами (описанная выше AIDA64 вполне подойдет для этих целей).

Если все параметры в норме, то можно приступать к разгону. Разгон для каждого процессора может быть разным, поэтому ниже представлена универсальная инструкция проведения данной операции через BIOS:

1.Произведите вход в BIOS при помощи клавиши **Del** или клавиш от **F2** до **F12** (зависит от версии БИОСа, материнской платы).

2.В меню BIOS найдите раздел с одним из таких наименований (зависит от вашей версии БИОСа и модели материнской платы) – “**MB Intelligent Tweaker”**, “**M.I.B, Quantum BIOS”**, “**Ai Tweaker”**.

3.Теперь вы можете видеть данные о процессоре и вносить некоторые изменения. Перемещаться по меню можно при помощи клавиш со стрелочками. Переместитесь до пункта “**CPU Host Clock Control”**, нажмите**Enter** и поменяйте значение с “**Auto”** на “**Manual”**, чтобы можно было самостоятельно изменять настройки частоты.

4.Спуститесь на пункт ниже к “**CPU Frequency”**. Чтобы внести изменения, нажмите **Enter**. Далее в поле “**Key in a DEC number”** введите значение в диапазоне от того, что написано в поле “**Min”** до “**Max”**. Не рекомендуется применять сразу максимальное значение. Лучше наращивать мощности постепенно, дабы не нарушить работу процессора и всей системы. Для применения изменений нажмите **Enter**.

5.Чтобы сохранить все изменения в БИОСе и выйти, найдите пункт в меню “**Save & Exit”** или несколько раз нажмите на **Esc**. В последнем случае система сама спросит, требуется ли сохранять изменения.

Способ 4: оптимизация работы ОС

Это самый безопасный метод увеличения производительности ЦП путём очистки автозагрузки от ненужных приложений и дефрагментации дисков. Автозагрузка – это автоматическое включение той или иной программы/процесса при загрузке операционной системы. Когда в этом разделе скапливается слишком много процессов и программ, то при включении ОС и дальнейшей работе в ней, на центральный процессор может быть оказана слишком высокая нагрузка, что нарушит производительность.

Очистка Автозагрузки

В автозагрузку приложения можно добавлять как самостоятельно, так и приложения/процессы могут добавляться сами. Чтобы второго случая не было, рекомендуется внимательно читать все пункты, которые отмечены галочкой во время установки того или иного софта. Как убрать уже имеющиеся элементы из Автозагрузки:

1.Для начала перейдите в “**Диспетчер задач”**. Чтобы перейти туда, используйте комбинацию клавиш**Ctrl+SHIFT+ESC** или в поиске по системе вбейте “**Диспетчер задач”** (последнее актуально для пользователей на Windows 10).

2.Перейдите в окно “**Автозагрузка”**. Там будут представлены все приложения/процессы, которые запускаются вместе с системой, их состояние (включено/отключено) и общее влияние на производительность (Нет, низкое, среднее, высокое). Что примечательно – здесь вы можете отключить все процессы, при этом не нарушите работу ОС. Однако, отключив некоторые приложения, вы можете сделать работу с компьютером немного неудобной для себя.

В первую очередь, рекомендуется отключать все пункты, где в колонке “**Степень влияния на производительность”** стоят отметки “**Высокое”**. Чтобы отключить процесс, кликните по нему и в правой нижней части окна выберите “**Отключить”**.

Чтобы изменения вошли в силу рекомендуется выполнить перезагрузку компьютера.

Проведение дефрагментации

Дефрагментация диска увеличивает не только скорость работы программ на этом диске, но также немного оптимизирует работу процессора. Происходит это потому, что ЦП обрабатывает меньше данных, т.к. в ходе дефрагментации обновляется и оптимизируется логическая структура томов, ускоряется обработка файлов. Инструкция проведения дефрагментации:

1.Нажмите правой кнопкой мыши по системному диску (вероятнее всего, это **(C:)**) и перейдите в пункт“**Свойства”**.
2.В верхней части окна найдите и перейдите во вкладку “**Сервис”**. В разделе “**Оптимизация и дефрагментация диска”** нажмите “**Оптимизировать”**.

3.В открывшемся окне можно выбрать сразу несколько дисков. Перед дефрагментацией рекомендуется провести анализ дисков, нажав на соответствующую кнопку. Анализ может идти до нескольких часов, в это время не рекомендуется запускать программы, которые могут вносить какие-либо изменения на диске.

4.После анализа система напишет, требуется ли дефрагментация. Если да, то выделите нужный диск (диски) и нажмите на кнопку “**Оптимизировать”**.

5.Рекомендуется также назначить автоматическую дефрагментацию дисков. Для этого перейдите по кнопке“**Изменить параметры”**, далее отметьте галочкой “**Выполнять по расписанию”** и задайте нужное расписание в поле “**Частота”**.

Оптимизировать работу ЦП не так сложно, как кажется на первый взгляд. Однако, если оптимизация не дала сколь-нибудь заметных результатов, то в этом случае центральный процессор потребуется разогнать самостоятельно. В некоторых случаях разгон не обязательно производить через БИОС. Иногда производитель процессора может предоставить специальную программу для увеличения частоты той или иной модели.

Практическое задание к работе.

1. Зайдите в BIOS и найдите настройки, которые относятся к управлению питанием. Составьте два списка: а) настроек, которые редактировать нельзя; б) настройки, которые поддаются редактированию.
2. Поясните пункты списка, на что они влияют и что они означают.
3. Установите количество ядер процессора, необходимое в разное время работы ПК.
4. Установите действия при нажатии кнопок Power, Sleep, WakeUp (через BIOS или Windows).
5. Установите автоматическое выключение дисков через 1 час и монитора через 20 минут и сохраните схему энергосбережения под собственным именем.
6. Установите авторегулировку частоты вращения кулера на процессоре в зависимости от загрузки процессора.
7. Проверьте наличие опции обязательного отключения процессора при чрезмерной нагрузке или фатальной системной ошибке.
8. Настройте автоотключение ПК ровно в 19-00.

Лабораторная работа №15. Оптимизация использования памяти. Оптимизация использования жесткого диска. Оптимизация использования сети. Инструменты повышения производительности программного обеспечения.

Теоретический материал.

Оптимизация оперативной памяти Windows
Самыми сильными качествами оперативной памяти являются ее высокие скорости чтения и записи информации. К сожалению, это достигается только за счет физических свойств и незначительности размеров модулей памяти.

Тем не менее, если вы будете «правильно» записывать данные в память и удалять оттуда ненужную информацию, ничто не сможет помешать вам насладиться высокими скоростями работы оперативной памяти:

С помощью утилиты Dataram RAMDisk вы можете использовать часть вашей оперативной памяти в качестве жесткого диска.

В нашей пошаговой инструкции ниже «**Настраиваем бесплатный RAMDisk**» мы показываем, как вы можете создать и настроить небольшого размера флеш-диск в

оперативной памяти. О том, как вы можете использовать такой диск, например, для хранения кэша браузера Firefox, мы расскажем в нижеприведенной инструкции «**Размещаем в RAM-диске кэш браузера**».

С помощью бесплатной программы CleanMEM вы можете автоматизированно или вручную удалять более ненужные данные из вашей оперативной памяти.

Увеличиваем размеры оперативной памяти

Если ваш компьютер, несмотря на все усилия по оптимизации, продолжает оставаться медлительным, вы можете предпринять еще кое-что, чтобы повысить производительность системы. Для этого вам потребуется заменить уже установленные модули памяти или добавить к ним новые.

- В случае с ноутбуками замена отдельных компонент может оказаться несколько более сложной. Однако, именно к жестким дискам и модулям памяти, как правило, вы можете получить доступ через специальные «сервисные крышки», открывающиеся без необходимости в полной разборке корпуса.

- Обратите внимание на условия гарантии на ваш компьютер — не потеряется ли она при вскрытии корпуса. В том случае, если у вас есть сомнения, для установки компонент или их замены лучше будет привлечь специалиста.

Размещаем в RAM-диске кэш браузера:

- После того, как вы с помощью утилиты Dataram RAMDisk выделите часть вашей оперативной памяти под использование в качестве «жесткого диска», введите в адресной строке браузера Firefox команду «about:config» и подтвердите, что ознакомились с соответствующим предупреждением

Кликните правой кнопкой мыши и создайте новую строку. Введите в соответствующем поле открывшегося окна строку «browser.cache.disk.parent_directory».

- Выберите букву, соответствующую названию созданного вами RAM-диска.

- И в других программах тоже укажите созданный вами RAM-диск в качестве хранилища для временных данных, кэша и скачиваемых файлов. Прежде всего в случае с программами для обработки видео и графики, временными интернет-файлами и вычислительными приложениями высокая скорость чтения и записи оперативной памяти будет особенно заметной. Но будьте внимательны: при выключении ПК или его перезагрузке, данные, хранящиеся на таком диске, будут потеряны. Храните на нем только временные и легко восстанавливаемые файлы.

Настраиваем бесплатный RAMDisk

Бесплатная утилита

С помощью Starwind RAMDisk вы можете бесплатно настроить под Windows один или несколько RAM-дисков. Единственное ограничение: размеры каждого такого RAM-диска не могут превышать 1 Гбайт.

Может понадобиться драйвер

Windows должна думать, что в случае с RAM-диском речь идет как бы о жестком диске. Этого программное обеспечение добивается с помощью нового драйвера.

Обзор RAM-дисков

Windows может работать даже с несколькими RAM-дисками. Утилита представит вам обзор всех имеющихся в системе RAM-дисков.Добавляем RAM-диск

Для добавления и настройки нового RAM-диска кликните на пункт «Add Device».

При настройке поможет ассистент

Сразу же вам на помощь приходит «ассистент», который поможет пройти самые важные шаги.

Ограничение в 1 Гбайт

В случае со свободно распространяемой версией программы размеры RAM-диска не могут превышать 1 Гбайт. Тем не менее, в большинстве случаев этого должно быть достаточно. Для разных действий вы можете использовать разные RAM-диски. Важно: обязательно поставьте

галочку перед «Automount this Device», чтобы при каждой перезагрузке системы RAM-диск появлялся в «Проводнике» автоматически.

Вот и все. Создание RAM-диска занимает не больше минуты.

Новый жесткий диск

Новый «жесткий диск» сразу же появится в обзоре Starwind RAMDisk.

Привод в «Проводнике»

Пользователи операционной системы Windows могут заполнить диск с помощью «Проводника».

Оптимизация использования жесткого диска

Систематизация файлов

Чтобы на диске всегда был порядок, вы знали где и какие файлы у вас находятся и сколько они занимают места, приучите себя к их правильному размещению.

Не храните файлы и папки на рабочем столе, он предназначен в основном для ярлыков. Сохраняйте все файлы сразу же в нужные папки с интуитивными именами. Группируйте файлы одной и той же тематики в одну папку. При скачивании новой версии файла или программы удаляйте старые версии, чтобы они не дублировались и не занимали двойной объем.

В общем старайтесь себя дисциплинировать, не скидывайте файлы куда попало и не оставляете их разбор на потом. Иначе вы постоянно будете сталкиваться с проблемами поиска файлов и в конце концов нехваткой места на диске, какого бы объема он не был.

Ускорение работы диска

Кроме свободного места на диске есть еще несколько важных факторов, влияющих на скорость и стабильность его работы.

Обновление драйвера контроллера дисков

От драйвера контроллера дисков зависит не только скорость его работы, но и стабильность всей системы. Я рекомендую обновить драйвер контроллера дисков до последней версии. Это особенно важно если у вас SSD диск. Также обратите на это пристальное внимание если у вас установлена Windows 10, где устаревший драйвер SATA контроллера может приводить к фризам (подвисаниям изображения на несколько секунд).

Для того, чтобы проверить текущую версию установленного у вас драйвера, зайдите в диспетчер устройств. Для этого нажмите сочетание клавиш «Win+R», введите

«devmgmt.msc» и нажмите «Enter». Кликните правой кнопкой мыши на SATA контроллере и выберите «Свойства».

Если у вас установлен старый драйвер от Microsoft, то найдите новый драйвер на сайте производителя вашей материнской платы или ноутбука и установите его.

Также драйвер можно обновить с помощью одной из специальных утилит для обновления драйверов. Я рекомендую утилиту «Driver Booster».

Включение функции TRIM на SSD

Если у вас еще нет SSD диска и ваш компьютер работает не так быстро как хотелось, то установите его и вы получите значительный прирост быстродействия и отзывчивости системы!

Все современные диски SSD поддерживают функцию TRIM, которая предназначена для оптимизации их скорости. Эта функция обязательно должна быть включена в операционной системе.

Windows XP и Vista не поддерживают функцию TRIM. Как решить эту проблему мы рассмотрим чуть ниже.

Для того, чтобы проверить включена ли функция TRIM в Windows 7, 8.1, 10, скачайте файл «Проверка TRIM» в разделе «Ссылки» и запустите его от имени Администратора.

Если функция TRIM включена, то вы должны увидеть строку «DisableDeleteNotify=0».

Если вы увидите «DisableDeleteNotify=1», значит функция TRIM не активна и ее нужно включить с помощью файла «Включение TRIM», который также нужно скачать в разделе «Ссылки» и запустить от имени Администратора.

Но это еще не все, то что функция TRIM включена в операционной системе не говорит о том, что она реально работает. Для того, чтобы наверняка убедиться выполняется ли функция TRIM на вашем SSD диске, воспользуйтесь утилитой «TRIMcheck», которую можно скачать в разделе «Ссылки».

Скопируйте утилиту на ваш SSD диск (обычно это диск «C»), запустите ее и нажмите клавишу «Enter» для начала теста.

По окончании теста снова нажмите клавишу «Enter» и утилита закроется.

Подождите несколько минут и снова запустите утилиту.

Если вы видите сообщение о том, что TRIM работает (WORKING), значит все в порядке. Если вы увидите сообщение о том, что TRIM не работает (NOT WORKING), значит что-то не так и нужно предпринять следующие шаги.

Если вы еще не обновили драйвер контроллера SATA до последней версии, что мы рассматривали выше, то сделайте это. Если вы уже обновили драйвер контроллера SATA, то попробуйте выполнить откат к прежней версии драйвера (старый драйвер Microsoft поддерживает TRIM).

После обновления или отката драйвера компьютер нужно перезагружать.

После этого заново проверьте включена ли функция TRIM в операционной системе. Если нет, то включите ее и выполните проверку реальной работы с мощью утилиты «TRIMcheck».

Для задействования функции TRIM в Windows XP и Vista, чтобы скорость SSD не деградировала, есть несколько вариантов.

Некоторые SSD диски поддерживают аппаратный TRIM не зависящий от операционной системы. Такие модели есть, например у Crucial.

Некоторые SSD имеют специальную утилиту, которая может выполнять TRIM в любой операционной системе. В частотности это SSD от Intel.

Наиболее универсальный вариант это использование дефрагментатора «O&O Defrag Professional», который может выполнять TRIM для любого SSD в Windows XP и Vista. Для этого достаточно настроить автоматическую оптимизацию SSD (обычно диска «C») раз в неделю.

Но все же лучшим вариантом будет переход на Windows 7 или выше. Обратите внимание, что при установке системы SATA контроллер должен быть в режиме AHCI, что настраивается в BIOS.

Отключение лишних служб

Во всех версиях Windows есть множество не совсем нужных системных служб. Некоторые из них замедляют работу диска и всей системы, а также приводят к повышенному износу HDD и SSD дисков. Сейчас я покажу вам где и как отключаются службы.

Нажмите сочетание клавиш «Win+R», введите «services.msc» и нажмите «Enter». Кликните правой кнопкой мыши на требуемой службе и выберите «Свойства».

Для отключения службы установите тип запуска «Отключена» и нажмете «ОК».

Теперь при загрузке компьютера служба запускаться не будет.

Отключение автозагрузки программ

Лишние программы в автозагрузке не только замедляют загрузку компьютера, но и используют его память и процессор. Поэтому их отключение положительно влияет не только на скорость работы диска, но и на производительность системы в целом.

Управления автозагрузкой в разных версиях Windows находится в разных местах, кроме того там бывает недостаточно информации чтобы понять что из себя представляет та или иная программа в автозагрузке. Поэтому мы воспользуемся утилитой «CCleaner», которую мы использовали для автоматической очистки диска. Запустите программу и перейдите в раздел «Сервис / Автозагрузка».

Оставляете в автозагрузке только те программы, которые вам известны и действительно нужны каждый раз при загрузке компьютера, такие как Skype, Антивирус и т.п. Остальные компоненты лучше отключить с помощью кнопки «Выключить» в правой панели.

При этом любую из отключенных программ вы всегда сможете запустить вручную. А если пропадет какой-то нужный значок из системного трея, то вы сможете просто вернуть его с помощью кнопки «Включить».

Дефрагментация диска

В процессе работы диска происходит его фрагментация, т.е. файлы разбиваются на множество мелких фрагментов. При обращении к тому или иному файлу, диску приходится собирать все его фрагменты в единое целое, что значительно снижает производительность самого диска и всей системы.

Для ускорения работы диска, после его тщательной очистки, необходимо выполнить его дефрагментацию. В этом процессе все файлы, разбитые на фрагменты, соединяются. Это значительно облегчает и ускоряет работу диска.

Производить дефрагментацию дисков SSD не требуется, так как они работают по другому принципу и не подвержены проблемам фрагментации. Для них это даже плохо, так как приводит к дополнительному износу.

Для дефрагментации рекомендую использовать простую и быструю утилиту «Defraggler», скачать которую можно ниже в разделе «Ссылки».

Запустите утилиту, выберите нужный раздел диска и нажмите кнопку «Дефрагментация».

Дефрагментировать нужно прежде всего системный раздел (диск «C»), так как он в первую очередь отвечает за скорость работы системы. Желательно, до выполнения дефрагментации очистить диск, чтобы на нем было не менее 15% свободного места.

Дефрагментация раздела с системой и основными программами обычно занимает не много времени (15-30 минут). Но если на разделе большой объем пользовательских файлов, то это может занять длительное время (2-3 часа и больше). Поэтому такие файлы лучше хранить на другом разделе (например, диске «D») и запускать его дефрагментацию на ночь.

Производить дефрагментацию рекомендуется только после значительных изменений на диске (удаление/установка нескольких программ и игр), не чаще 1 раза в месяц. Это будет значительно ускорять работу диска (на 15-30% и более).

Оптимизация работы сетевой карты

И так нажимаем одновременно на клавишу «R» и «WIN». Далее прописываем «mmc devmgmt.msc» и жмём «ОК». Теперь находим раздел «Сетевые адаптеры» и далее переходим в свойства того устройства, который вы хотите настроить.

Переходим во вкладку «Дополнительно». И так смотрите, у нас есть определённые свойства, которые мы можем включать (Enebled) или выключать (Disable). На новых версиях «Виндовс» может быть написано «Вкл» или «Выкл». А теперь разбёрем каждое свойство:

ВНИМАНИЕ! Параметры адаптера могут в какой-то степени улучшить показатели, в каком-то моменте ухудшить. Изменяя установки сетевого адаптера, лучше возьмите листочек и выпишите – что именно вы изменили, чтобы в случаи чего вернуть параметры обратно. Также я рекомендую скачать последнюю версию драйвера для вашей сетевой карты или Wi-Fi модуля и установить его. Только после этого заходим в характеристики

- **ARP Offload** – данная функция включена автоматом. Позволяет игнорировать все ARP запросы. Нужна в качестве защиты. Но иногда в некоторых организациях ее включают для более детальной настройки сети.
- **Large Send Offload IPv4/IPv6 – Giant Send Offload** – функция перекладывает фрагментацию пакетов именно на адаптер. Включаем обязательно, чтобы снизить нагрузку на центральный процессор.
- **Auto Disable Gigabit** – если у вас роутер или коммутатор подключен с 4 жилами – 100 Мбит в секунду, то выключаем эту функцию. Она включает и отключает поддержку 1 Гбита. Если вы пользуетесь кабелями по 100 Мбит её можно также отключить. Для роутеров с портом на 1 Гбит – включаем.
- **Energy Efficient Ethernet** – включает энергосбережение – можно включить на ноутбуках, но если скорость станет ниже или будут проблемы с интернетом – сразу выключаем. Для игроманов – вообще ничего по энергосбережению включать не нужно, так как при этом будет сильно садиться показатель производительности обработки сетевого трафика.
- **Flow Control** – если пакеты данных не успели обработаться они стают в очереди. При этом на сервер отсылается команда, чтобы он подождал с отправкой данных. Так как если будет переполнен буфер памяти сетевой карты, информация может потеряться. В общем если сильные лаги, тормоза при просмотре видео – можете включить.
- **Network Address** – виртуальный MAK-адрес. Можно поменять, но бессмысленно, так как физический MAK остается. Этот пункт игнорируем.
- **TCP/UDP Checksum Offload IPv4/IPv6** (контрольная сумма) – для обработки контрольной суммы будет выступать процессор, а не сетевая карта. Нужно включать, если есть интенсивная передача пакетов. Опять же для игр может уменьшить нагрузку на сетевую карту и уменьшить лаги. (Rx & Tx Включить)
- **Transmit Buffers — это** как раз тот самый буфер. Если будет сильно маленькое значение могут быть лаги в играх, так что лучше выставить значение по умолчанию – 147.
- **Green Ethernet** – опять сокращение энергопотребление, на ПК эту функцию лучше выключить. На ноутбуках – по ситуации.

- **TCP Checksum Offload (IPv4)/ (IPv6)** – Та же контрольная сумма, но для TCP. Ставим в режим «Вкл».
- **Interrupt Moderation** – если много качаете, включите. Если много играете, может повышать пинг в игре, из-за простоя пакетов – тогда вырубаем.
- **Receive Side Scaling – RSS** – для обработки нескольких потоков сразу всеми свободными ядрами процессора. Нужно для многоядерных процессоров. Если функция выключена, то все потоки по очереди будут обрабатываться одним ядром. В общем будут лаги и прерывания. Если при включении вы видите, что стало ещё хуже, значит нужно обновить драйвер на адаптер. Качаем только с официального сайта. Вообще это свойство обязательно включается по умолчанию.
- **Priority & VLAN** – при отправке пакета дополнительно записываем информацию, о важности и приоритете пакета. Можно включить. Если будут лаги и тормоза – выключаем.
- **WOL & Shutdown Link Speed** – стандартная скорость коннекта при отключении. Ставим на сотку.
- **NS Offload** – Включаем. Таким образом соседские Wi-Fi сети при отправке запроса к вам, не будут получать ответа.
- **Jumbo Frame – Jumbo Packet** – я бы эту функцию выключил, так как она снижает частоту обработки пакетов в 6 раз.
- **Speed & Duplex** – выставляет режим в «Дуплекс» или «Полудуплекс». Первый позволяет одновременно принимать и отправлять данные – ставим, если играем. Второй режим может одновременно принимать или отправлять данные. Ставим второй, если много качаем. Но честно сказать, полудуплекс сильно снижает нормальную работу очень многих сервисов. Поэтому лучше всегда ставить или «Автосогласование» или «Дуплекс».
- **Wake on pattern match и Wake on Magic Packet** – включаем.
- **UDP Checksum Offload (IPv6)/ (IPv4)** – включает обработку контрольной суммы пакетов UDP. Включаем для обработки процессором, а не «сетевухой».

После изменения, следует перезагрузить компьютер или ноутбук, чтобы некоторые изменения вступили в силу. Установки сетевого адаптера всегда можно откатить обратно, самое главное не потеряйте тот листок с настройками.

Практическое задание к работе.

1. Выполните проверку логических дисков на наличие ошибок.
2. Выполните полную дефрагментацию всех логических дисков.
3. Увеличьте размер виртуальной оперативной памяти до 6 Гб через настройки системы.

Используйте НЕ системный жёсткий диск.

4. С помощью утилиты msconfig проверьте список автозагрузки и удалите оттуда ненужные программы.
5. Просмотрите вкладку «Службы» панели «Администрирование» и остановите службы, которые не являются жизненно важными в работе ПК.
6. Согласно теоретической части, проверьте настройки сетевой карты на Вашем ПК и установите оптимальные.
7. Запишите названия свойств сетевой карты и их значения после изменений.
8. Запретите отключение сетевой карты при переходе в спящий режим.

Лабораторная работа №18. Составление коммерческого предложения на внедрение программного обеспечения.

Теоретический материал.

Изучите прилагаемый к работе PDF-файл «Пример коммерческого предложения по внедрению ПО».

Практическое задание к работе.

1. Используя ранее выбранную предметную область, составьте по образцу коммерческое предложение на внедрение вашего продукта в эксплуатацию.

Выполнить задания в рабочей тетради.

Лабораторная работа №19. Составление дорожной карты работ по внедрению ПО.

Теоретический материал.

Пример дорожной карты см. в приложении к лабораторной работе.

Как создать идеальную дорожную карту продукта и что для этого нужно?

Дорожная карта или roadmap в руках умелого менеджера продукта – настоящее стратегическое оружие. Как большинство стратегов умеют грамотно обращаться со своими рабочими инструментами, так и менеджер продукта должен уметь тактически применить дорожную карту и использовать доступные сервиса для этой цели.
Если раньше для этих целей обходились простыми функциями Excel или Powerpoint, то современные менеджеры продуктов могут получать реальную пользу и удовольствие от работы с качественными инструментами для создания дорожной карты.

Зачем нужна дорожная карта?

Цель дорожной карты, как основного документа менеджера продукта, – донести главные идеи и прогресс в задачах до членов команды и внешних заинтересованных сторон (акционеров, заказчиков, партнеров).

Продуктовая дорожная карта состоит из инициативы глобального уровня и всех ее запланированных шагов. В нее не должна входить каждая функция продукта и подробные списки багов. Этот стратегический документ предназначен для отдельного планирования. Обязательно следует актуализировать roadmap продукта на протяжении всего его жизненного цикла. Включенные функции, инициативы и требования должны создаваться и инициироваться многими сторонами: руководством, клиентами, sales-менеджерами, партнерами, службой поддержки, разработчиками, финансистами и, конечно, продуктовиками.

Дорожные карты не ограничиваются продуктами: их цели аналогичны для разных видов (например, маркетинговые и IT-роадмапы).

Любая дорожная карта, ориентированная на свою аудиторию, имеет свои особенности.

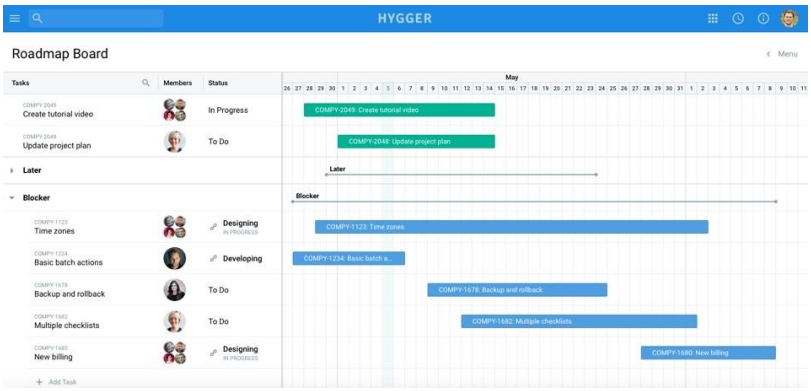
7. **Дорожные карты для разработчиков** обычно фокусируются на функциях, спринтах, релизах и майлстоунах. Они довольно короткие и, как правило, более масштабные.
8. **Дорожные карты для продавцов** сфокусированы на сочетании функций и преимуществ для клиентов.
9. **Внешние дорожные карты** (для клиентов или партнеров) ориентированы на основных преимуществах продукта для них. Как любой внешний документ, этот вид дорожной карты продукта должен быть привлекательным, визуально понятным и доступным.

Также дорожные карты отличаются в разных командах. К примеру, роадмап в Agile-команде будет отличаться от типовой дорожной карты в Waterfall.

Отличия дорожные карты в Agile и Waterfall

- Команды Waterfall обычно бизнес-ориентированы, основаны на финансовых метриках. В Agile цели ориентированы на клиента (например, рост пользователей и удовлетворенность клиентов).

- Дорожные карты в Waterfall отражают завершения в срок год или два года, а дорожная карта Agile обычно отражает квартальные завершенности. Планирование в компаниях Waterfall и Agile также отличается в зависимости от сроков.
- Различия также связаны с принципом взаимодействия. Взаимодействия в командах Waterfall последовательны, а члены Agile-команд работают в соответствии с кросс-функциональностью и одновременностью действий.
- Наконец, дорожные карты Waterfall имеют ограниченную гибкость, а дорожные карты Agile гораздо более гибки, как и сама методология.



Не существует идеального подхода, как визуально создавать дорожную карту; вы можете использовать разные шаблоны для отображения основных данных:

- Стратегические инициативы глобального уровня
- Релизы по периодам (кварталам)
- Детализированные функции
- Информация о баг фиксинге

Как создать идеальную дорожную карту?

Один из простейших способов создать дорожную карту — использовать электронные таблицы. Например, с помощью Excel можно скомпилировать продуктовые идеи, инициативы, выставить сроки и дедлайны. Их достаточно просто обновлять.

Однако дорожные карты в таблицах имеют значительные недостатки. Таблицы не обладают достаточной визуализацией и их недостаточно для представления стратегического плана.

Кроме того, тот же Excel — это статичный документ, который после шаринга сложно контролировать и синхронизировать версии со всеми участниками команды.

Презентации

Гораздо проще представить визуально дорожную карту в ПО, предназначенном для создания презентаций. Здесь у менеджера продукта больше возможностей и свободы действий. Но и в этом случае, презентация — это статичный документ, требующий ручных обновлений, как и электронная таблица, что может создать путаницу с контролем версий. В идеале дорожная карта должна синхронно обновляться у всех членов команды. Именно поэтому сегодня все популярнее становится сервис для управления продуктами с функционалом для создания дорожных карт.

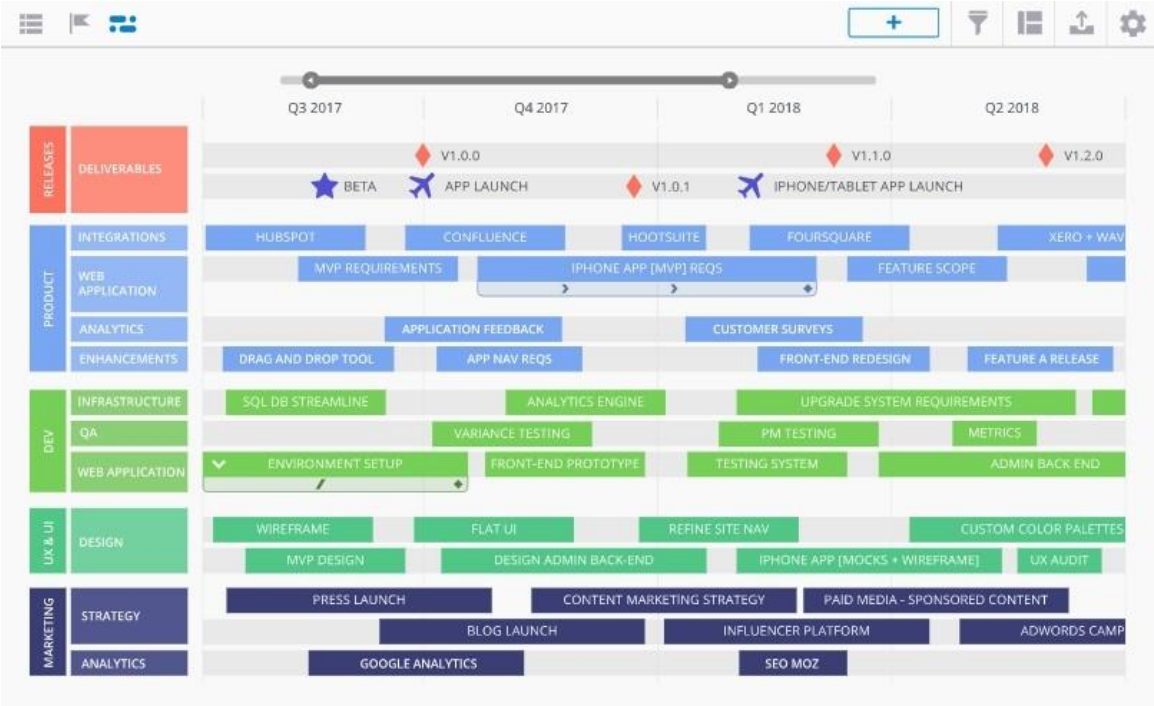
Почему специальные сервисы лучше простых способов создания дорожной карты?

