МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

ФАКУЛЬТЕТ «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

Допущен к защите:

зав. кафедрой «Компьютерные технологии»

д. ф.-м. н., профессор _____ Куралбаев З.К.

«___»_____20 __г.

Магистерская диссертация

Система автоматического обновления программного обеспечения

специальность: 6М070400 - Вычислительная техника и программное обеспечение

Магистрант Рябинин Артём Владиславович

_____ Рябинин А.В.

Научный руководитель,

к. т. н., доцент _____ Сатимова Е.Г.

АЛМАТЫ 2014

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ 2

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 3

ВВЕДЕНИЕ 4

1 ОСОБЕННОСТИ КЛИЕНТ-СЕРВЕРНОЙ АРХИТЕКТУРЫ В ИНФОРМАЦИОННЫХ СИСТЕМАХ 7

1.1 Понятие информационной системы 7

1.2 История развития информационных систем 8

1.2 Процессы информационной системы 9

1.3 Двухуровневая архитектура 11

1.4 Трехуровневая или многоуровневая архитектура 15

1.5 Различные модели технологии «клиент-сервер» 16

1.6 Программное обеспечение 23

Выводы по первому разделу 25

2 ИССЛЕДОВАНИЕ СПОСОБОВ РЕШЕНИЯ ПРОБЛЕМ ОБНОВЛЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 27

2.1 Системы управления версиями 31

2.2 Основные понятия систем управления версиями 32

Выводы по второму разделу 44

3 АНАЛИЗ ИНФОРМАЦИОННОЙ СИСТЕМЫ «РЕГИСТРАЦИОННЫЙ ПУНКТ ДОКУМЕНТИРОВАНИЯ И РЕГИСТРАЦИИ» 45

3.1 Производственная подсистема ввода и передачи данных для оформления заявлений на документирование 46

3.2 Информационная система «Регистрационный пункт документирования и регистрации населения» 47

3.3 Структура ИС «РП ДРН» 49

3.4 Функциональные возможности клиентского программного обеспечения ... 51

Выводы к третьему разделу 54

4 РЕШЕНИЕ ПРОБЛЕМЫ АКТУАЛИЗАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИС «РП ДРН» 55

4.1 Архив версий программного обеспечения 56

4.2 Механизм обновления программного обеспечения 58

Выводы к четвертому разделу 64

ЗАКЛЮЧЕНИЕ 65

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 66

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ИС – информационная система
ОС – операционная система
ПО – программное обеспечение
ЭВМ – электронная вычислительная машина
ПК – персональный компьютер
БД – база данных
СУБД – система управления базами данных
РП – регистрационный пункт
ДРН – документирование и регистрация населения

АҢДАТПА

Диссертациялық жұмыс ақпараттық жүйелердегі бағдарламалық қамтаманы жаңартумен байланысты мәселелерді қарастырады. Қазіргі уақытта әрбір бағдарламалық қамтама уақытылы жаңартылуын талап етеді. Жұмыстың мақсаты – үлестірілген ақпараттық жүйелердің ерекшеліктерін және «ХҚТ ТП» АЖ мысалы негізінде бағдарламалық қамтаманы жаңартумен байланысты мәселелерді зерттеу. Зерттеу нәтижелері негізінде «ХҚТ ТП» АЖ-нде бағдарламалық қамтаманы автоматты түрде жаңарту мәселелерін шешетін жүйе ұсынылды.

АННОТАЦИЯ

Диссертационная работа рассматривает проблемы, связанные с обновлением программного обеспечения в информационных системах. В настоящее время практически каждое программное обеспечение требует своевременного обновления. Цель данной работы – исследование особенностей распределенных информационных систем и проблем связанных с обновлением программного обеспечения на примере ИС «РП ДРН». На основе результатов исследования была предложена система, которая способна решить проблемы автоматического обновления программного обеспечения в ИС «РП ДРН».

ABSTRACT

The dissertation work examines the problems associated with the software update in information systems. Currently, each software requires timely updates. The aim of this work – study features of distributed information systems and problems associated with the software update on the example IS "RP DRP". Based on the results of research proposed a system, which can solve the problem of automatic software updates, IS "RP DRP".

ВВЕДЕНИЕ

Общая характеристика работы. За последние десятилетия программное обеспечение претерпело значительные изменения, пройдя путь от примитивных программ, способных выполнять только простейшие логические и арифметические операции, до сложных систем обработки информации.

Сегодня благодаря растущим темпам развития науки и техники в большинстве компаний и предприятий функционируют различные информационные системы, позволяющие повысить производительность и эффективность работы. Изменение каких-либо бизнес-процессов и постоянное появление новых технологий приводит к тому, что программное обеспечение, являющееся частью информационной системы, постепенно теряет свою актуальность и в конечном итоге уже просто не способно выполнять возложенные на него функции. В результате возникает естественная необходимость в обновлении программного обеспечения во избежание различных сбоев и не состыковок в работе информационной системы. В настоящее время практически любое серьезное программное обеспечение имеет функцию автоматического обновления. Вместо того чтобы пользователю или администратору постоянно сверять текущую версию программы с новой и затем вручную её загружать и устанавливать, программа может сама свериться с новой версией и затем при необходимости обновить саму себя. В достаточно крупных информационных системах автоматизация актуализации программного обеспечения является особенно важным моментом. Поскольку в случае обновления программ «вручную» потребовались бы выделять достаточно большие человеческие ресурсы, что в свою очередь неминуемо привело бы к повышению человеческого фактора, различным задержкам и сбоям в системе.

Актуальность темы. На сегодняшний день в сфере документирования в подавляющем большинстве используются либо локальные системы либо информационные системы с распределенной структурой, которые используют технологию «тонкий клиент». Эта технология лишена недостатков, связанных с актуализацией ПО.

В некоторых информационных системах использование архитектуры «толстый клиент» технологически обусловлено. Это может быть связано с большими объемами информации, со сложностью обработки информации на местах, использованием специализированного ПО и т.д.

Одна из основных особенностей технологии «клиент сервер» заключается в том, что любое программное обеспечение требует обновления непосредственно на рабочих местах. Эта особенность иногда причисляется к разряду недостатков. Поэтому в настоящее время, в связи с наличием данных информационных систем проблема обновления ПО «толстого клиента» весьма актуальна.

Целью настоящей работы является разработка системы автоматического обновления программного обеспечения, которая в настоящий момент отсутствует в ИС «РП ДРН».

Задачи исследования. Важность разработки системы автоматического обновления способной функционировать в рамках ИС «РП ДРН» требует ставить следующие задачи, решение которых является целью настоящей диссертационной работы.

1. Исследование особенностей распределенных информационных систем, методов их построения.
2. Исследование проблем обновления ПО в распределенных системах использующих архитектуру «толстый клиент» на примере ИС «РП ДРН».
3. Реализация возможного решения проблемы обновления ПО.

Научная новизна. В Республике Казахстан функционирует информационная система документирования, построенная на технологии «толстый клиент» и не имеющая аналогов в мире. Она обладает весьма расширенным технологическим функционалом и сопряжена с большим набором технических средств и производственного оборудования. В связи с этим данная информационная имеет проблему связанную с обновлением ПО.

В диссертационной работе предлагается система автоматического обновления, которая на данный момент отсутствует в Казахстане и может решить проблему актуализации ПО в информационной системе документирования.

Структура и объем диссертации. Данная диссертационная работа содержит список обозначений и сокращений, введение, основная часть из пяти разделов, заключения и списка использованных источников. Объем диссертации составляет 64 страницы машинописи, включая 24 рисунка, 2 таблицы.

1 ОСОБЕННОСТИ КЛИЕНТ-СЕРВЕРНОЙ АРХИТЕКТУРЫ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Технология «клиент-сервер» появилась, чтобы сменить существующую на тот момент, централизованную систему, которая была способна управлять вычислительными процессами на ЭВМ в 80-х годах прошлого века.

Благодаря своей высокой отказоустойчивости и надежности вычислительной системы и легкости масштабирования, высокой скорости обработки информации, возможности одновременной работы пользователя с несколькими приложениями, обеспечению пользователя качественным интерфейсом и другим возможностям эта далеко не исчерпавшая себя и весьма перспективная технология получила свое дальнейшее развитие.

Вместо системы баз данных, технология «клиент-сервер» стала основной технологией всемирной сети Internet. В последствии, в результате перенесения идей сети Internet в среду корпоративных систем, появилась технология Intranet. В отличие от технологии «клиент-сервер» эта технология направлена не на данные, а на готовую к использованию информацию. Вычислительные системы, основанные на Intranet, состоят из центральных серверов информации и распределенных компонентов представления информации конечному пользователю (программы-навигаторы, или браузеры). Клиентом и сервер взаимодействуют в Intranet при помощи web – технологий.