У менеджеров продуктов сегодня есть возможность визуализировать дорожные карты с помощью лучших управленческих инструментов, которые помогают:

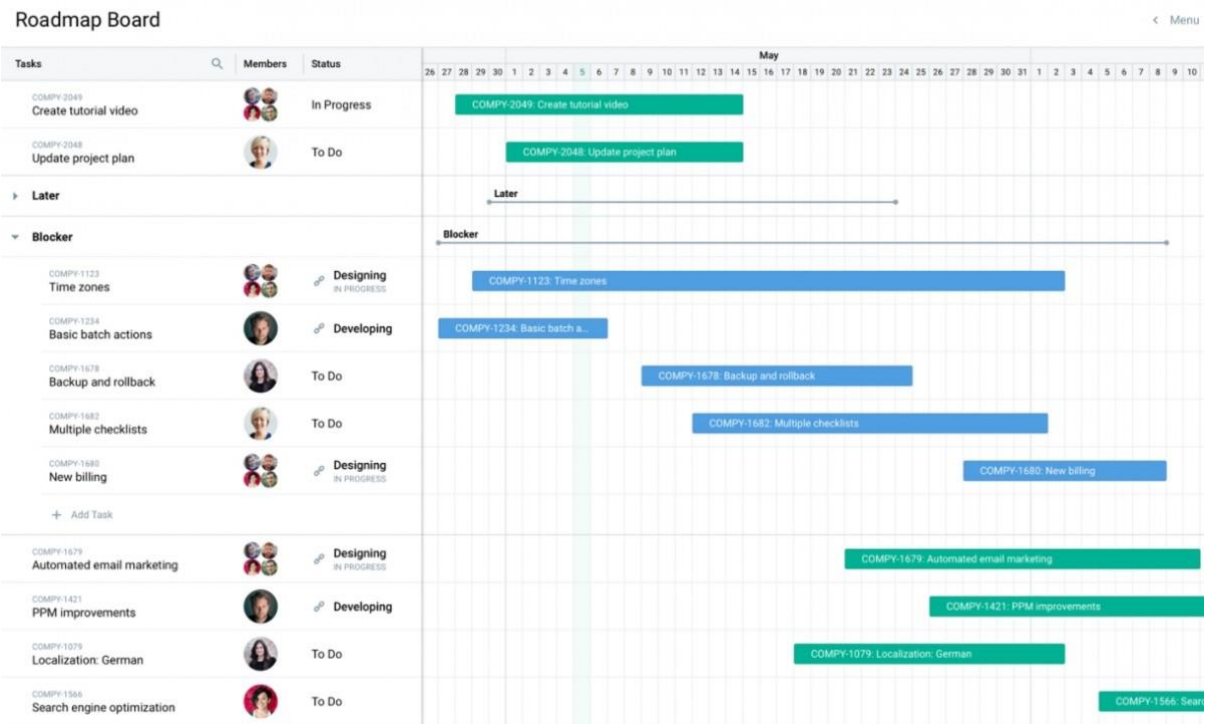
- Представить визуально дорожную карту продукта
- Связать глобальную стратегию с процессами дорожной карты
- Определить и оценить идеи
- Сотрудничать со всеми заинтересованными сторонами (включая клиентов и нетехнических коллег)
- Интегрироваться со сторонними системами

Какой сервис выбрать? Вот **ТОП-7 платформ для менеджеров продуктов**, которые заботятся о качественной визуализации дорожной карты:

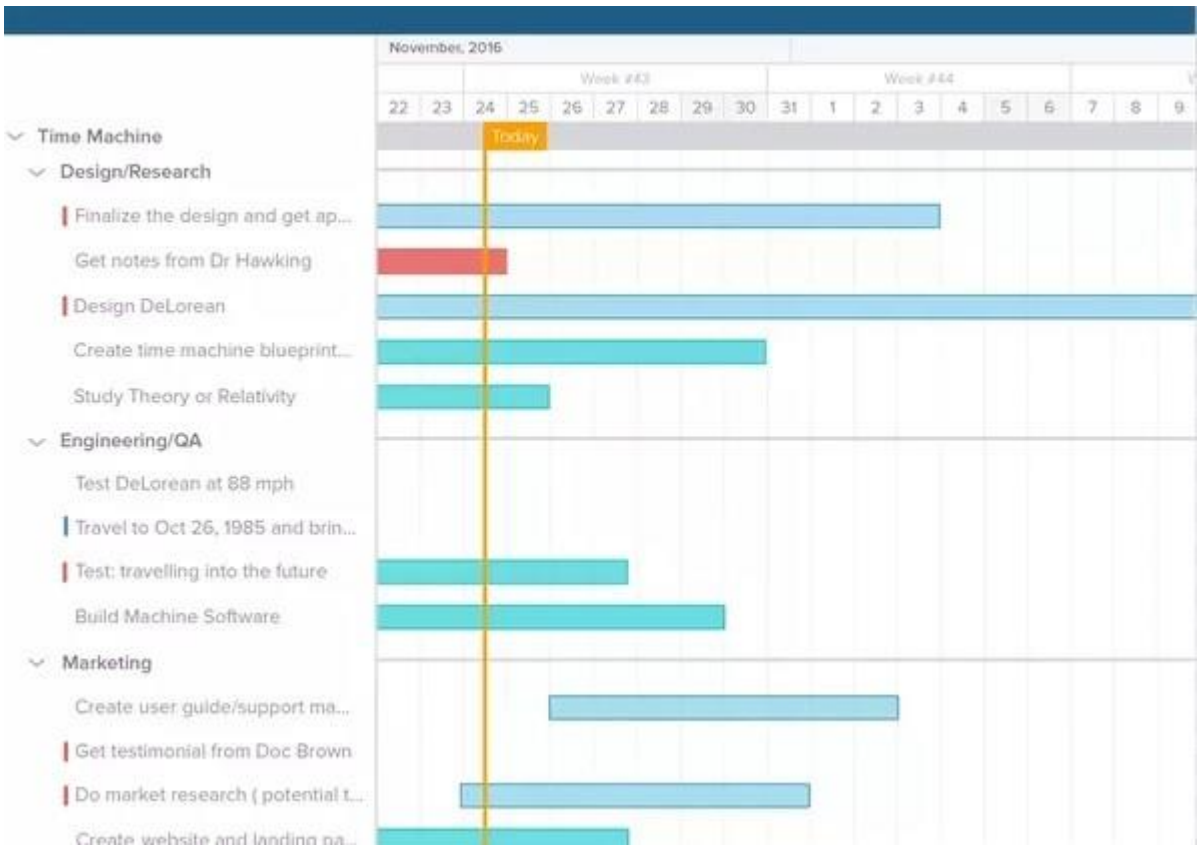
Roadmunk



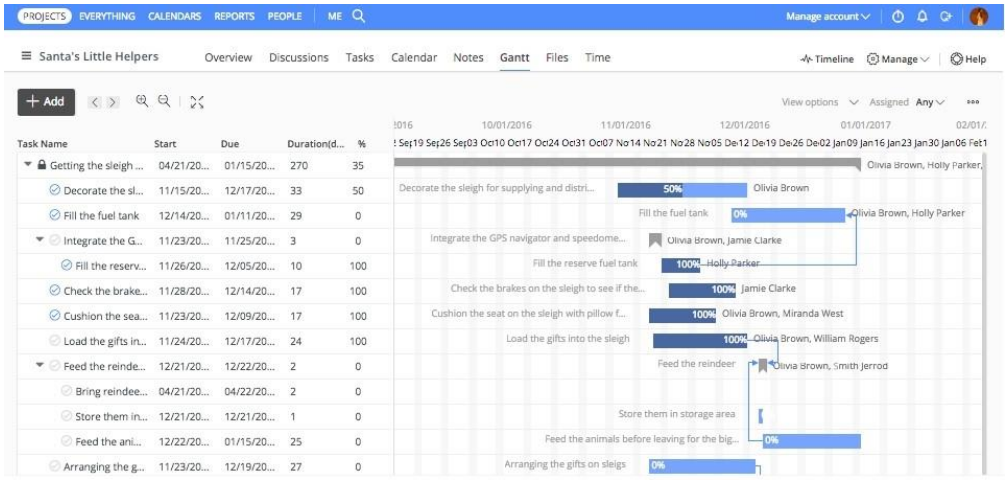
Hygger



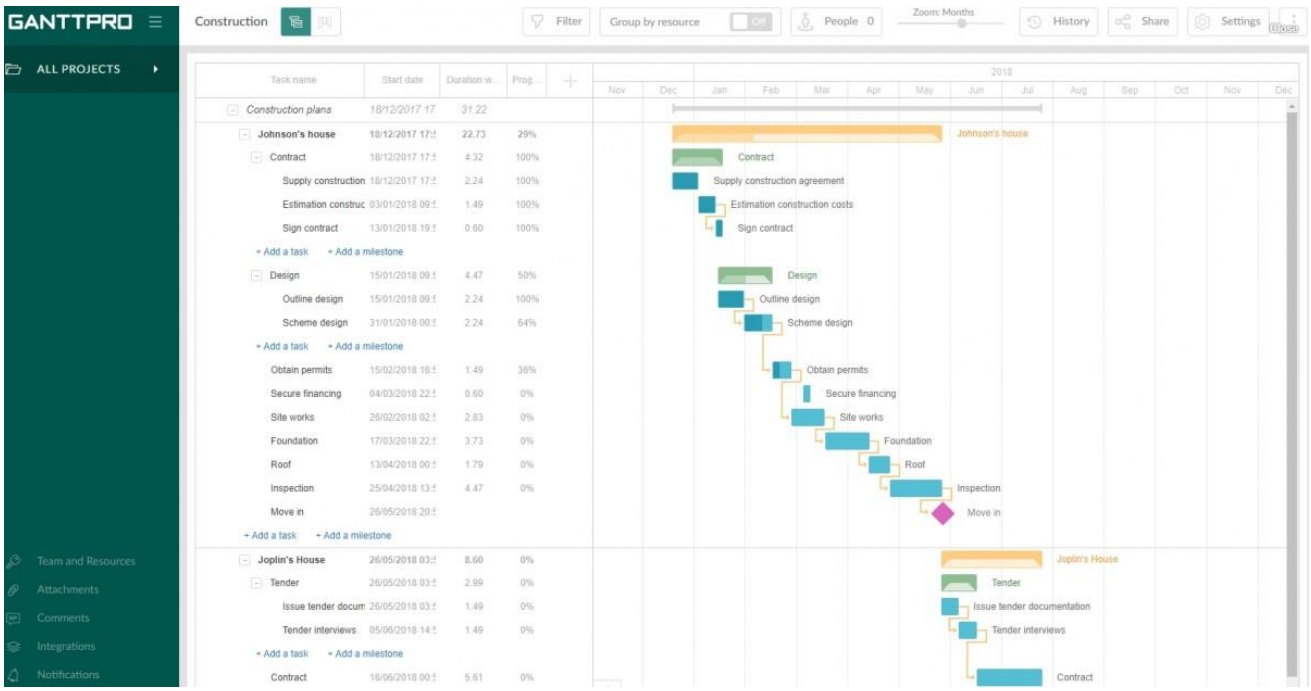
Taskworld



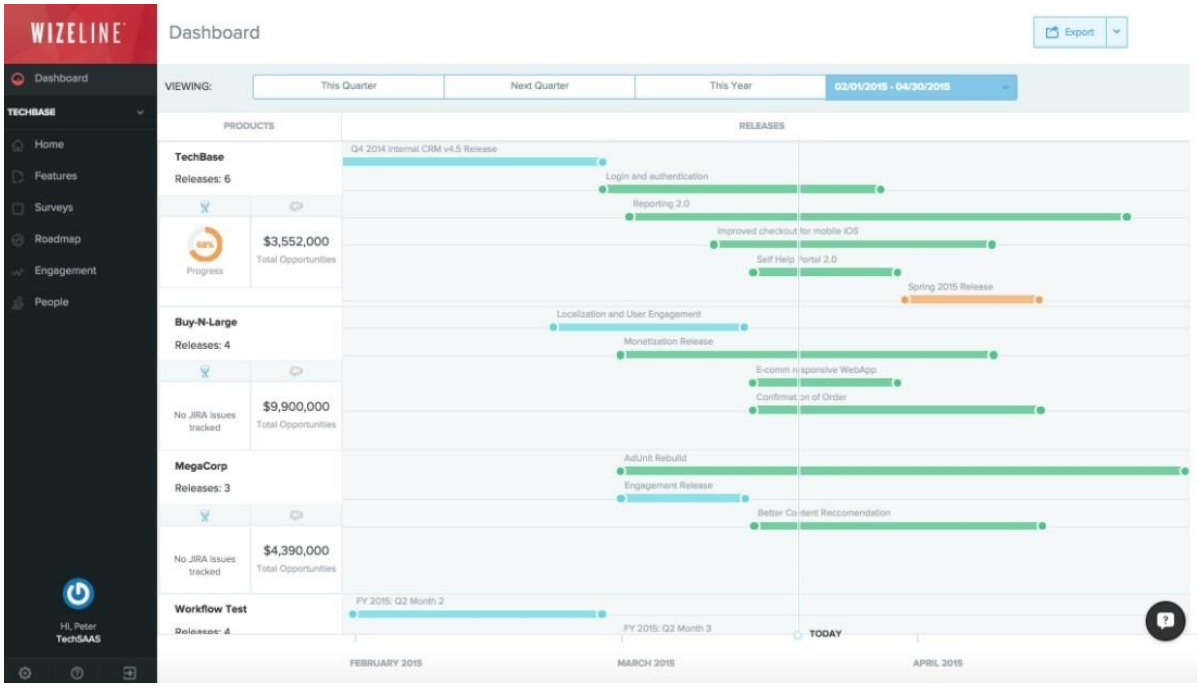
Proofhub



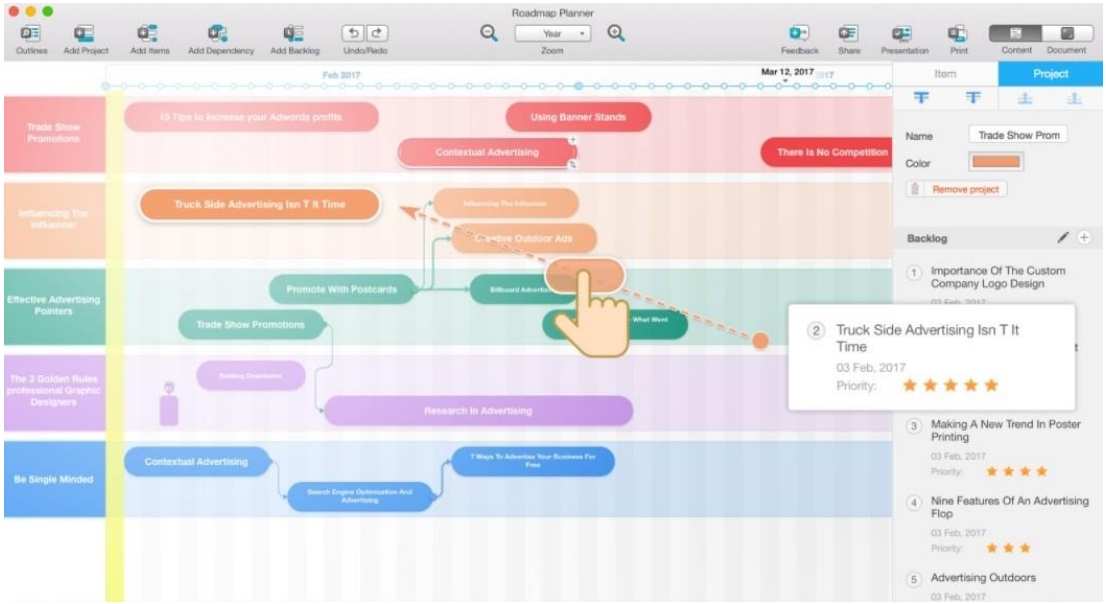
GanttPRO



Wizeline



Roadmap planner



Если сервис для управления продуктом определен, можно приступать к созданию дорожной карты. С чего начать? Как создать понятную всем дорожную карту?

Основные этапы создания дорожной карты

Вероятно, нет необходимости напоминать лишний раз о ключевых бизнес-целях, которые напрямую связаны с созданием дорожных карт. Четко понимая их, а также ваши инициативы, в которые вы собираетесь инвестировать, вы можете определить, какие функции добавить в свою дорожную карту (подумав о том, что будет иметь наибольшее влияние на ваш бизнес). Вот 4-шаговая стратегия, которая будет полезна всем:

1. Определение стратегии

Обычно глобальные стратегии основаны на ключевых целях. Это общее видение целей определяет ваш прогноз для всего продукта. Сильное видение продукта поддерживается деталями, связанными с вашими клиентами и их потребностями. Оно отражает суть того, что вы хотите получить. Убедитесь, что ваша команда понимает все на этом этапе, чтобы развить свой будущий шедевр.

2. Кастомизация релизов

Здесь вы выбираете функции, которые нужно выделить, и решаете, должны ли внутренние или внешние данные представляться в каждом релизе или нет. Даты внешних и внутренних релизов могут быть разными.

3. Приоритизация фич

Помните, что запросы клиентов всегда должны оцениваться в соответствии с вашей стратегией.

Существуют различные метрики, которые помогают оценить вашу стратегию. Нетрудно создать свою собственную оценочную карту по вашему виду продукта, поскольку каждый продукт является чем-то уникальным. Имея свою оценочную карту, вы сможете объективно расставить приоритеты в своих дорожных картах. Не забывайте об общих правилах по установлению приоритетов и известных методологиях приоритизации.

4. Совместное использование дорожной карты

Создание отличных продуктов невозможно без коммуникации, обратной связи и прозрачности взаимоотношений. Без них не обойтись и в вашей стратегии.Когда вы получаете желаемый результат, вы можете сохранить его и поделиться с заинтересованными сторонами. С помощью программного обеспечения для управления продуктами, вы можете легко делиться своими роадмапами, постоянно обновляя их.

Практическое задание к работе №19.

1. Используя ранее выбранную предметную область, составьте по образцу коммерческое предложение на внедрение вашего продукта в эксплуатацию.

Выполнить задания в рабочей тетради.

Лабораторная работа №21. Составление портфеля проектной документации по внедрению.

Теоретический материал.

Этап подготовки проекта

Начальным этапом проекта внедрения КИС является подготовка. В контексте данной фазы формулируются цели и задачи, а также готовятся шаблоны документов и укрупненный план график проекта. Основным документом этапа служит устав, определяющий цели проекта, а также содержащий функциональный, организационный, технический и методологический объемы проекта. Кроме того документ описывает участников проекта и задает порядок согласования проектной документации. Подготавливается концепция обучения проектной группы, включающая предлагаемый подход к обучению команды внедрения КИС заказчика. Шаблоны документов, используемые для подготовки документации на последующих этапах проекта, формируются здесь же. Содержащийся в уставе объем проекта необходим для определения сроков выполнения проекта. Последние отражаются в укрупненном плане графике, который позже уточняется для каждой фазы. Таким образом, устав является главенствующим документом этапа подготовки.

Этап проектирования

Завершив подготовку проекта, переходим к этапу проектирования системы. Качество, взаимосвязь и детализация проектируемых решений являются определяющим фактором успеха внедрения КИС. Допущенные на этапе проектирования ошибки устранить достаточно трудоемко. Начало этапа проектирования сопровождается подготовкой обучающих материалов и плана проведения обучения команды заказчика. Сформированная ранее концепция обучения проектной группы содержит лишь поверхностное содержание указанных документов. Далее проектная команда заказчика совместно со специалистами подрядчика участвует в обследовании бизнес-процессов клиента. Результатом анализа процессов являются функционально-технические требования, предъявляемые к проектируемой системе.

Требования заказчика сопоставляются со стандартным решением КИС (Fit-анализ) для выявления функционального дефицита (GAP-анализ). Функциональный дефицит требует доработки системы, для чего готовятся спецификации на разработку, содержащие постановку задачи и предлагаемый вектор решения. Разрабатывается концепция ролей и полномочий, определяющая перечень ролей пользователей и правила их создания и присвоения сотрудникам. Стандартный функционал КИС, спецификации на разработку и концепция ролей и полномочий необходимы для формирования проектных решений. Проектные решения содержат бизнес-процессы заказчика в моделях «как есть» и «как будет» [8-9] с указанием доработок системы и ролей пользователей.

Проектные решения создаются на основе данных Fit/GAP-анализа функционально-технических требований клиента. Проектный опыт показывает, решения чаще всего формируются для каждого бизнес-процесса заказчика. Кроме того, отдельно выделяются решения по ведению основных данных, организационной структуре и миграции. Вопрос миграции исторических данных системы рассматривается отдельно в соответствующей концепции. Концепция включает описание подхода миграции данных, используемые механизмы миграции согласно проектным решениям и предполагаемый план миграции. Концепции обучения конечных пользователей и перехода к использованию системы тоже создаются на данном этапе. Концепция обучения пользователей определяет порядок и плановые сроки проведения тренингов, необходимые обучающие материалы, а также перечень выполняемых упражнений.

В концепции перехода к использованию системы описывается порядок применения нового КИС решения и работы предыдущей системы, задается перечень шагов для обеспечения пользователям возможности работы с новым решением, и определяется набор операций, выполняемых вручную техническими специалистами в КИС. Все создаваемые документы данного этапа взаимосвязаны. Проектные решения можно отнести к наиболее значимым

документам, так как они являются основой для реализации системы, обучения пользователей, миграции данных и перехода к применению предлагаемого КИС решения.

Этап реализации

Реализация системы ведется согласно подготовленным на этапе проектирования документам. Ошибки проектирования неминуемо приводят к неправильной настройке и доработке системы, именно по этой причине фаза проектирования имеет столь существенное значение. Следуя проектным решениям, спецификациям на разработку и концепции ролей и полномочий ведется реализация системы, готовятся описания выполненных настроек, технической реализации спецификаций и настройки ролей и полномочий соответственно. Не вошедшие в описание выполненных настроек операции требуют ручного ввода специалистами КИС. Поэтому описание подобных операций ведется в инструкции по переходу к использованию системы, ссылка на которую содержится в соответствующей концепции.

Согласно концепции миграции данных были подготовлены проектные решения, реализованные в КИС на данном этапе. Здесь же готовятся инструкции, включающие описание процедур загрузки и контроля данных, а также примеры шаблонов загрузки. Настроенная и доработанная система используется для проведения внутреннего тестирования. Тестирование ведется специалистами КИС на основе сценариев функционального тестирования. Сценарии содержат упражнения, отражающие бизнес-процессы проектных решений. Цель функционального тестирования заключается в проверке корректности работы отдельных программ. Интеграционное тестирование в отличие от функционального позволяет рассмотреть правильность взаимодействия программ, вовлеченных в единый бизнес-процесс.

К интеграционному тестированию привлекаются как специалисты КИС, так и ключевые пользователи клиента. Ошибки функционального и интеграционного тестирования фиксируются в журнале регистрации проблем для последующего их устранения. Количество ошибок в журнале проблем свидетельствует о глубине понимания бизнес-требований клиента. Если журнал содержит слишком большое число критических замечаний, высока вероятность приостановки проекта (так как замечания должны быть устранены до этапа ОПЭ).

Этап подготовки к опытно-промышленной эксплуатации

Реализация системы выполнена, и журнал проблем содержит незначительное число замечаний. Начинается подготовка к ОПЭ. Первоочередной задачей данного этапа является обучение конечных пользователей. Готовятся инструкции конечных пользователей (в разрезе бизнес-процессов или операций). Далее на их основе формируются сценарии обучения пользователей, включаемые в окончательный план обучения. Предполагаемый план обучения был создан ранее в контексте концепции обучения. Обучение пользователей проводится в условиях близких к реальным. Поэтому необходимо подготовить список участников и присвоить им реальные роли для выполнения тестовых упражнений. Тренинги являются своего рода тестированием системы, тем самым обновляется журнал проблем.

Далее анализируются полученные в ходе обучения замечания. Продолжение проекта возможно, если журнал проблем содержит замечания, не тормозящие проведение ОПЭ. В этом случае готовится список пользователей участвующих в ОПЭ, присваиваются необходимые роли. Формируется план перехода к использованию системы в режиме ОПЭ, включающий перечень необходимый шагов для обеспечения работы КИС и сроки их выполнения. Конкретные шаги плана содержат ссылки на операции из инструкции по переходу к использованию системы. План миграции данных аналогичен плану перехода к использованию системы, однако, содержит ссылки на инструкцию по миграции. Клиент обеспечивает заполнение и проверку данных в шаблонах загрузки. Этап подготовки

завершается заведением пользователей в системе проведения ОПЭ, а также миграцией основных и переменных данных.

Этап опытно-промышленной эксплуатации

Опытно-промышленная эксплуатация позволяет проверить работоспособность системы при выполнении реальных бизнес-операций с использованием исторических данных из предыдущей системы. Загрузка переменных данных на этапе подготовки к ОПЭ ограничивается одним периодом. Поэтому первое, что пользователи выполняют в системе, – проверка корректности загрузки остатков. Далее сотрудниками осуществляется ввод движений материалов и операций по счетам на основе первичных документов заданного периода. Замечания пользователей при работе с системой заносятся в журнал проблем. Этап ОПЭ завершается закрытием периода в модулях логистики, бухгалтерского учета и контроллинга.

Этап перехода к продуктивной эксплуатации

Успешное завершение этапа ОПЭ позволяет говорить о переходе к ПЭ. Основное условие перехода – отсутствие замечаний в журнале проблем и обновление всей проектной документации по результатам исправления замечаний. Аналогично этапу подготовки к ОПЭ готовятся списки пользователей системы, планы перехода к ПЭ и миграции данных. Заполняются шаблоны загрузок данных. Создав пользователей в КИС, выполнив все операции из плана перехода и миграцию данных, начинается работа в режиме ПЭ. Начиная с этого момента, возникающие замечания и проблемы разрешаются силами группы поддержки клиента. На этапах же реализации, подготовки к ОПЭ и ОПЭ ошибки системы регистрировались в журнале проблем и исправлялись специалистами подрядчика.

Шаблоны документации смотрите в приложении к лабораторной работе.

Типичные документы на различных этапах внедрения и поддержки ПО приведены в конце лабораторной работы в виде схем.

Практическое задание к работе №21.

1. Используя ранее выбранную предметную область, составьте портфель документов на внедрение вашего программного продукта в эксплуатацию.

Выполнить задания в рабочей тетради.

Лабораторная работа №22. «Разработка руководства системного администратора»

Теоретический материал.

ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению (с Изменением N 1)

ГОСТ 19.503-79

Группа Т55

МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ

Единая система программной документации

РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

Требования к содержанию и оформлению

Unified system for program documentation. System programmer's guide. Requirements for contents and form of presentation

МКС 35.080

Дата введения 1980-01-01

Постановлением Государственного комитета СССР по стандартам от 12 января 1979 г. N 74 дата введения установлена 01.01.80

ИЗДАНИЕ (январь 2010 г.) с Изменением N 1, утвержденным в сентябре 1981 г. (ИУС 11-81).

Настоящий стандарт устанавливает требования к содержанию и оформлению программного документа "Руководство системного программиста", определенного ГОСТ 19.101-77.

Стандарт полностью соответствует СТ СЭВ 2094-80*.

(Измененная редакция, Изм. N 1).

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Структура и оформление программного документа устанавливаются в соответствии с ГОСТ 19.105-78.

Составление информационной части (аннотации и содержания) является обязательным.

1.2. Руководство системного программиста должно содержать следующие разделы:

общие сведения о программе;
структура программы;
настройка программы;

проверка программы;
дополнительные возможности;
сообщения системному программисту.

В зависимости от особенностей документа допускается объединять отдельные разделы или вводить новые.

В обоснованных случаях допускается раздел "Дополнительные возможности" не приводить, а в наименованиях разделов опускать слово "программа" или заменять его на "наименование программы".
(Измененная редакция, Изм. N 1).

2. СОДЕРЖАНИЕ РАЗДЕЛОВ

2.1. В разделе "Общие сведения о программе" должны быть указаны назначение и функции программы и сведения о технических и программных средствах, обеспечивающих выполнение данной программы.

2.2. В разделе "Структура программы" должны быть приведены сведения о структуре программы, ее составных частях, о связях между составными частями и о связях с другими программами.

(Измененная редакция, Изм. N 1).

2.3. В разделе "Настройка программы" должно быть приведено описание действий по настройке программы на условия конкретного применения (настройка на состав технических средств, выбор функций и др.).

При необходимости приводят поясняющие примеры.

2.4. В разделе "Проверка программы" должно быть приведено описание способов проверки, позволяющих дать общее заключение о работоспособности программы (контрольные примеры, методы прогона, результаты).

2.5. В разделе "Дополнительные возможности" должно быть приведено описание дополнительных разделов функциональных возможностей программы и способов их выбора.

2.6. В разделе "Сообщения системному программисту" должны быть указаны тексты сообщений, выдаваемых в ходе выполнения настройки, проверки программы, а также в ходе выполнения программы, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

2.7. В приложении к руководству системного программиста могут быть приведены дополнительные материалы (примеры, иллюстрации, таблицы, графики и т.п.).

Практическое задание. Используя вышеприведенный шаблон, выполните следующие действия:

1. Выберите для руководства системного администратора своей программы (см. предметную область) только необходимые разделы и пункты.
2. Изучите приложение к лабораторной работе (документ «Руководство системного программиста»).
3. Используя стандарт и пример, составьте руководство системного администратора.

Лабораторная работа №23. «Разработка руководства оператора»

Теоретический материал.

РД 50-34.698-90 Руководство пользователя (пример оформления)

Ниже представлен пример (образец) документа "**Руководство пользователя**", разработанного на основании методических указаний РД 50-34.698-90.
Данный документ формируется IT-специалистом, или функциональным специалистом, или техническим писателем в ходе разработки рабочей документации на систему и её части на стадии «Рабочая документация».

Для формирования руководства пользователя в качестве примера был взят инструмент *Oracle Discoverer* информационно-аналитической системы «Корпоративное хранилище данных».
Ниже приведен состав руководства пользователя в соответствии с ГОСТ. Внутри каждого из разделов кратко **приведены требования к содержанию и текст примера заполнения** (выделен вертикальной чертой).
Разделы руководства пользователя:

1. Введение.
2. Назначение и условия применения.
3. Подготовка к работе.
4. Описание операций.
5. Аварийные ситуации.
6. Рекомендации по освоению.

1. Введение

В разделе "Введение" указывают:

- область применения;
- краткое описание возможностей;
- уровень подготовки пользователя;
- перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю.

1.1. Область применения

Требования настоящего документа применяются при:

- предварительных комплексных испытаниях;
- опытной эксплуатации;
- приемочных испытаниях;
- промышленной эксплуатации.

1.2. Краткое описание возможностей

Информационно-аналитическая система Корпоративное Хранилище Данных (ИАС КХД) предназначена для оптимизации технологии принятия тактических и стратегических управленческих решений конечными бизнес-пользователями на основе информации о всех аспектах финансово-хозяйственной деятельности Компании.

ИАС КХД предоставляет возможность работы с регламентированной и нерегламентированной отчетностью.

При работе с отчетностью используется инструмент пользователя Oracle Discoverer Plus, который предоставляет следующие возможности:

- формирование табличных и кросс-табличных отчетов;
- построение различных диаграмм;
- экспорт и импорт результатов анализа;
- печать результатов анализа;
- распространение результатов анализа.

1.3. Уровень подготовки пользователя

Пользователь ИАС КХД должен иметь опыт работы с ОС MS Windows (95/98/NT/2000/XP), навык работы с ПО Internet Explorer, Oracle Discoverer, а также обладать следующими знаниями:

- знать соответствующую предметную область;
- знать основы многомерного анализа;
- понимать многомерную модель соответствующей предметной области;
- знать и иметь навыки работы с аналитическими приложениями.

Квалификация пользователя должна позволять:

- формировать отчеты в Oracle Discoverer Plus;
- осуществлять анализ данных.

1.4. Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю

- Информационно-аналитическая система «Корпоративное хранилище данных». ПАСПОРТ;
- Информационно-аналитическая система «Корпоративное хранилище данных». ОБЩЕЕ ОПИСАНИЕ СИСТЕМЫ.

2. Назначение и условия применения Oracle Discoverer Plus

В разделе "Назначение и условия применения" указывают:

10. виды деятельности, функции, для автоматизации которых предназначено данное средство автоматизации;
11. условия, при соблюдении (выполнении, наступлении) которых обеспечивается применение средства автоматизации в соответствии с назначением (например, вид ЭВМ и конфигурация технических средств, операционная среда и общесистемные программные средства, входная информация, носители данных, база данных, требования к подготовке специалистов и т. п.).

Oracle Discoverer Plus в составе ИАС КХД предназначен для автоматизации подготовки, настройки отчетных форм по показателям деятельности, а также для углубленного исследования данных на основе корпоративной информации хранилища данных.

Работа с Oracle Discoverer Plus в составе ИАС КХД возможна всегда, когда есть необходимость в получении информации для анализа, контроля, мониторинга и принятия решений на ее основе.

Работа с Oracle Discoverer Plus в составе ИАС КХД доступна всем пользователям с установленными правами доступа.

3. Подготовка к работе

В разделе "Подготовка к работе" указывают:

5. состав и содержание дистрибутивного носителя данных;
6. порядок загрузки данных и программ;
7. порядок проверки работоспособности.

3.1. Состав и содержание дистрибутивного носителя данных

Для работы с ИАС КХД необходимо следующее программное обеспечение:

6. Internet Explorer (входит в состав операционной системы Windows);
7. Oracle JInitiator устанавливается автоматически при первом обращении пользователя к ИАС КХД.

3.2. Порядок загрузки данных и программ

Перед началом работы с ИАС КХД на рабочем месте пользователя необходимо выполнить следующие действия:

8. Необходимо зайти на сайт ИАС КХД ias-dwh.ru.
9. Во время загрузки в появившемся окне "Предупреждение о безопасности", которое будет содержать следующее: 'Хотите установить и выполнить "Oracle JInitiator" ...' Нажимаем на кнопку "Да".
10. После чего запуститься установка Oracle JInitiator на Ваш компьютер. Выбираем кнопку Next и затем OK.

3.3. Порядок проверки работоспособности

Для проверки доступности ИАС КХД с рабочего места пользователя необходимо выполнить следующие действия:

5. Открыть Internet Explorer, для этого необходимо кликнуть по ярлыку «Internet Explorer» на рабочем столе или вызвать из меню «Пуск».
6. Ввести в адресную строку Internet Explorer адрес: ias-dwh.ru и нажать «Переход».
7. В форме аутентификации ввести пользовательский логин и пароль. Нажать кнопку «Далее».
8. Убедиться, что в окне открылось приложение Oracle Discoverer Plus.

В случае если приложение Oracle Discoverer Plus не запускается, то следует обратиться в службу поддержки.

4. Описание операций

В разделе "Описание операций" указывают:

7. описание всех выполняемых функций, задач, комплексов задач, процедур;
8. описание операций технологического процесса обработки данных, необходимых для выполнения функций, комплексов задач (задач), процедур.

Для каждой операции обработки данных указывают:

9. наименование;

10. условия, при соблюдении которых возможно выполнение операции;
11. подготовительные действия;
12. основные действия в требуемой последовательности;
13. заключительные действия;
14. ресурсы, расходуемые на операцию.

В описании действий допускаются ссылки на файлы подсказок, размещенные на магнитных носителях.

4.1. Выполняемые функции и задачи

Oracle Discoverer Plus в составе ИАС КХД выполняет функции и задачи, приведенные в таблице ниже:

Функции	**Задачи**	**Описание**
Обеспечивает многомерный анализа в табличной и графической формах	Визуализация отчетности	В ходе выполнения данной задачи пользователю системы предоставляется возможность работы с выбранным отчетом из состава преднастроенных.
	Формирование табличных и графических форм отчетности	В ходе выполнения данной задачи пользователю системы предоставляется возможность формирования собственного отчета в табличном или графическом виде на базе преднастроенных компонентов.