В последнее время технология «клиент-сервер» все более и более востребована, но сама по себе она не предполагает решения всех проблем информационных систем. Она позволяет понять, как должна быть организована современная распределенная информационная система. Однако функционирование этой технологии в определённых программных продуктах и в видах программного обеспечения порой значительно отличаются [1].

1.1 Понятие информационной системы

Система - это любой объект, который может рассматривается и как одно целое, но в то же время и как объединенная едиными поставленными целями совокупность различных элементов. Системы сильно отличаются как своим составом, так и основными целями.

Само понятие «система» в информатике очень обширно и многозначно. В основном оно используется, когда речь идёт о технических средствах и программах. Под системой можно обозначить аппаратную часть компьютера. Системой могут быть названы также несколько программ для решения конкретных прикладных задач, совмещённых с операциями ведения документации и проведения расчётов.

Сочетание с понятием «система» слова «информационная» показывает цель ее создания и функционирования. Задачами информационных систем являются сбор, хранение, обработка, поиск и выдача информации, в которой нуждается процесс принятия решений задач в различных областях. Системы позволяют анализировать проблемы и создавать новые продукты.

В последнее время понимание информационной системы подразумевает использование в качестве основного технического средства переработки информации персонального компьютера. Иногда наряду с персональным компьютером в состав технической базы информационной системы входит мэйнфрейм или суперкомпьютер. Но несмотря на это, техническая составляющая информационной системы сама по себе будет не актуальна, если не принимать во внимание роль пользователя, для которого предназначена искомая информация и без которого невозможно ее получение и представление.

Но есть ли отличие между компьютерами и информационными системами? Компьютеры, оснащенные специализированными программными средствами, являются технической базой и инструментом для информационных систем. Для функционирования информационной системы необходим человеческий фактор, взаимодействующий с компьютерами и телекоммуникациями [2].

1.2 История развития информационных систем

Впервые об информационных системах заговорили в 50-х гг. Тогда они в основном использовались для обработки счетов и расчета зарплаты, а функционировали на электромеханических бухгалтерских счетных машинах. Это значительно сократило затраты и время на подготовку бумажных документов.

В следующее десятилетие отношение к информационным системам поменялось. Информация, полученная с их помощью, стала применяться для регулярной отчетности по многим показателям. Для реализации этих задач организациям надо было приобретать компьютерное оборудование широкого назначения, которое позволяло поддерживать разные функции, а не только обрабатывать счета и считать зарплату.

В 70-х - начале 80-х гг. информационные системы используются в качестве средства управленческого контроля, позволяющего ускорять процесс принятия решений.

К началу 90-х гг. концепция использования информационных систем получает новое направление. Системы становятся важнейшим источником информации и широко используются на всех уровнях организации любого профиля. Информационные системы, из которых вовремя извлекалась необходимая информация, помогали своим пользователям оптимизировать свою деятельность, создавать новые товары и услуги, находить новые рынки сбыта, находить себе выгодных партнеров, снижать цены на свою продукцию без потери качества и многое другое [3].

Основные этапы развития информационных систем и цели их использования на разных периодах представлены в таблице 1.1

Т а б л и ц а 1.1 – Изменение подхода к использованию информационных систем

Период времени	Концепция использования информации	Вид информационных систем	Цель использования
1950 – 1960 гг.	Бумажный поток расчетных документов	ИС обработки расчетных документов на электромеханических бухгалтерских машинах	Повышение скорости обработки документов.
1960 – 1970 гг.	Основная помощь в подготовке отчетов	Управленческие ИС для производственной информации	Ускорение процесса подготовки отчетности
1970 – 1980 гг.	Управленческий контроль реализации (продаж)	Системы поддержки принятия решения; системы для высшего звена управления	Выработка наиболее рационального решения
1980 – 2000 гг.	Информация – ресурс, обеспечивающий конкурентное преимущество	Стратегические информационные системы; автоматизированные офисы	Выживание и процветание фирмы

1.2 Процессы информационной системы

Процессы, которые обеспечивают работу любой информационной системы, можно представить в виде схемы изображенной на рисунке 1.1, состоящей из следующих блоков:

- ввод информации из внешних или внутренних источников;
- обработка первичной информации и придание ей оптимальной конфигурации;
- вывод информации непосредственно для потребителя или передачи в другую систему;
- обратная связь - это информация, проанализированная и изменённая персоналом данной организации для коррекции входной информации.

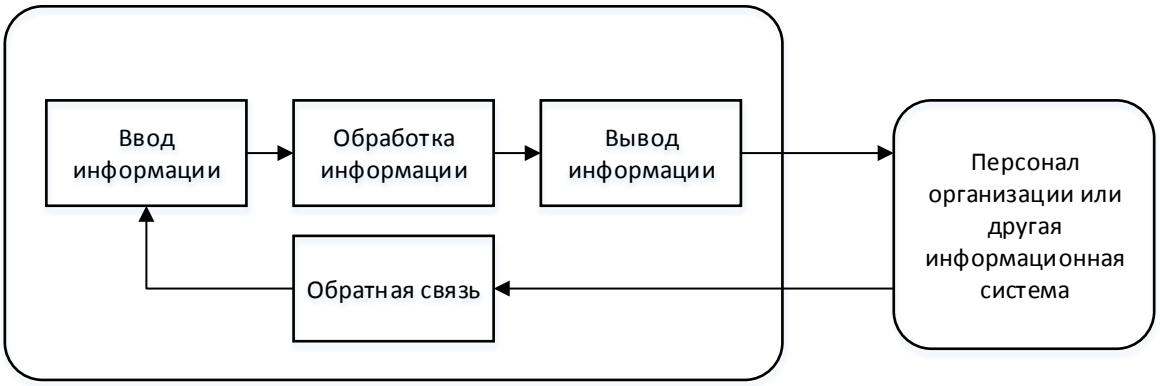

Рисунок 1.1 – Процессы в информационной системе

Информационная система обладает определёнными свойствами:

- любую информационную систему можно проанализировать, она строится и управляется в соответствии с основными принципами построения систем;
- информационная система стремится к прогрессу;
- при построении информационной системы используется системный подход;
- конечным продуктом информационной системы является информация, на которой основываются окончательные решения;
- информационная система прежде всего это совместная работа человека и компьютера, целью которой является обработка информации.

Сейчас распространено мнение об информационной системе как о системе, созданной с помощью компьютерной техники. Но в более широком понимании информационная система может существовать не только в компьютерном варианте [4].

Для того, чтобы разобраться в работе информационной системы, необходимо вникнуть в задачи, которые она решает, а также организационные процессы, в которые она внедрена. Если вы хотите определить возможности компьютерной информационной системы для поддержки принятия решений, то вам необходимо определить:

- структурированность решаемых организационных задач;
- уровень иерархии организации, на котором должно быть принято решение;
- принадлежность искомой задачи к определённой функциональной сфере бизнеса;
- категорию используемой информационной технологии.

Внедрение информационных систем может обеспечить:

- более оптимальные варианты решения административных задач за счет использования математических методов и интеллектуальных систем и т.д.;
- упрощение работы персонала за счет ее автоматизации;
- обеспечение точности информации;

- оптимизацию носителей информации, что обеспечит более выгодную организацию переработки информации на компьютере и сокращение бумажных документов;
- рационализацию потоков информации и системы документооборота в фирме;
- снижение затрат на производство продуктов и услуг;
- предложение потребителям абсолютно новых услуг;
- совершенствование рыночных отношений;
- совершенствование отношений фирмы с партнёрами за счет предоставления им разных скидок и услуг.

Принцип работы в компьютерной информационной системе доступен для понимания неспециалистом в компьютерной области и его вполне можно использовать для контроля процессов управления деятельностью организации.

1.3 Двухуровневая архитектура

Как известно, компоненты сети имеют разные возможности: одни имеют доступ к ресурсам (например, принтер, процессор, система управления базой данных (СУБД), файловая система и т.д.), другие могут обращаться к этим ресурсам.

Технология «клиент-сервер» – это архитектура программного комплекса, в которой прикладная программа рассредоточивается по двум логически различным компонентам, взаимодействующим по принципу «запрос-ответ» и решающим свои определенные задачи, как показано на рисунке 1.2.

Рисунок 1.2 – Архитектура «клиент-сервер»

В схеме двухуровневой архитектуры сервером является компьютер, который управляет и/или владеет определённым ресурсом, а клиентом – компьютер, которому необходим этот ресурс.

Клиент и сервер могут работать как на одном компьютере (ПК), так и на разных ПК в сети. При необходимости определённый программный блок может одновременно выполнять функции сервера по отношению к одному блоку и клиента по отношению к другому. Главный принцип технологии «клиент-сервер» представляет из себя разделение функций приложения на три или более группы.

Первая группа – это логика представления. Она обеспечивает взаимодействие пользователей с приложением. Конкретные характеристики логики представления могут быть различные (интерфейс командной строки, сложные графические пользовательские интерфейсы, интерфейсы через посредника) но задача группы состоит в том, чтобы обнаружить средства для более результативного обмена информацией между пользователем и информационной системой.

Вторая группа называется бизнес-логикой. Она определяет конкретную задачу приложения (например, прикладные функции, характерные для данной предметной области). Разделение приложения по программным направлениям обозначает основу для распределения приложения на нескольких компьютерах.

Третья группа - это логика доступа к данным или алгоритмы доступа к данным. Данные критерии всегда рассматривались как специфический для определённого приложения интерфейс к механизму постоянного хранения данных наподобие файловой системы или СУБД. При помощи модулей обработки данных образуется определённый для приложения интерфейс к СУБД. При помощи интерфейса приложение управляет соединениями с базой данных и запросами к ней (перевод специфических для конкретного приложения запросов на язык SQL, анализ результатов и перемещение их обратно в специфические для данного приложения структуры данных).

Каждая из этих групп может работать независимо от двух других. Не изменяя программ, используемых для хранения и обработки данных, можно изменить интерфейс с пользователем так, что одни и те же данные будут отображаться в виде таблиц, графиков или гистограмму. Простые приложения иногда могут собрать все три части в единственную программу, и данное разделение соответствует функциональным границам [6; 7].

В связи с наличием различных функций в любом приложении присутствуют следующие компоненты:

- компонент представления данных;
- прикладной компонент;
- компонент управления ресурсом.

В классической архитектуре клиент-сервер чаще всего распределяет три основные части приложения по двум физическим модулям. Обычно прикладной компонент располагается на сервере, компонент представления данных – на стороне клиента, а компонент управления ресурсом распределяется между клиентской и серверной частями. Это является основной проблемой классической двухуровневой архитектуры.

В подобной архитектуре при разборе алгоритмов обработки данных разработчики должны знать всё о последних изменениях, внесенных в систему, что осложняет разработку клиент-серверных систем, их установку и сопровождение, потому что нужно координировать действия разных групп специалистов. У разработчиков часто возникают спорные моменты, а это затрудняет развитие системы и иногда приходится изменять уже разработанные элементы.

Чтобы решить проблему разногласия различных элементов архитектуры были созданы две модификации двухзвенной архитектуры «клиент-сервер»: «толстый клиент» и «тонкий клиент». В данных архитектурах разработчики попытались выполнять обработку данных на одной из двух физических частей – либо на стороне клиента, либо на сервере.

Каждый вариант не идеален. В первом необоснованно перегружается сеть, потому что по ней передаются необработанные, то есть, избыточные данные. Также затрудняется поддержка системы и ее изменение, так как замена алгоритма вычислений или исправление ошибки требует синхронной замены всех интерфейсных программ, а иначе могут возникнуть ошибки или несогласованность данных. Если обработку информации осуществляет сервер, то возникает проблема описания встроенных процедур и их отладки. Систему с обработкой информации на сервере практически невозможно перенести на другую платформу, что является существенной проблемой.

«Толстый клиент» в архитектуре «клиент-сервер» — это компьютер или приложение, обеспечивающее, в противовес тонкому клиенту, расширенную возможность независимо от центрального сервера. Практически всегда на сервере только хранятся данные, а вся работа по обработке и представлению этих данных осуществляется машиной клиента. Архитектура «толстый клиент» изображена на рисунке 1.3.

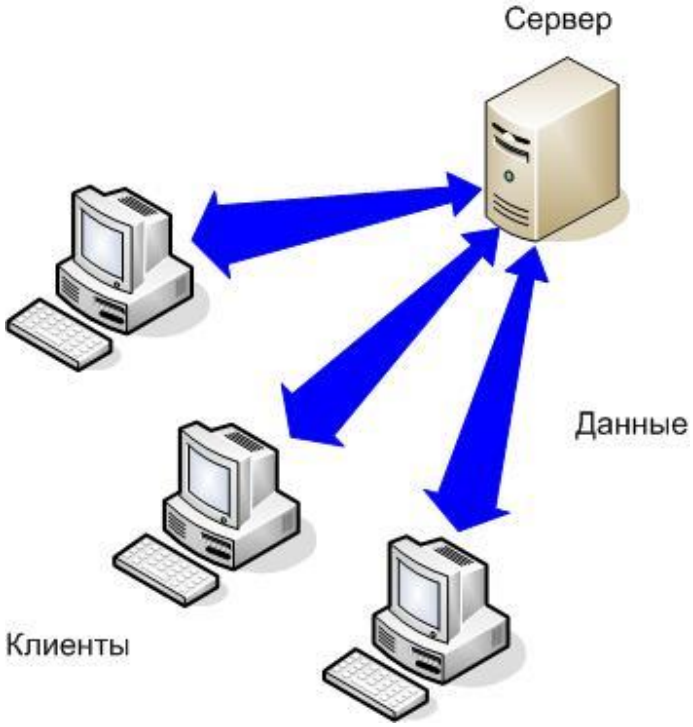

Рисунок 1.3 – Архитектура «Толстый клиент»

Сильные стороны архитектуры «толстый клиент»:

- толстый клиент обладает широким диапазоном в отличие от тонкого клиента;
- режим многопользовательской работы;
- работает даже при нарушениях связи с сервером;
- высокая скорость действия
- Возможные проблемы архитектуры «толстый клиент»:
- большой размер дистрибутива;
- многое в работе клиента зависит от того, для какой платформы он разрабатывался;
- при работе с ним бывают проблемы с удаленным доступом к данным;
- сложности процесса установки и настройки;
- сложность обновления и зависящая от неё неактуальность данных.

«Тонкий клиент» в архитектуре «клиент-сервер» — компьютер или приложение, способное переносить всю или большую часть задач по обработке информации на сервер. Примером тонкого клиента может служить компьютер с браузером, использующийся для работы с веб-приложениями, как изображено на рисунке 1.4.

Рисунок 1.4 – Архитектура «Толстый клиент»

Архитектура «тонкий клиент» не имеет недостатков, которые есть у архитектуры «толстый клиент», но, тем не менее, иногда встречаются сложности:

- усложняется реализация, так как языки типа SQL не приспособлены для разработки подобного программного обеспечения;
- производительность таких программ, значительно ниже, чем написанных на других языках, что актуально для сложных систем;

- программы, в которых использованы СУБД-языки, часто работают не вполне чётко;
- ошибка в них может привести к выходу из строя всего сервера баз данных;
- подобные программы абсолютно непереносимы на другие системы и платформы.

Для исключения недостатков двухуровневой архитектуры используются многоуровневые (три и более уровней) архитектуры «клиент-сервер» [6; 7].

1.4 Трехуровневая или многоуровневая архитектура

Примерно в середине 90-х годов двадцатого века появилась трехзвенная архитектура «клиент-сервер», которая разделила информационную систему по функциональным возможностям на три разных компонента: логика представления, бизнес-логика и логика доступа к данным. В отличие от двухзвенной архитектуры в трехзвенной имеется ещё одно звено – сервер приложений, задача которого - осуществление бизнес-логики, при этом полностью освобождается клиент, который запрашивает промежуточное программное обеспечение, и максимально использует все возможности серверов.

В трехуровневой архитектуре клиент обычно не перегружается обработками данных, а делает свою основную работу представления информации, поступающей с сервера приложений. Данный интерфейс реализуется с помощью стандартных средств Web-технологии, а то есть браузера, CGI и Java. Это сокращает объем данных, перемещаемых между клиентом и сервером приложений, в результате чего возможно подключать клиентские компьютеры даже по медленным линиям типа телефонных каналов. Важный момент: клиентская часть может быть элементарной, поэтому часто ее осуществляют с помощью универсального браузера. Но если менять ее все-таки придется, то это производится очень просто.

Сервер приложений – это программное обеспечение, являющееся прослойкой между клиентом и сервером, как показано на рисунке 1.5.

Рисунок 1.5 - Сервер приложений

Использование сервера приложений дает больше возможностей, например, уменьшается нагрузка на клиентские компьютеры, потому что сервер приложений распределяет нагрузку и обеспечивает защиту от сбоев. Так как бизнес-логика хранится на сервере приложений, то при каких-либо изменениях в отчетности или расчетах клиентские программы никоим образом не затрагиваются [8].

Существует несколько серверов приложений от известных компаний: IBM, Oracle и у каждого имеется набор предоставляемых сервисов. Эти сервисы делают программирование комфортным и упрощают развертывание приложений масштаба предприятия.

Многоуровневые клиент-серверные системы просто перевести на Web-технологию – надо заменить клиентскую часть универсальным или специализированным браузером, а сервер приложений дополнить Web-сервером и небольшими программами вызова процедур сервера. Для разработки этих программ можно использовать как Common Gateway Interface (CGI), так и более современную технологию Java.