4.2. Описание операций технологического процесса обработки данных, необходимых для выполнения задач

Ниже приведено описание пользовательских операций для выполнения каждой из задач.

Задача: «Визуализация отчетности»

Операция 1: Регистрация на портале ИАС КХД

Условия, при соблюдении которых возможно выполнение операции:

4. Компьютер пользователя подключен к корпоративной сети.
5. Портал ИАС КХД доступен.
6. ИАС КХД функционирует в штатном режиме.

Подготовительные действия:

На компьютере пользователя необходимо выполнить дополнительные настройки, приведенные в п. 3.2 настоящего документа.

Основные действия в требуемой последовательности:

6. На иконке «ИАС КХД» рабочего стола произвести двойной щелчок левой кнопкой мышки.
7. В открывшемся окне в поле «Логин» ввести имя пользователя, в поле «Пароль» ввести пароль пользователя. Нажать кнопку «Далее».

Заключительные действия:

Не требуются.

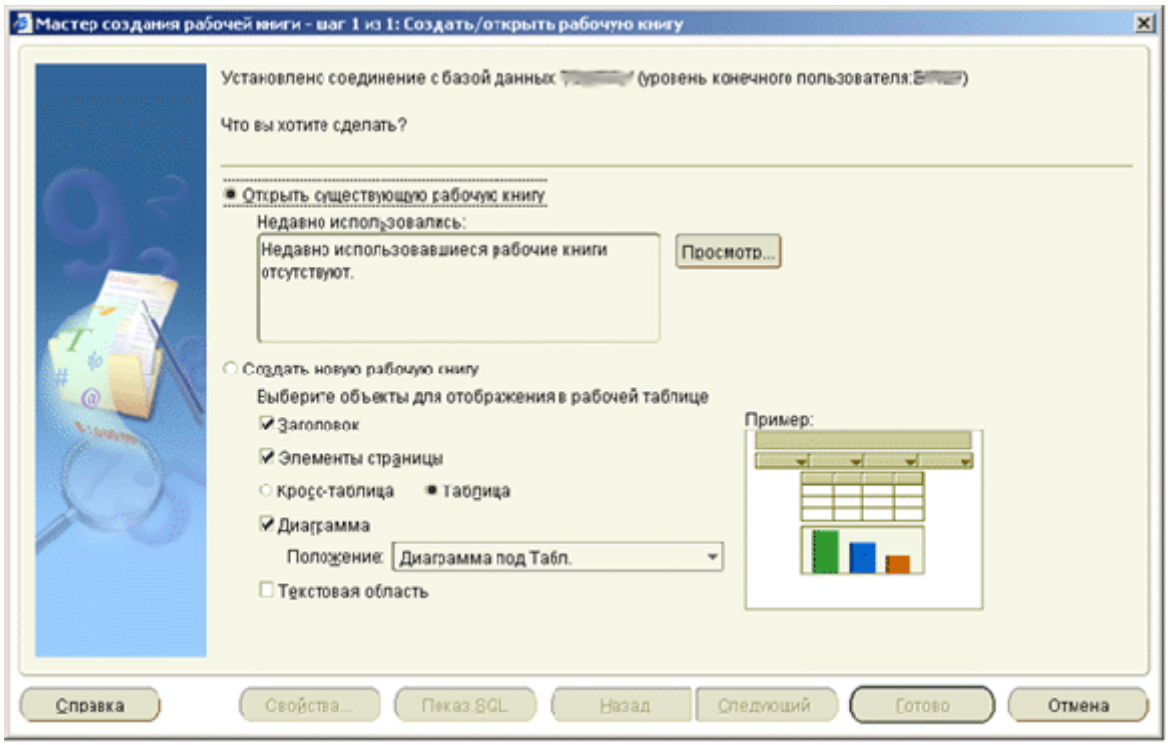
Ресурсы, расходуемые на операцию:
15-30 секунд.

Операция 2: Выбор отчета

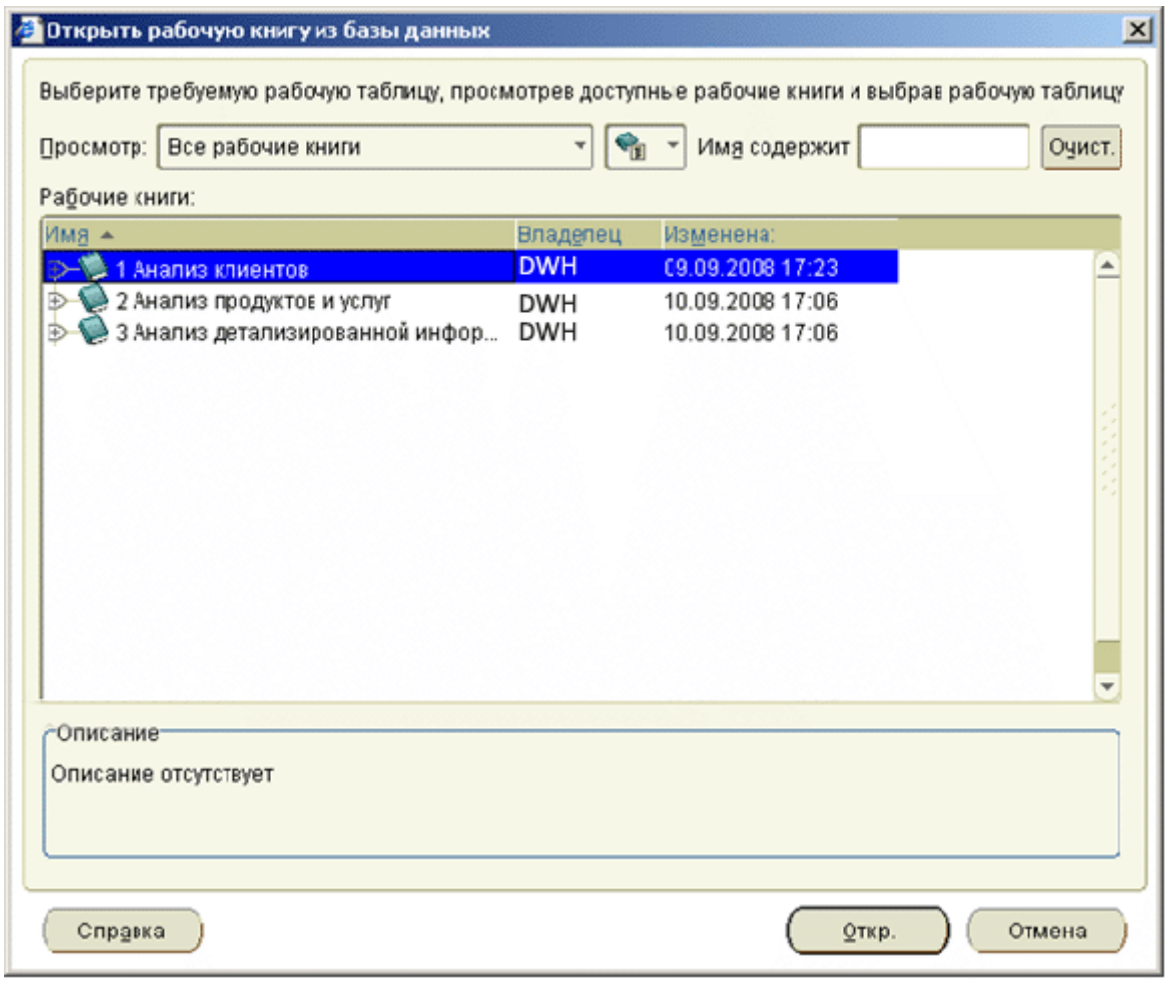
Условия, при соблюдении которых возможно выполнение операции:
Успешная регистрация на Портале ИАС КХД.

Подготовительные действия:
Не требуются.

Основные действия в требуемой последовательности:
1. В появившемся окне «Мастер создания рабочих книг» поставить точку напротив пункта «Открыть существующую рабочую книгу».



2. Выбрать нужную рабочую книгу и нажать кнопку «Откр.»:



Заключительные действия:

После завершения работы с отчетом необходимо выбрать пункт меню «Файл», далее выбрать пункт «Закрыть».

Ресурсы, расходуемые на операцию:

15 секунд.

Задача: «Формирование табличных и графических форм отчетности»

Заполняется по аналогии.

5. Аварийные ситуации

В разделе "Аварийные ситуации" указывают: 1. действия в случае несоблюдения условий выполнения технологического процесса, в том числе при длительных отказах технических средств; 2. действия по восстановлению программ и/или данных при отказе магнитных носителей или обнаружении ошибок в данных; 3. действия в случаях обнаружении несанкционированного вмешательства в данные; 4. действия в других аварийных ситуациях.

В случае возникновения ошибок при работе ИАС КХД, не описанных ниже в данном разделе, необходимо обращаться к сотруднику подразделения технической поддержки ДИТ (HelpDesk) либо к ответственному Администратору ИАС КХД.

Класс ошибки	Ошибка	Описание ошибки	Требуемые

			действия пользователя при возникновении ошибки
Портал ИАС КХД	Сервер не найден. Невозможно отобразить страницу	Возможны проблемы с сетью или с доступом к порталу ИАС КХД.	Для устранения проблем с сетью обратиться к сотруднику подразделения технической поддержки (HelpDesk). В других случаях к администратору ИАС КХД.
	Ошибка: Требуется ввести действительное имя пользователя	При регистрации на портале ИАС КХД не введено имя пользователя.	Ввести имя пользователя.
	Ошибка: Требуется ввести пароль для регистрации	При регистрации на портале ИАС КХД не введен пароль.	Ввести пароль.
	Ошибка: Сбой аутентификации. Повторите попытку	Неверно введено имя пользователя или пароль, либо такая учетная запись не зарегистрирована.	Нужно повторить ввод имени пользователя и пароля, однако после третей неудачной попытки регистрации учетная запись блокируется. Если учетная запись заблокирована, нужно обратиться к администратору ИАС КХД.
Сбой в электропитании рабочей станции	Нет электропитания рабочей станции или произошел сбой в электропитании.	Рабочая станция выключилась или перезагрузилась.	Перезагрузить рабочую станцию. Проверить доступность сервера ИАС КХД по порту 80, выполнив следующие команды: - нажать кнопку «Пуск» - выбрать пункт «Выполнить» - в строке ввода набрать команду telnet ias_dwh.ru 80 - если открылось окно Telnet, значит соединение возможно. Повторить попытку подключения (входа) в ИАС КХД
Сбой локальной сети	Нет сетевого взаимодействия между рабочей станцией и сервером приложений ИАС КХД	Отсутствует возможность начала (продолжения) работы с ИАС КХД. Нет сетевого подключения к серверу ИАС КХД	Перезагрузить рабочую станцию. Проверить доступность сервера ИАС КХД по порту 80, выполнив следующие команды: - нажать кнопку «Пуск» - выбрать пункт «Выполнить» - в строке ввода набрать команду telnet ias_dwh.ru 80 - если открылось окно Telnet, значит соединение возможно.

			После восстановления работы локальной сети повторить попытку подключения (входа) в ИАС КХД.

6. Рекомендации по освоению

В разделе "Рекомендации по освоению" указывают рекомендации по освоению и эксплуатации, включая описание контрольного примера, правила его запуска и выполнения.

Практическое задание. Используя вышеприведенный шаблон, выполните следующие действия:

1. Выберите для руководства пользователя своей программы (см. предметную область) только необходимые разделы и пункты.
2. Составьте руководство оператора (пользователя) программы.
3. Составьте руководство пользователя.

Лабораторная работа №24. Разработка (подготовка) документации и отчетных форм для внедрения программных средств

Теоретический материал.

Разработка программной документации выполняется в соответствии с ГОСТ 19.106-78 «Требования к программным документам, выполненным печатным способом»

Этот стандарт устанавливает правила выполнения программных документов для вычислительных машин, комплексов и систем независимо от их назначения и области применения и предусмотренных стандартами ЕСПД.

Общие требования. Вписывать в программные документы, выполненные машинописным, машинным и рукописным способами, отдельные слова, формулы, условные знаки (от руки чертежным шрифтом), буквы латинского и греческого алфавитов, а так же выполнять схемы и рисунки необходимо черными чернилами или тушью.

Опечатки и графические неточности, обнаруженные в процессе выполнения допускается исправлять подчисткой некачественно выполненной части текста (чертежа) и нанесением на том же листе исправленного текста (графики) машинописью или черной тушью в зависимости от способа выполнения документа.

Повреждение листов документов, помарки и следы не полностью удаленного текста (графики) не допускаются.

Программные документы оформляют на листах формата А4. Кроме того:

10. - допустимо оформление на листах формата А3;

11. - при машинном способе выполнения документа допускаются отклонения размеров листов, соответствующих форматам А4 и А3, определяемые возможностями применяемых технических средств; на листах форматов А4 и А3, предусматриваемых

выходными характеристиками устройств вывода данных, при изготовлении документа машинным способом;

12. - при изготовлении документа типографским способом возможно использование листов типографических форматов.

Расположение материалов программного документа осуществляется в следующей последовательности:

- 1. титульная часть:
  - - лист утверждения (не входит в общее количество листов документа);
  - - титульный лист (первый лист документа);
  - - информационная часть:
  - - аннотация;
  - - лист содержания;
- 2. основная часть:
  - - текст документа (с рисунками, таблицами и т.п.);
  - - перечень терминов и их определений;
  - - перечень сокращений;
  - - приложения;
  - - предметный указатель;
  - - перечень ссылочных документов;
- 3. часть регистрации изменений:
  - - лист регистрации изменений.

Построение документа. При необходимости допускается делить документ на части. Деление на части осуществляется на уровне не ниже раздела. Каждую часть комплектуют отдельно, при этом в конце содержания первой части следует перечислить названия остальных частей.

Допускается включение в документ частей текста программы, оформляемых в соответствии с правилами языка, на котором написан текст программы.

Аннотацию размещают на отдельной странице (страницах), снабжают заголовком "АННОТАЦИЯ", нумеруют и включают в содержание документа.

Содержание документа размещают на отдельной странице (страницах), снабжают заголовком "СОДЕРЖАНИЕ" и включают в общее количество страниц документа. В содержании документа дается перечисление наименований разделов и подразделов и номеров страниц.

Текст каждого документа, при необходимости, разбивается на пункты, а пункты - на подпункты, независимо от того, разделен документ на части, разделы и подразделы или нет.

Заголовки разделов пишут прописными буквами и размещают симметрично относительно правой и левой границ текста. Заголовки подразделов записывают с абзаца

строчными буквами (кроме первой прописной). Переносы слов в заголовках не допускаются. Точку в конце заголовка не ставят. Каждый раздел рекомендуется начинать с нового листа.

Разделы, подразделы, пункты и подпункты следует нумеровать арабскими цифрами с точкой. Разделы должны иметь порядковый номер (1, 2 и т.д.)

Текст документа.

Текст документа должен быть кратким, четким, исключающим возможность неверного толкования. Термины и определения должны быть едиными и соответствовать установленным стандартам, а при их отсутствии - общепринятым в научно-технической литературе, и приводиться в перечне терминов.

Необходимые пояснения к тексту документа могут оформляться сносками. Сноска обозначается цифрой со скобкой, вынесенной на уровень линии верхнего обреза шрифта.

Если сноска относится к отдельному слову, знак сноски помещается непосредственно возле этого слова, если же к предложению целом, то в конце предложения. Текст сноски располагают в конце страницы и отделяют от основного текста линией длиной 3 см, проведенной в левой части страницы.

Иллюстрации.

Иллюстрации могут быть расположены в тексте документа и (или) в приложениях. Иллюстрации, если их в данном документе более одной, нумеруют арабскими цифрами в пределах всего документа.

В приложениях иллюстрации нумеруются в пределах каждого приложения в порядке, установленном для основного текста документа. Ссылки на иллюстрации дают по типу: "рис.12" или "(рис.12)". Иллюстрации могут иметь тематический заголовок и подрисуночный текст, поясняющий содержание иллюстрации.

Формулы.

Формулы в документе, если их более одной, нумеруются арабскими цифрами, номер ставят с правой стороны страницы, в скобках на уровне формулы. В пределах всего документа или его частей, в случае деления документа на части, формулы имеют сквозную нумерацию.

Ссылки в тексте на порядковый номер формулы дают в скобках, например: "в формуле (3)". При делении документа на части номер части ставится перед порядковым номером формулы и отделяется от последней точкой, например: "в формуле (1.4)".

Значение символов, входящих в формулу, должны быть приведены непосредственно под формулой. Значение каждого символа печатают с новой строки в той последовательности, в какой они приведены в формуле. Первая строка расшифровки должна начинаться со слова "где", без двоеточия после него.

Ссылки.

В программных документах допускаются ссылки на стандарты и другие документы. Ссылаться следует на документ в целом или на его разделы (с указанием обозначения и наименования документа, номера и наименования раздела или приложения).

Допускается указывать только обозначение документа и (или) разделов без указания их наименований. Ссылки на отдельные подразделы, пункты и иллюстрации другого документа не допускаются. Допускаются ссылки внутри документа на пункты, иллюстрации и отдельные подразделы.

Примечания.

В примечаниях к тексту и таблицам указывают только справочные и пояснительные данные. Одно примечание не нумеруется. После слова "Примечание" ставят точку. Несколько примечаний следует нумеровать по порядку арабскими цифрами с точкой. После слова "Примечание" ставят двоеточие. Текст примечаний допускается печатать только через один интервал.

Сокращения.

Сокращения слов в тексте и надписях под иллюстрациями не допускаются, за исключением:

- - сокращений, установленных в ГОСТ 2.316-68, и общепринятых в русском языке;
- - сокращений, применяемых для обозначения программ, их частей и режимов работы, в языках управления заданиями, в средствах настройки программы и т.п., обозначаемых буквами латинского алфавита.

Приложения.

Иллюстрированный материал, таблицы или текст вспомогательного характера допускается оформлять в виде приложений. Приложения оформляют как продолжение данного документа на последующих страницах или выпускают в виде отдельного документа.

Каждое приложение должно начинаться с новой страницы с указанием в правом верхнем углу слова "Приложение" и иметь тематический заголовок. При наличии в документе более одного приложения все приложения нумеруют арабскими цифрами (без знака №), например: Приложение 1, Приложение 2 и т.д.

При выпуске приложения отдельным документом, на титульном листе под наименованием документа следует указывать слово "Приложение", а при наличии нескольких приложений указывают также его порядковый номер.

Практическое задание.

1. Используя ранее выбранную предметную область и результаты работ №21, 22 и 23, составьте реестр документов, которые были написаны в ходе внедрения программного обеспечения.
2. Оформите реестр согласно стандарту ГОСТ 19.106-78.
3. Зафиксируйте, какие ещё документы могли быть написаны в ходе внедрения программного продукта и в ваш реестр не попали.
4. Составьте акт по внедрению программного обеспечения согласно предметной области.

Лабораторная работа №25. Описание процесса сопровождения.

Основные виды работ по сопровождению

Теоретический материал.

Сопровождение программного обеспечения (или поддержка) – это процесс улучшения и оптимизации программного обеспечения (ПО) перед сдачей в эксплуатацию. Сопровождение ПО – это одна из фаз процесса разработки программного обеспечения, следующая за фазой реализации основного технического задания.

В ходе поддержки исправляются обнаруженные дефекты и недоработки. Также добавляется новая функциональность, вносятся изменения для повышения удобства использования (юзабилити) программы.

Услуги по поддержке программного обеспечения включают в себя такие работы как:

Исправление ошибок и устранение неполадок, не выявленных ранее.

Оптимизация работы программы при различных условиях эксплуатации.

Обновление и доработка по требованиям Заказчика.

Профилактические работы по обслуживанию баз данных информационной системы.

Подготовка технической и пользовательской документации.

Обновление модулей программы и используемых библиотек с учетом современных технологий.

Работы по сопровождению программного обеспечения проводятся в тесном контакте с сотрудниками заказчика, что позволяет более динамично развивать программное обеспечение, оперативно изменяя приоритеты разработки. Также сокращается время, необходимое на согласование плана работ, поскольку дополнения и исправления обычно несут менее глобальный характер, чем при разработке ядра программы.

Управление обслуживанием программных продуктов

Сопровождение программного обеспечения — процесс улучшения, оптимизации и устранения дефектов программного обеспечения (ПО) после передачи в эксплуатацию. Сопровождение ПО — это одна из фаз жизненного цикла программного обеспечения, следующая за фазой передачи ПО в эксплуатацию. В ходе сопровождения в программу вносятся изменения, с тем, чтобы исправить обнаруженные в процессе использования дефекты и недоработки, а также для добавления новой функциональности, с целью повысить удобство использования (юзабилити) и применимость ПО.

В модели водопада, сопровождение ПО выделяется в отдельную фазу цикла разработки. В спиральной модели, возникшей в ходе развития объектно-ориентированного программирования, сопровождение не выделяется как отдельный этап. Тем не менее, эта деятельность занимает значительное место, учитывая тот факт, что обычно около 2/3 жизненного цикла программных систем занимает сопровождение.

Сопровождаемость программного обеспечения — характеристики программного продукта, позволяющие минимизировать усилия по внесению в него изменений:

· для устранения ошибок;

· для модификации в соответствии с изменяющимися потребностями пользователей.

Структура ИТ-сопровождения

Принято выделять несколько линий сопровождения (структура приведена на примере внешнего сопровождения ПО):

• 0 линия (call-center, информационный центр, горячая линия) - обработка телефонных обращений от клиентов, передача обращений техническим специалистам (1-ая линия сопровождения)

• 1 линия (инженер по сопровождению, инженер технической поддержки, support engineer) – консультация/настройка/устранение ошибок в работе ПО/наполнение базы знаний, составление мануалов

• 2 линия (инженер по сопровождению, инженер технической поддержки, support engineer) функциональное сопровождение/проектная деятельность на этапе запуска ПО на машинах заказчика

• 3 линия (инженер по сопровождению, инженер технической поддержки, support engineer) - системное сопровождение/проектная деятельность на этапе запуска ПО на оборудовании заказчика

Работу инженера по сопровождению ошибочно сравнивают с работой информационного центра. Однако по функционалу эти специалисты принципиально различаются – если call-center фактически аккумулирует обращения пользователей, то сопровождение является центральным звеном в цепочке разработки и доработки ПО, которое решает проблемы, возникающие в период эксплуатации ПО (системы, сервиса).

Современная ИТ инфраструктура крупной компании включает в себя все большее количество программных систем, необходимых для эффективной реализации большинства бизнес-процессов. В программное обеспечение входит как набор прикладных программ, использующихся в работе каждого отдельного сотрудника, так и комплексные корпоративные информационные системы, автоматизирующие процессы предприятия. Информационные системы класса CRM, ERP, SCM, и другие, корпоративные порталы, системы управления персоналом и аналитические системы управления предприятием, а также специализированные информационные системы становятся важным элементом, обеспечивающим эффективность компании. Поэтому задача обеспечения бесперебойной работы ключевых приложений, администрирование информационных систем и техническая поддержка являются одной из важнейших задач ИТ подразделения. Обслуживание каждой системы требует наличия высококвалифицированного и узкоспециализированного персонала, значительных организационных усилий и затрат.

Программные средства являются одним из наиболее гибких видов промышленных изделий и эпизодически подвергаются изменениям в течение всего времени их использования.

Иногда достаточно при корректировке программного обеспечения внести только одну ошибку для того, чтобы резко снизилась его надежность или его корректность при некоторых исходных данных.

Для сохранения и повышения качества программного обеспечения необходимо регламентировать процесс модификации и поддерживать его соответствующим тестированием и контролем качества. В результате программное изделие со временем обычно улучшается как по функциональным возможностям, так и по качеству решения отдельных задач.

Работы, обеспечивающие контроль и повышение качества, а также развитие функциональных возможностей программ, составляют процесс сопровождения.

В процессе сопровождения в программное обеспечение вносятся следующие изменения, значительно различающиеся причинами и характеристиками;

- исправление ошибок - корректировка программ, выдающих неправильные результаты в условиях, ограниченных техническим заданием и документацией. Исправление ошибок требуют около 20% общих затрат на сопровождение.

- регламентированная документами адаптация программного обеспечения к условиям конкретного использования, с учетом характеристик внешней среды или конфигурации аппаратуры, на которой предстоит функционировать программам. Адаптация занимает около 20% общих затрат на сопровождение.

- модернизация - расширение функциональных возможностей или улучшение характеристик решения отдельных задач в соответствии с новым или дополнительным техническим заданием на программное изделие. Модернизация занимает до 60% общих затрат на сопровождение.

Первый вид изменений (исправление ошибок) является непредсказуемым и его трудно регламентировать.

Остальные виды корректировок носят упорядоченный характер и проводятся в соответствии с заранее подготавливаемыми планами и документами. Эти корректировки в наибольшей степени изменяют программные изделия и требуют наибольших затрат.

Поэтому изменения, обусловленные ошибками, в большинстве случаев целесообразно по возможности накапливать и реализовывать их, приурочивая к изменениям, регламентированным модернизациями.

Однако некоторые ошибки вызывают необходимость срочного исправления программ. В этих случаях допустимо некоторое отставание корректировки документации при более срочном и регистрируемом исправлении самих программ.

Сопровождение программ — это «ложка дегтя» для каждого программиста, всегда помеха при начале разработки какого-либо нового проекта, заставляющая отвлекаться от его разработки и возвращаться к старым программам и старым проблемам.

Что делает сопровождение программного обеспечения крайне непривлекательным? Это плохо документированный код, недостаточно полное начальное проектирование и отсутствие внешней документации.

Если все этапы жизненного цикла разработки программного обеспечения выполнялись правильно, то сопровождение не будет вызывать серьезных проблем, а будет элементарной технической поддержкой и модификацией внедренного программного продукта.

Со временем, иногда через десятки лет, сопровождение программного обеспечения прекращается. Это может быть обусловлено: разработкой более совершенных программных средств; прекращением использования сопровождаемого программного продукта; нерентабельным возрастанием затрат на его сопровождение.

Отметим, однако, что программное изделие может долго применяться кем-либо и после прекращения его. сопровождения от лица разработчика, потому, что этот некто может плодотворно использовать программное изделие у себя самостоятельно, без помощи разработчика.

Для того чтобы со временем прийти к обоснованному решению о прекращении сопровождения программного обеспечения, необходимо периодически оценивать эффективность его эксплуатации, возможный ущерб от отмены сопровождения. В некоторых случаях решение о прекращении сопровождения принимается при противодействии со стороны отдельных пользователей.

Процессы поддержки

RUP предусматривает три процесса поддержки:

· Управление проектом;

· Управление конфигурацией;

· Управление средой.

Управление проектом. Цели

Основной целью процесса управления проектом является обеспечение руководства проектом, направленное на успешную сдачу программного продукта. RUP акцентирует внимание на планировании жизненного цикла и отдельных итераций, управлении рисками, наблюдаемости хода проекта и метриках проекта.

Планирование предполагает созданий двух видов планов:

· План фаз – крупномасштабный план проекта, показывающий основные вехи жизненного цикла (даты завершения больших этапов – фаз, выпуска эволюционных прототипов и т. д.) и требуемые ресурсы. Он создается в начале фазы исследования и обновляется по мере необходимости.

· План итераций создается для каждой итерации и предназначается для определения и распределения задач между участниками проекта.

Риском будем называть все то, что может стоять на пути к успеху проекта и на данный момент является неизвестным или неопределенным. Главная идея управления рисками заключается в том, что не нужно пассивно ждать, пока риск станет проблемой, но требуется заблаговременно определять линию поведения. Управление рисками означает определение и оценку рисков, принятие линии поведения, направленной на устранение, снижение вероятности риска, а также выбор действий на случай реализации риска.

Для контроля и управления проектом используются **измерения** . Они проводятся для того чтобы установить, насколько отдалено текущее состояние проекта от поставленной цели, спланировать работы и определить, как можно повысить эффективность процесса.

Управление проектом. Деятельности

Открытие нового проекта. Эта деятельность выполняется только в первой итерации. Осуществляется инициализация проекта, определяются и оцениваются риски, разрабатывается бизнес-план. Цель – перевод проекта в стадию, когда возможно принятие решения о продолжении или об отказе от проекта.

Оценка области действия проекта и рисков. Целью является пересмотр и уточнение возможностей и характеристик проекта, а также связанных с ним рисков.

Создание плана разработки ПС. Создается план разработки ПС, включающий перечень рисков, планы измерений, управления рисками, разрешения проблем, принятия продукта. Определяется структура и ресурсы проекта. Разрабатывается план фаз проекта.

Планирование итерации. Разрабатывается план очередной итерации, уточняется и корректируется бизнес-план и план разработки ПС. План итерации подробно описывает, что должно произойти за время итерации. Он определяет исполнителей и выполняемые ими работы. При создании плана итерации необходимо:

· Сформулировть объективные критерии успеха итерации. Они будут использоваться при ее оценке;

· Определить конкретные, измеримые артефакты, которые требуется разработать или изменить, а также выполняемые для этого работы;

· Использовать типичную итерационную декомпозицию работ для реальных действий, которые должны быть произведены;

· Использовать смету при определении продолжительности и объема работ для каждого вида деятельности, удерживая все значения в пределах бюджета.

Наблюдение и контроль. Эта деятельность включает распределение работ и создание графика работ, наблюдение за состоянием проекта, обработку исключительных ситуаций, и создание отчета о состоянии. Выполняется обработка предложенных изменений и включение их в график работ текущей или последующих итераций, производится наблюдение за активными рисками, дается оценка прогресса и качества проекта. В RUP постоянно выполняемые оценки состояния проекта являются основой для решения разных проблем и управления рисками проекта. Если команда разработчиков выявила проблему, назначается сотрудник, ответственный за ее разрешение, и определяется дата, когда проблема должна быть разрешена. Ход процесса должен регулярно контролироваться, а обновления должны выполняться по мере необходимости.

Управление итерацией. Целью этой деятельности является получение достаточных для выполнения итерации ресурсов, разделение требуемых работ, оценка результатов итерации.

Завершение фазы. Выполняются работы, завершающие выполнение фазы. Даются ответы на следующие основные вопросы:

· Решены ли все основные проблемы предыдущей итерации?

· Известно ли состояние всех основных артефактов (см. ниже управление конфигурацией)?

· Рассмотрены ли все проблемы развертывания?

При удовлетворительном состоянии проекта выдается разрешение на переход к следующей фазе.

Завершение проекта. Руководитель проекта подготавливает проект к завершению. Готовится заключительная оценка состояния. При успешной сдаче проекта заказчик получает программный продукт в пользование. Высвобождающие ресурсы могут быть перераспределены (использованы в других проектах).

Управление конфигурацией и изменениями. Цели

Целью управления конфигурацией и изменениями является поддержание целостности артефактов с учетом возможности внесения в них изменений в соответствии с запросами на изменения. Этот вспомогательный процесс распространяется на весь жизненный цикл программного продукта и складывается из трех отдельных процессов.

Управление конфигурацией (Configuration management).

Процесс управления конфигурацией (УК) связан с определением артефактов, зависимостей между ними и конфигураций, являющихся непротиворечивыми множествами взаимосвязанных артефактов. Сюда же относятся вопросы распределения рабочих сред между участниками проекта с целью предотвращения ненужного дублирования документов. УК декларирует, что все артефакты подчиняются **версионному управлению.** По мере развития проекта у каждого артефакта появляются

множество **версий** . Все они сохраняются в **архиве проекта** , причем ведется история изменений, в которой указывается причины и цели создания каждой версии каждого артефакта. Записанную в архив проекта версию нельзя изменить, можно лишь создать новую версию. Кроме того, знания о зависимостях между артефактами позволяют точно повторять действия при создании наборов артефактов (например, при сборке программы из отдельных файлов). Создание версий и связей поддерживается с помощью инструментальных средств УК.

УК позволяет:

· исключить возможность потери артефактов,

· обеспечить возможность «возврата назад» в случаях, когда принятые решения и полученные при их реализации версии артефактов не устраивают заказчика или разработчиков проекта,

· гарантировать точное повторение действий над группой связанных артефактов в итерациях,

· избегать дублирования артефактов и связанной с этим возможности привнесения дополнительных дефектов,

· поддерживать наблюдаемость хода выполнения работ по проекту путем предоставления нужной информации.

Управление внесением изменений

Деятельности этого процесса направлены на сбор и сохранение **запросов на внесение изменений** , поставляемых как участниками разработки ПС, так и внешними сторонами. Запросы на внесение изменений могут появляться по разным причинам (исправление дефекта, улучшение параметра качества продукта, например, производительности, задание дополнительных требований и т. д.) Каждый запрос на внесение изменений сохраняется в **базе данных**проекта и может находиться в разных **состояниях** в зависимости от хода выполнения работ по этому запросу (новый, зарегистрированный, принятый, завершенный и др.). Состояния запросов изменяются участниками проекта в соответствии с выделенными правами. По мере работы с запросом в него добавляется информация (например, сначала это может быть только описание дефекта, затем добавляется результаты анализа, указываются затрагиваемые артефакты, назначается исполнитель). С запросом может быть связан **приоритет** , позволяющий определить порядок выполнения работ по запросам на изменения.

Управление состоянием и измерениями

Этот процесс связан с управлением проектом и направлен на предоставление информации, полезной для оценки:

· Состояния, прогресса, общих тенденций и качества продукта;

· Издержек и расходов;

· Проблемных областей, требующих внимания;

· Того, что сделано и что необходимо сделать.

Вся необходимая информация находится в базе данных и архиве проекта. Руководитель работ оценивает состояние дел путем непосредственного просмотра запросов, истории версий артефактов или на основе анализа различных отчетов, формируемых инструментальным средством УК. Анализ позволяет руководителю оценить имеющиеся ресурсы и принять необходимые управленческие решения.

Деятельности

Планирование конфигурации проекта и управления изменениями. Описываются все виды деятельности, связанные с УК, которые надо выполнить. Описывается процесс контроля над изменениями.

Создание среды УК. Целью этой деятельности является создание рабочей среды, в которой будут доступны все артефакты, будет обеспечена разработка, интеграция и архивирование продукта.

Создание рабочей среды исполнителей. В пределах своей рабочей среды исполнитель получает доступ к артефактам проекта, имеет возможность вносить в них изменения и предлагать внесение изменений в проект в целом. Изменения, сделанные отдельными исполнителями, направляются в рабочую среду интеграции.