По сравнению с двухуровневой или файл-серверной архитектурой трёхуровневая архитектура предпочтительнее, потому что имеет:

- масштабируемость;
- конфигурируемость;
- разобщённость уровней друг относительно;
- высокая безопасность;
- высокая надёжность;
- низкие требования к скорости канала (сети) между терминалами и сервером приложений;
- низкие требования к производительности и техническим характеристикам терминалов, как результат снижения их стоимости.

Но проблемы всё же имеются. По сравнению с двухуровневой или файл-серверной архитектурой обнаруживаются следующие недостатки трёхуровневой архитектуры:

- более высокая сложность создания приложений;
- сложнее в разворачивании и администрировании;
- высокие требования к производительности серверов приложений и сервера базы данных, что ведёт за собой повышение стоимости серверного оборудования;
- высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений.

Но допуская наличие недостатков, трехуровневая архитектура однозначно превосходит двухуровневую и сейчас очень популярна.

1.5 Различные модели технологии «клиент-сервер»

Первой базовой технологией для локальных сетей была модель файлового сервера. Эта технология была широко распространена среди

отечественных разработчиков, использовавших такие системы, как FoxPro, Clipper, Clarion, Paradox и так далее.

Модель файлового сервера является наиболее удобной и характеризует не столько способ возникновения фактографической информационной системы, сколько способ общения компьютеров в локальной сети. Один из компьютеров сети выделяется и опознаётся файловым сервером, т. е. общим хранилищем любой информации [9].

В модели файлового сервера все главные составляющие размещаются на клиентской установке. При обращении к данным ядро СУБД запрашивает ввод-вывод данных у сервиса файловой системы. С помощью функций операционной системы в оперативную память клиентской установки полностью или частично на время сеанса работы копируется файл базы данных. Следовательно, сервер выполняет только пассивную функцию. Модель файлового сервера показана на схеме, приведенной на рисунке 1.5.

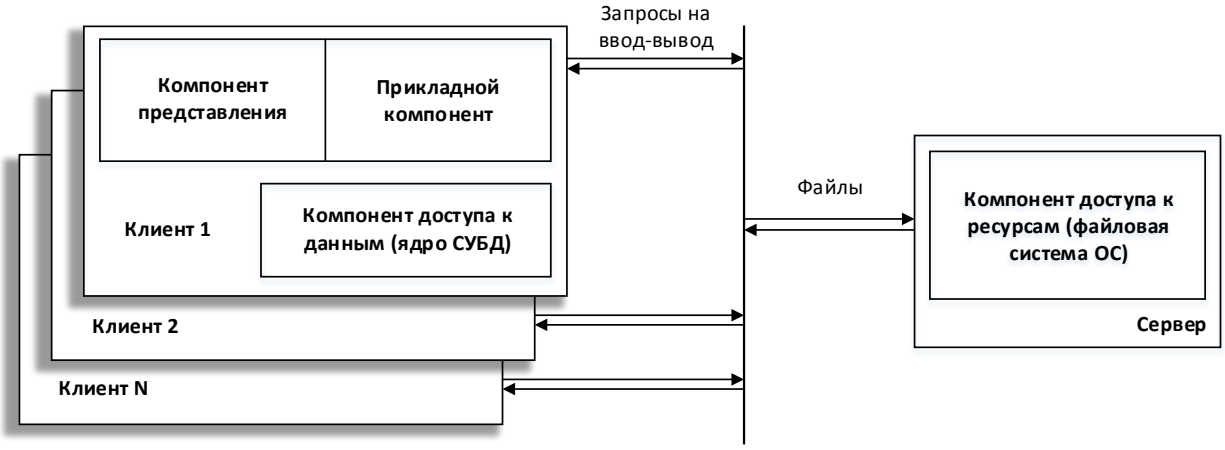

Рисунок 1.5 – Модель файлового сервера

Эта модель очень проста, в ней отсутствуют высокие требования к производительности сервера. Так же важно то, что программные компоненты СУБД в данном случае не распределены, т. е. никакая часть СУБД на сервере не инсталлируется и не размещается.

Несомненно, модель имеет и недостатки. Самый очевидный - это высокий сетевой трафик, имеющий максимальное значение в момент массового вхождения в систему пользователей. Более значимый недостаток с точки зрения работы с общей базой данных это отсутствие специальных механизмов безопасности файла (файлов) базы данных со стороны СУБД. Другими словами, распределение данных между пользователями (параллельная работа с одним файлом данных) осуществляется только средствами файловой системы ОС для синхронной работы нескольких прикладных программ с одним файлом.

Со временем недостатки, присущие технологии «файл-сервер», были ликвидированы и используемая в современных СУБД архитектура «клиент-сервер» стала практически стандартом. Если предполагается, что проектируемая сетевая технология будет иметь архитектуру «клиент-сервер»,

то это значит, что прикладные программы, реализованные в ее рамках, будут иметь распределенный характер, то есть какая-то часть функций приложений будет реализована в программе-клиенте, а другая - в программе-сервере.

Отличия в функционировании приложений в рамках технологии «Клиент-сервер» отвечают на четыре вопроса:

- какие виды программного обеспечения в логических компонентах;
- какие механизмы программного обеспечения используются для реализации функций логических компонентов;
- как логические компоненты распределяются компьютерами в сети;
- какие способы используются для связи компонентов друг с другом.

Таким образом, существуют три подхода, каждый из которых реализован в определённой модели технологии «Клиент – сервер»:

- модель доступа к удаленным данным (Remote Date Access - RDA);
- модель сервера базы данных (DateBase Server - DBS);
- модель сервера приложений (Application Server - AS).

Рассмотрим функции и характеристики различных моделей технологии «Клиент-сервер» [9].

Модель доступа к удаленным данным (RDA) – сетевая архитектура технологии «Клиент – сервер», при которой коды компонента представления и прикладного компонента совмещены и выполняются на компьютере-клиенте. Доступ к информационным ресурсам осуществляется при помощи непроцедурного языка или вызовами функций специальной библиотеки, если есть определённый интерфейс прикладного программирования - API.

Запросы к информационным ресурсам направляются по сети к удаленному компьютеру, который анализирует их, возвращая клиенту блоки данных, как на рисунке 1.6.

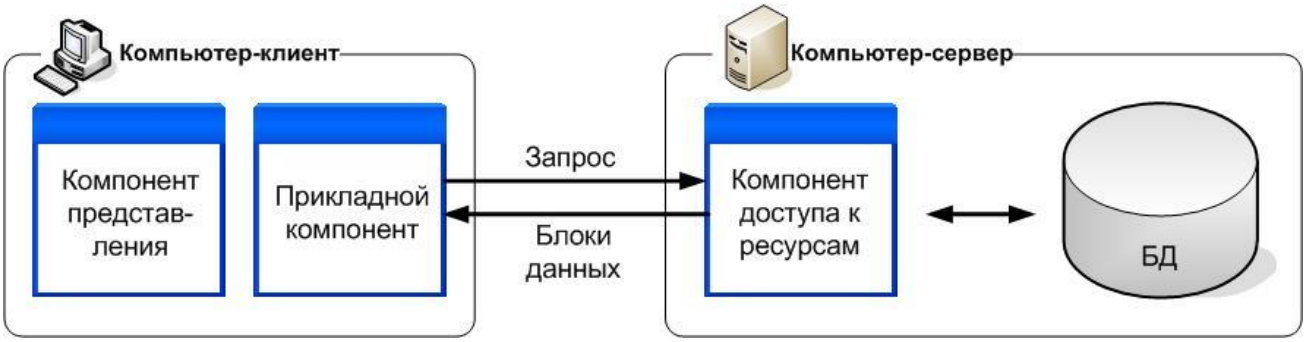

Рисунок 1.6 – Модель доступа к удаленным данным

Главным преимуществом RDA-модели является широкий выбор инструментальных средств разработки приложений, позволяющих быстро создать приложения, работающие с SQL-ориентированными СУБД. Обычно инструментальные средства поддерживают графический интерфейс пользователя с ОС, а также средства автоматической генерации кода, в которых

смешаны прикладные функции и функции представления. При этом RDA-модель имеет ряд ограничений.

Во-первых, взаимодействие клиента и сервера посредством SQL-запросов значительно загружает сеть. Приложение является нераспределенным, и вся его логика расположена на компьютере-клиенте, поэтому взаимодействие его с сервером посредством SQL-запросов приводит к передаче по сети данных большого объема, возможно, избыточных. Если увеличивается число пользователей, сеть становится проблемным местом, снижая быстродействие всей информационной системы.

Во-вторых, адекватное администрирование приложений в RDA-модели практически невозможно. Если отличающиеся по своей природе функции (функции представления и чисто прикладные функции) совмещены в одной и той же программе, написанной на языке четвертого поколения, то если нужно изменить прикладные функции, приходится переделывать всю программу целиком.

В-третьих, при коллективной работе над проектом, каждый разработчик отвечает за отдельные прикладные функции, что значительно затрудняет контроль за их взаимной непротиворечивостью. Каждому из авторов приходится программировать интерфейс с пользователем, что может нарушить единый стиль интерфейса и его целостность. Сложность обновления программного обеспечения возникает еще и потому, что замену ПО необходимо производить одновременно на всех компьютерах-клиентах.

В-четвертых, из-за невозможности реализации разграничения доступа по функциям на стороне сервера, а только на стороне клиента, возникают проблемы с безопасностью. При этом разграничение производится только по таблицам базы данных, что снижает защищенность.

Несмотря на широкое распространение, RDA-модель вытесняется более технологичной DBS-моделью.

Модель сервера баз данных (DBS) - сетевая архитектура технологии «Клиент – сервер», основу которой составляет механизм хранимых процедур, реализующий прикладные функции. В DBS – модели понятие информационного ресурса сведено до базы данных из-за того же механизма хранимых процедур, который осуществлён в СУБД, даже не во всех.

В DBS-модели приложение является распределенным. Компонент представления выполняется на компьютере-клиенте, в то время как прикладной компонент (реализующий бизнес-функции) представлен как набор хранимых процедур и функционирует на компьютере-сервере БД [7]. Хранимые процедуры также называют компилируемыми резидентными процедурами или процедурами базы данных, как на рисунке 1.7.

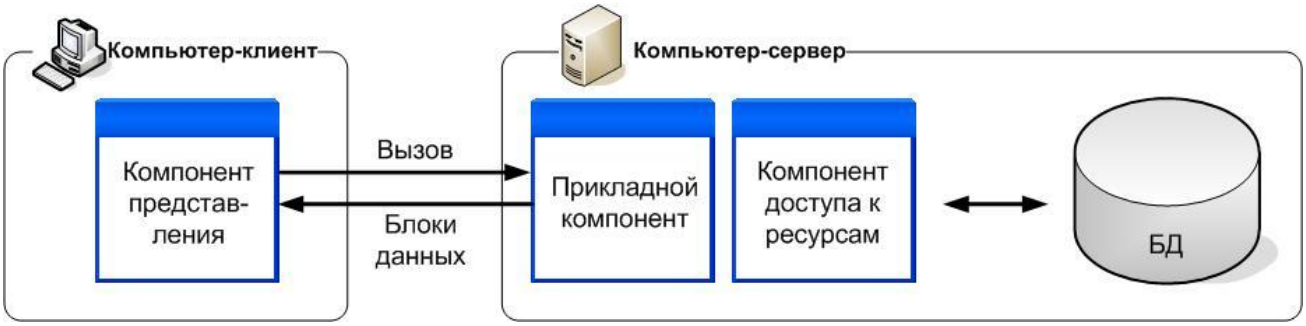

Рисунок 1.7 – Модель сервера базы данных

Преимущества DBS-модели перед RDA-моделью очевидны: это и возможность централизованного администрирования различных функций, и снижение трафика сети из-за того, что вместо SQL-запросов по сети передаются вызовы хранимых процедур, и возможность распределения процедуры между несколькими приложениями, и экономия ресурсов компьютера за счет использования первоначального плана выполнения процедуры. Но есть недостатки и у этой модели.

Во-первых, разнообразные процедурные расширения SQL, используемые для написания хранимых процедур, не являются языками программирования в полном смысле слова. Они встроены в конкретные СУБД и имеют ограниченные возможности. Таким образом, система, в которой прикладной компонент реализован при помощи хранимых процедур, не является мобильной относительно СУБД. В большинстве СУБД невозможно отладить и тестировать хранимые процедуры, что приводит не просто к сбою, а к полной неработоспособности всей базы данных.

Во-вторых, в DBS-модели не предусмотрены различные варианты взаимодействия клиента и сервера, необходимые для децентрализации приложений, например хранимые очереди, асинхронные вызовы и другие.

В-третьих, DBS-модель не обеспечивает необходимой эффективности использования вычислительных ресурсов. Ограничения в ядре СУБД не позволяют обеспечивать эффективный баланс загрузки, миграцию процедур на другие компьютеры-серверы БД и осуществлять многое другое, например запросы с приоритетом.

Чаще всего используется частичный синтез RDA- и DBS-моделей для построения многопользовательских информационных систем: поддержка целостности базы данных и некоторых простейших прикладных функций хранимых процедур (DBS-модель), а более сложные задачи осуществляются непосредственно в прикладной программе, которая выполняется на компьютере-клиенте (RDA-модель).

Все недостатки DBS - модели учтены в AS-модели, которая в отражает преимущества технологии «клиент-сервер».

Модель сервера приложений (AS) - сетевая архитектура технологии «Клиент – сервер», представляющая собой процесс, выполняемый на компьютере-клиенте и отвечающий за интерфейс с пользователем (ввод и

отображение данных). Главным элементом этой модели является прикладной компонент, называющийся сервером приложения, работающий на удаленном компьютере (или нескольких компьютерах). Сервер приложений реализован как группа прикладных функций, представленных в виде сервисов (служб). Каждый сервис предоставляет некоторые услуги всем программам, которые могут ими воспользоваться, как изображено на рисунке 1.8.

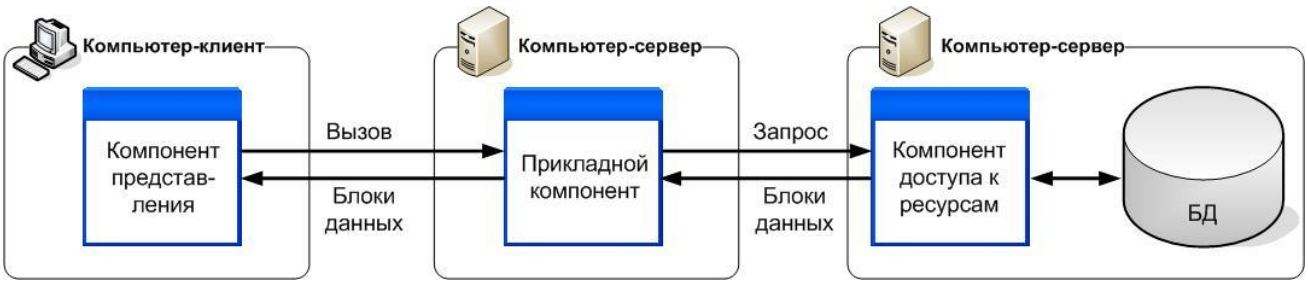

Рисунок 1.8 – Модель сервера приложений

Серверов приложений может быть несколько, и каждый их них предоставляет определенные услуги. Любая программа, которая пользуется ими, рассматривается как клиент приложения.

Подробности реализации прикладных функций в сервере приложений полностью недоступны для клиента приложения. Клиент, которым может быть стандартный браузер, обращается с запросом к определённой службе, при этом серверы приложений обезличены и служат для создания графического интерфейса с пользователем, что приводит к эффективному управлению балансом загрузки. Запросы поступают от клиента по очереди к AS-процессу, который извлекает и передает их для обработки службе в соответствии с приоритетами.

Так как данные «спрятаны» за сервером приложений, в котором обычно встроена проверка полномочий клиента, в СУБД обеспечивается защита данных.

Доступ к информационным ресурсам обеспечивается как и в RDA-модели менеджером ресурсов (например, SQL-сервер). Из прикладных компонентов доступны такие ресурсы как, базы данных, очереди, почтовые службы и другие. Серверы приложений выполняются, как правило, на том же компьютере, где функционирует менеджер ресурсов, что избавляет от необходимости направления SQL-запросов по сети и повышает производительность системы, Также серверы приложений могут выполняться и на других компьютерах.

AS-модель является универсальной системой, в которой может быть сколько угодно уровней, сообщающихся между собой. Четкое разделение логических компонентов, возможность баланса загрузки между несколькими серверами, и оптимальный выбор программных средств для их осуществления обеспечивают модели такой уровень гибкости, защиты данных и открытости, который пока недостижим в RDA- и DBS-моделях. В AS-модели возможна работа по медленным линиям связи, что определённо снижает трафик между

клиентом и сервером приложений. Следовательно, именно AS-модель является основой для мониторов обработки транзакций.

Проанализировав все модели технологии «Клиент – сервер», можно сказать, что RDA- и DBS-модели имеют в основе двухзвенную схему разделения функций. В RDA-модели прикладные функции отданы клиенту, в DBS-модели их реализация осуществляется через ядро СУБД. В RDA-модели прикладной компонент совмещается с компонентом представления, в DBS-модели интегрируется в компонент доступа к ресурсам.