Управление версиями. При записи артефакта в архив проекта формируется его новая версия. Управление версиями предусматривает обязательное указание причин и целей создания версии. При выборе артефакта из архива можно указать конкретную версию.

Управление запросами на внесение изменений. Целью этой деятельности является обеспечение последовательного внесения изменений в проект и информирование о состоянии продукта, его изменениях и о влиянии изменений на стоимость и сроки выполнения проекта.

Наблюдение за состоянием конфигурации. Наблюдаемость процесса разработки обеспечивается путем доступа руководства проекта к хранимым артефактам и запросам на изменения, а также путем предоставления отчетов о состоянии конфигурации. При этом инструментальные средства, поддерживающие управление конфигурацией и изменениями, могут выполнять требуемые измерения. Например, можно показать процент завершенных запросов, число открытых запросов высокого приоритета и т. д.

Управление средой. Цели

Многие виды деятельностей, предусмотренные RUP, могут быть автоматизированы посредством инструментальных средств, что позволит избежать наиболее трудоемких, напряженных и подверженных ошибкам аспектов разработки ПС.

Цель процесса управления средой – процедурная и инструментальная поддержка организации, разрабатывающей ПС. Она включает:

· Выбор и приобретение инструментальных средств;

· Настройку инструментальных средств под требования организации-разработчика;

· Конфигурирование процесса;

· Усовершенствование процесса;

· Создание технических служб поддержки.

Роли и артефакты

Основным исполнителем процесса является **технолог** . В его обязанности входит конфигурирование процесса перед запуском проекта и усовершенствование процесса в ходе проектных работ. Поддержка аппаратной и программной среды разработки ложится на плечи **системного администратора** .

Главным создаваемым артефактом является **план разработки** , описывающий процесс, используемый в данном проекте. В нем указывается, какие артефакты, предусматриваемые в RUP, будут использоваться в проекте и каким образом.

Деятельности

Подготовка среды к реализации проекта. Выполняется оценка текущего состояния организации-разработчика и имеющейся инструментальной поддержки. Создается предварительный план разработки. Составляется перечень инструментальных средств, которые предполагается использовать при разработке. Составляется перечень шаблонов для производства основных артефактов.

Подготовка среды к итерации. Производится дополнение и уточнение плана разработки, выполняется подготовка и настройка инструментальных средств, которые будут использоваться в итерации. Составляется набор шаблонов для артефактов, создаваемых в итерации. Осуществляется подготовка сотрудников к использованию инструментальных средств.

Подготовка руководящих директив. Для каждого основного технологического процесса подготавливается набор директив на основе анализа проблем и дефектов предыдущих итераций.

В состав документации сопровождения управления конфигурацией входят:

13. ?отчеты пользователей о?выявленных дефектах и?предложения по корректировке программ;
14. ?журнал выявленных дефектов и?предложений по совершенствованию и?развитию версии ППП;
15. ?журнал подготовленных и?утвержденных корректировок, а?также реализованных изменений в?новой версии пакета;
16. ?отчет о?результатах эксплуатации снятой с?сопровождения версии пакета;
17. ?журнал тиражирования и?характеристик базовых версий, поддерживаемых сопровождением [2].

Пользовательская документация включает в?себя:

- ?паспорт на программное средство, где содержатся общие сведения о?ППП, его основные характеристики, комплектность, акт о?приемке, гарантии изготовителя (поставщика);

- ?общее описание информационной системы (ИС), в?составе которой будет использоваться ППП (назначение и?описание ИС, описание взаимосвязей ППП с?другими составляющими ИС);
- ?руководство администратора программного средства, которое регламентирует функции администрирования, процедуры по инсталляции и?подготовке ППП к?эксплуатации, порядок и?средства ведения базы данных и?восстановления информации при сбоях;
- ?руководства оперативных пользователей с?требованиями к?уровню подготовки пользователя, описание функций. Описан порядок подготовки ППП к?работе и?действия пользователя в?аварийных ситуациях, приведены рекомендации по освоению пакета, включая описание контрольного примера, правила его запуска и?выполнения [8].
-

В процессе сопровождения в?программы вносятся различные изменения:

- ?исправление ошибок - корректировка программ, выдающих неправильные результаты в?условиях, ограниченных техническим заданием и?документацией;
- ?модернизация - расширение функциональных возможностей или улучшение качества решения отдельных задач в?соответствии с?новым или дополнительным техническим заданием на ППП;
- ?адаптация, регламентированная документацией, к?условиям конкретного использования, обусловленным характеристиками внешней среды или конфигурацией аппаратуры [2].

Документация по сопровождению ПС (*system documentation*) описывает ПС с точки зрения ее разработки. Эта документация необходима, если ПС предполагает изучение того, как оно устроена (сконструирована), и модернизацию его программ. Как уже отмечалось, сопровождение - это продолжающаяся разработка. Поэтому в случае необходимости модернизации ПС к этой работе привлекается специальная команда разработчиков-сопроводителей. Этой команде придется иметь дело с такой же документацией, которая определяла деятельность команды первоначальных (основных) разработчиков ПС, - с той лишь разницей, что эта документация для команды разработчиков-сопроводителей будет, как правило, чужой (она создавалась другой командой). Чтобы понять строение и процесс разработки модернизируемого ПС, команда разработчиков-сопроводителей должна изучить эту документацию, а затем внести в нее необходимые изменения, повторяя в значительной степени технологические процессы, с помощью которых создавалось первоначальное ПС.

Документация по сопровождению ПС можно разбить на две группы:

1) документация, определяющая строение программ и структур данных ПС и технологию их разработки;

2) документацию, помогающую вносить изменения в ПС.

Документация первой группы содержит итоговые документы каждого технологического этапа разработки ПС. Она включает следующие документы:

· Внешнее описание ПС (Requirements document).

· Описание архитектуры ПС (description of the system architecture), включая внешнюю спецификацию каждой ее программы (подсистемы).

· Для каждой программы ПС - описание ее модульной структуры, включая внешнюю спецификацию каждого включенного в нее модуля.

· Для каждого модуля - его спецификация и описание его строения (design description).

· Тексты модулей на выбранном языке программирования (program source code listings).

· Документы установления достоверности ПС (validation documents), описывающие, как устанавливалась достоверность каждой программы ПС и как информация об установлении достоверности связывалась с требованиями к ПС.

Документы установления достоверности ПС включают, прежде всего, документацию по тестированию (схема тестирования и описание комплекта тестов), но могут включать и результаты других видов проверки ПС, например, доказательства свойств программ. Для обеспечения приемлемого качества этой документации полезно следовать общепринятым рекомендациям и стандартам.

Документация второй группы содержит:

· *Руководство по сопровождению ПС* (system maintenance guide), которое описывает особенности реализации ПС (в частности, трудности, которые пришлось преодолевать) и как учтены возможности развития ПС в его строении (конструкции). В нем также фиксируются, какие части ПС являются аппаратно- и программно-зависимыми.

Общая проблема сопровождения ПС - обеспечить, чтобы все его представления шли в ногу (оставались согласованными), когда ПС изменяется. Чтобы этому помочь, связи и зависимости между документами и их частями должны быть отражены в руководстве по сопровождению, и зафиксированы в базе данных управления конфигурацией.

Практическое задание.

1. Используя ранее выбранную предметную область, составьте по образцу руководство по сопровождению вашего программного продукта в ходе эксплуатации.
2. Используя ранее выбранную предметную область, определите, какие должны быть обновления вашего программного продукта.
3. Составьте план сопровождения вашего программного продукта.
4. Определите, какие в программе предусмотрены настройки для оптимизации её работы в различных условиях эксплуатации (разные характеристики рабочих ПК и возможность коллективного использования в качестве клиента или сервера).
5. Выполните основные операции по обслуживанию ПК (оптимизацию системного реестра, проверку и дефрагментацию жесткого диска, создание контрольных точек восстановления системы).

Лабораторная работа №26. Составление плана работ по сопровождению

Теоретический материал.

План обеспечения сопровождения и поддержки — подробные сведения

В рамках плана обеспечения сопровождения и поддержки клиентам предоставляются следующие возможности:

Приоритетная техническая поддержка по электронной почте, телефону или факсу по вопросам установки и настройки систем.

Бесплатные лицензии на обновления для активации эквивалентных функций в любой новой версии программного обеспечения с датой выпуска в рамках периода действия плана обеспечения сопровождения и поддержки.

При отказе от обновления до новой версии пользователи продолжают получать приоритетную техническую поддержку до тех пор, пока компания Cogent не прекратит предоставлять поддержку для данной версии продукта.

На индивидуальной основе возможно предоставление особых условий по обеспечению технической поддержки для продуктов, снятых с производства.

План обеспечения сопровождения и поддержки не распространяется на следующие позиции:

Разработка пользовательских скриптов DataHub, страниц WebView, ASP, AJAX или апплетов Java.

Отладка скриптов, макросов Excel или других пользовательских приложений.

Услуги консалтинга по проектированию систем и планированию проектов.

Техническая поддержка на объекте.

Сборы за подписку и продление

- Сбор за план обеспечения сопровождения и поддержки составляет 20% от прейскурантной цены приобретенных программных лицензий.
- Датой начала срока действия плана обеспечения сопровождения и поддержки является дата приобретения первой лицензии.
- Дата продления плана обеспечения сопровождения и поддержки наступает через 12 месяцев с даты начала предоставления поддержки. План продлевается ежегодно в один и тот же день.

- Ежегодный сбор за продление также рассчитывается по ставке 20% от текущей прейскурантной цены приобретенных программных лицензий.
- Уведомления о продлении отправляются приблизительно за 30 дней до даты продления.

Добавление будущих приобретений в план обеспечения сопровождения и поддержки

- Для обеспечения эффективного управления технической поддержкой и обновления продуктов после подписки на план обеспечения сопровождения и поддержки вы должны добавлять в него все приобретенные впоследствии программные лицензии.
- Все новые лицензии регистрируются с той же датой продления в плане обеспечения сопровождения и поддержки, что и исходная покупка программного обеспечения соответственно. Таким образом, для всех лицензий поддержка будет продлена в один день.
- Когда подойдет время продления плана обеспечения сопровождения и поддержки, плата за продление на следующий год будет рассчитываться на основе общей стоимости программных лицензий, зарегистрированных в плане.

Добавление прошлых приобретений в план обеспечения сопровождения и поддержки

- Вы можете без труда добавлять в план обеспечения сопровождения и поддержки лицензии, приобретенные ранее.
- Сбор за добавление в план обеспечения сопровождения и поддержки старых лицензий составляет 20% от текущей прейскурантной цены за программное обеспечение.
- Сбор за добавление в план обеспечения сопровождения и поддержки старых лицензий пропорционально распределяется с учетом даты приобретения лицензий.

Прочие условия

- Компенсация клиентам, отменившим подписку раньше даты продления, не выплачивается.

Будущие изменения плана обеспечения сопровождения и поддержки

Компания сохраняет за собой право вносить изменения в условия плана обеспечения сопровождения и поддержки в любое время. Для существующих клиентов все изменения вступают в силу с момента следующего продления соглашения.

Организация процесса сопровождения подразумевает выполнение следующих действий:

- определение цели и состава процессов сопровождения;
- определение причин и видов изменения программного средства в процессе его сопровождения;
- организация процессов и передача на сопровождение разработанного программного средства;
- заключение договора между заказчиком и исполнителем на сопровождение программного средства;
- разработка концепции методов и процессов сопровождения ПП;
- разработка спецификации требований на модификации при сопровождении программного средства;
- утверждение заказчиком концепции, договора и технического задания на сопровождение ПП;
- организация контроля реализации концепции и договора на сопровождение программного средства.

Данная последовательность действий рассматривается с позиции концепции сопровождения, в которой учитывается и область сопровождения и изменений программной системы, выраженная в виде совокупности тех или иных видов работ.

Существует несколько *подходов к организации процесса сопровождения.* Один из них можно представить в виде схемы.

Здесь пп. 1.1 — 1.4 фактически относятся к этапу разработки и подразумевают под собой попытку предугадать направления будущих изменений требований к ПП и учесть их в плане. Пункт 1.1 можно реализовать посредством применения соответствующих образцов проектирования. Пункт 1.2 легко реализуется через подробные комментарии и должное форматирование кода, облегчающее сопровождение.

Пункт 2 предназначен для оценки объема и классификации работ на сопровождение. Пункт 3 соответствует выбору службы поддержки: это может быть своя собственная служба или специальная сторонняя компания. План сопровождения (п. 4) регламентирует поток запросов на сопровождение внутри организации. Типичный план сопровождения приведен на рис. 8.5.

Жирной линией отмечена номинальная последовательность обработки запросов. Отдел сопровождения получает от пользователей

Организация процесса сопровождения

СОПРОВОЖДЕНИЕ

ТИПИЧНАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ ДЕЙСТВИЙ ПРИ СОПРОВОЖДЕНИИ

1. Подготовка ведения учета необходимых метрик.
2. Убедиться в подтверждении запроса.
3. Изучение проблемы.
4. Классификация запроса на сопровождение (исправление или усовершенствование.
5. Определить, требует ли реализация переработки на более высоком уровне.
6. Осуществление запрошенной модификации.
7. Планирование перехода с учетом текущего состояния.
8. Контроль влияния небольших изменений в масштабе всего приложения.
9. Реализация изменения.
10. Модульное тестирование.
11. Регрессионное тестирование.
12. Системное тестирование.
13. Обновление документации.

или заказчиков предложения и жалобы, которые оформляются как запросы на сопровождение. В дальнейшем разработчик принимает решение о последующей реализации запросов и присваивает каждому из них приоритет. Затем запросы обслуживаются техническим персоналом службы сопровождения. Тонкие линии означают другие последовательности появления и исчезновения запросов.

В общем случае порядок и длительность работ определяются масштабами проекта, а сам процесс реализации запросов сопряжен с двумя проблемами: доставкой готового кода пользователям и борьбой с дефектами. Для устранения дефектов можно применять так

называемые *исправления* или *заплатки* (patch), суть которых заключается в изменениях кода, позволяющих либо устранить дефект, либо обойти его. Возможный вариант работы с исправлениями представлен на рис. 8.6.

Сопровождение и исправление

Преимущества и недостатки исправлений

Преимущества	Недостатки
Быстрая удовлетворенность заказчиков	Дублирование работ: необходима реализация как временного, так и постоянного исправления
Возможность непрерывной работы и тестирования без широкого распространения дефектов	Иногда patch не заменяется соответствующей версией программного продукта
Исключено скрытие других дефектов	Затруднен выпуск постоянного исправления, предназначенного для устранения временного
Возможность тестировать исправление	Затруднен процесс документирования

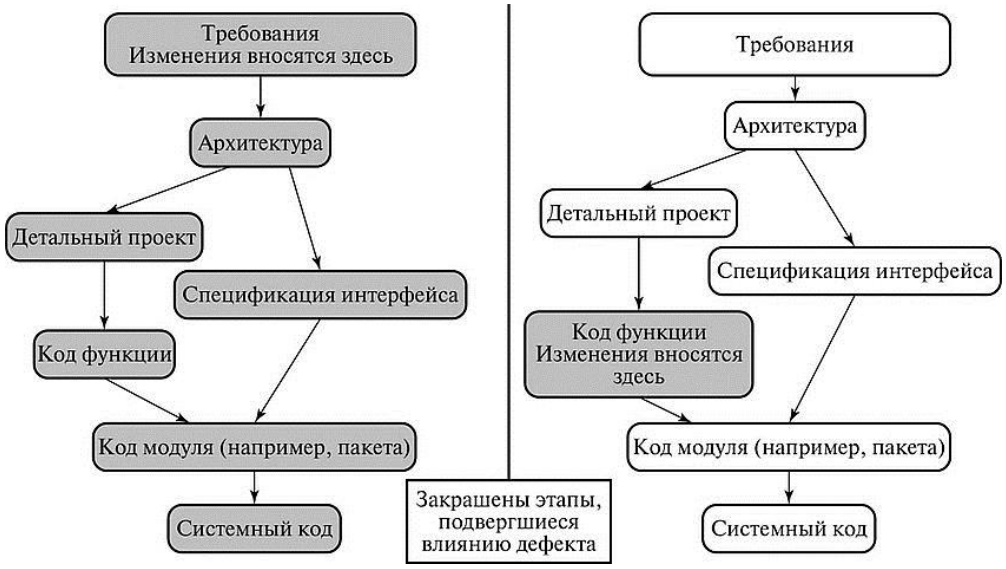
используются до получения полноценной версии программного продукта с внесенными изменениями в рамках запроса на сопровождение.

Обработка запросов на сопровождение включает в себя все стадии разработки ПО, однако становится необходимым процесс анализа влияния изменений на артефакты

системы. Согласно проведенным исследованиям 19% дефектов в приложении возникают на этапе определения требований, 52% — на стадии проектирования и 7% — в процессе программирования. Влияние устранения дефекта на артефакты иллюстрирует рис. 8.7.

Самый простейший случай соответствует внесению изменений в единственный артефакт (например, некорректное именование переменной или освобождение ресурсов после использования). Наиболее сложный случай характеризуется распространением изменений на все артефакты и этапы разработки (например, изменение требований).

Работы по сопровождению рассмотрим в контексте некоторых стандартов. В частности, *стандарт 1EEE1219* определяет следующие работы: корректировка, адаптация и совершенствование, состоящие из семи стадий, которые приблизительно соответствуют стадиям процесса разработки. Это определение задачи, анализ, проектирование, реализация, системное тестирование, приемо-сдаточное тестирование и поставка. Каждая из стадий оперирует шестью одинаковыми артефактами.



Примеры влияния дефектов на процесс сопровождения

Работы по сопровождению согласно стандарту IEEE 1219 и их атрибуты

Стандарт ISO/1EC14764 {Standard for Software Engineering) оперирует четырьмя составляющими:

1) *корректирующее сопровождение* — «реактивная» модификация ПП, выполняемая после передачи в эксплуатацию для устранения сбоев;
2) *адаптирующее сопровождение* — модификация ПП на этапе эксплуатации для обеспечения продолжения его использования с заданной эффективностью в условиях изменений бизнес-окружения, порождающих новые требования к системе;
3) *совершенствующее сопровождение* — модификация ПП на этапе эксплуатации с целью повышения характеристик производительности и удобства сопровождения;
4) *профилактическое сопровождение* — модификация ПП на этапе эксплуатации для идентификации и предотвращения скрытых дефектов до их проявления в реальных сбоях.

Стандарт ISO/IEC14764 классифицирует адаптивное и совершенствующее сопровождение как работы по расширению функциональности продукта, а корректирующую и профилактическую деятельности — как корректировку системы. Таким образом, согласно ISO/ IEC14764 все работы по сопровождению можно разбить на категории, реализующие «проактивный» и «реактивный» подходы (табл. 8.2). Исследования показали, что 60—80% объема работ относятся к усовершенствованию приложения.

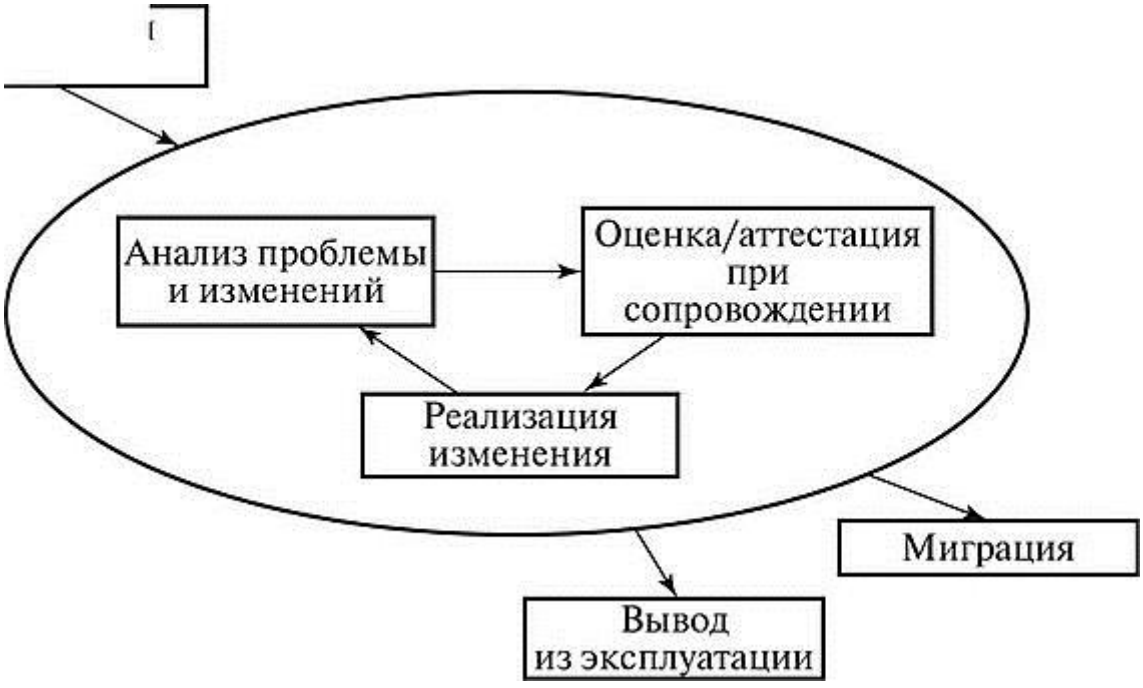
Стандарт ISO/IEC14764 уточняет положения по сопровождению ПП стандарта ЖЦ ПП ISO/IEC 12207, а в отличие от IEEE1219 работы в нем сгруппированы несколько иначе (рис. 8.9).

Виды сопровождения

Подход	Корректирующие работы	Работы по расширению
«Проактивный»	Профилактическое сопровождение	Совершенствующее сопровождение
«Реактивный»	Корректирующее сопровождение	Адаптирующее сопровождение

Технические вопросы процесса сопровождения. Особое внимание следует уделить ряду технических вопросов, касающихся процесса сопровождения, которые можно разделить на следующие группы: ограниченное понимание ПС, тестирование, анализ влияния, возможность сопровождения. Рассмотрим их подробнее.

Ограниченное понимание ПС связано в первую очередь с тем, что специалисты по сопровождению, как правило, не занимались разработкой сопровождаемого ПО. Поэтому до 60% усилий может быть потрачено на анализ и понимание сопровождаемого ПП. На данном этапе определяющим фактором является наличие качественной (в плане соответствия реальному состоянию кода и полноте содержания) документации к ПП. Затраченные на понимание программной системы усилия можно уменьшить, если использовать соответствующие методологии (например, иМЬ 2.0) или инструменты (например, обеспечивающие обратный инжиниринг для полного соответствия модели и программного кода). Важно также приводить в согласованность документацию с изменениями, внесенными при реализации запросов на сопровождение.



Процесс сопровождения по стандарту 180/1EC14764

При работах по сопровождению возможно распространение так называемой ряби, когда многочисленные незначительные изменения различных частей ПП при определенных

условиях приводят к неадекватной работе системы в целом. В этом случае есть смысл провести выборочное регрессионное *тестирование.*

Анализ влияния необходим для оценки всех возможных последствий и влияний изменений, вносимых в существующую систему. Для качественного анализа персонал сопровождения должен обладать как можно большей (в идеале — на уровне разработчиков) информацией о системе, ее специфике, содержании и структуре. При этом следует также учесть влияние изменений на окружение сопровождаемого ПП в виде других программных систем, функционирующих в том же операционном или системном пространстве.

Цели анализа влияния можно сформулировать следующим образом:

- определение содержания изменений для работ по планированию и реализации;
- получение максимально возможной оценки ресурсов, необходимых для проведения соответствующих работ;
- анализ стоимости и выгоды от внесения запрошенных изменений (обычно касается пожеланий в запросах на расширение системы);
- обсуждение сложности вопросов, связанных с внесением соответствующих изменений.

Сложность реализации соответствующего запроса на сопровождение часто является основным фактором, определяющим сроки и способы решения проблемы. Если система изначально разрабатывалась с учетом дальнейшей поддержки (например, использовались соответствующие образцы проектирования), то осуществить анализ влияния значительно легче.

Возможность сопровождения в ШЕЕ (стандарт 610.12-90, обновленный в 2002 г.) определяется как легкость сопровождения, расширения, адаптации и корректировки для удовлетворения заданных требований. Стандарт 180/1EC9126-01 определяет возможность сопровождения как одну из характеристик качества.

Для уменьшения стоимости дальнейшего сопровождения на протяжении всего цикла процесса разработки необходимо специфицировать, оценивать и контролировать характеристики, влияющие на возможность сопровождения. Одной из ключевых проблем сопровождения является отсутствие системной документации к ПП.

Управление процессом сопровождения. *Управленческие вопросы* можно разделить на следующие группы: согласование с организационными целями, кадровое обеспечение, организация процесса сопровождения, организационные аспекты сопровождения, аутсорсинг.

Согласование с организационными целями описывает, как осуществить возврат инвестиций от деятельности по сопровождению. Организационные цели сопровождения направлены на максимальное продление срока эксплуатации ПП, а деятельность по сопровождению есть обновление и расширение программной системы как отклик на изменяющиеся потребности пользователей. Такая постановка задачи приводит к трудности выявления прибыли на инвестированный капитал.

Проблемы кадрового обеспечения связаны с тем, что инженеры по сопровождению, как правило, считаются в компаниях-разработ-чиках специалистами «второго класса», поэтому часто возникают проблемы с удержанием квалифицированного персонала в отделах сопровождения.

Организация процесса сопровождения во многом схожа с организацией процесса создания ПО. В общем случае результатом итерации сопровождения является новая версия ПП, которая проходит практически все этапы разработки.

Организационные аспекты сопровождения связаны в первую очередь с тем, кто (какая организация) будет осуществлять сопровождение системы. Если сопровождение будет осуществляться силами разработчика, то необходимо определиться со структурными подразделениями, участвующими в этом процессе. Возможно также привлечение

сторонних организаций. Преимуществами последнего подхода являются возможность выбора сопровождающей организации из нескольких альтернатив (что позволяет выбрать подходящую стоимость сопровождения), а также появление у разработчика возможности заниматься другими видами работ (не сопровождением). Основным недостатком считается постепенная потеря разработчиками контроля над кодом созданной системы.

Аутсорсинг подразумевает полную передачу непрофильных работ сторонним организациям. Лишь незначительная часть крупных корпораций использует аутсорсинг и только для некритичных компонентов, выполняющих не очень важные бизнес-функции, поскольку они не хотят терять контроль над ассоциированными с этими системами данными или функциональностью. К тому же сама процедура передачи системы на внешнее сопровождение не отражена в стандартах, что затрудняет документальное определение предоставляемых аутсорсером сервисов.

Поскольку реализация запроса на сопровождение есть процесс, охватывающий практически все аспекты разработки, то применение общих (для всего ЖЦ ПП) метрик, разработанных SEI CMU (Software Engineering Institute, Carnegie-Mellon University), считается адекватным. Эти метрики охватывают такие аспекты, как размер, усилия, расписание и качество.

Инструменты процесса сопровождения. Важным моментом при организации процесса сопровождения является*выбор инструментов***,** поддерживающих работы по сопровождению.

Выделяют две категории инструментов сопровождения. Во-первых, *инструменты облегчения понимания*, которые помогают человеку в понимании программ. Примерами могут служить различные средства визуализации. Во-вторых,*инструменты реинжиниринга*, которые поддерживают деятельность по реинжинирингу.

Средства «обратного» инжиниринга помогают в процессе восстановления таких артефактов, как спецификация и описание дизайна (архитектуры) существующего ПО, которые в дальнейшем могут быть трансформированы для генерации нового продукта на основе функциональности существующего.

Следует отметить, что функциональность современных средств проектирования, поддерживающих анализ исходного кода (в случае объектно-ориентированных систем) и его визуализацию (в том числе и поведенческую, например в виде соответствующих диаграмм UML), позволяет объединить упомянутые категории инструментов в единый класс «инструментов реинжиниринга». В то же время деятельность по сопровождению и поддержке, в частности касающаяся сбоев и исправления обнаруженных в ПО ошибок, требует в определенной степени отнесения к этой категории и средств конфигурационного управления (например, в части обработки запросов на изменения).

Особенностью инструментов реинжиниринга является сочетание в одной среде инструментов для специалистов двух областей: непосредственно реинжиниринга бизнес-процессов и разработчиков программного обеспечения, поддерживающего модифицируемый бизнес-процесс.

Для контроля изменений в рамках всего ЖЦ ПП используются средства управления запросами на изменение, которые также могут быть использованы для поддержки процесса реализации запроса на сопровождение, поскольку и устранение дефектов, и расширение функциональности связаны с изменением многих артефактов ПП.

В общем случае любой из продуктов, поддерживающих процесс «понимания» программных систем и контроля изменений, можно использовать как инструмент сопровождения.

Весь процесс разработки программной системы можно назвать термином «инжиниринг», поскольку система отражает необходимую специфику предметной области, детально изученную разработчиками. Частью процесса эволюции ПП является его реинжиниринг.

Существует следующие *подходы к определению реинжиниринга.*

1. Реинжиниринг — это повторная реализация наследуемой системы с целью повышения удобства ее эксплуатации и сопровождения.
2. Реинжиниринг — это детальная оценка и перестройка ПО для формирования понимания, воссоздания (на уровне модели и, в ряде случаев, требований) и дальнейшей реализации его функциональности в новой форме. Сам реинжиниринг проводится не столько для улучшения возможностей сопровождения, сколько для замены старого ПО новыми версиями.
3. Реинжиниринг ПО в ряде случаев связывается с реинжинирингом какого-либо бизнес-процесса (BPR — Buisness Process Reengineering). Последний характеризуется совершенствованием внутренних процессов, протекающих внутри фирмы или ее структурных подразделений, что обязательно приведет к модификации обслуживающего их ПО.

В рамках первого подхода функциональность системы и ее архитектура не изменяются, а работы по сопровождению в рамках реинжиниринга касаются перевода системы на более современный язык программирования, ее повторного документирования, реорганизации и реструктуризации, модификации и модернизации структуры системы и системных данных. Поэтому есть смысл рассматривать реинжиниринг как один из способов сохранения наследуемых систем в эксплуатации, зарекомендовавших себя хорошо с коммерческих позиций, особенно если сама система обладает определенной коммерческой ценностью, а ее сопровождение дорого.

Значительными плюсами реинжиниринга являются снижение рисков (система не разрабатывается заново, поэтому все крупные риски уже учтены) и снижение затрат (ренжиниринг обходится дешевле разработки новой системы примерно в 4 раза). Основное различие между инжинирингом и разработкой новой системы связано со стартовой точкой начала работ.

Один из возможных *вариантов организации реинжиниринга* представлен на рисунке, где выделены следующие основные этапы:

Спецификация	*(* Проектирование _	-	Новая система
системы	V и реализация *Г*		

Традиционная разработка ПО

Существующая	*((* Исследование	Система после
система	Ги преобразование*)*	реинжиниринга

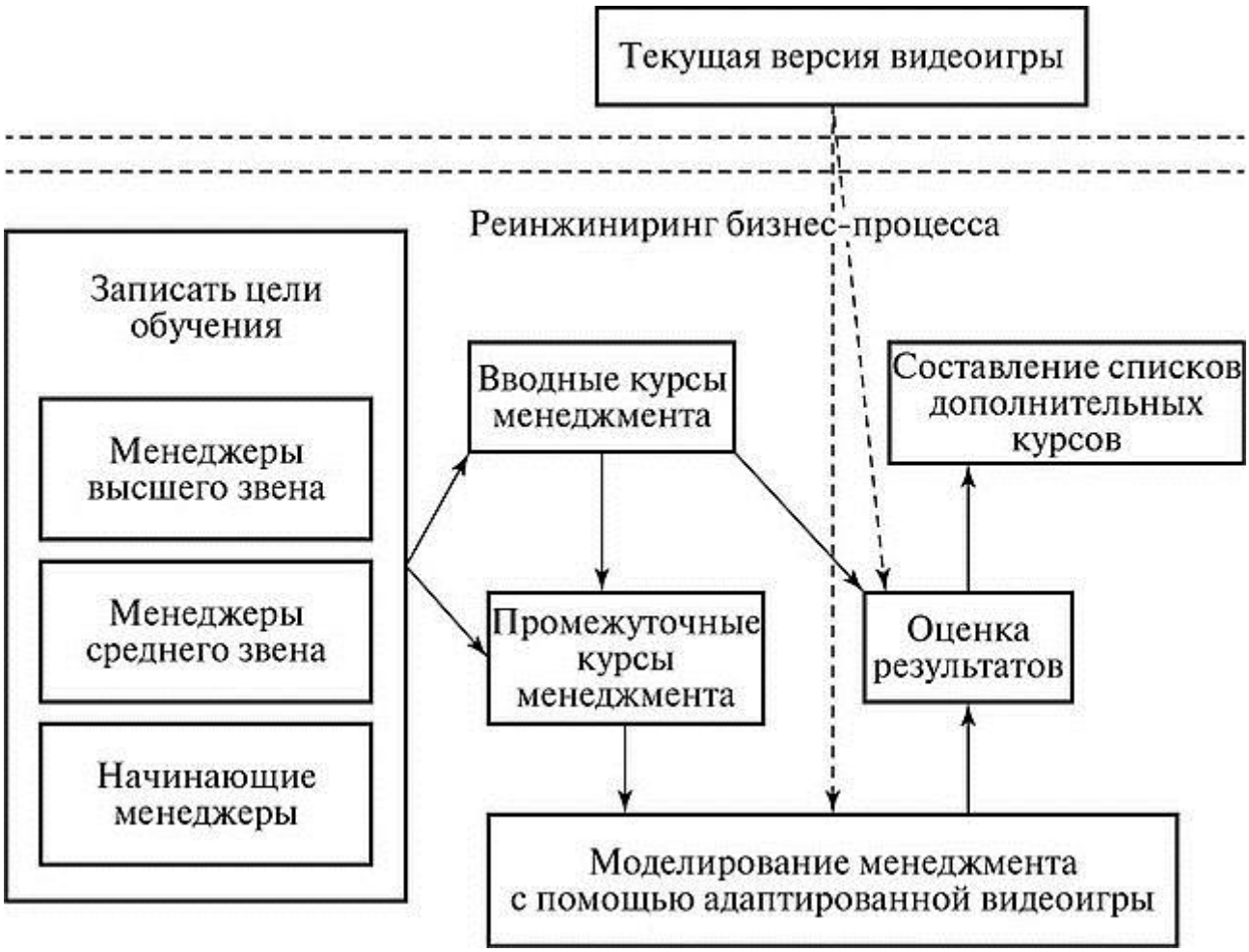
Реинжиниринг ПО

Традиционная разработка и реинжиниринг

Процесс реинжиниринга

- перевод исходного кода — конвертирование программы со старого языка программирования на его новую версию или другой язык;
- анализ программ — документирование структуры и функциональных возможностей программ на основе их анализа;
- модификация структуры программ — изменение управляющей структуры программ для их упрощения и лучшего понимания;
- разбиение программ на модули — группировка взаимосвязанных частей программ в модули для устранения избыточности и оптимизации их структуры, функциональности и интерфейса;
- изменение системных данных — приведение данных, с которыми работает программа, в соответствие с нововведениями, например изменением типа БД, с которой работает система.