В AS-модели осуществлена трехзвенная схема разделения функций, где прикладной компонент определён как важнейший изолированный элемент приложения, имеющий стандартизированные интерфейсы с двумя другими компонентами.

Результаты анализа моделей технологий «файловый сервер» и «Клиент-сервер» представлены в таблице 1.2.

Таблица 1.2 – Результаты анализа моделей технологий «файловый сервер» и «Клиент- сервер»

Критерии	«Файловый сервер»	«Клиент – сервер»		
	FS - модель	RDA-модель	DBS-модель	AS-модель
Сложность разработки приложений	Низкая	Низкая	Высокая	Высокая
Сложность администрирования	Низкая	Высокая	Высокая	Высокая
Степень защиты данных	Высокая	Низкая	Высокая	Высокая
Требования к характеристикам сервера	Высокие	Низкие	Высокие	Высокие
Трафик, создаваемый в сети	Низкий	Очень высокий	Низкий	Низкий
Сложность обновления ПО	Низкая	Высокая	Низкая	Низкая
Требования к характеристикам сети	Низкие	Очень высокие	Низкие	Низкие
Распределение загрузки	Нет	Есть	Есть	Есть
Требования к характеристикам рабочих станций	-	Очень высокие	Низкие	Низкие

Окончание таблицы 1.2

Использование графического интерфейса	-	+	+	+
Использование символьного интерфейса	+	+	+	+

Несмотря на свое названия технология «клиент - сервер» также является системой распределенных вычислений. В этом случае распределенные вычисления рассматриваются как архитектура «клиент - сервер» с вовлечением нескольких серверов. Термин «сервер» означает просто программу, отвечающую на запросы и выполняющую определённые действия по запросу клиента. Из-за того, что распределенные вычисления – это один из видов систем «клиент - сервер», пользователи получают такие же преимущества, например, увеличение общей пропускной способности и возможность многозадачной работы. Кроме того, интеграция дискретных сетевых компонентов и обеспечение их функционирования как единого целого способствует увеличению эффективности и снижению расходов.

Следовательно, если предстоит работа с небольшими информационными системами, не требующими графического интерфейса с пользователем, можно выбрать FS - модель. Проблему графического интерфейса легко решает RDA-модель. DBS-модель является предпочтительным вариантом для СУБД. AS-модель является лучшим вариантом для создания больших информационных систем, а также в случае использования низкоскоростных каналов связи.

При разработке любой информационной системы можно использовать модели как в чистом виде, так и их объединение. Для этого необходимо правильно выбрать не только модель, но и платформу, на которой она будет реализована, так как ошибки, допущенные на этом этапе, могут уничтожить все преимущества прикладной части информационной системы. Причем большая часть недостатков, вызванных этими ошибками, выявляется, как правило, только на этапе эксплуатации.

1.6 Программное обеспечение

В традиционной архитектуре «клиент-сервер» серверы должны иметь большую вычислительную мощность, чем у клиентов, потому как на серверы предназначены для более серьезных задач, таких как администрирование, обеспечение защиты информации, выполнение множества одновременных запросов и так далее.

Успешного применение технологии «клиент-сервер» невозможно без использования соответствующего программного обеспечения, состоящего из клиентской и серверной части.

Установленное на сервере программное обеспечение необходимое для управления базами данных, отвечает на запросы клиентов и начинает поиск

информации. Как часть системы «клиент-сервер» оно возвращает только результаты поиска.

В состав обработки данных на сервере входят такие операции как сортировка, нахождение и извлечение затребованной информации, и последующая её отправка обратно пользователю.

Серверное программное обеспечение так же предусматривает и другие операции над информацией: добавление, удаление, обновление, её защита и так далее.

Помимо всего прочего на сервере размещены так называемые хранимые процедуры (предварительно написанные процедуры обработки данных), которые могут быть использованы любым клиентом и помогают обрабатывать данные, уменьшая размер исполняемого кода и используемого дискового пространства на компьютерах-клиентах. Хранимая процедура может быть вызвана любым количеством клиентов. Нет никакой необходимости включать её в код каждой программы. Помимо частичной обработки данных, хранимые процедуры также уменьшают сетевой трафик. Это обусловлено тем, что единичное обращение клиента приводит к выполнению серии команд хранимой процедуры. Каждая команда хранимой процедуры требует отдельного запроса к базе данных, и может проводить контроль безопасности, чтобы предотвратить несанкционированный запуск клиентами некоторых процедур.

Возможности технологии «клиент-сервер» дополняются инструментальными средствами, приложениями и утилитами для интерфейсной части. К ним относятся:

- средства запросов, упрощающие доступ к данным сервера, используя написанные запросы и встроенные возможности для создания отчетов;
- различные пользовательские приложения, которые могут работать в качестве интерфейсной части и обеспечивают доступ к серверу базы данных.

Прочие приложения имеют у себя встроенный SQL, который позволяет взаимодействовать с системами управления базами данных от всевозможных производителей. Важным моментом в реализациях систем «клиент-сервер» заключается в необходимости разработки специализированых интерфейсных частей. Эта задача значительно упрощается благодаря использованию программистами программистам и администраторам информационных систем, отвечающими за доступ к серверам базы данных средств разработки программного обеспечения.

Необходимое программное обеспечение определяется в зависимости от типа операционной системы, а так же поставленных задач. Например, если используется ОС семейства Windows, то, скорее всего, на компьютере-клиенте используется программный пакет Microsoft Office, в состав которого входят текстовый процессор Microsoft Word, табличный процессор Microsoft Excel, система подготовки презентаций Microsoft PowerPoint, система управления базами данных Microsoft Access и многое другое. Хотя не исключено использование других офисных пакетов, таких как OpenOffice или LibreOffice, которые вдобавок ко всему прочему еще и бесплатны.

В качестве сетевой операционной системы используют Windows Server, которая способна обеспечить совместное использование файлов, различных печатающих устройств, множества процессов и ресурсов многими пользователями. ОС также позволяет предоставить услуги по соединению с рабочими станциями (клиентскими компьютерами) для реализации дополнительных возможностей.

Remote Access Service (RAS) – набор служб удаленного доступа относятся к любой комбинации аппаратного и программного обеспечения, чтобы включить инструменты удаленного доступа или информацию, которые обычно находятся в сети устройств IT предприятия. Сервер RAS – специализированный компьютер, который объединяет каналы передачи. Провайдеры доступа часто используют серверы RAS, чтобы ограничить физические соединения с клиентами, которые имеют к интернету через модем.

В качестве системы управления реляционными базами данных зачастую используют Microsoft SQL Server. Microsoft SQL Server построена на технологии «клиент-сервер», поддерживает систему обработки транзакций, механизм распределенных транзакций, систему сохранения ссылочной целостности и тиражирование данных.

Microsoft Systems Network Architecture (SNA) Server представляет собой комплект интегрированных служб, который обеспечивает доступ к данным на сервере и ресурсам приложения от клиентской рабочей станции. Сервер SNA обеспечивает необходимую связь с сервером и интегрирует настольную среду с системой универсального компьютера типа IBM, для поддержания работы приложения, требующих оперативной обработки транзакций и веб-доступ.

Microsoft Exchange Server позволяет организовать доступ к корпоративной электронной почте, календарю и контактам через браузер, телефон и ПК. Exchange Server работает на Windows Server и является частью линейки продуктов. Microsoft Servers так же построен на принципах технологии «клиент-сервер» и позволяет масштабироваться в соответствии с возрастанием вычислительных возможностей сети.

Microsoft Internet Information Services (IIS) обеспечивают безопасную, легкую в управлении, модульную и расширяемую платформу для того, чтобы надежно размещать вебсайты, службы и приложения. IIS позволяет настраивать, и добавлять новые опции, такие как маршрутизация запроса приложения и выравнивание нагрузки посредством бесплатных расширений IIS. Этот программный продукт так же позволяет разворачивать и выполнять веб-приложения ASP.NET и PHP и обеспечивает возможность создания Web-, FTP- и Gopher-серверов для сети Internet.

Выводы по первому разделу

В заключение можно сделать следующие выводы. В настоящее время существует множество различных архитектур информационных систем. Наиболее популярная это клиент-серверная архитектура, которая сейчас используется во многих организациях и предприятиях.

Клиент-серверная архитектура, в настоящее время, активно развивается. В результате чего появилось множество различных типов этой архитектуры. Каждый тип имеет свои преимущества и недостатки, в том числе, связанные с проблемами обновления программного обеспечения, которые были описаны выше.

2 ИССЛЕДОВАНИЕ СПОСОБОВ РЕШЕНИЯ ПРОБЛЕМ ОБНОВЛЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В современном мире компьютерные технологии приобретают все более важное значение. Персональные компьютеры наполнили все сферы жизни человека. Вычислительная техника широко применяется как в области начального образования, так и в исследованиях новейших технологий. Невозможно представить какое-либо современное предприятие без серьезного серверного оборудования и информационных систем. Программные комплексы обеспечивают работу сложнейших бизнес-процессов.

В условиях бурного развития информационных технологий программное обеспечение претерпевает постоянные модификации, с целью исправления каких-либо ошибок или совершенствования функциональных возможностей и функций. Каждая модификация программного продукта или компонента может сопровождаться необходимостью добавления новых файлов, а также замены имеющихся файлов файлами более новых версий. После того, как разработчик выявил и устранил ошибку в своем программном продукте и создал реализацию решения данной проблемы, требуется включить эту модификацию в соответствующее обновление, и предоставить новую реализацию продукта его пользователям. Разработчик весьма заинтересован в получении произведенных обновлений программного обеспечения всеми пользователями в кратчайшие сроки и с максимальной гарантией.

Интернет предоставляет пользователям замечательную возможность для получения различных обновлений программных продуктов. Непрерывное развитие Интернета, а также повышение скорости передачи данных по каналам связи создает предпосылки для предоставления пользователям различного программного обеспечения и его обновлении в онлайн режиме, с осуществлением загрузок и инсталляций по сети. Разработчики программных средств также весьма заинтересованы в использовании Интернета для распространения различного рода обновлений, так как это ощутимо снижает затраты на доставку. В свою очередь Интернет предоставляет пользователям возможность получать модификации для устранения ошибок сразу после того, как они были произведены разработчиком и опубликованы в сети для загрузки. Сайты производителей программного обеспечения в сети Интернет могут быть организованы таким образом, чтобы сделать достаточно простым определение местоположения и обнаружение файлов обновлений. Технические аспекты загрузки файлов по большей части исчезли из поля зрения пользователя программного обеспечения, и теперь чаще всего обрабатываются операционной системой.

При традиционном подходе производитель программного продукта создает обновление программного обеспечения в виде специального так называемого «пакета» для загрузки. Данный пакет является саморазворачивающимся исполняемым файлом, сжатым с целью уменьшения размера. Как правило, внутри пакета содержится программа установка и все

обновленные файлы программного продукта. Размер пакета определяется суммой размеров каждого сжатого измененного файла и непосредственно самой программы извлечения.

При выполнении обновления пакет сначала извлекает все содержащиеся в нем файлы во временное местоположение на диске компьютера, затем запускает установочную программу. Установочная программа размещает каждый обновленный файл в надлежащее местоположение в системном каталоге. В виду того, что обновленные файлы поставляются в сжатом виде, по мере их установки производится необходимая декомпрессия. Все имеющиеся файлы с одним и тем же наименованием, в одном и том же каталоге в случае необходимости будут заменены обновленными файлами.

Не смотря на то, что Интернет предоставляет широкие возможности для удобного и быстрого распространения программных обновлений, ограничения скорости передачи данных по сети вызывают определенные сложности. Размеры приложений современного программного обеспечения привели к тому, что объемы загружаемых обновлений стали необоснованно большими. Чаще всего разработчики группируют целый ряд модификаций для расширения функционала и устранения различных ошибок программного продукта в одно обновление. Если производитель обновляет свои программные продукты на систематической основе, то объемы загружаемых пакетов обновлений будут постоянно увеличиваться. Это связано с тем, что разработчики не могут не включать в состав обновлений измененные ранее файлы, исходя из предположения, что пользователь уже имеет файлы с необходимыми модификациями из более ранних обновлений. Вследствие того, что пакет обновления может содержать не только последние измененные файлы, но и некоторое количество файлов измененных в более ранних обновлениях, его размер может оказаться весьма значительным, даже при условии сжатия содержимого. Поэтому в некоторых случаях, даже при использовании скоростных соединений эффективность загрузки при обновлениях заметно снижается.

Большие затраты времени при загрузке обновлений конечно весьма нежелательны. Особенно в тех случаях, когда пользователи имеют очень дорогой и низкоскоростной Интернет или ограниченное время соединения для загрузки файлов. Поэтому сокращение размеров файла обновления, а как следствие времени его загрузки будет сокращать финансовые затраты пользователей. Производители программного обеспечения как правило также имеют некоторые (иногда весьма ощутимые) затраты, связанные с размерами обновлений, которые они поставляют. Поэтому сокращение размеров обновлений также может позволить производителям сократить свои денежные затраты. Уменьшение размеров файлов обновлений позволит производителям программного обеспечения значительно повысить пропускную способность доступной им сети, позволяя обслуживать соединения с большим количеством пользователей на имеющемся оборудовании сетевого сервера.

Длительное время, которое необходимо для загрузки большого обновления, также делает процесс обновления более зависимым от различного рода проблем с сетевым соединением. Имеется также ряд причин, по которым сеанс связи с Интернет может быть разъединен преждевременно. Это могут быть различные помехи в линиях связи, разрывы соединения и ошибочные действия пользователя. Иногда провайдеры Интернет используют ограничение времени соединения, в течение которого пользователь может оставаться подсоединенным к сети в одном сеансе. Если во время загрузки большого файла обновления произойдет разрыв соединения, то в большинстве случаев пользователю придется осуществить повторный запуск сеанса обновления. Это связано с тем, что большинство обычных операционных систем и протоколов передачи данных не позволяют производить дозагрузку файла, поэтому промежуток времени будет потерян, и загрузку нужно будет запустить повторно. В некоторых ситуациях вероятность отказа столь велика, что для многих пользователей получение обновления в оперативном режиме становится почти невозможным. Если размер пакета обновления окажется слишком большим, то редкие пользователи смогут загрузить его полностью.

Одна из попыток сократить размер обновлений программного обеспечения и повысить эффективность использования относительно медленных каналов связи относится к применению небольших корректирующих изменений. Это использование так называемых дельта-корректировок или двоичных корректировок. Специалисты в данной области информационных технологий знают, что понятие дельта-корректировки соответствует специальному программному коду, который может изменять имеющийся файл непосредственно на компьютере пользователя. Поскольку дельта-корректировка включает в себя специальный программный код, то необходима уникальная дельта-корректировка для каждой уникальной версии файла. В применении к процедуре обновлений программного обеспечения служба обновления может передавать пользователю дельта-корректировку значительно меньшего размера вместо передачи полного обновленного файла. Затем обновленная дельта-корректировка используется для того, чтобы модифицировать существующий файл в обновленный файл.

Несмотря на то, что применение дельта-корректировок может потенциально сократить объемы передаваемых данных, необходимых для осуществления процедуры обновления файлов, современные подходы к использованию дельта-корректировок являются несовершенными. Это особенно становится заметно в процедуре управления выбором применимых дельта-файлов в ситуациях, когда существует большое количество версий файла. Поскольку для каждой версии файла необходима своя уникальная дельта-корректировка, то большинство систем обновления программного обеспечения в некоторых случаях могут требовать, чтобы сотни, если не тысячи, уникальных дельта-корректировок соответствовали своей уникальной версии файла.

Некоторые службы обновления, поддерживающие осуществление дельта-корректировок, передают все возможные дельта-корректировки на клиентское вычислительное устройство. Такой подход обычно приводит к значительному увеличению объемов данных, необходимых для выполнения обновления программного обеспечения. Это связано с тем, что с течением времени количество возможных дельта-корректировок для обновлений неизбежно увеличивается. Следовательно, в некоторых ситуациях быстрый рост числа потенциально применимых дельта-корректировок может привести к тому, что размер такого файла обновлений станет соизмеримым с размером полного обновленного файла. При другом подходе сетевая служба обновления программного обеспечения просматривает клиентское устройство, чтобы выбрать, какая дельта-корректировка может быть применима именно для него. Это, безусловно, позволяет значительно уменьшить объем передаваемых данных дельта-корректировок, однако это требует использования дополнительного логического модуля в службе обновления программного обеспечения. Данный модуль должен осуществлять анализ обновляемого приложения непосредственно на рабочем месте пользователя с целью определения перечня необходимых дельта-корректировок. Использование дополнительного логического блока увеличивает использование системных ресурсов, которые должны быть обеспечены службой обновления. К тому же данный подход, как правило, препятствует применению кэширования сети, которое обычно используют традиционные Web-серверы.

Необходимо также отметить, что данные технологии существующих систем обновления не способны поставлять некоторые типы обновлений программного обеспечения, в частности драйверов для аппаратного обеспечения. Как известно в данной области техники, обновления специального программного обеспечения, такого как драйверы аппаратного обеспечения, являются трудной задачей для предоставления пользователям в широком распространении. Это связано с тем, что обновления специального программного обеспечения должны осуществляться на клиентских компьютерах, имеющих специфические аппаратные средства. В большинстве случаев, если клиентский компьютер получит несовместимую версию обновленного драйвера аппаратного обеспечения, установка такой версии драйвера может вызвать фатальную системную ошибку, и даже препятствовать нормальной работе компьютера.