Второй подход характеризуется перепланированием приложения, когда программные продукты, взятые за базу, перепроектируются в соответствии с изменившимися требованиями, например адаптированная ролевая игра может являться частью системы для обучения менеджменту, выполняющей моделирование взаимодействия обучаемых в рамках тестовых примеров (рис. 8.12).



Реинжиниринг ролевой видеоигры для обучения менеджменту

Третий подход рассматривает реинжиниринг как самостоятельный проект, включающий в себя формирование концепции, применение соответствующих инструментов и техник, анализ и использование опыта проведения реинжиниринга, оценку рисков и преимуществ, связанных с такими работами. В результате продукт реализуется в новом качестве при сохранении функциональности оригинального продукта.

Практическое задание.

Пользуясь предыдущими наработками и требованиями конкретной предметной области, выполните следующие действия:

1. Составьте план работ по сопровождению ПО в произвольной форме.
2. Определите приоритет каждого вида деятельности по сопровождению (вспомогательный, низкий, основной, наивысший).
3. Определите количество и состав сотрудников на каждый вид работ по сопровождению.

Лабораторная работа №27. Реализация плана работ по сопровождению

Теоретический материал.

В зависимости от объема работ по сопровождению, должно быть принято решение о

внесении тех или иных разделов в конкретный план сопровождения.

Разделы, рекомендованные для включения в план сопровождения:

1. ВВЕДЕНИЕ

1.1. Описание сопровождаемого программного продукта

1.2. Определение исходных состояний программного продукта

1.3. Описание уровня требуемой поддержки

1.4. Определение организации (подразделений), проводящей сопровождение

1.5. Описание условий (протоколов), согласованных между заказчиком и поставщиком

2. ОРГАНИЗАЦИОННЫЕ РАБОТЫ И РАБОТЫ ПО СОПРОВОЖДЕНИЮ

2.1. Роли и обязанности сопроводителя до поставки программного продукта

2.1.1. Реализация процесса сопровождения

2.1.2. Определение инфраструктуры процесса сопровождения

2.1.3. Установление процесса обучения

2.1.4. Установление процесса сопровождения

2.2. Роли и обязанности сопроводителя после поставки программного продукта

2.2.1. Реализация процесса сопровождения

2.2.2. Анализы проблем и модификаций

2.2.3. Реализация (внесение) модификаций

2.2.4. Рассмотрение и принятие модификаций

2.2.5. Перенос программного продукта в новые условия

2.2.6. Изъятие программного продукта из эксплуатации

2.2.7. Решение проблем (включая службу поддержки клиентов)

2.2.8. При необходимости - обучение персонала (сопроводителя и пользователя)

2.2.9. Усовершенствование процесса сопровождения

2.3. Роль пользователя

2.3.1. Приемочные испытания

2.3.2. Взаимосвязи с другими организациями

3. РЕСУРСЫ

3.1. Персонал

3.1.1. Состав персонала для конкретного проекта

3.2. Программные средства

3.2.1. Определение программных средств, необходимых для поддержки эксплуатации системы (с учетом системных требований)

3.3. Технические средства

3.3.1. Определение технических средств, необходимых для поддержки эксплуатации системы (с учетом системных требований)

3.4. Оборудование (аппаратура)

3.4.1. Определение требований к оборудованию (аппаратуре) системы (помимо технических средств вычислительной техники)

3.5. Документы

3.5.1. План обеспечения качества

3.5.2. План управления конфигурацией (может быть совместным с процессом разработки)

3.5.3. Документы разработки (передаются из процесса разработки)

3.5.4. Инструкции по сопровождению

3.5.5. Порядок проведения верификации и аттестации программного продукта

3.5.6. Процедуры тестирования и акты о тестировании

3.5.7. План обучения

3.5.8. Инструкции по эксплуатации программных продуктов (передаются из процесса разработки)

3.6. Информационные ресурсы

4. ПРОЦЕСС ВЫПОЛНЕНИЯ КОНКРЕТНОЙ ДЕЯТЕЛЬНОСТИ

4.1. Процессы, выполняемые сопроводителем

5. ОБУЧЕНИЕ

5.1. Определение уровня обучения, необходимого для сопроводителя и пользователя

СТРУКТУРА КОНЦЕПЦИИ СОПРОВОЖДЕНИЯ

Концепция сопровождения должна быть разработана сразу после окончания опытной эксплуатации программного продукта.

Концепцию сопровождения разрабатывает сопроводитель для каждого программного продукта, полученного для сопровождения, которую затем утверждает заказчик.

Концепция сопровождения должна содержать следующие разделы:

8. область сопровождения программного продукта;
9. практическое применение процесса сопровождения;
10. определение ответственных за сопровождение;
11. оценка стоимости сопровождения.

1. Область сопровождения

Данный раздел должен определять обязанности сопроводителя, а также, какую поддержку программного продукта он обязан обеспечить. Область сопровождения может зависеть от наличия бюджетных средств. В данном разделе должны быть описаны:

18. типы выполняемого сопровождения;
19. порядок сопровождения программной документации;
20. действия сопроводителя при разных типах сопровождения;
21. необходимый уровень обученности персонала сопровождения;
22. организация службы поддержки клиентов («Hot line»);
23. порядок поставки программного продукта.

2. Практическое применение процесса сопровождения

Концепция сопровождения должна отражать задачи сопровождения программного продукта после его поставки. В данном разделе необходимо определить все организации (подразделения), участвующие в процессе сопровождения, их роли и основные задачи. Также необходимо описать выбранный процесс сопровождения для этого программного продукта.

3. Определение ответственных за сопровождение

В данном разделе необходимо определить организации (подразделения) и конкретных физических лиц, отвечающих за сопровождение программного продукта. Требуется определить роли физических лиц и уточнить их обязанности.

При выполнении сопровождения по соглашению с третьей стороной, это должно быть отмечено в концепции сопровождения.

4. Оценка стоимости сопровождения

В данном разделе должна быть проведена оценка стоимости сопровождения. При наличии соглашения на сопровождение программного продукта для внешней организации необходимо учитывать:

- проезд до места расположения пользователя;
- обучение как сопроводителей, так и пользователей;
- необходимая техническая поддержка приобретенных (стандартных) программных и технических средств;
- зарплаты и премии персонала.

При разработке концепции стоимость оценивают на основе ограниченных данных. Эта стоимость может быть уточнена в процессе сопровождения.

В качестве исходных данных могут быть использованы данные по аналогичным проектам.

Практическое задание.

Используя примерный план работ по сопровождению, описанный в теоретической части работы, а также предыдущие наработки и выбранную предметную область, выполните следующие задания:

1. Составьте детальный план работ по обслуживанию, которые предполагаете проводить непосредственно в вашей предметной области (используйте структуру примерного плана).
2. Рассчитать зарплату персонала, который будет осуществлять сопровождение.
3. Рассчитать стоимость оборудования, необходимого для сопровождения ПП.
4. Определить необходимость обучения сотрудников компании новому ПО, в ходе сопровождения.
5. Определить, какая форма технической поддержки будет осуществляться в ходе сопровождения ПП.

Лабораторная работа №28. Управление сопровождением

Теоретическая часть.

Управление SCM-процессом (Management of SCM Process)

SCM-деятельность контролирует эволюцию и целостность продукта, идентифицируя его элементы, управляя и контролируя изменения, а также, проверяя, записывая и обеспечивая отчетность по конфигурационной информации. С инженерной точки зрения, SCM способствует разработке и реализации изменений. Успешное внедрение SCM требует точного планирования и управления. Это, в свою очередь, предполагает понимание организационного контекста и тех ограничений, которые связаны с проектированием и реализацией процесса конфигурационного управления.

Для планирования SCM-процесса необходимо понимать организационный контекст и связи между организационными элементами. SCM-работы предполагают взаимодействие с другими аспектами проектной деятельности (не путайте с управлением проектами – это, при всей своей значимости, лишь один из видов проектной деятельности) и организационными элементами.

Организационные элементы, отвечающие за процессы поддержки программной инженерии, могут быть структурированы несколькими способами. Несмотря на то, что ответственность за выполнение определенных SCM-задач может быть назначена (принята или ассоциирована, в зависимости от управленческих принципов и установок, т.е. общего менеджмента - general management) различным частям (лицам, группам, подразделениям и т.п.) организации, например, структуре, отвечающей за разработку программного обеспечения, общая ответственность за конфигурационное управление часто возлагается на отдельный (специализированный) организационный элемент или назначенную персону.

Программное обеспечение часто разрабатывается как составная часть б*о*льшей системы, содержащей аппаратные и программно-аппаратные/встраиваемые элементы. В этом случае, SCM-деятельность ведется параллельно с работами по конфигурационному управлению (CM) в отношении аппаратной или программно-аппаратной части, строго согласуясь с общим конфигурационным управлением на уровне системы, в целом. Ряд источников (см. библиографию SWEBOK, связанную с данной областью знаний) описывает SCM в сочетании с контекстом, в рамках которого проводится такая деятельность.

SCM может играть роль интерфейса к работам, направленным на обеспечение качества (quality assurance), вытекающим, например, из отслеживания записей <по изменениям> и несогласующихся элементов (например, выявленным в процессе сборки очередной версии системы, *прим. автора*). С точки зрения составителей

<данной области знаний SWEBOK>, некоторые элементы, находящиеся под управлением SCM <процесса>, могут также служить объектами рассмотрения в рамках организационных программ по обеспечению качества. Управление несогласующемися элементами обычно относится к работам по управлению качеством. Однако, SCM может обеспечить существенную помощь в отслеживании (трассировке) и создании отчетности по элементам программных конфигураций, попадающих в такую <проблемную> категорию.

SWEBOK отмечает, что возможно тесное взаимодействие между организационными структурами, отвечающими за разработку и сопровождение (и SCM играет роль инфраструктуры, обеспечивающей такую связь).

В зависимости от контекста, существует множество подходов и практик в части выполнения задач конфигурационного управления в приложении к программному обеспечению. Часто, одни и те же инструменты поддерживают и разработку, и сопровождение, обеспечивая достижение целей и содержания SCM.

Ограничения и правила SCM (Constraints and Guidance for the SCM Process)

Ограничения и правила в отношении процесса конфигурационного управления порождаются различными источниками. Политики и процедуры, формулируемые на корпоративном или другом организационном уровне, могут влиять или предписывать структуру и реализацию SCM-процесса для заданного проекта. Кроме того, контракт между заказчиком и поставщиком может содержать положения, затрагивающие процесс конфигурационного управления. Например, может требоваться проведение определенных процедур проверки (аудита) или специфицирован набор элементов (активов, артефактов), передаваемых под управление <процедур и системы> конфигурационного управления (или в части формализации обработки и контроля реализации запросов на изменения, поступающих от заинтересованных лиц). Когда разрабатываемый программный продукт потенциально затрагивает аспекты публичной безопасности, могут налагаться определенные ограничения со стороны соответствующих регулирующих органов (например, USNRC Regulatory Guide 1.169, “Configuration Management Plans for Digital Computer Software Used in Safety Systems of Nuclear Power Plants”, U.S. Nuclear Regulatory Commission, 1997). Наконец, на структуру и реализацию SCM-процесса в проекта влияют выбранные (с точки зрения модели и адаптированных характеристик) процессы жизненного цикла и инструменты, применяемые для реализации программной системы.

Рекомендации по структуре и реализации SCM-процесса могут быть также результатом применения лучших практик (best practices), представленных в стандартах, выпущенных соответствующими стандартизирующими организациями. Лучшие практики также отражены в моделях совершенствования и оценки процессов, например, в CMMI – Capability Maturity Model Integration Института программной инженерии (SEI – Software Engineering Institute) университета Карнеги-Меллон (Carnegie-Mello University) и ISO/IEC 15504 (SPICE) “Software Engineering – Process Assessment”.

Планирование процесса конфигурационного управления для заданного проекта должно согласовываться с организационным контекстом, соответствующими ограничениями, общепринятыми рекомендациями, а также характеристиками и природой самого проекта (например, его размером или значимостью). Основные работы, проводимые при планировании SCM-деятельности включают:

Идентификацию программных конфигураций (Software Configuration Identification)

Контроль конфигураций (Software Configuration Control)

Учет статусов конфигураций (Software Configuration Status Accounting)

Аудит конфигураций (Software Configuration Auditing)

Управление выпуском и поставкой (Software Release Management and Delivery)

Кроме этого, необходимо принимать во внимание и такие аспекты конфигурационного управления, как организационные вопросы, обязанности, ресурсы и расписание, выбор инструментов и реализация, контроль поставщиков и субподрядчиков, а также, контроль интерфейсов <взаимодействия программных модулей>. Результаты планирования сводятся в *план конфигурационного управления* (*SCM Plan - SCMP*), обычно, являющийся объектом оценки и аудита в рамках деятельности по обеспечению качества (SQA – Software Quality Assurance).

1.3.1 Организация и обязанности (SCM organization and responsibilities)

Для предотвращения путаницы в том, кто будет выполнять заданные работы и задачи конфигурационного управления, должны быть четко идентифицированы организации (организационные структуры), вовлеченные в SCM-процесс. Конкретные обязанности по выполнению заданных работ и задач SCM должны быть назначены соответствующим организационным сущностям. Также, должны быть идентифицированы общие полномочия и пути отчетности, даже если это выполняется в процессе планирования управления проектом или деятельности по обеспечению качества.

1.3.2 Ресурсы и расписание (SCM resources and schedules)

В процессе планирования конфигурационного управления идентифицируется персонал и инструменты, привлекаемые для выполнения соответствующих работ и задач SCM. Планирование касается вопросов определения расписания, устанавливая последовательность задач конфигурационного управления и идентифицируя их связь с расписанием проекта и его вехами, определенными на стадии планирования проекта. Также должны быть специфицированы требования по обучению персонала, необходимые для реализации планов.

1.3.3 Выбор инструментов и реализация (Tool selection and implementation)

SCM-деятельность поддерживается различными типами инструментальных средства и процедур по их использованию. В зависимости от ситуации, эти инструменты могут включать комбинацию различных возможностей – автоматизированные средства могут решать отдельные задачи SCM, интегрированные средства могут обслуживать потребности многих участников процесса программной инженерии (например, SCM, разработку, проверку и аттестацию и т.п.). Значимость инструментальной поддержки конфигурационного управления (как и других аспектов деятельности в области программной инженерии) растет с каждым днем вместе со сложностью внедрения, ростом размера проектов и сложности проектного окружения. Возможности инструментальных средств развиваются для обеспечения поддержки:

12. SCM-библиотек (проектно-ориентированных баз знаний, *прим. автора*)
13. Запросов на изменения (software change request - SCR) и процедур утверждения (approval)

14. Управления кодом (и связанных рабочих продуктов) и изменениями
15. Отчетности по статусу конфигураций и сбору соответствующих метрических показателей
16. Аудиту конфигураций
17. Управлению и отслеживанию <состояния и полноты> программной документации
18. Выполнению задач по сборке программных продуктов и их модулей
19. Управлению, контролю и поставке выпусков (релизов) программных продуктов

Инструменты, используемые для обеспечения конфигурационного управления, могут также предоставлять метрики, необходимые для совершенствования процессов. SWEBOK обращает внимание (рекомендуя соответствующий первоисточник) на следующие ключевые индикаторы: работы и прогресс <по их выполнению> (Work and Progress) и индикаторы качества – поток изменений (Change Traffic), стабильность <конфигураций> (Stability), раздробленность (Breakage), модульность (Modularity), переработка (Rework), адаптируемость (Adaptibility), среднее время между сбоями (MTBF – Mean Time Between Failures), зрелость/полнота <информации> (Maturity). Отчетность по этим индикаторам может быть организована различным образом, например, по элементам конфигураций или по типу запросов на изменения.

Рисунок 3 демонстрирует отображение инструментальных возможностей и процедур на работы по конфигурационному управлению.

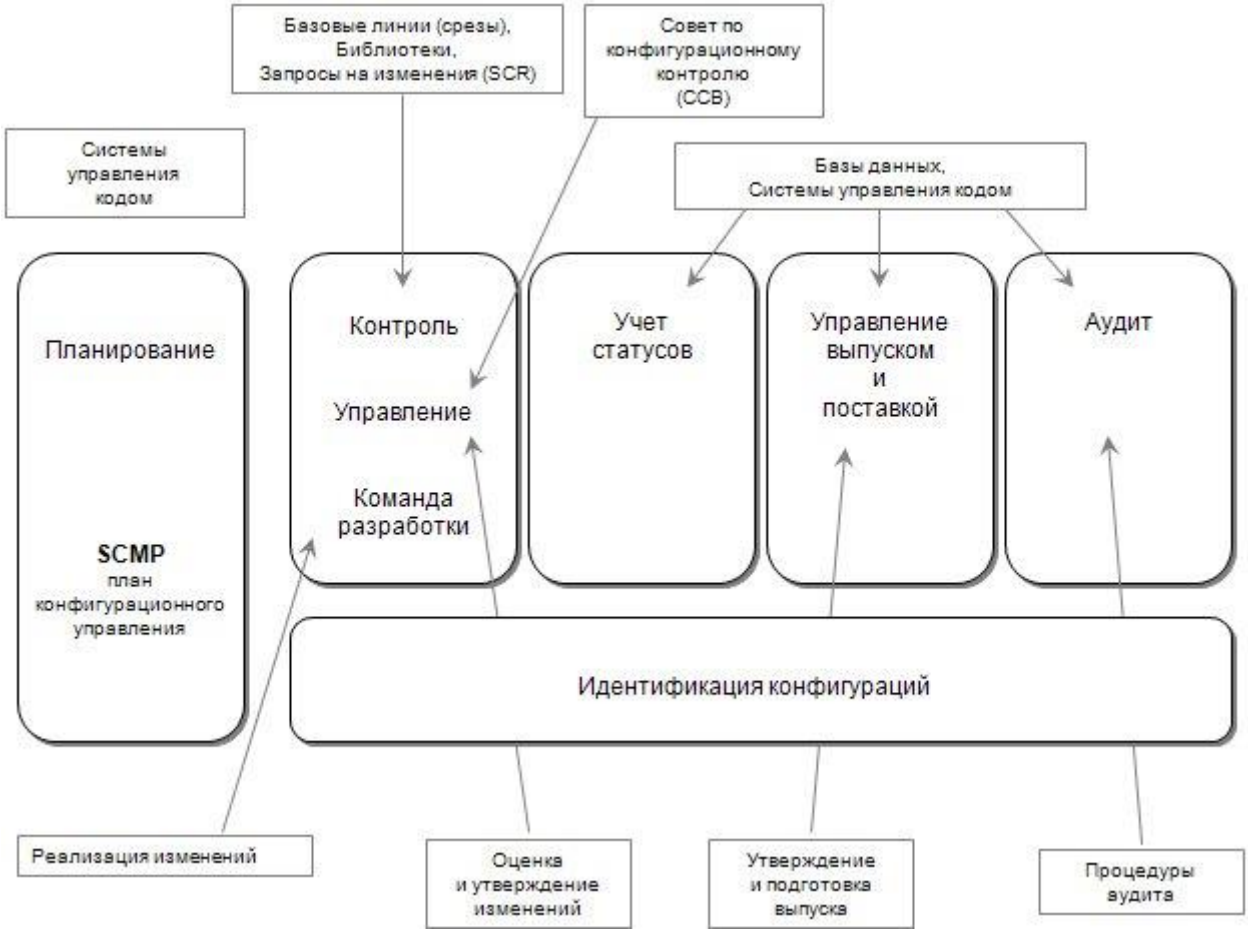


Рисунок 3. Характеристики SCM-инструментов и связанные процедуры. [SWEBOK, 2004, с.7-4, рис. 3]

В этом примере система управления кодом поддерживает программные библиотеки контролируя доступ к элементам библиотек, координирует действия множества пользователей и помогает в проведении рабочих процедур. Другие инструменты поддерживают процесс сборки и выпуска программного обеспечения и документации на основе программных элементов, содержащихся в библиотеках. Инструменты для управления запросами на изменения программного обеспечения используются для контролируемых <системой конфигурационного управления> программных элементов. Другие инструменты могут обеспечивать управление базой данных и необходимыми менеджменту отчетными средствами, а также деятельностью по разработке и обеспечению качества. Как уже упоминалось выше, в рамках SCM-системы может быть объединен целый ряд инструментов различных типов. При этом сама система конфигурационного управления может быть тесно связана и поддерживать другие виды работ, касающиеся не только SCM.

В процессе планирования инженеры выбирают те SCM-средства, которые применимы для решения стоящих перед ними задач.

Вопрос выбора SCM-системы должен решаться исходя из целей, сформулированных в отношении используемых процессов программной инженерии и уровня зрелости этих процессов. Кроме того, необходимо учитывать и вопросы унификации программных средств, используемых для поддержки инфраструктуры разработки и сопровождения всего портфеля программных проектов, выполняемых в организации. В силу фундаментальной значимости SCM-системы для обеспечения базовых процессов программной инженерии и управления всеми проектными активами, принимать решение об использовании той или иной SCM-системы для каждого отдельно взятого проекта выглядит необоснованным. SCM-система, система управления требованиями (более чем желательно, связанная с SCM), средства бизнес-моделирования и проектирования, среды разработки – все это должно быть стандартизировано в рамках организации, за исключением тех случаев, когда требования в отношении тех или иных инструментальных средств сформулированы со стороны заказчика и являются составной частью требований, предъявляемых к проекту. Возвращаясь к вопросу выбора SCM-системы, безусловно, необходимо учитывать мнение инженеров, однако, сложившиеся привычки не должны “перевешивать” функциональность предлагаемых к унификации SCM-средств, обеспечиваемую ими доступность и прозрачность информации о состоянии проекта в любой момент времени и, конечно, возможность эффективного администрирования активов проекта, в том числе, в контексте необходимых для этого трудозатрат.

В процесс планирования рассматриваются аспекты, которые могут “всплыть” в процессе внедрения (и, даже, на этапе эксплуатации) выбираемой системы конфигурационного управления. В частности, обсуждаются и вопросы возможных “культурных” изменений, если это необходимо (с точки зрения поставленных целей – проекта и/или совершенствования процессов). Дополнительная информация, затрагивающая SCM-инструментарий, представлена в области знаний SWEBOK “Software Engineering Tools and Methods”.

1.3.4 Контроль поставщиков/подрядчиков (Vendor/Subcontractor Control)

Программные проекты могут требовать необходимости приобретать или использовать уже приобретенное программное обеспечение – компиляторы и другие инструменты (среды разработки, библиотеки компонент). Планирование должно касаться вопросов – надо и, если надо, то как помещать эти инструменты

(например, интегрируя в программные библиотеки проекта) под управление SCM-системы и как их изменения и обновления будут оцениваться и управляться.

Аналогичные соображения существуют и в отношении программного обеспечения, создаваемого подрядчиками. В этом случае, в отношении SCM-процесса подрядчика предъявляются специальные требования со стороны заказчика и они вносятся в контракт, предполагая не только возможность мониторинга, но и соответствие его возможностей заданным требованиям. В последнее время все чаще отмечается важность доступности SCM-информации для эффективного мониторинга соответствия (compliance monitoring).

1.3.5 Контроль интерфейсов (Interface Control)

Когда программные элементы должны связываться с другими программными или аппаратными элементами, изменения в одних элементах могут влиять на другие элементы. Планирование SCM-процесса рассматривает, в частности, как будут идентифицироваться связанные элементы и как будут управляться и сообщаться их изменения. Конфигурационное управление может быть частью более масштабного процесса системного уровня (т.е. в рамках всей системы, к которой относятся соответствующие программные элементы) по определению и контролю интерфейсов, включая описание в соответствующих спецификациях интерфейсов, планах контроля интерфейсов и других документах. В этом случае, SCM-планирование контроля интерфейсов проводится в контексте процесса системного уровня.

План конфигурационного управления (SCM Plan)

Результаты SCM-планирования для заданного проекта определяются в *плане конфигурационного управления* (*Software Configuration Management Plan, SCMP*), который является документом, используемом в качестве описание SCM-процесса. Он всегда поддерживается в актуальном состоянии (обновляясь и утверждаясь по мере внесения в него необходимых изменений) на протяжении всего жизненного цикла. При описании SCM-плана обычно необходимо разработать ряд детальных процедур, определяющих как конкретные требования будут выполняться в повседневной деятельности.

Создание и сопровождение плана конфигурационного управления основывается на информации, получаемой в процессе работ по планированию. Рекомендации по созданию и сопровождению SCMP можно найти, например, в одном из ключевых SCM-стандартов *IEEE 828-98 "Standard for Software Configuration Management Plans"*. Этот стандарт описывает требования к информации, содержащейся в плане конфигурационного управления, а также определяет шесть категорий SCM-информации, содержащейся в плане (обычно, представленных в виде соответствующих разделов, *прим. автора*):

24. Введение (Introduction) – описывает цели, содержание и используемые термины.
25. Управление (SCM Management) – описывает структуру, обязанности, полномочия, политики, директивы (указания, обязательные для исполнения) и процедуры.
26. Работы (SCM Activities) – определяет идентификацию конфигураций, их контроль и т.п.
27. Расписание (SCM Schedule) – определяет связь работ по конфигурационному управлению с другими аспектами и процессами проектной деятельности.

28. Ресурсы (SCM Resources) – описывает инструменты, физические ресурсы, персонал и т.п.
29. Сопровождение плана (SCMP Maintenance) – определяет правила, по которым в план вносятся изменения и описывает как эти изменения внедряются в повседневный SCM-процесс.

После того, как внедрен процесс конфигурационного управления, может быть необходимо контролировать (проводить надзор) над SCM-процессом для обеспечения того, что SCM-план исполняется надлежащим образом. В ряде случаев определяются конкретные требования по обеспечению качества (SQA), контролирующие исполнение процессов и процедур конфигурационного управления. Для этого может быть необходимо введение соответствующих полномочий и назначение обязанностей по контролю выполнения задач SCM. Аналогичные полномочия и обязанности по надзору над SCM-процессом могут существовать в контексте SQA-деятельности.

Использование интегрированных SCM-инструментов с возможностью контроля процесса может сделать процедуру надзора более легкой и прозрачной. Некоторые инструменты предоставляют высокий уровень настраиваемости для обеспечения гибкой адаптации процессов. Другие инструменты являются менее гибкими, диктуя те или иные процессы и их характеристики. Требования контроля (надзора), с одной стороны, и уровень гибкости и адпатируемости, с другой, являются определяющими критериями выбора того или иного инструмента.

1.5.1 Метрики и процесс количественной оценки в SCM (SCM measures and measurement)

Количественные показатели (метрики) могут определяться для обеспечения информации о разрабатываемом продукте или для оценки исполнения самого процесса конфигурационного управления. Связанной целью SCM-мониторинга может быть и раскрытие возможностей по совершенствованию процесса (не только SCM-процесса, но и других процессов программной инженерии). Количественная оценка SCM-процессов предоставляет хорошие средства для мониторинга эффективности деятельности по конфигурационному управлению на постоянной основе. Эти измерения полезны для оценки текущего состояния процесса и проведения сравнений во времени (как прогресса в отношении развития продукта, так и качества выполнения процесса, как такового). Анализ измерений позволяет понять причины изменения процесса и внести соответствующие коррективы в план конфигурационного управления (SCMP).

Программные библиотеки и различные возможности SCM-средств предоставляют источники для получения информации о характеристиках SCM-процесса (наравне с проектной информацией и данными, необходимыми для принятия тех или иных управленческих решений). Например, информация о времени, необходимом для выполнения различных типов изменений, может быть полезна для оценки критериев того, какой уровень полномочий оптимален для утверждения определенных типов изменений.

Необходимо сохранять фокус на проведении анализа измерений и формировании соответствующих выводов, вместо проведения “измерений ради измерений” (к сожалению, последнее встречается слишком часто, чтобы не отметить этот факт). Обсуждение количественных оценок в отношении процесса и продукта представлено в области знаний “Процесс программной инженерии”. Программа

проведения количественных оценок обсуждается в области знаний “Управление программной инженерией”.

1.5.2 Аудит в рамках SCM (In-process* audits of SCM)

Аудит может проводится на протяжении всего процесса программной инженерии для определения текущего статуса заданных элементов конфигураций или оценки реализации процесса конфигурационного управления. SCM-аудит предоставляет более формальный механизм мониторинга выбранных аспектов процесса и может координироваться с работами в области обеспечения качества (SQA; см. секцию “5. Software Configuration Auditing”).

* “in-process” подчеркивает непрерывность/периодичность аудита и его позиционирование как неотъемлемой составной части конфигурационного управления.

Практическое задание.

1. Перечислите все функции сопровождения.
2. Исходя из предыдущих наработок и своей предметной области:
3. Определите типы сопровождения (корректирующие и т.п.), которые могут понадобиться.
4. Составьте повременной и пооперационный план сопровождения ПО
5. Составьте SCM-план для вашего ПО.

Лабораторная работа №30.

Реинжиниринг бизнес-процессов при внедрении ПО.

Теоретическая часть.

Понятие «реинжиниринга ИС», его содержание и место в ЖЦ ИС.

Сразу следует признать, что в настоящий момент понятие «реинжиниринг ИС» не является повсеместно устоявшимся. Как следствие довольно часто возникает определенная терминологическая путаница. Авторами исследуются одни и те же проблемы, подходы, методы и технологии их решения, однако в качестве базовых понятий, наряду с «реинжинирингом ИС» употребляются «эволюция ИС» , «миграция ИС», «модернизация ИС» «реструктуризация ИС» .

Нельзя отрицать, что деятельность по миграции ИС имеет определенную специфику (окраску) по отношению к деятельности по модернизации ИС.

«Реинжиниринг представляет собой систематическую трансформацию существующей системы с целью улучшения ее характеристик качества, поддерживаемой ею функциональности, понижения стоимости ее сопровождения, вероятности возникновения значимых для заказчика рисков, уменьшения сроков работ по сопровождению системы»,

становится очевидным, что и миграция, и модернизация ИС являются частью деятельности по реинжинирингу ИС. Как результат, подходы, методы и технологии миграции, модернизации, эволюции ИС, следует считать частью методологического и

инструментально - технологического обеспечения процесса реинжиниринга ИС.

Такой взгляд на реинжиниринг ИС согласуется с таксономией, вводимой в ряде работ. В этих работах авторами делается попытка выстроить систему понятий, соотносимых с данным видом деятельности. Так, реинжиниринг ИС определяется как «исследование (изучение, обследование) и перестройка исходной системы с целью ее воссоздания в новой форме с последующей реализацией этой новой формы. Далее, в контексте деятельности по реинжинирингу вводятся и определяются такие важные понятия, как

1. прямой инжиниринг (Forward engineering);
2. редокументирование (Redocumentation);
3. рефакторинг (Refactoring);
4. реструктуризация (Restructuring);
5. переориентация (Retargeting);
6. обратный инжиниринг (обратное проектирование) (Reverse engineering);
7. сопровождение программных продуктов (Software maintenance);
8. трансляция исходного кода (Source Code Translation);
9. и т.д.

Перечисленные понятия раскрывают понятие «реинжиниринг ИС», а соотносимая с ними деятельность рассматривается либо как одна из форм реинжиниринга ИС, либо как подпроцесс процесса реинжиниринга. Определения этих и других понятий приводятся в приложении к данной статье в глоссарии понятий и терминов. При этом в тех случаях, когда проявлялись отличия в определении того или иного понятия, в глоссарий включались альтернативные варианты определения, а при необходимости, и поясняющие определение комментарии.

Основное содержание реинжиниринга ИС и его место в ЖЦ ИС.

Следующим шагом на пути исследования подходов, методов и технологий реинжиниринга ИС следует считать определение основного содержания деятельности по реинжинирингу ИС, места реинжиниринга в ЖЦ ИС.

Так, придерживаются следующей позиции при определении границ деятельности по реинжинирингу, и, как следствие, места реинжиниринга в ЖЦ ИС.

Утверждается, что реинжиниринг ИС занимает промежуточное местоположение по отношению к разработке и сопровождению ИС. При этом сопровождение ИС рассматривается как деятельность, предусматривающая выполнение изменений, направленных на коррекцию, усовершенствование и адаптацию ИС, а разработка ИС как деятельность, включающая реализацию новых возможностей, добавление новой функциональности, осуществление таких существенных улучшений, как переход на использование новых компьютеров, внедрение новых информационных технологий. Авторами правомерно утверждается, что реинжиниринг характеризуется деятельностью, как по сопровождению, так и по разработке ИС. При этом эти два вида деятельности в контексте реинжиниринга ИС могут существенно перекрываться.

В контексте исследований, связанных с эволюцией ИС, авторами выделяются деятельности по сопровождению, модернизации и замещению ИС.

Определяется, что сопровождение представляет собой инкрементальный итеративный процесс, в рамках которого выполняются малые изменения в системе, не затрагивающие структурной организации системы (архитектуры системы).

В отличие от сопровождения модернизация характеризуется как деятельность, которая предусматривает значительные изменения существующей системы (в том числе в ее структуре), но не ее утилизацию или замещение новой системой.

И, наконец, замещение рассматривается как процесс, который заключается во внедрении

новой системы, способной полностью заменить существующую ИС. Замещение обычно применяется к системам, которые недокументированны, устарели или не расширяемы, расходятся с требованиями бизнеса и для которых модернизация невозможна или экономически не выгодна.

Первоначально, осуществляется разработка (построение) ИС. Далее выполняется деятельность по ее сопровождению. В процессе сопровождения возникает необходимость в реструктуризации системы и как следствие выполняется деятельность по ее модернизации. После этого снова осуществляется деятельность по сопровождению системы. В тот момент, когда ИС перестает удовлетворять заказчика, осуществляется ее замещение на новую систему и последовательность выполняемых видов деятельности повторяется. Безусловно, в реальной жизни может возникать и другая последовательность, когда, сразу, не прибегая к модернизации, осуществляется замещение одной ИС другой. Однако общий подход к последовательности выполнения сопровождения, модернизации и замещения останется неизменным.

Приводимые виды деятельности основаны на таксономии. В отличие от предыдущей работы, в качестве базового понятия используется именно «реинжиниринг», а вводимая таксономия рассматривается как множество видов деятельности, соотносимых с реинжинирингом ИС. При этом применительно к унаследованным ИС выделяются соответственно деятельности по их оценке, сопровождению, трансформации, замещению, а так же смешанные стратегии, предусматривающие совместное выполнение перечисленных ранее видов деятельности.

Обеспечивая концептуальное понимание процесса реинжиниринга ИС, в ряде работ определяются основные виды деятельности (фазы) соотносимые с этим процессом.

Так, рассматриваются следующие основные фазы:

1. оценки показателей проекта по реинжинирингу, в том числе характеристик унаследованной информационной системы (фаза оценки);
2. анализа решений по реинжинирингу, в том числе принятие решения о необходимости проведения работ по реинжинирингу или сопровождению/разработке ИС;
3. осуществления реинжиниринга (выполнения работ по реинжинирингу);
4. внедрения системы, трансформированной в результате проведения реинжиниринга.

Другой подход к определению деятельности по реинжинирингу базируется на так называемой модели «подковы».

В основу данной модели положены следующие процессы (виды деятельности), соотносимые с реинжинирингом ИС:

- анализ существующей системы, основанный на одном или более ее логических описаний;
- трансформация этих логических описаний в новое, улучшенное логическое описание системы.
- разработка новой системы, основанной на новых логических описаниях системы.

Эти три основных процесса соединяются в модели в виде «подковы» (см Рис. 2).

Модель «подковы».

Первый процесс (Architecture Recovery) предусматривает восстановление архитектуры существующей системы посредством извлечения на основании исходного кода характеризующих ее артефактов. Полученная архитектура анализируется на предмет соответствия требованиям к изменяемости, надежности, защищенности и так далее.

Второй процесс (Architecture Transformation) заключается в трансформации (реинжиниринге) восстановленной архитектуры к желаемой новой архитектуре. Полученная в результате трансформации новая архитектура оценивается с позиции ее качества с учетом накладываемых на нее организационных и экономических ограничений.

И, наконец, третий процесс (Architecture-based Development) включает деятельность по разработке системы, соответствующей новой архитектуре. Здесь решаются вопросы декомпозиции элементов системы по пакетам, осуществляется выбор стратегий взаимодействия элементов/компонентов системы. В рамках данного процесса так же обеспечивается интеграция в новую систему артефактов унаследованной системы, например, посредством переписывания части унаследованного кода и/или применения технологии построения оболочек для компонентов унаследованной системы.

С моделью «подкова» соотносится три уровня, на каждом из которых может осуществляться трансформация существующей системы.

Представление на уровне структуры кода (Code-Structure Representation) включает программный код, а так же такие артефакты, как абстрактные синтаксические деревья и графы потоков (flow graphs), получаемые на основании выполнения различного рода аналитических операций (например, разбора кода). Текстуальный перевод, а так же перевод на основе синтаксических деревьев являются примерами трансформации системы на этом уровне.

В отличие от предыдущего уровня, представление на уровне функциональности системы (Function-Level Representation) предусматривает определение связей между функциями программ (например, последовательности вызовов), данными (как частный случай, связей между сущностями данных, соотношений данных и функций) и файлами (к примеру, распределение функций и данных по файлам). Трансформация системы на данном уровне осуществляется на основании пересмотра функциональности системы и заключается, например, в переходе от функционального подхода к объектному подходу к анализу и проектированию, в переходе от реляционной БД к объектной БД.

На архитектурном уровне артефакты предыдущих уровней, объединяются в подсистемы в терминах архитектурных компонентов архитектуры ИС. Трансформация системы на этом уровне предусматривает коренные изменения в структуре программы, в том числе в части применения основных образцов взаимодействия компонентов: типов программных компонентов, используемых соединителей (connectors), вариантов декомпозиции

функциональности, образцов управления и обмена данными времени выполнения.

Согласно модели «подкова» трансформация системы, в зависимости от ситуации, может осуществляться на каждом из представленных уровней, при этом более низкий уровень поддерживает трансформацию на более высоком уровне.

Следует признать, что модель «подкова» находит широкое применение в рамках деятельности, связанной с реинжинирингом ИС. Так, на ее основе определяются требования и основной каркас для интеграции инструментальных средств реинжиниринга на архитектурном уровне и уровне программного кода. С учетом соотносимых с ней процессов и уровней представления, осуществляется расширение модели CORUM (Common Object-based Reengineering Unified Model). Эта модель, в свою очередь, разрабатывалась:

- для представления требуемой для систем управления на основе программного кода (Code-based Management System) информации о программных средствах;
- для обеспечения интероперабельности между системами данного класса (в том числе между средствами реинжиниринга программ на уровне программного кода).

В контексте вводимых расширений в этой же работе для модели «подкова» определяется семантическая модель, обеспечивающая на основании формулируемых унифицирующих свойств семантическую связь между архитектурным уровнем и уровнем программного кода.

Подход очень близок к подходу, основанному на модели «подкова». Характеризуя жизненный цикл реинжиниринга ИС, авторы определяют следующие шаги процесса реинжиниринга:

- анализ требований для выявления конкретных целей реинжиниринга унаследованной системы;
- восстановление модели, в том числе документирование и понимание структуры унаследованной системы;
- выявление проблем, связанных с унаследованной системой;
- анализ проблем, включающий выбор архитектуры, позволяющий устранить обнаруженные в унаследованной системе дефекты;
- реорганизация, включающая выбор и применение оптимального подхода трансформации унаследованной системы;
- распространение изменений.

В отличие от предыдущих работ, деятельность по реинжинирингу рассматривается в качестве одной из форм деятельности по эволюции унаследованных ИС, когда, требуется более сильнодействующее «лекарство» для «лечения» ИС, нежели серия локальных инкрементальных изменений.

Так, описывается каркас (enterprise framework), характеризующий:

- глобальную среду, в которой осуществляется эволюция системы;
- действия, процессы и рабочие продукты (артефакты), которые сопровождают деятельность по эволюции системы.

Авторами подчеркивается, что помимо технических вопросов, связанных с эволюцией унаследованных ИС, существует так же множество организационных вопросов. Например, «Как планировать эволюцию большой сложной системы, включая ее реинжиниринг?», «Какие существуют критичные факторы успеха эволюции системы?», «Как оценить, что люди, осуществляющие эволюцию системы, на правильном пути?». Кроме этого, важным

является необходимость учета стратегических, организационных, и других аспектов бизнеса, влияющих на эволюцию унаследованной ИС.

Поэтому основной целью создания каркаса (enterprise framework) следует считать необходимость оценки среды, в рамках которой осуществляется эволюция унаследованной ИС, необходимость обеспечения понимания широкого спектра вопросов, как технического, так и управленческого характера, которые соотносятся с эволюцией унаследованных ИС. Важнейшим здесь становится выявление факторов, определяющих успех деятельности по эволюции ИС, а так же разработка согласованного множества технических и управленческих инструкций (действий), требуемых для эффективного планирования, оценки и управления инициативами по эволюции систем.

Отражая применительно к эволюции системы пространство проблем и пространство решений, предлагаемый каркас включает такие элементы, как организация, проект, унаследованная система, инженерия систем и программных средств, технологии, целевая система. Между этими элементами определяется связь. При этом для каждого из них приводится список вопросов (checklists), позволяющий охарактеризовать (исследовать и оценить) интересующий элемент.

Следует признать, что положенная в основу каркаса концепция является расширением концепции разработки ИС с учетом деятельности по внедрению, повседневных операций и деятельности по сопровождению. Каркас может быть использован для выполнения следующих видов деятельности:

- исследование и анализ пространства проблем и пространства решений в контексте инициатив по эволюции системы;
- разработка руководств по составлению стратегических и тактических планов по реинжинирингу унаследованных систем;
- выявление технологических вопросов и потенциальных проблем на протяжении всего пути по эволюции систем;
- рецензирование планов, ранжирование (приоритезация) технических вопросов, разработка рекомендаций по улучшениям процессов эволюции систем и результатов их выполнения (рабочих продуктов).

В рамках подхода выделяются две значительные составляющие: основная фаза и фаза эволюции. Основная фаза сфокусирована на таких элементах, как организация, проект и унаследованная система. Фаза эволюции концентрируется на целевой системе, инженерии систем и программных средств, технологиях, используемых для построения целевой системы. Другими словами первая фаза сфокусирована на пространстве проблем, а вторая на пространстве решений.

Предлагается комплексный, основанный на рассмотренном ранее каркасе, подход к эволюции систем, который определяется в контексте унаследованных систем и современных программных технологий.

В основу подхода положены следующие положения (принципы):

1. различие между эволюцией систем и сопровождением программных средств;
2. использование описанного ранее каркаса (enterprise framework) при поддержке принятия решений в процессе эволюции системы;
3. достижение технического понимания систем на высоком уровне абстракции;
4. применение технологий распределенных объектов, технологии «wrapping» для эволюции системы;
5. применение «net-centric» вычислений для эволюции системы.

Определяя различие между сопровождением программных средств и эволюцией систем в части реинжиниринга ИС, авторы рассматривают сопровождение как «мелкозернистую»,

«краткосрочную» деятельность, направленную на планирование и внесение локализованных изменений. При сопровождении структура (архитектура) системы остается относительно неизменной, а требуемая «порция» вносимых за определенный промежуток изменений, как правило, связана с изменением какого-либо одного требования к системе. Такие изменения, обычно, не сопровождаются существенным изменением значений характеристик и атрибутов качества программных средств.

В отличие от сопровождения, эволюция систем рассматривается как «крупнозернистая», «высокоуровневая», форма изменений на уровне структуры (архитектуры) системы. Вносимые в процессе эволюции изменения, приводят к изменениям значений атрибутов качества, что, как правило, существенно упрощает сопровождение систем. Для этого при внесении изменений в структуру системы могут использоваться стратегии «снизу вверх» и «сверху вниз», применение которых основано на ревизии программного кода, восстановлении описания архитектуры унаследованной системы, проектировании новой структуры, новой формы документации и т.д.

Авторами подчеркивается, что эволюция позволяет адаптировать систему сразу с учетом большого количества выдвигаемых к ней требований (включая приобретение новых возможностей), что в конечном итоге увеличивает стратегическую и экономическую значимость программных средств. При этом акцент смещается от понимания программ к пониманию систем, от сопровождения к эволюции и миграции, от стратегий «снизу вверх» к стратегиям «сверху вниз».

В отличие от рассмотренных ранее работ, основное внимание уделяется исследованию и решению технических проблем, связанных с миграцией унаследованных систем. Характеризуя понятие «миграция ИС» в данной работе выделяются отдельно эволюция, сопровождение и миграция ИС. Основываясь на том факте, что объектом эволюции и сопровождения является унаследованная ИС, а объектом миграции как унаследованная, так и новая (целевая) системы, авторы разделяют миграцию и другие процессы ЖЦ ИС. При этом миграция рассматривается как деятельность, которая начинается с унаследованной ИС и заканчивается сопоставимой целевой ИС.

В основу предлагаемого авторами подхода положена декомпозиция структуры системы на компоненты пользовательского интерфейса, компоненты – приложения и компоненты управления базами данных. При этом в качестве основных инкрементально выполняемых шагов миграции выступают:

1. анализ унаследованной ИС;
2. декомпозиция структуры унаследованной ИС;
3. проектирование интерфейсов (пользовательских и системных) целевой системы;
4. проектирование приложений (функциональной логики) целевой системы;
5. проектирование базы данных целевой системы;
6. развертывание среды, требуемой для эксплуатации и сопровождения целевой системы;
7. создание и развертывание необходимых шлюзов;
8. миграция унаследованных данных;
9. миграция унаследованных приложений (компонентов, реализующих функциональную логику);
10. миграция унаследованных интерфейсов (пользовательских и системных);
11. переход к использованию целевой системы.

3. Классификация подходов, методов и технологий.

В настоящий момент существует значительное количество работ, посвященных проблемам, методам и технологиям реинжиниринга ИС. Эти работы охватывают данную

проблематику с различных точек зрения, рассматривая и исследуя в различной степени как проблемы концептуального уровня (иногда даже философского характера), так и конкретные методы, и инструментальные средства, предназначенные для реинжиниринга ИС.

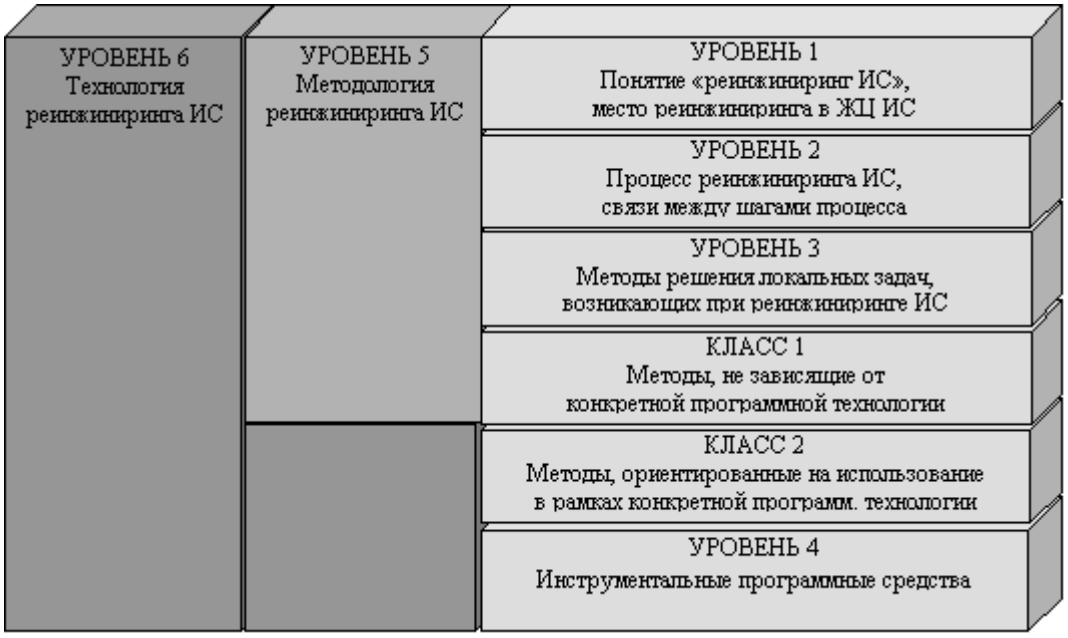
Несмотря на наличие множества различных решений, их исследование и комплексное применение на практике бывает затруднено. Причинами возникающих трудностей следует считать:

1. понятие «реинжиниринг ИС» различными исследователями до сих пор трактуется по разному, существует множество близких понятий, наличие которых приводит к появлению внешне отличающихся, но по сути схожих подходов, методов и технологий;
2. предлагаемые решения не позиционируются в контексте других существующих решений;
3. решения не интегрированы на уровне методологий и технологий, большое количество методов и инструментальных средств направлено на решение отдельных локальных задач, связанных с реинжинирингом ИС;
4. наблюдается разрыв между решениями концептуального характера и решениями, направленными на решение конкретных прикладных задач.

Следует признать, что на множестве существующих решений (подходов, методов и технологий) отсутствует их систематизация, обеспечивающая позиционирование и определяющая взаимосвязь решений на различных уровнях рассмотрения задач реинжиниринга, в том числе, уровнях методологии и инструментальных средств.

В сложившейся ситуации целесообразным видится определение классификации, определяющей структуризацию на множестве существующих подходов, методов и технологий реинжиниринга ИС.

Так, классифицируя существующие подходы, методы и технологии, можно выделить следующие уровни рассмотрения и исследования аспектов, соотносимых с деятельностью по реинжинирингу ИС (см Рис).



Уровни рассмотрения и исследования аспектов, соотносимых с реинжинирингом ИС.

Первый уровень включает исследования, направленные на достижение концептуального понимания деятельности по реинжинирингу ИС. Именно на этом уровне исследуются вопросы адекватного определения понятия «реинжиниринг ИС», определения места реинжиниринга в жизненном цикле (ЖЦ) ИС, в том числе выявление связей процесса

реинжиниринга ИС в целом с другими процессами ЖЦ ИС. Следует считать, что большая часть рассмотренных ранее аспектов, относятся к этому уровню.

В отличие от первого, *второй уровень* содержит исследования, основная цель которых заключается в выявлении основных шагов (действий), реализуемых в процессе реинжиниринга и в определении связей между основными шагами процесса. Здесь в сферу рассмотрения попадают потоки управления и потоки данных между основными шагами процесса, основные роли, соотносимые с исполнителями процесса, а так же правила распределения ролей среди команды исполнителей. Исследования и разработки на этом уровне проводятся как без учета, так и с учетом вводимых ограничений (например, архитектурных решений, которым должны соответствовать подлежащие реинжинирингу ИС).

Так, выделяются основные фазы процесса реинжиниринга ИС, а для каждой фазы определяются действия (деятельности) и соотносимые с ними потоки управления. Процесс дается в самом общем виде и не зависит от каких-либо ограничений, например используемых программных технологий. Авторами так же определяется выполняемый в рамках процесса реинжиниринга поток работ. Однако здесь основное внимание уделяется вопросам технического характера, а выполняемые при реинжиниринге шаги предусматривают декомпозицию структур, соответственно, унаследованной и целевой системы на компоненты пользовательского интерфейса, компоненты – приложения и компоненты управления базами данных. В отличие от предыдущих работ авторами основной акцент сделан на решение задач оценки унаследованных систем и поддержки принятия решений при реинжиниринге ИС. Авторами определяется процесс оценки, состоящий из следующих входящих в его состав процессов (подпроцессов):

1. технической оценки (Reengineering Technical Assessment),
2. экономической оценки (Reengineering Economic Assessment),
3. принятия управленческих решений (Reengineering Management Decision).

Для каждого из процессов описывается его область применения, поток входящих в него работ (шагов процесса), определяются связи с другими процессами оценки. Следует признать, что потенциально эти процессы могут быть интегрированы в любой процесс разработки. Однако стоит заметить, что оценка унаследованных систем является лишь одной из задач по реинжинирингу ИС.

Еще одним примером процесса, направленного на решение локальной задачи, является итеративный процесс, определяющий следующую последовательность шагов, которые должны быть выполнены при планировании проектов реинжиниринга ИС:

1. Определение целей, направлений деятельности организации и информационной системы.
2. Формирование объединенной команды, которая будет осуществлять реинжиниринг унаследованной системы.
3. Определение среды разработки и сопровождения, базирующейся на применении CMM (Capability Maturity Model).
4. Выбор стандартного множества метрик для оценки программных средств.
5. Анализ унаследованной системы.
6. Определение процесса реализации деятельности по реинжинирингу.
7. Разработка/Обновление стандартных средств тестирования и валидации.
8. Анализ средств реинжиниринга.
9. Обучение.

На третьем уровне рассматриваются (исследуются и разрабатываются) методы, каждый из которых направлен на решение некоторой локальной задачи, возникающей в процессе реинжиниринга ИС, например, выполнения определенного шага процесса. По сути, эти методы воплощают собой некоторые вполне конкретные решения, с которыми

соотносится определенная область применения. Как правило, в проектах по реинжинирингу применятся некоторая комбинация таких методов, при этом каждый из них может стать частью методологии реинжиниринга ИС, но в отдельности таковой не является. Более того, объединение этих методов так же нельзя рассматривать в качестве методологии, поскольку между ними не определены связи, обеспечивающие их интегральную целостность. Другими словами отсутствуют системообразующие факторы, делающие набор методов целостным образованием – системой.

С некоторой условностью все методы реинжиниринга ИС можно разделить на два класса.

Методы, относящиеся к первому классу, определены на концептуальном уровне и в целом не зависят от какой-то одной программной технологии. Так, дается обзор методов модернизации и миграции унаследованных систем. Среди всего множества рассматриваемых методов к первому классу относятся метод репликации баз данных и основанный на объектно-ориентированном подходе метод построения оболочек для компонентов унаследованной системы (object – oriented wrapping), методы «белого» и «черного» ящика модернизации системы. Другими представителями данного класса являются: методы оценки вариантов реинжиниринга ИС, метод планирования миграции программных средств, методы извлечения знаний о существующей системе, методы трансформации (реконструкции) архитектуры ИС, методы автоматизации реинжиниринга программ и т.д. В целом следует считать, что область применения таких методов, как правило, характеризуется некоторым классом программных технологий. А для применения в реальных проектах каждый из них адаптируется с учетом используемых в проекте технологий и инструментальных средств.

Отдельным направлением исследований, относящимся к данному классу и получившим развитие в последние годы, является исследование и разработка образцов реинжиниринга ИС. Каждый из образцов реинжиниринга ИС нацелен на решение некоторой типовой задачи (проблемы), которая сопровождает деятельность по реинжинирингу ИС. Не смотря на некоторые отличия, авторы в целом придерживаются единого подхода к описанию таких образцов. Так, определяется следующий шаблон описания.

1. Имя образца.
2. Цели применения.
3. Область приложения.
4. Мотивация к применению.
5. Структура системы до и после применения образца.
6. Процесс применения образца.
7. Обсуждение образца.
8. Особенности, зависящие от языка программирования.

Стоит заметить, что хотя последний из разделов шаблона «привязывает» шаблон к определенной среде реализации, языкам программирования, все же каждый из образцов представляет некоторое концептуальное решение проблемы. Подробную информацию об образцах реинжиниринга можно найти в книге «Object-Oriented Reengineering Patterns».

В отличие от первого класса, методы второго изначально ориентированы на использование определенных программных технологий. Ко второму классу относятся так же адаптации методов из первого класса. Здесь методы в наибольшей степени приспособлены к их непосредственному (прямому) применению в конкретных проектах. Примерами методов данного класса следует считать: методы интеграции с использованием CGI, методы интеграции на основе технологии XML, метод построения оболочек для компонентов унаследованной системы с использованием технологии CORBA.

И наконец, *четвертый уровень* включает исследование и разработку инструментальных программных средств, автоматизирующих применение подходов, методов и технологий, рассматриваемых на предыдущих уровнях. Следует признать, что в настоящий момент таких средств существует большое количество, среди которого выделяются средства:

1. переноса приложений написанных на устаревших языках на современные языки и платформы (например, с языков PL/1, Кобол на языки C++, Java, Visual Basic);
2. средства интеграции унаследованных приложений, к примеру, на основе метода построения оболочек для компонентов унаследованной системы;
3. средства автоматизированного извлечения данных из унаследованных систем;
4. средства автоматизированного извлечения знаний об унаследованных системах;
5. средства оптимизации (реструктуризации) унаследованных систем при их переносе на современные языки и платформы (как на уровне программного кода, так и на уровне архитектуры ИС);
6. различные средства анализа программного кода.

В рамках этой классификации инструментальных средств выделяются такие типы средств, как

1. реинжиниринга бизнес процессов;
2. преобразования имен данных;
3. реинжиниринга данных (БД);
4. прямого инжиниринга;
5. преобразования форматов;
6. реструктуризации;
7. обратного проектирования;
8. трансляции исходного кода;
9. и др.

В этой же работе авторами приводятся характеристики инструментальных средств реинжиниринга ИС. Для каждого из них специфицируется имя программного продукта; требования к аппаратной и программной платформе; координаты компаний – поставщиков; поддерживаемые языки программирования, СУБД, платформы; принадлежность к тому или иному типу средств.

Осуществляя классификацию и исследование существующих подходов, методов и технологий реинжиниринга ИС, дополнительно к уже выделенным уровням, следует добавить еще два, являющихся по своей природе интегральными. Это уровень методологии и уровень технологии реинжиниринга ИС. Первый из них обеспечивает целостное рассмотрение и применение подходов и методов без учета среды их реализации, специфики конкретных проектов. Это соответствует интеграции первых трех уровней, причем на третьем уровне в сферу рассмотрения, в первую очередь, попадают методы первого класса. Уровень технологии обеспечивает адаптацию (конкретизацию) методологии реинжиниринга ИС с учетом среды реализации, специфики конкретных проектов посредством применения методов и инструментальных средств, соответствующих третьему и четвертому уровням. При этом в отличие от методологии, на третьем уровне, прежде всего, рассматриваются методы второго класса.

Следует признать, в процессе проводимых авторами настоящей статьи исследований не было обнаружено целостных методологий и технологий реинжиниринга ИС, соответствующих уровню проработки и области охвата RUP. В

Представленный подход к классификации обеспечивает систематизацию на множестве существующих подходов, методов и технологий реинжиниринга ИС. В качестве его основных областей применения следует рассматривать:

1. оценку состояния в области методологического и технологического обеспечения реинжиниринга ИС;

2. адаптацию и разработку методологий и технологий реинжиниринга ИС.

В завершении стоит отметить, что возможны и другие полезные подходы к систематизации методов и технологий реинжиниринга ИС. Так, предлагаемые и исследуемые различными авторами процессы реинжиниринга, как правило, охватывают систему в целом, в то время как методы могут соотноситься с некоторыми ее составляющими, с отдельными шагами процесса. Последний факт обуславливает возможность классификация многих методов:

- по объектам применения (например, в зависимости от их типа (компонент пользовательского интерфейса, компонент данных, вычисляющий компонент (компонент бизнес - логики)));
- по видам деятельности процесса реинжиниринга (методы извлечения знаний о существующих ИС (методы обратного проектирования), методы оценки (анализа) существующих ИС и т.д.).

Практическое задание.

1. Исходя из предыдущих наработок и своей предметной области, проанализируйте бизнес-процессы ещё раз и найдите в них:
    a. «узкие места»;
    b. Неправильные связи;
    c. Разницу между запланированными процессами и их конечной реализацией.
2. Внесите минимально необходимые изменения после обсуждения с преподавателем.

Лабораторная работа №31.

Рефакторинг бизнес-процессов при внедрении ПО.

Теоретическая часть.

Концепция «рефакторинга» (refactoring) возникла в кругах, связанных со Smalltalk, но вскоре нашла себе дорогу и в лагеря приверженцев других языков программирования. Поскольку рефакторинг является составной частью разработки структуры приложений (framework development), этот термин сразу появляется, когда «структурщики» начинают обсуждать свои дела. Он возникает, когда они уточняют свои иерархии классов и восторгаются тем, на сколько строк им удалось сократить код. Структурщики знают, что хорошую структуру удается создать не сразу — она должна развиваться по мере накопления опыта. Им также известно, что чаще приходится читать и модифицировать код, а не писать новый. В основе поддержки читаемости и модифицируемости кода лежит рефакторинг — как в частном случае структур (frameworks), так и для программного обеспечения в целом.

Так в чем проблема? Только в том, что с рефакторингом связан известный риск. Он требует внести изменения в работающий код, что может привести к появлению трудно находимых ошибок в программе. Неправильно осуществляя рефакторинг, можно потерять дни и даже недели. Еще большим риском чреват рефакторинг, осуществляемый без формальностей или эпизодически. Вы начинаете копаться в коде. Вскоре обнаруживаются новые возможности модификации, и вы начинаете копать глубже. Чем больше вы копаете, тем больше вскрывается нового и тем больше изменений вы производите. В конце концов, получится яма, из которой вы не сможете выбраться. Чтобы не рыть самому себе могилу, следует производить рефакторинг на систематической основе. В книге «Design Patterns» сообщается, что проектные модели создают целевые объекты для рефакторинга. Однако указать цель — лишь одна часть задачи; преобразовать код так, чтобы достичь этой цели, — другая проблема.

Существует несколько методов рефакторинга. Каждый метод описывает мотивацию и технику испытанного на практике преобразования кода. Некоторые виды рефакторинга, такие как «Выделение метода» или «Перемещение поля», могут показаться очевидными, но пусть это не вводит вас в заблуждение. Понимание техники таких методов рефакторинга важно для организованного осуществления рефакторинга. С помощью методов рефакторинга можно поэтапно модифицировать код, внося каждый раз небольшие изменения, благодаря чему снижается риск, связанный с развитием проекта. Эти методы рефакторинга и их названия быстро займут место в вашем словаре разработчика.

Что такое рефакторинг?

Рефакторинг представляет собой процесс такого изменения программной системы, при котором не меняется внешнее поведение кода, но улучшается его внутренняя структура. Это способ систематического приведения кода в порядок, при котором шансы появления новых ошибок минимальны. В сущности, при проведении рефакторинга кода вы улучшаете его дизайн уже после того, как он написан.

«Улучшение кода после его написания» — непривычная фигура речи. В нашем сегодняшнем понимании разработки программного обеспечения мы сначала создаем дизайн системы, а потом пишем код. Сначала создается хороший дизайн, а затем происходит кодирование. Со временем код модифицируется, и целостность системы, соответствие ее структуры изначально созданному дизайну постепенно ухудшаются. Код медленно сползает от проектирования к хакерству.

Рефакторинг представляет собой противоположную практику. С ее помощью можно взять плохой проект, даже хаотический, и переделать его в хорошо спроектированный код. Каждый шаг этого процесса прост до чрезвычайности. Перемещается поле из одного класса в другой, изымается часть кода из метода и помещается в отдельный метод, какой-то код перемещается в иерархии в том или другом направлении. Однако суммарный эффект таких небольших изменений может радикально улучшить проект. Это прямо противоположно обычному явлению постепенного распада программы.

При проведении рефакторинга оказывается, что соотношение разных этапов работ изменяется.Проектирование непрерывно осуществляется во время разработки, а не выполняется целиком заранее. При реализации системы становится ясно, как можно улучшить ее проект. Происходящее взаимодействие приводит к созданию программы, качество проекта которой остается высоким по мере продолжения разработки.

Правила рефакторинга

20. Обнаружив, что в программу необходимо добавить новую функциональность, но код программы не структурирован удобным для добавления этой функциональности образом, сначала произведите рефакторинг программы, чтобы упростить внесение необходимых изменений, а только потом добавьте функцию.
21. Перед началом рефакторинга убедитесь, что располагаете надежным комплектом тестов. Эти тесты должны быть самопроверяющимися.
22. При применении рефакторинга программа модифицируется небольшими шагами. Ошибку нетрудно обнаружить.
23. Написать код, понятный компьютеру, может каждый, но только хорошие программисты пишут код, понятный людям.

Самый важный урок, который должен преподать данный пример, это ритм рефакторинга: тестирование, малые изменения, тестирование, малые изменения, тестирование, малые изменения. Именно такой ритм делает рефакторинг быстрым и надежным.

Принципы рефакторинга

Рефакторинг (Refactoring): изменение во внутренней структуре программного обеспечения, имеющее целью облегчить понимание его работы и упростить модификацию, не затрагивая наблюдаемого поведения.
Производить рефакторинг (Refactor): изменять структуру программного обеспечения, применяя ряд рефакторингов, не затрагивая его поведения.

Рефакторинг не меняет видимого поведения программного обеспечения. Оно продолжает выполнять прежние функции. Никто — ни конечный пользователь, ни программист — не сможет сказать по внешнему виду, что что-то изменилось.

Зачем нужно проводить рефакторинг?

30. Рефакторинг улучшает композицию программного обеспечения
31. Рефакторинг облегчает понимание программного обеспечения
32. Рефакторинг помогает найти ошибки
33. Рефакторинг позволяет быстрее писать программы

Когда следует проводить рефакторинг?

Рефакторингом следует заниматься постоянно понемногу. Надо не решать проводить рефакторинг, а проводить его, потому что необходимо сделать что-то еще, а поможет в этом рефакторинг.

- Правило трех ударов — Вот руководящий совет, который дал мне Дон Роберте (Don Roberts). Делая что-то в первый раз, вы просто это делаете. Делая что-то аналогичное во второй раз, вы морщитесь от необходимости повторения, но все-таки повторяете то же самое. Делая что-то похожее в третий раз, вы начинаете рефакторинг.
- Применяйте рефакторинг при добавлении новой функции
- Применяйте рефакторинг, если требуется исправить ошибку
- Применяйте рефакторинг при разборе кода

Почему рефакторинг приносит результаты

Из-за чего бывает трудно работать с программами? В данный момент мне приходят в голову четыре причины:

- Программы, трудные для чтения, трудно модифицировать.
- Программы, в логике которых есть дублирование, трудно модифицировать.
- Программы, которым нужны дополнительные функции, что требует изменений в работающем коде, трудно модифицировать.
- Программы, реализующие сложную логику условных операторов, трудно модифицировать.

Итак, нам нужны программы, которые легко читать, вся логика которых задана в одном и только одном месте, модификация которых не ставит под угрозу существующие функции и которые позволяют выражать условную логику возможно более простым способом. Рефакторинг представляет собой процесс улучшения работающей программы не путем изменения ее функций, а путем усиления в ней указанных качеств, позволяющих продолжить разработку с высокой скоростью.

Когда рефакторинг не нужен?

В некоторых случаях рефакторинг вообще не нужен. Основной пример — необходимость переписать программу с нуля. Иногда имеющийся код настолько запутан, что подвергнуть его рефакторингу, конечно, можно, но проще начать все с самого начала.
Явный признак необходимости переписать код — его неработоспособность. Это обнаруживается только при его тестировании, когда ошибок оказывается так много, что сделать код устойчивым не удается. Помните, что перед началом рефакторинга код должен выполняться в основном корректно.

Другой случай, когда следует воздерживаться от рефакторинга, это близость даты завершения проекта. Рост производительности, достигаемый благодаря рефакторингу,

проявит себя слишком поздно — после истечения срока. Правильна в этом смысле точка зрения Уорда Каннингема (Ward Cunningham). Незавершенный рефакторинг он сравнивает с залезанием в долги. Большинству компаний для нормальной работы нужны кредиты. Однако вместе с долгами появляются и проценты, то есть дополнительная стоимость обслуживания и расширения, обусловленная чрезмерной сложностью кода. Выплату каких-то процентов можно вытерпеть, но если платежи слишком велики, вы разоритесь. Важно управлять своими долгами, выплачивая их часть посредством рефакторинга.

Однако приближение срока окончания работ — единственный случай, когда можно отложить рефакторинг, ссылаясь на недостаток времени. Опыт работы над несколькими проектами показывает, что проведение рефакторинга приводит к росту производительности труда. Нехватка времени обычно сигнализирует о необходимости рефакторинга.

Рефакторинг и проектирование

Рефакторинг играет особую роль в качестве дополнения к проектированию. Если заранее подумать об архитектуре программы, то можно избежать последующей дорогостоящей переработки. Многие считают, что проектирование важнее всего, а программирование представляет собой механический процесс. Аналогией проекта служит технический чертеж, а аналогией кода — изготовление узла. Но программа весьма отличается от физического механизма. Она значительно более податлива и целиком связана с обдумыванием. Как говорит Элистер Кокберн (Alistair Cockburn): «При наличии готового дизайна я думаю очень быстро, но в моем мышлении полно пробелов».

Существует утверждение, что рефакторинг может быть альтернативой предварительному проектированию. В таком сценарии проектирование вообще отсутствует. Первое решение, пришедшее в голову, воплощается в коде, доводится до рабочего состояния, а потом обретает требуемую форму с помощью рефакторинга. Такой подход фактически может действовать. Мне встречались люди, которые так работают и получают в итоге систему с очень хорошей архитектурой. Тех, кто поддерживает «экстремальное программирование» [Beck , XP], часто изображают пропагандистами такого подхода. Подход, ограничивающийся только рефакторингом, применим, но не является самым эффективным. Даже «экстремальные» программисты сначала разрабатывают некую архитектуру будущей системы. Они пробуют разные идеи с помощью CRC-карт или чего-либо подобного, пока не получат внушающего доверия первоначального решения. Только после первого более или менее удачного «выстрела» приступают к кодированию, а затем к рефакторингу. Смысл в том, что при использовании рефакторинга изменяется роль предварительного проектирования. Если не рассчитывать на рефакторинг, то ощущается необходимость как можно лучше провести предварительное проектирование. Возникает чувство, что любые изменения проекта в будущем, если они потребуются, окажутся слишком дорогостоящими. Поэтому в предварительное проектирование вкладывается больше времени и усилий — во избежание таких изменений впоследствии. С применением рефакторинга акценты смещаются. Предварительное проектирование сохраняется, но теперь оно не имеет целью найти единственно правильное решение. Все, что от него требуется, — это найти приемлемое решение. По мере реализации решения, с углублением понимания задачи становится ясно, что наилучшее решение отличается от

того, которое было принято первоначально. Но в этом нет ничего страшного, если в процессе участвует рефакторинг, потому что модификация не обходится слишком дорого. Рефакторинг предоставляет другой подход к рискам модификации. Возможные изменения все равно надо пытаться предвидеть, как и рассматривать гибкие решения. Но вместо реализации этих гибких решений следует задаться вопросом: «Насколько сложно будет с помощью рефакторинга преобразовать обычное решение в гибкое?» Если, как чаще всего случается, ответ будет «весьма несложно», то надо просто реализовать обычное решение. Рефакторинг позволяет создавать более простые проекты, не жертвуя гибкостью, благодаря чему процесс проектирования становится более легким и менее напряженным. Научившись в целом распознавать то, что легко поддается рефакторингу, о гибкости решений даже перестаешь задумываться. Появляется уверенность в возможности применения рефакторинга, когда это понадобится. Создаются самые простые решения, которые могут работать, а гибкие и сложные решения по большей части не потребуются.

Рефакторинг и производительность

С рефакторингом обычно связан вопрос о его влиянии на производительность программы. С целью облегчить понимание работы программы часто осуществляется модификация, приводящая к замедлению выполнения программы. Рефакторинг, несомненно, заставляет программу выполняться медленнее, но при этом делает ее более податливой для настройки производительности. Секрет создания быстрых программ, если только они не предназначены для работы в жестком режиме реального времени, состоит в том, чтобы сначала написать программу, которую можно настраивать, а затем настроить ее так, чтобы достичь приемлемой скорости.

Второй подход предполагает постоянное внимание. В этом случае каждый программист в любой момент времени делает все от него зависящее, чтобы поддерживать высокую производительность программы. Это распространенный и интуитивно привлекательный подход, однако он не так хорош на деле. Модификация, повышающая производительность, обычно затрудняет работу с программой. Это замедляет создание программы. На это можно было бы пойти, если бы в результате получалось более быстрое программное обеспечение, но обычно этого не происходит. Повышающие скорость усовершенствования разбросаны по всей программе, и каждое из них касается только узкой функции, выполняемой программой.

С производительностью связано то интересное обстоятельство, что при анализе большинства программ обнаруживается, что большая часть времени расходуется небольшой частью кода. Если в равной мере оптимизировать весь код, то окажется, что 90% оптимизации произведено впустую, потому что оптимизировался код, который выполняется не слишком часто. Время, ушедшее на ускорение программы, и время, потерянное из-за ее непонятности — все это израсходовано напрасно.

Третий подход к повышению производительности программы основан как раз на этой статистике. Он предполагает создание программы с достаточным разложением ее на компоненты без оглядки на достигаемую производительность вплоть до этапа оптимизации производительности, который обычно наступает на довольно поздней стадии разработки и на котором осуществляется особая процедура настройки программы. Начинается все с запуска программы под профайлером, контролирующим программу и сообщающим, где расходуются время и память. Благодаря этому можно обнаружить тот

небольшой участок программы, в котором находятся узкие места производительности. На этих узких местах сосредоточиваются усилия, и осуществляется та же самая оптимизация, которая была бы применена при подходе с постоянным вниманием. Но благодаря тому, что внимание сосредоточено на выявленных узких местах, удается достичь больших результатов при значительно меньших затратах труда. Но даже в этой ситуации необходима бдительность. Как и при проведении рефакторинга, изменения следует вносить небольшими порциями, каждый раз компилируя, тестируя и запуская профайлер. Если производительность не увеличилась, изменениям дается обратный ход. Процесс поиска и ликвидации узких мест продолжается до достижения производительности, которая удовлетворяет пользователей.

Разработка тестов

При проведении рефакторинга важным предварительным условием является наличие надежных тестов.

Правила разработки тестов

- Делайте все тесты полностью автоматическими, так чтобы они проверяли собственные результаты.
- Комплект тестов служит мощным детектором ошибок, резко сокращающим время их поиска.
- Чаще запускайте тесты. Запускайте тесты при каждой компиляции — каждый тест хотя бы раз в день.
- Получив сообщение об ошибке, начните с создания теста модуля, показывающего эту ошибку.
- Лучше написать и выполнить неполные тесты, чем не выполнить полные тесты.
- Подумайте о граничных условиях, которые могут быть неправильно обработаны, и сосредоточьте на них свои тесты.
- Не забывайте проверять, чтобы в случае возникновения проблем генерировались исключительные ситуации.
- Опасение по поводу того, что тестирование не выявит все ошибки, не должно помешать написанию тестов, которые выявят большинство ошибок.

Проблемы рефакторинга

- Потребность вносить изменения в существующий код
- Необходимость строго придерживаться поставленной задачи
- Покрывать код проверочными тестами

Признаки, что Вам нужен рефакторинг

- Ваш программный продукт работает, но внесение новой функциональности иногда затягивается на недели;
- В определенных местах Ваш код работает совершенно не так, как Вы того ожидали;
- Вы часто ошибаетесь в сроках реализации поставленной задачи;
- Вам приходится вносить однотипные изменения в разных местах.

Методы рефакторинга

1. Инкапсуляция поля (Encapsulate Field);

2. Выделение класса (Extract Class);
3. Выделение интерфейса (Extract Interface);
4. Выделение локальной переменной (Extract Local Variable);
5. Выделение метода (Extract Method);
6. Генерализация типа (Generalize Type);
7. Встраивание (Inline);
8. Введение фабрики (Introduce Factory);
9. Введение параметра (Introduce Parameter);
10. Подъём поля/метода (Pull Up);
11. Спуск поля/метода (Push Down);
12. Замена условного оператора полиморфизмом (Replace Conditional with Polymorphism);
13. и так далее.

Практическое задание.

1. Перечислите все функции сопровождения.
2. Исходя из предыдущих наработок и своей предметной области
    1. Определите типы сопровождения (корректирующие и т.п.), которые могут понадобиться.
    2. Составьте повременной и пооперационный план сопровождения ПО
3. Составьте план управления конфигурациями своего ПО.

Лабораторная работа №32.

Рефакторинг бизнес-процессов при внедрении ПО.

Теоретическая часть.

Документация по сопровождению ПС (system documentation) описывает ПС с точки зрения ее разработки. Эта документация необходима, если ПС предполагает изучение того, как оно устроена (сконструирована), и модернизацию его программ. Как уже отмечалось, сопровождение - это продолжающаяся разработка. Поэтому в случае необходимости модернизации ПС к этой работе привлекается специальная команда разработчиков-сопроводителей. Этой команде придется иметь дело с такой же документацией, которая определяла деятельность команды первоначальных (основных) разработчиков ПС, - с той лишь разницей, что эта документация для команды разработчиков-сопроводителей будет, как правило, чужой (она создавалась другой командой). Команда разработчиков-сопроводителей должна будет изучать эту документацию, чтобы понять строение и процесс разработки модернизируемого ПС, и внести в эту документацию необходимые изменения, повторяя в значительной степени технологические процессы, с помощью

которых создавалось первоначальное ПС.

Документация по сопровождению ПС можно разбить на две группы:

(1) документация, определяющая строение программ и структур данных ПС и технологию их разработки;

(2) документацию, помогающую вносить изменения в ПС.

Документация первой группы содержит итоговые документы каждого технологического этапа разработки ПС. Она включает следующие документы:

· Внешнее описание ПС (Requirements document).

· Описание архитектуры ПС (description of the system architecture), включая внешнюю спецификацию каждой ее программы.

· Для каждой программы ПС - описание ее модульной структуры, включая внешнюю спецификацию каждого включенного в нее модуля.

· Для каждого модуля - его спецификация и описание его строения (design description).

· Тексты модулей на выбранном языке программирования (program source code listings).

· Документы установления достоверности ПС (validation documents), описывающие, как устанавливалась достоверность каждой программы ПС и как информация об установлении достоверности связывалась с требованиями к ПС.

Документы установления достоверности ПС включают прежде всего документацию по тестированию (схема тестирования и описание комплекта тестов), но могут включать и результаты других видов проверки ПС, например, доказательства свойств программ.

Документация второй группы содержит

· Руководство по сопровождению ПС (system maintenance guide), которое описывает известные проблемы вместе с ПС, описывает, какие части системы являются аппаратно- и программно-зависимыми, и как развитие ПС принято в расчет в его строении (конструкции).

Пользовательская документация программных средств

Пользовательская документация ПС (*user documentation*) объясняет пользователям, как они должны действовать, чтобы применить разрабатываемое ПС. Она необходима, если ПС предполагает какое-либо взаимодействие с пользователями. К такой документации относятся документы, которыми должен руководствоваться пользователь при *инсталляции* ПС (при установке ПС с соответствующей настройкой на среду применения ПС), при применении ПС для решения своих задач и при управлении ПС (например, когда разрабатываемое ПС будет взаимодействовать с другими системами). Эти документы частично затрагивают вопросы сопровождения ПС, но не касаются вопросов, связанных с модификацией программ.

В связи с этим следует различать две категории пользователей ПС: ординарных пользователей ПС и администраторов ПС. *Ординарный пользователь ПС* (*end-*

user)использует ПС для решения своих задач (в своей предметной области). Это может быть инженер, проектирующий техническое устройство, или кассир, продающий железнодорожные билеты с помощью ПС. Он может и не знать многих деталей работы компьютера или принципов программирования. *Администратор ПС* (*system administrator*) управляет использованием ПС ординарными пользователями и осуществляет сопровождение ПС, не связанное с модификацией программ. Например, он может регулировать права доступа к ПС между ординарными пользователями, поддерживать связь с поставщиками ПС или выполнять определенные действия, чтобы поддерживать ПС в рабочем состоянии, если оно включено как часть в другую систему.

Состав пользовательской документации зависит от аудиторий пользователей, на которые ориентировано разрабатываемое ПС, и от режима использования документов. Под *аудиторией* здесь понимается контингент пользователей ПС, у которого есть необходимость в определенной пользовательской документации ПС [13.2]. Удачный пользовательский документ существенно зависит от точного определения аудитории, для которой он предназначен. Пользовательская документация должна содержать информацию, необходимую для каждой аудитории. Под *режимом использования* документа понимается способ, определяющий, каким образом используется этот документ. Обычно пользователю достаточно больших программных систем требуются либо документы для изучения ПС (использование в виде *инструкции*), либо для уточнения некоторой информации (использование в виде *справочника*).

В соответствии с работами можно считать типичным следующий состав пользовательской документации для достаточно больших ПС:

1. *Общее функциональное описание ПС*. Дает краткую характеристику функциональных возможностей ПС. Предназначено для пользователей, которые должны решить, насколько необходимо им данное ПС.

2. *Руководство по инсталяции ПС*. Предназначено для администраторов ПС. Оно должно детально предписывать, как устанавливать системы в конкретной среде, в частности, должно содержать описание компьютерно-считываемого носителя, на котором поставляется ПС, файлы, представляющие ПС, и требования к минимальной конфигурации аппаратуры.

3. *Инструкция по применению ПС*. Предназначена для ординарных пользователей. Содержит необходимую информацию по применению ПС, организованную в форме удобной для ее изучения.

4. *Справочник по применению ПС*. Предназначен для ординарных пользователей. Содержит необходимую информацию по применению ПС, организованную в форме удобной для избирательного поиска отдельных деталей.

5. *Руководство по управлению ПС*. Предназначено для администраторов ПС. Оно должно описывать сообщения, генерируемые, когда ПС взаимодействует с другими системами, и как должен реагировать администратор на эти сообщения. Кроме того, если ПС использует системную аппаратуру, этот документ может объяснять, как сопровождать эту аппаратуру.

Разработка пользовательской документации начинается сразу после создания внешнего описания. Качество этой документации может существенно определять успех ПС. Она должна быть достаточно проста и удобна для пользователя (в противном случае это ПС, вообще, не стоило создавать). Поэтому, хотя черновые варианты (наброски) пользовательских документов создаются основными разработчиками ПС, к созданию их окончательных вариантов часто привлекаются профессиональные технические писатели. Кроме того, для обеспечения качества пользовательской документации разработан ряд стандартов, в которых предписывается порядок разработки этой документации, формулируются требования к каждому виду пользовательских документов и определяются их структура и содержание.

Практическое задание.

1. Перечислите все функции сопровождения.
2. Исходя из предыдущих наработок и своей предметной области
    1. Определите типы сопровождения (корректирующие и т.п.), которые могут понадобиться.
    2. Составьте повременной и пооперационный план сопровождения ПО
3. Составьте план управления конфигурациями своего ПО.

Лабораторная работа №33. Анализ задачи сопровождения

Теоретический материал.

В настоящее время, по мере усложнения и роста стоимости используемых программных систем, все более актуальной становится проблема их сопровождения. С одной стороны, наблюдается ускоренное развитие информационных технологий, требующее постоянных изменений в используемом программном обеспечении, с другой стороны жизненный цикл сложных программных систем должен быть достаточно длительным, чтобы успеть окупить затраты на их создание. По некоторым оценкам стоимость сопровождения современной информационной системы (ИС) может достигать 80% всех затрат жизненного цикла ИС. В то же время задачи этапа сопровождения ИС до настоящего времени остаются мало исследованными по сравнению с задачами этапов анализа требований, планирования и оценки проекта, проектирования, реализации и тестирования.

Сопровождение, по ГОСТ Р ИСО/МЭК 12207-99, – *это внесение изменений в ПО в целях исправления ошибок, повышения производительности или адаптации к изменившимся условиям работы или требованиям*.

Цель сопровождения *является обеспечение функционально-технического состояния системы актуального потребностям Заказчика для эффективной эксплуатации.*

Являясь неотъемлемой частью функционирования программных систем любого масштаба, особое значение процесс сопровождения приобретает в корпоративных системах. Яркий **пример** подобных программ – банковские ИС. Их разветвлённая модульная структура со сложными механизмами сопряжения и высокими требованиями к надёжности данных не может оставаться работоспособной без систематического сопровождения, как внутреннего, так и внешнего.

В стандарте ГОСТ Р ИСО/МЭК 14764–2002 описана **структура модификаций**, производимых в ходе сопровождения ИС, однако эта структура не учитывает специфики задач сопровождения корпоративных информационных систем. Заимствованный принцип разделения производимых в системе изменений на исправление ошибок и изменение функционала был учтён при разработке собственной классификации.

Информационная инфраструктура включает в себя аппаратное и программное обеспечение. На рис. 1 показана иерархия задач сопровождения, учитывающая специфику банковских информационных систем. Все задачи сопровождения разбиты на три класса: развитие ИС, корректирующее сопровождение и сопровождение данных.

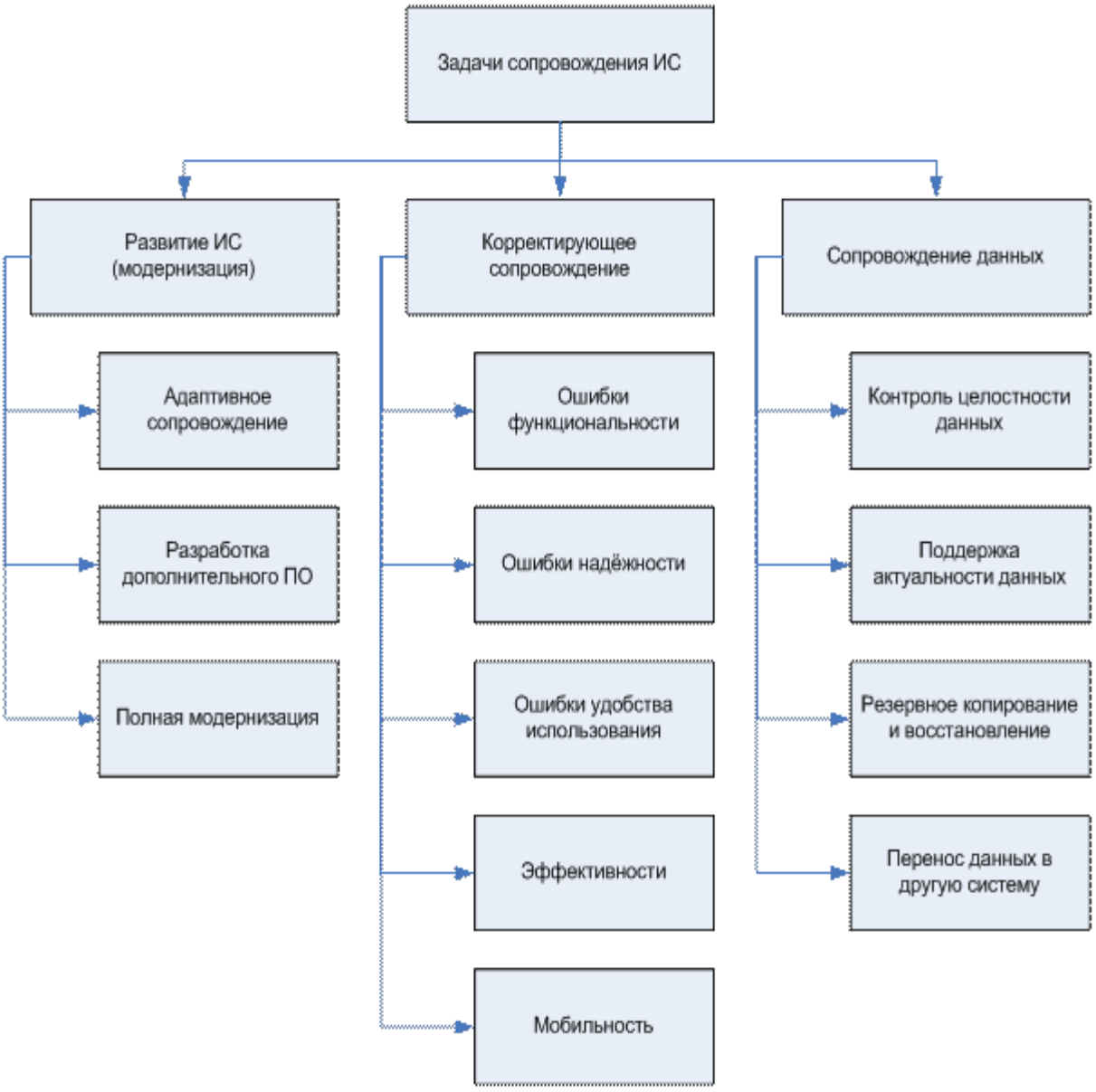


Рисунок 1. Классификации задач сопровождения ИС.

Развитие ИС *предполагает частичную или полную модернизацию*.

В связи с этим, развитие ИС можно разделить:

- на доработку ИС,
- разработку дополнительного ПО;
- на замену ИС на более современную и функциональную.

Адаптивное сопровождение - это доработка программного продукта после поставки, позволяющее адаптировать его к новым условиям эксплуатации.

Полная модернизация является наиболее дорогостоящим этапом в развитии ИС. Активное развитие рынка и все возрастающие потребности банковского бизнеса предъявляют новые требования к системам автоматизации. Сегодня по некоторым данным свыше 13% коммерческих банков абсолютно не удовлетворены функционирующим ИТ-решением и в связи с этим планируют сменить АБС. Понимая, что переход на новую систему требует больших финансовых сложений, затрагивает весь персонал и заставляет на протяжении нескольких месяцев работать параллельно в двух АБС, можно утверждать, что переход на новую АБС в разы труднее, чем ее первоначальное внедрение при открытии банка.

Основным фактором, объясняющим стремление банка сменить функционирующую АБС, является моральное устаревание системы. Для половины опрошенных ИТ-специалистов это вполне объективная причина модернизации информационного обеспечения банка. Кроме того, банки нуждаются в надежном партнере, способном не только разработать, но и обеспечить качественное сопровождение и развитие решения, которое способно опережать рост бизнеса банка.
В случае разработки **дополнительного ПО** очевидны немалые преимущества: *функционал дорабатывается под нужды персонала, необходимые требования будут квалифицированно оценены и выполнены, интерфейс и принципы работы останутся неизменными, доработка будет производиться с использованием существующих средств и методов.* Данный вид модернизации может быть осуществлён в случае, если необходимый функционал требуется лишь на относительно короткий срок, или стандартное решение от разработчика отсутствует.
Корректирующее сопровождение *направлено на выявление и устранение несоответствий и ошибок после поставки программного продукта.*

Информационная система, как и любое ПО, не всегда даёт желаемый результат работы. Применительно к сопровождению ИС **ошибка**, это искажение кода программы или искажение данных, которые в ходе функционирования этой программы могут вызвать отказ или снижение эффективности функционирования.

Под отказом ИС *в общем случае понимают событие, заключающееся в нарушении работоспособности объекта*. При этом критерии отказов, как признак или совокупность признаков нарушения работоспособного состояния программного обеспечения,

определяются в зависимости от функционального назначения той или иной системы или модуля.

В качестве показателя степени тяжести ошибки, позволяющего дать количественную оценку тяжести проявления последствий ошибки, можно использовать условную вероятность отказа программного обеспечения при проявлении ошибки. Оценку степени тяжести ошибки как условной вероятности возникновения отказа, можно производить согласно **ГОСТ 28195 – 89**, используя метрики и оценочные элементы, характеризующие устойчивость программного обеспечения. При этом оценку необходимо производить для каждой ошибки в отдельности, а не для всей ИС.

Среди основных критериев работы банковской ИС можно обозначить надёжность и предсказуемость, которые трактуются как отсутствие недостатков, сбоев и явных ошибок. Недостатки зависят от субъективной оценки качества ИС банковскими служащими – основными пользователями системы, клиентами, получающими документы, подготовленные с использованием ИС, Центральным Банком и другими надзорными органами, в которые предоставляются отчётные данные.
При этом даже при наличии спецификации ошибок, выявленные на конечном этапе недостатки, говорят о низком качестве всей системы в целом. При таком подходе преодоление недостатков ИС, особенно на заключительном этапе проектирования, может приводить к снижению надёжности. Очевидно, что для разработки ответственного и безопасного ПО такой подход не годится, однако проблемы наличия ошибок в спецификациях, субъективного оценивания пользователем качества программы существуют и не могут быть проигнорированы[4]. В банке должна быть разработана система некоторых ограничений, которая бы учитывала эти факторы при разработке и сопровождении ИС. Для обычных программ все проблемы, связанные с субъективным оцениванием их качества и наличием ошибок, скорее всего, неизбежны.

Возникающие ошибки ИС предлагается разделить на **ошибки функциональности, надёжности, удобства использования, эффективности, мобильности.** При этом сопровождение каждой из них может быть как реактивным – в виде реакция на выявленные ошибки, так и профилактическим, которое применяется в особо ответственных модулях и системах во избежание возможных (ещё не возникших) проблем.

В настоящее время банковские ИТ-структуры осознали преимущества централизованной системы отслеживания и решения проблем и ошибок. Система отслеживания ошибок – это прикладная программа, разработанная с целью помочь разработчикам программного обеспечения учитывать и контролировать ошибки, найденные в программах, а также

следить за процессом устранения этих ошибок и выполнения или невыполнения пожеланий. Главный элемент такой системы – это заявка, содержащая основные параметры ошибки и этапы её устранения. База данных заявок является одновременно и классификатором найденных ошибок и базой знаний по исправлениям и доработкам. Система позволяет организовать эффективный процесс сопровождения с сильной обратной связью.

Сопровождение данных. Важной спецификой корпоративных ИС является значительно превышение продолжительности жизненного цикла (ЖЦ) данных над продолжительностью ЖЦ программной среды, технологий обработки, бизнес логики и т.д. Поэтому целесообразно вынести поддержку данных в отдельный класс задач, состоящий из контроля целостности данных, поддержки актуальности данных и резервного копирования и восстановления, а также переноса данных из одной системы в другую.

Информационная инфраструктура коммерческого банка изобилует разнородными базами данных, справочниками, классификаторами, каталогами, файловыми хранилищами. Объёмы информации исчисляются десятками и сотнями гигабайт, поэтому задача сопровождения данных сводится к поддержанию их целостности, актуальности и высокому уровню защиты от воздействий извне.

Понятие **целостности** используется в контексте терминологии информационной безопасности, при этом объектами, по отношению к которым он применяется, могут быть информация, специализированные данные, ресурсы автоматизированной системы и пр. Целостность информации определяется как состояние информации, при котором её изменение осуществляется только преднамеренно субъектами, имеющими на него право, либо таковое отсутствует.

Актуальность – это свойство данных в указанный момент времени адекватно отображать состояние объектов предметной области.

Если целостность данных поддерживается с помощью СУБД, то наиболее популярным средством поддержания актуальности является механизм создания резервных копий. Теоретически должна существовать возможность восстановления (получения актуализированной версии системы) на любой день, минуту и секунду. Практика же диктует свои требования, в частности в силу высокой ресурсоёмкости процесса создания резервной копии, он должен проводиться в период отсутствия нагрузки на сервер ИС. Как правило, таким периодом выбирают время с окончания операционного дня (обычно 21:00 текущего календарного дня) по начало следующего (8:00 следующего календарного дня).

Нескольких часов оказывается достаточно для создания архива и записи его в хранилище. В редких случаях, когда БД настолько велика, что оценочное время процесса архивации превышает допустимое, вместо упаковки данных применяют резервное копирование всего программного комплекса на аналогичный компьютер или виртуальную машину, откуда уже производят запись в архив. Таким образом, в случае программно-аппаратного сбоя актуальность данных АБС может быть восстановлена в пределах одного операционного дня.

Существуют программные комплексы, для которых необходимо поддерживать актуальность в течение дня. Например, программный комплекс, взаимодействующий с Расчётным Центром Информатизации Банка России, с использованием которого производится отправка и получение платёжных документов. Транспортной единицей в данном ПК является рейс – защищённый ключами шифрования и аутентификации блок данных с информационными полями. В крупных банках за день может быть отправлено до нескольких сотен рейсов. В случае непредвиденной ситуации (сбой, удаление пакета), восстановить состояние системы, используя сведения на конец прошлого дня, практически невозможно. Во избежание подобных происшествий, каждый пакет перед отправкой и при поступлении добавляется в архив. Правила именования и нумерации файлов позволяют однозначно определить дату и время пакета, его направление (входящий/исходящий) и однозначно позиционировать в очереди сообщений. Иначе обстоит дело с системами денежных переводов. Принципы обработки данных тесно связаны с ведением бухгалтерского учёта в банках, который строго регламентируется нормативной документацией Центрального Банка РФ, поэтому в части создания, подтверждения, выплаты, отправки системы практически не различаются. Чего нельзя сказать о принципах хранения и доступа к данным. Здесь сопровождение напрямую связано с местом хранения и способам работы с информацией.
На сегодняшний день существует коммерческое и открытое ПО, позволяющее контролировать точность и неизменность версий файлов. Большинство инструментов мониторинга за актуальностью системы основаны на сравнении текущего состояния системы и состоянию, которому можно доверять. Это состояние часто называют базовой целостностью, получаемой с заведомо исправной системы.

Сетевые инструменты мониторинга привлекают особое внимание, потому что они обеспечивают наблюдение за несколькими системами на уровне пересылаемых пакетов. Но способность видеть пакеты, исходящие от атакующего к уязвимой системе, всего лишь предупреждает об опасности и часто слишком поздно. Для того чтобы узнать, как система ответила и была ли атака успешной, необходимо исследовать вредоносный объект.

Инструменты целостного мониторинга позволяют детально изучить атаки на систему, на которой они установлены.

Существует много способов установить наблюдение за отдельно взятой системой. Но стоит помнить, что ни одна программа или приложение не может сама по себе гарантировать полный контроль над целостностью всей системы. Лучше всего иметь несколько инструментов обнаружения подозрительного поведения и несанкционированных изменений.

Инструменты мониторинга за целостностью системы могут существенно улучшить политику безопасности банка. Наряду со встроенными средствами контроля целостности, защиты и правильным ведением логов, такие инструменты служат хорошим препятствием для злоумышленника и являются хорошим средством обеспечения безопасности банковской ИС.

В отдельную группу задач сопровождения данных вынесен перенос данных в другую систему. Проблема связана с сохранением существующих связей и семантики. Процесс переноса данных не должен порождать дублирование записей, равно как и их потерю.

Практическое задание. Используя предметные области (см. список ниже), выполните следующие действия:

1. Определите, какие виды работ по сопровождению вашего программного продукта требуется выполнить.
2. Проанализируйте, с помощью каких средств вы сможете выполнить эти работы.
3. Составьте список ресурсов и необходимых действий.
4. Составьте план сопровождения программного продукта.

Лабораторная работа №34. Реализация запроса на сопровождение

Теоретический материал.

В договорах и регламентах сопровождения поставщики ПО предлагают поочередно выделить приоритеты, правда, в контексте постановки запросов к службе сопровождения. Для рынка WMS это неприемлемо. Наш «самолет» при полном отказе, в отличие от их «танка», попросту разобьется. Я предлагаю рассматривать приоритеты с другой точки зрения: что резервировать, что оставить своей IT-службе, что должно быть элементом гарантии, а что частью собственно сопровождения, в том числе и платного.

Приоритет 1. Программное обеспечение полностью неработоспособно. Сбой при запуске или зависание системы. Большинство функций системы не выполняется, что существенно влияет на бизнес заказчика.

Приоритет 2. Программное обеспечение функционирует частично либо с низкой производительностью, что влияет на бизнес заказчика, при этом часть работ может выполняться.

Приоритет 3. Программное обеспечение функционирует, но встречаются незначительные проблемы или дефекты, или некоторое снижение производительности.

Приоритет 4. Услуги по установке и настройке программного обеспечения. Обновления системы. Вопросы, возникающие при эксплуатации программного обеспечения, не оказывающие влияния на бизнес заказчика. Консультации по изменению конфигурации программного обеспечения. Запросы на доработку программного обеспечения. Ошибки в документации.

Элементы гарантии

Приоритет 1. В случае эксплуатации WMS – неработоспособная система не только «существенно влияет на бизнес», она его парализует. Поэтому все оборудование комплекса должно иметь резерв для замены при выходе из строя, все данные должны иметь постоянно обновляемые резервные копии, а системное ПО должно быть корректно настроено. И это бремя ложится на заказчика. Если все это изначально должно быть обеспечено поставщиком, то есть полное право требовать работоспособности системы «бесплатно и мгновенно». Включать в договор сопровождения первый приоритет – самоубийство.

Приоритет 2. Объем настроек должен четко определяться в проектной документации. Если выявляется, что какая-то часть системы функционирует не так, как предполагалось, то это – опять же гарантийные обязательства поставщика. Если даже в договоре будет программа испытаний и акт их проведения, а также акт приемки в эксплуатацию, дефекты могут выявляться и спустя значительное время. Протестировать все возможные варианты взаимодействия процессов в системе невозможно, но это проблема производителя ПО.

Другой стороной медали частичной функциональности может быть то, что персонал заказчика может изменить конфигурацию системы и использовать систему не так, как планировалось, тестировалось и принималось. В этом случае поставщик, во-первых, вправе сравнивать текущие настройки с теми, которые были на момент приемки работ.

Во-вторых, он вправе требовать от заказчика ведения журнала изменений конфигурации. Это поможет быстро локализовать проблему. Не всегда возврат к предыдущей конфигурации автоматически восстанавливает работоспособность, при работе с конфликтной конфигурацией данные могут быть изменены так, что программа и с корректными настройками будет капризничать.

В договорах внедрения и сопровождения необходимо предусмотреть все описанные варианты и их последствия. Корни конфликтов в большинстве случаев скрыты здесь. Внесение изменений в одной части программы может повлечь проблемы в другой. Такова специфика класса программ WMS.

Снижение производительности

Приоритет 3 не имеет четких границ с приоритетом 2. Что может означать формулировка «незначительные проблемы»? Требуется более четкое определение в договорах, например, речь может идти об ошибках интерфейса пользователя, которые вызывают некоторый дискомфорт в работе, но никак на алгоритмы и данные не влияют.
Проблемы снижения производительности – это отдельный разговор. Любые системы при длительном использовании начинают «подтормаживать», растет объем базы данных, и запросы начинают выполняться медленнее. При продаже разработчики всегда заявляют требования к аппаратным средствам. Эти требования должны выдвигаться осознанно, и случай, если потребитель выполнил условия, а уровень быстродействия низок, является гарантийным. Поставщик может ограничивать в договоре максимальный объем хранимых данных. Это связано с тем, что многие бизнесмены стремятся хранить всю историю перемещений товаров по всему складу за весь период работы программы в рабочей базе «на всякий случай». Объективной необходимости в этом нет. Если вы торгуете золотыми слитками, то объем базы будет небольшим.

Полное исследование гигабайта информации с целью выявления, кто и когда три года назад что-то сделал не так, обойдется дороже, чем ущерб, нанесенный ошибкой. Обычно оптимальным бывает срок от 3 месяцев до 1 года. Этот срок может быть связан с регламентом инвентаризаций. При необходимости хранения данные могут быть перенесены из рабочей базы в архивную.

Приоритет 4 – единственный, который попадает под определение сопровождения, вернее – никак не попадает под гарантийные обязательства. Есть ли необходимость в таком сопровождении? Есть, но это точно не абонентское обслуживание, ни одна из услуг не прогнозируема. Рамочный договор на сопровождение просто удобнее, чем постоянное

заключение отдельных договоров на какую-либо услугу, но объем работ и оплата могут и должны определяться в каждом случае отдельно.

Что касается услуг по установке и настройке программного обеспечения, то они могут иметь место, если речь идет о:

а) настройке ПО-радиотерминалов;

б) настройке рабочих станций в клиент-серверной модели, но WMS с такой архитектурой не так много, большинство – трехуровневые;

в) создании резервного сервера уже в процессе эксплуатации и настройки схемы резервирования.

Во всех случаях эффективнее делать это с помощью собственного персонала. Поставщик обязан научить его выполнять такие работы, создать четкую инструкцию по установке и, конечно, консультировать по вопросам установки. Ему приходится делать это чаще и опыт безусловно больше.

Обновления системы

Человек, работающий в области эксплуатации систем, должен быть максимально консервативен. Экспериментаторство – это удел разработчиков. Делать обновление за обновлением не стоит. В них исправляются не ваши проблемы, но иногда при таком исправлении создаются «ваши». Если обновление делалось специально как ответ на ваш запрос об исправлении ошибки, тогда – да. В остальных случаях администратор обязан наложить вето. Поставщику выгоднее держать всех своих пользователей на одной и той же версии, так меньше затраты на сопровождение. Но ведь это его проблемы. Вот пусть и делает обновление бесплатно и гарантирует при этом, что все возникающие при этом ошибки он тоже устранит. Вопросы по использованию ПО – это бесплатная форма сопровождения де-факто на рынке WMS. За советы денег не берут, если, конечно, ответ на ваш вопрос не заставит весь офис поставщика неделю искать на него ответ.

Консультации по изменению конфигурации и запросы на доработку WMS – это по большей части внедренческий вопрос или вопрос разработчику. Сопровождение только зафиксирует запрос и переадресует его выше. Время реакции здесь спрогнозировать трудно. Ошибки в документации тоже должны исправляться бесплатно.

«Мелко нашинкованные услуги»

Дайте определения как можно большему числу понятий в пакете проектных договоров. В этот пакет войдут:

лицензионный договор, по которому поставляется ПО и, конечно, гарантийные обязательства по нему;

договор на внедрение, в котором (или в приложениях к которому) определяются границы проекта, объем и состав работ по проекту, процедуры внесения изменений в ходе проекта;

договор на сопровождение.

В пакете договоров необходимо дать определения всем понятиям, которыми оперируют эти договоры. Например, должно быть четко определено, что такое «ошибка программного обеспечения». На первый взгляд вопрос выглядит тривиальным, но WMS – это сложный «многослойный» комплекс аппаратных и программных средств, взаимодействующих между собой и с пользователями (по причине этой «многослойности» следовало бы дать определение и самому понятию «программное обеспечение»). Поэтому ошибки могут быть на любом из уровней, и не всякая ошибка проявляется и легко обнаруживается. Первое, что приходит в голову, – это записать: «ошибкой считается любое поведение системы, не соответствующее документации на систему». Но это нежизнеспособное определение, в документации не могут быть описаны все возможные последовательности действий в системе во всех вариантах. Поэтому придется расширять это определение некоторым «контекстом использования», и этому контексту надо будет тоже давать определение. **«Полной неработоспособности» надо тоже давать определение.** В системах управления складом вполне может работать, к примеру, приемка и не работать отгрузка. С точки зрения бизнеса – это полная неработоспособность, склад не может функционировать, а с формальной точки зрения программа все же работает! Чтобы не было потом ненужных споров, лучше определить в договоре, «невозможность каких бизнес-процессов в системе считается полным отказом».

Совершенно необходимо дать определение «времени реакции поставщика», а для этого необходимо определить, с какого события это время начинается, например, со времени отправки электронного письма определенного содержания на определенный адрес, определенным персоналом заказчика. Здесь лучше использовать каналы, которые регистрируют время независимо от контрагентов, например, операторы мобильной связи, почтовые провайдеры и т. п. Также необходимо определить, каким событием это время заканчивается, например, это может быть тот же самый ответ по почте, или время появления сотрудника сопровождения на объекте, или момент восстановления работоспособности.

Время реакции может зависеть от степени критичности проблемы, при полной неработоспособности оно должно быть минимальным. Придется оговаривать и время реакции в рабочее (для поставщика) и нерабочее время. Поэтому в договоре надо давать и определение рабочего времени.

Переговорный процесс по договорам позволяет понять степень моральной ответственности сторон по обсуждаемым вопросам и уровень компетенций. Если сторона уверена в своем продукте и своих сотрудниках, то более четкое определение понятий она только приветствует. Хороший договор, четко регламентирующий отношения, это залог того, что возникающие проблемы (а они неизбежны) будут устраняться, а не расти как снежный ком.

Практическое задание.

1. Установите на вашем ПК Apache Server. Проанализируйте возникающие при этом проблемы и зафиксируйте их.
2. Устраните возникшие проблемы, выполните отчёт о ходе устранения.
3. Установите на вашем ПК пакет Apache OpenOffice, при этом версия программного продукта должна быть не последней и не локализованной.
4. Сделайте так, чтобы установленный Open Office не изменил ассоциацию файлов, он должен остаться на ПК в качестве вспомогательного продукта.
5. Воспользуйтесь составленным ранее планом сопровождения программного продукта или выполните следующие действия: обновите программный продукт до последней версии; установите пакет локализации; зарегистрируйте программный продукт; сделайте невозможным его использование всеми пользователями данного компьютера, кроме одного-единственного.
6. Установите на ваш ПК любое ПО для удалённого администрирования и выполните деинсталляцию вышеуказанных программ на ПК других студентов.

Лабораторная работа №35. Определение качества сопровождения

Теоретический материал.

Качество сопровождения

Сопровождение нередко кажется тяжким грузом, но его можно рассматривать и как возможность продемонстрировать высокое качество обслуживания потребителей. Беннетт пишет, что такое отношение к сопровождению широко распространено в Японии. Качественное сопровождение можно считать необходимым условием для достижения постоянной удовлетворенности покупателя и для получения его будущих заказов. Некоторые предприятия выделяют сопровождение в самостоятельный вид деятельности, потому что оно может стать надежным долгосрочным источником дохода.

Метрики сопровождения

Поскольку на сопровождение уходит значительная часть общей суммы трудозатрат за время жизни приложения (иногда для предприятий это оказывается неожиданностью), особенно важным становится наличие адекватной методики оценки затрат на сопровождение и доходов от него. Перечислим основные метрики сопровождения:

- количество строк сопровождаемого кода;
- трудозатраты на решение задач по сопровождению;
- количество дефектов.

Прежде всего следует определить цели сопровождения конкретного приложения, после чего выбрать дополнительные метрики, с помощью которых можно будет оценивать успех в достижении поставленных целей. Зависимость выбора метрик от поставленных целей демонстрирует таблица.

Выбор метрик в соответствии с целью сопровождения

Цель	Вопрос	Выбор метрики
Максимальное удовлетворение потребителя	Какие недостатки сказываются на удовлетворении потребителя?	* (1) частота отказов
		* (30) среднее время до отказа
		* отношение дефектов и исправлений ([Количество дефектов, появившихся в результате сопровождения]/[Количество устраненных дефектов])
	Сколько времени уходит на устранение недостатков?	* устранение отказов (среднее время на исправление недостатка с момента начала работ по исправлению)
		* длительность существования дефектов (среднее время от обнаружения дефекта до валидации исправления)
	Каковы основные препятствия?	* коэффициент использования персонала по видам задач (среднее количество человеко-месяцев на обнаружение и исправление каждого дефекта)
		* коэффициент использования компьютеров (среднее рабочее время / среднее системное время на один дефект)
Оптимизация трудозатрат и графика	На что уходят силы?	Трудозатраты и затраты календарного времени в расчете на дефект по категориям сложности
		* планирование
		* воспроизведение ситуации
		* отчет об ошибке
		* устранение недостатков
		* усовершенствование
Минимизация количества дефектов (продолжение сосредоточенного тестирования по схеме разработки)	Где наиболее вероятно обнаружение дефектов?	* (13) Количество точек входа и выхода для каждого модуля
		* (16) Цикломатическая сложность (см. раздел 7.6.1.2)

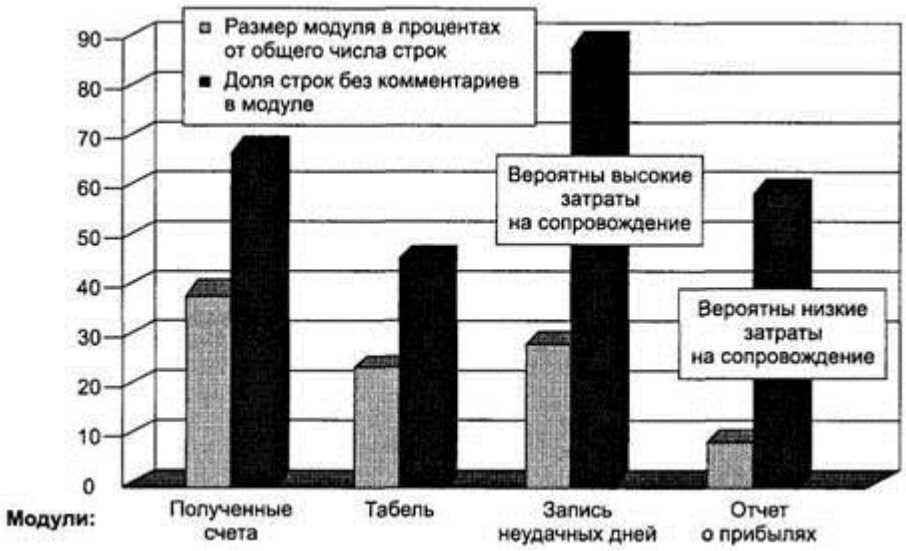
* Для метрик, взятых из стандарта IEEE, в таблице приведены соответствующие номера

Рассмотрим эти метрики более подробно.

(1) Частота отказов. Отказы — это дефекты, обнаруженные во время тестирования или в процессе эксплуатации приложения. Вычисляется как отношение количества найденных дефектов к величине NCSLOC. Последняя аббревиатура расшифровывается как «тысяч строк исходного кода не считая комментариев». (30) Среднее время до отказа. Среднее время получения отказа после запуска приложения. Эта метрика требует введения определения отказа для тестируемого приложения. Определение зависит от того, что будет восприниматься потребителем как отказ. Отказом может считаться как аварийное завершение приложения, так и возникновение определенных конкретных проблем. Для финансового приложения как отказ может быть определена ошибка в вычислениях на величину не менее 1 доллара.

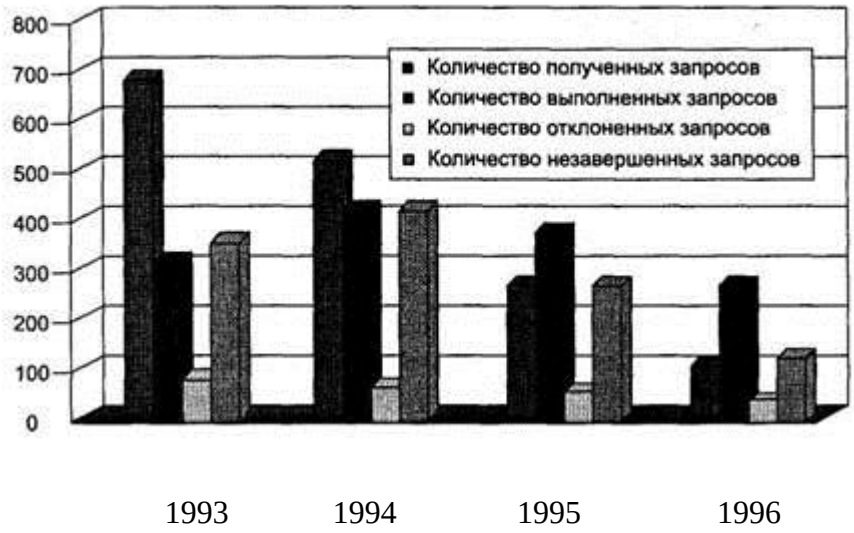
Применение метрик сопровождения

В этом разделе обсуждаются вопросы применения метрик для управления действиями по сопровождению. Доля комментариев в общем числе строк исходного кода позволяет предсказать масштаб трудозатрат на сопровождение. По сравнению с тремя другими модулями модуль «Запись неудачных дней» создаст больше всего трудностей при сопровождении из-за большой доли не комментированных строк и своего большого объема. Сопровождать модуль «Отчет о прибылях» будет проще всего, потому что он имеет наименьшие размеры и высокую долю комментариев. Долю комментариев можно вычислить при помощи специальной программы или путем изучения взятых наугад участков кода.



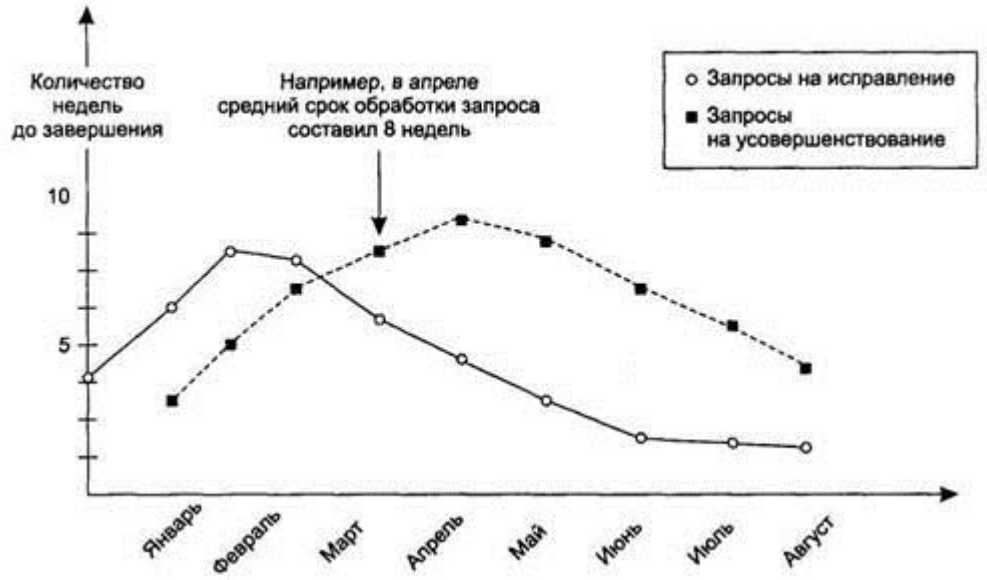
Оценка трудозатрат на сопровождение

Для управления затратами на сопровождение полезны графики, аналогичные приведенному на рисунке выше. В соответствии с этим графиком большое количество запросов на исправления и усовершенствования прибывает в первые два года, в результате чего на второй год эксплуатации возникает пик задержек выполнения запросов. Постепенно задолженность устраняется. Общий вид кривых на этом графике достаточно типичен, изменяется только масштаб по оси времени (годы, месяцы, недели).



Профиль количества запросов на устранение недостатков

Ранее в этой лекции мы подчеркивали разницу между исправлением недостатков и усовершенствованием приложения. Обычно руководитель отдела сопровождения старается учитывать затраты на эти два вида деятельности отдельно друг от друга, чтобы усовершенствования оплачивались заказчиком. Для достижения эффективного распределения объема работ полезны графики, подобные рисунку ниже. На графике показано среднее количество недель ожидания решения по запросу на сопровождение. Время отсчитывается с момента поступления первого отчета. Для исправлений средний срок составляет около одной недели, а для усовершенствований — около четырех недель.



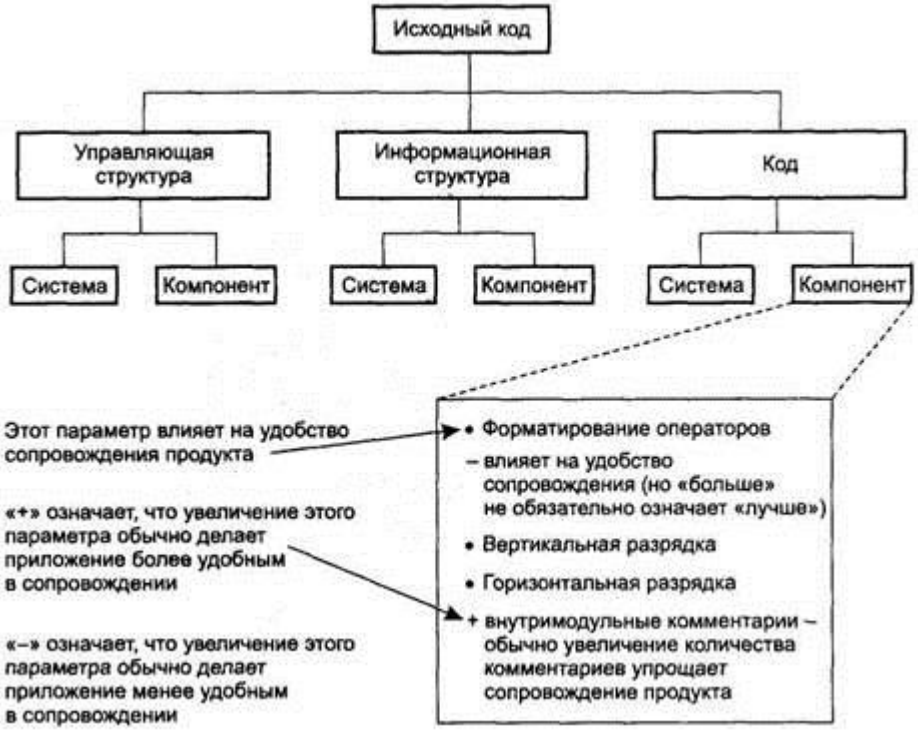
Пример профиля задержки до принятия решения о выполнении запроса

Удобство сопровождения

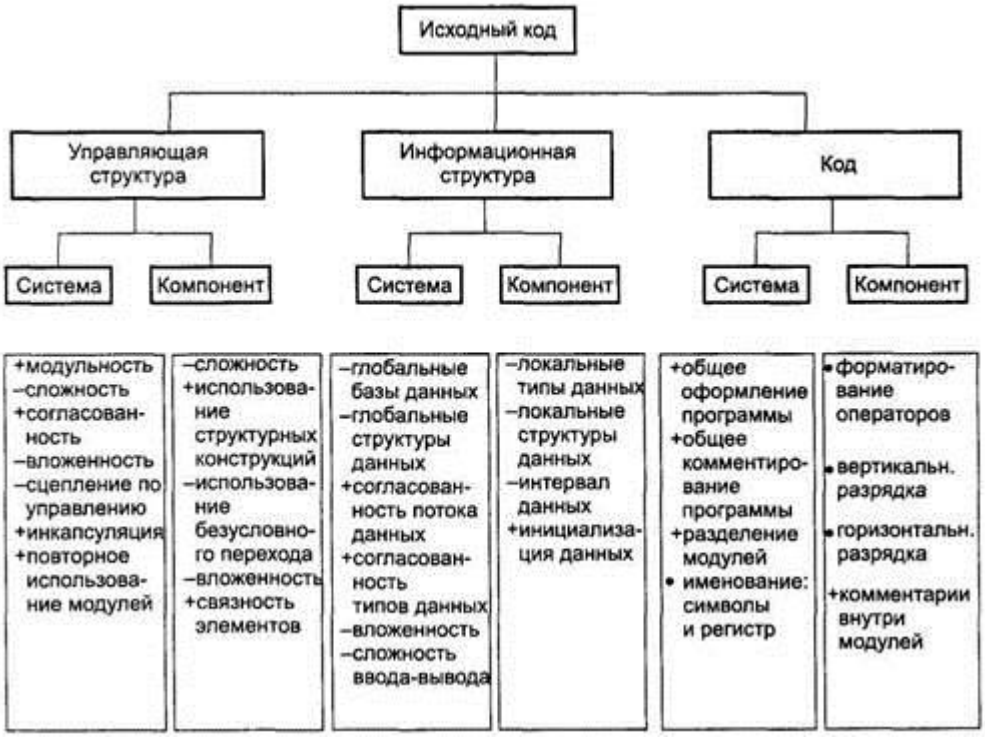
Оман выделил основные параметры исходного кода, влияющие на удобство сопровождения приложения. Проведенное им разбиение исходного кода по типам показано на рисунке ниже. Автор изменил предложенное Оманом представление, чтобы сделать рисунок доступнее.

Например, чем лучше система разбита на модули, тем проще ее сопровождать (Исходный код ► Управляющая структура ► Система). Чем лучше данные инициализируются, тем проще их сопровождать (Исходный код ► Информационная структура ► Компонент). Читатель, несомненно, обратит внимание на то, что большинство перечисленных качеств уже рассматривались в этой книге с точки зрения качества проектирования и реализации.

Вспомните, что основным мотивом использования образцов проектирования является обеспечение удобства сопровождения приложений. Например, образец проектирования State позволяет с легкостью добавлять новые состояния, не изменяя функциональность имеющихся. К сожалению, усовершенствованные методы разработки систем обычно приводят к увеличению, а не к уменьшению затрат на сопровождение. Судя по всему, это связано с тем, что хорошо разработанные приложения проще изменять, поэтому мы чаще прибегаем к их адаптации к новым условиям.



Влияние параметров исходного кода на удобство сопровождения (1)



Влияние параметров исходного кода на удобство сопровождения (2)

10 важных метрик и KPI для оценки службы поддержки

В современном мире необходимо постоянно следить за эффективностью бизнеса, особенно по трем подразделениям, которые общаются с конечными клиентами: продажам, маркетингу и поддержке. К счастью, работа этих подразделений относительно легко

измерима. И есть несколько методов, позволяющих получить значимые цифры, описывающие ситуацию.

Если все делать правильно, эти подразделения окажут позитивное влияние на прибыль компании. Но как узнать, что есть правильно?

Как и в случае с продажами/ маркетингом, оценивать поддержку необходимо по метрикам и KPI. Если цифры показывают, что вы не достигаете поставленных целей, необходимо скорректировать свою стратегию. При этом не измерять нельзя: без объективных данных вы не можете улучшить сервис, особенно в современных условиях быстрой смены клиентских предпочтений. Исследование “2018 Consumer Intelligence Series: Experience is Everything” (доступно по адресу https://www.pwc.com/us/en/services/consulting/library/consumer-intelligence-series/future-of-customer-experience.html?WT.mc_id=CT11-PL1000-DM2-TR2-LS4-ND30-TTA5-CN_FutureofCXIEO-14&eq=infeditorial_hyken) показало, что **клиенты готовы тратить до 16% больше в компаниях, которые предлагают хорошее обслуживание и персонализированную поддержку.**

Далее мы сфокусируемся на метриках клиентского сервиса, KPI службы поддержки, а также измерении степени удовлетворенности клиентов. В этом нет ничего сложного. Но надо иметь в виду, что ключевые метрики, за которыми стоит следить, зависят от целей подразделения или бизнеса в целом.

Среднее время первого ответа — Average First Response Time (AFRT)

Или "время реакции". Метрика показывает, сколько в среднем клиенту приходится ждать, прежде чем он получит первый ответ на свой запрос в службу поддержки. Считается, что клиенты готовы подождать, если поддержка действительно качественная. Однако наблюдение за AFRT гарантирует, что клиенты получат ответ за приемлемое время. **Важно:**

- чем меньше время ответа, тем лучше;
- время ответа может быть разным в зависимости от ряда факторов, например, в зависимости от категории клиента (по обращениям VIP клиентом можно и нужно отвечать быстрее), от самого времени обращения (днем или ночью) и т.д.. При этом в некоторых Help Desk системах вы сами можете настраивать подобные "зависимости":
- стоит установить целевые показатели, в зависимости разных факторов и контролировать их;
- если реальность далека от цели, установите причины. Это может быть, к примеру, недостаточное количество сотрудников на первой линии.

Среднее время ответа — Average Reply Time (ART)

В отличие от AFRT, эта метрика показывает, как быстро решаются вопросы клиентов.

Важно:

- как и с AFRT, чем быстрее вы возвращаетесь к клиенту с итоговым ответом, тем лучше. Ставьте цели и достигайте их;

- если результат отрицательный, выясните, в чем причина, и примите меры для оптимизации процессов.

Общее количество заявок — Number of Support Tickets

Большое количество заявок — метрика двоякая. С одной стороны, хорошо, когда служба поддержки справляется с объемом. Но, с другой стороны, большое количество заявок может указывать на проблему с продуктами или услугами.

Важно:

- меньшее число заявок — к лучшему. Ставьте перед собой цель сократить их объем;
- наблюдайте за количеством заявок в единицу времени — следите за изменением этого показателя;
- выявляйте факторы, которые ведут к росту числа заявок.

Количество просроченных заявок — Number of Ticket Backlog

Как отмечено выше, клиенты не против подождать, если в итоге их сложная проблема будет действительно решена. И, понятно, есть некоторые проблемы, которые не удастся решить за условные 2-3 дня. Но важно следить за числом (а в идеале процентным соотношением) таких заявок и не допускать нарушения баланса между ними и заявками, которые решаются быстро.

Важно:

- чем меньше таких заявок, тем лучше. Ставьте соответствующие цели;
- возможно, для сокращения числа просроченных заявок требуется нанять больше сотрудников в службу поддержки?

Доля заявок, закрытых в ходе первого обращения — First Contact Resolution Rate

Клиенты не любят повторно связываться со службой поддержки, чтобы наконец-то получить ответ на свой вопрос. Поэтому количество заявок, решенных в ходе первого обращения (например, в одном сеансе чата или телефонном звонке), напрямую связано с клиентской удовлетворенностью.

Важно:

- для расчета метрики достаточно разделить количество заявок, закрытых в ходе первого обращения, на общее количество запросов в поддержку;
- ставьте цель получить большую долю заявок, закрытых в ходе первого обращения. Однако не любые средства ее достижения хороши (посадить на первую линию специалиста со второй линии — не оптимально);

- анализ ситуации позволит вам определить, в каких областях вашей деятельности чаще всего возникают проблемы, и можно ли их быстро решить. Кроме того, так вы можете оценить эффективность (или неэффективность) службы поддержки.

Доля решенных заявок — Resolution Rate

Понимание доли заявок, фактически решенных сотрудниками поддержки (от общего количества клиентских обращений) дает представление об индивидуальной и командной эффективности и продуктивности.

Важно:

- чем выше эта доля, тем лучше. Стоит стремиться к тому, чтобы решить больше проблем, сокращая издержки;
- прежде чем работать с этим показателем, стоит выяснить, есть ли глобальные (более высокоуровневые) проблемы, которые усложняют работу поддержки, отнимая большое количество времени;
- для повышения командной продуктивности можно нанять больше сотрудников поддержки.

Среднее время обработки заявки — Average Handle Time

Этот показатель позволяет количественно оценить эффективность повседневной работы поддержки.

Важно:

чем меньше среднее время обработки, тем лучше. Стоит периодически контролировать этот показатель и ставить перед собой цель уменьшать его со временем;

выявите факторы, которые увеличивают или сокращают время работы над заявкой. При необходимости внесите изменения в процессы управления заявками.

Лучшие специалисты — Top Agents

Понимание, кто именно работает лучше всех, позволяет сформировать сильный и клиенториентированный отдел поддержки. Оценка продуктивности каждого специалиста в отдельности позволяет создать атмосферу здоровой конкуренции и выявить тех, кому нужна дополнительная мотивация.

Важно:

основываясь на ранее обсуждавшихся метриках, есть несколько способов оценить эффективность каждого сотрудника. Кроме того, можно спросить самих клиентов;

определите, кто из сотрудников идет впереди коллектива (в будущем их можно

рассматривать, как кандидатов на руководящие должности). Возможно, отстающая часть коллектива нуждается в дополнительном обучении или контроле.

Индекс лояльности — Net Promoter Score

Индекс лояльности — это вероятность того, что ваши клиенты порекомендуют компанию другим. Обычно индекс вычисляется при помощи опроса клиентов. Достаточно задать всего два вопроса:

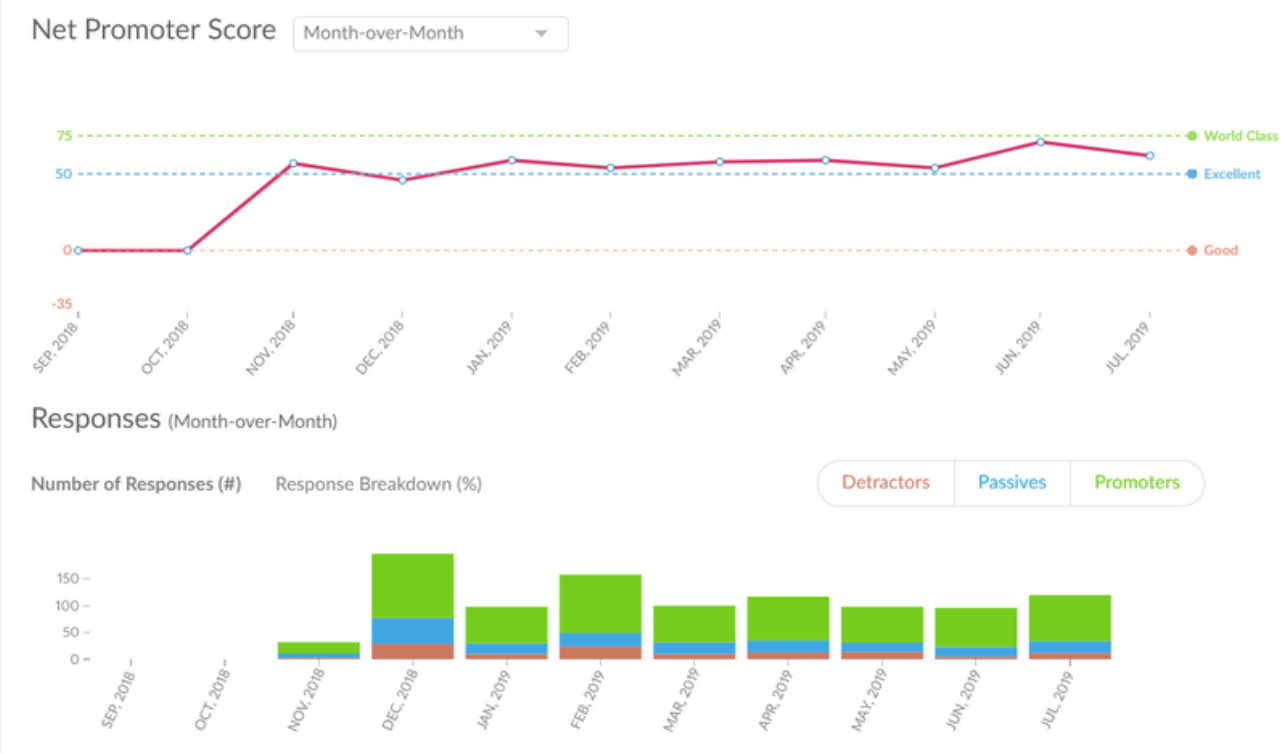
насколько вероятно, что клиент порекомендует компанию окружающим (оценка от 0 до 10);

почему?

Считается, что клиенты, давшие оценку от 0 до 6, упоминая компанию, скорее будут отзываться о ней негативно; с оценкой 7 — 8 будут действовать пассивно, а те, кто дают оценку 9 и 10 могут стать промоутерами вашего бизнеса.

Важно:

- что у этой стандартизованной метрики есть целевые значения для разных отраслей, полученные на основании данных сотен компаний по всему миру



- индекс лояльности — это доля промоутеров за вычетом доли негативно настроенных клиентов (в процентах от общего числа клиентов). Чем выше эта оценка, тем лучше;
- если доля негативно настроенных клиентов выше, вы рискуете столкнуться с оттоком (если клиенты еще не успели уйти к конкурентам). Нужно выяснять, почему так произошло, и искать способы решения.

Оценка удовлетворенности клиентов — Customer Satisfaction Score (CSAT)

Возможно, это самый важный показатель эффективности сервисного бизнеса, поскольку он напрямую связан с ростом дохода. Данный показатель измеряется с помощью опросов и изучения клиентских отзывов на сторонних ресурсах. Чтобы получить более полную картину, старайтесь оценивать клиентский опыт сразу после каждого взаимодействия с ним. Выясните, удовлетворен клиент или нет.

Важно:

чтобы получить уровень удовлетворенности, разделите количество довольных клиентов на общее число опрошенных. Чем выше эта оценка, тем лучше;

если оценка получилась низкой, нужно искать причины. Что не так с вашими услугами? Быть может, клиенту нахамили сотрудники поддержки? Данная метрика является индикатором того, что необходимо проверить процессы и людей.

Естественно, существуют и другие метрики, и ключевые показатели эффективности, которые могут дать представление об уровне обслуживания клиентов. Совершенно не обязательно следить за всеми. Нужно выбрать несколько метрик, которые имеют решающее значение именно для вашего бизнеса (и ваших клиентов) и внимательно следить за их изменениями.

Практическое задание.

1. Установите на вашем ПК пробную версию программы Help Desk.
2. Создайте в программе пользователей, подразделения, опробуйте систему авторизации.
3. Организуйте поступление заявок в вашу систему.
4. Отсортируйте заявки по статусу, по проценту выполнения.
5. Зафиксируйте все виды отчётов, которые имеются в системе HelpDesk.