Из всего этого следует, что в настоящее время имеется потребность в системах и способах усовершенствованного обмена обновлениями программного обеспечения между серверами и группами клиентов. Кроме того, имеется потребность в системах и способах обновления программного обеспечения, которые имеют усовершенствованные механизмы предоставления услуг обновления для специализированных конкретных типов клиентов при доставке специальных обновлений.

2.1 Системы управления версиями

Системы управления версиями представляют собой совокупность программных средств (приложений, библиотек функций), систем управления базами данных. Данный комплекс служит для контроля версий файлов обслуживаемого программного обеспечения, организации архива версий и непосредственно для обновления необходимых компонентов. Комплекс для управления версиями организует архив, который предоставляет возможность хранения различных версий одного и того же файла, при этом предоставляется возможность переходить на новые версии или возвращаться к более ранним. Также очень важна функция хранения различной системной информации и описания, как самого программного обеспечения, так и изменений, которые были с ним произведены [10].

В современных информационных системах очень часто возникает необходимость каких-либо обновлений версий программного обеспечения. Как правило, это связано с развитием функционала, оптимизацией различных бизнес-процессов и, конечно, исправлений различных ошибок выявленных в ходе эксплуатации программного продукта. При этом весьма важно бывает сохранить не только новую версию, но и несколько предыдущих. Это связно с тем, что при обновлении могут возникнуть непредвиденные ситуации, когда модификации вызовут сбои в работе системы. В этих случаях может появиться необходимость возврата к одной из предыдущих версий программного обеспечения. Самый простой вариант — хранить несколько версий программного продукта, нумеруя их по специальному алгоритму. Но данный способ малоэффективен, он в значительной мере зависит от человеческого фактора и требует повышенного внимания, что в свою очередь часто ведёт к ошибкам. Очевидно, что данная задача требует автоматизации.

Как правило, системы для управления версиями строятся по принципу централизованной модели. В этом случае для обновляемых файлов создается единое хранилище, которое управляется специальным программным обеспечением, которое и осуществляет процедуры по управлению версиями. На клиентском рабочем месте, чаще всего, должна быть установлена последняя версия программного обеспечения из хранилища, т. н. «рабочая копия». В конкретных ситуациях может быть получена какая-либо из предыдущих реализаций, которая должна быть выбрана по номеру версии.

В случаях внесения разработчиками каких-либо изменений в программное обеспечение, новая версия продукта помещается в хранилище. При этом предыдущая версия файла не удаляется из хранилища. В некоторых системах может использоваться т. н. дельта-компрессия. В таком случае храниться основная копия продукта, и все последующие изменения для каждой зарегистрированной версии, что позволяет значительно сократить объёмы хранимых данных. Некоторые виды систем могут поддерживать сохранение версий обоих видов, когда основная часть версий сохраняется в виде изменений (дельт) и в то же время осуществляется сохранение версий всех файлов в

полном виде. Это позволяет обеспечить возможность максимально полного восстановления истории изменений в случае повреждения хранилища.

Как правило, инсталляция новой версии программного обеспечения производится незаметно для пользователя. В этом случае пользователь просто работает на своем рабочем месте и у него всегда самая последняя версия программного обеспечения.

В некоторых системах предусматривается, что с одним проектом работает одновременно несколько работников. Если несколько человек изменяют одни и те же данные или файлы, то один из них может изменить данные введенные другим или вообще отменить изменения, сделанные другим. Программные комплексы для управления версиями могут контролировать возникновение таких конфликтов и предлагать пути их разрешения. В некоторых системах предусмотрена возможность автоматического объединения изменений, инициированных разными разработчиками. Но такая процедура чаще всего, возможна только для текстовых файлов и при соблюдении определенных условий. Такие возможности обычно закладываются в большинство систем управления версиями ориентированных на обеспечение процессов для разработки программного обеспечения. В случаях, когда не удается выполнить автоматическое объединение, информационная система обычно предлагает пользователю решить судьбу изменений.

Программные комплексы для управления версиями часто обладают дополнительным функционалом:

- возможность создания и хранения в репозитории различных вариантов одной и той же версии (создание «веток»);
- логирование производимых изменений в файле, с указанием даты, времени и автора тех или иных изменений;
- организация специального журнала, в котором пользователи могут записывать различные примечания и пояснения по поводу произведенных изменений в данной версии;
- осуществление контроля прав доступа различных пользователей, с возможностью управления функциями чтения и модификации файлов или каких-либо данных.

В некоторых случаях весьма затруднительно выполнить слияние, как в автоматическом, так и в ручном режиме. Такая ситуация может случиться, если формат файла очень трудно поддается анализу или его сложно определить. В некоторых системах для управления версиями предусмотрена возможность блокировки файла в хранилище каким-либо пользователем. Такая блокировка не дает другим пользователям получить последнюю версию или препятствует модификациям рабочей копии файла и обеспечивает, персональный доступ только тому пользователю, который в данный момент работает с документом.

2.2 Основные понятия систем управления версиями

Как и любая информационная система, программный комплекс для управления версиями имеет свои специфические особенности в архитектуре,

способах обработки и передачи данных и администрирования. Но все же для большинства систем управления версиями можно выделить сходные свойства. Чаще всего происходит так, что проект, который предполагается сопровождать, уже существует и возможно есть некое место на сервере, где размещен архив различных версий программного продукта.

Самым первым шагом, с которого должен начать разработчик, является определение актуальной рабочей копии проекта (совокупности файлов), с которой предстоит работать пользователям. Это может быть произведено при помощи стандартной процедуры получения версии файла либо специального алгоритма. Разработчик задаёт базовую версию программного продукта, которую будет необходимо скопировать в репозиторий. Обычно выбирается последняя версия программного обеспечения.

Вначале устанавливается соединение с сервером, и все файлы рабочей версии проекта копируются на компьютер разработчика. Дублирование рабочей копии является обычной практикой. На локальный либо сетевой диск помимо основного каталога с проектом дополнительно создается ещё одна его копия. В ходе работы с проектом, разработчик модифицирует только файлы рабочей копии. Вторая копия не изменяется и остается в качестве эталона. Она в любой момент может позволить без обращения к серверу определить изменения в проекте. Можно определить какие изменения были внесены в тот или иной файл или в проект в целом. При этом любая попытка модификации этой копии вручную приведет к ошибке в работе системы управления версиями [10].

2.2.1 Ветвление

Мелкие исправления в проекте могут быть осуществлены путём непосредственной правки рабочей копии и последующим переносам изменений прямо в главную ветвь на сервере. Однако если объём работ весьма значительный, то такая последовательность становится мало приемлемой. Это связано с тем, что отсутствие фиксации промежуточных изменений на сервере не позволяет работать с проектом в групповом режиме. При этом ощутимо повышается риск потери модификаций при системных сбоях, а также значительно затрудняется возможность анализа и возврата к предыдущим модификациям кода в пределах данного проекта. Для такого вида модификаций в проектах обычно применяют методику создания ветвей. В этом случае от основного ствола проекта производится ответвление нового варианта или его части, с целью ведения разработок параллельно с изменениями в основной версии. Обычно ветвь создаётся специальной командой системы управления версиями. Рабочая копия ветви создается обычным образом, либо путём переключения имеющейся рабочей копии на заданную ветвь.

При использовании ветвей чаще всего применяется следующая последовательность операций: разработчик периодически обновляет рабочую копию и сохраняет в ней свои модификации. Может случиться так, что ветвь разработки может стать самостоятельным вариантом проекта и развиваться

далее. Но чаще всего, после завершения всех работ, для которых была создана ветвь, она снова сливается с основной ветвью.

2.2.2 Слияние версий

Иногда некоторые процедуры, производимые в системе управления версиями, могут привести к необходимости объединения изменений:

- обновление рабочей копии, когда модификации произведенные в основной версии объединяются с локальными изменениями;
- фиксация изменений, когда локальные модификации объединяются с модификациями зафиксированными в основной версии;
- слияние ветвей, когда изменения, сделанные в разных ветвях, объединяются в одну.

Как правило, в качестве базовой версии для слияния используется та, в которой было осуществлено разделение сливаемых версий. Если это процедура фиксации изменений, то в качестве базовой версии используется версия последнего обновления перед фиксацией, если функция обновления — то версия предыдущего обновления, если слияние ветвей — то версия, в которой была создана соответствующая ветвь.

Большинство современных систем управления версиями в основном ориентировано на проекты разработки программного обеспечения, в файлах которого чаще всего содержится текст. Поэтому механизмы автоматического слияния изменений в первую очередь ориентированы на обработку текстовых файлов.

Так как основным содержимым файлов являются текстовые данные, при определении возможности слияния изменений в пределах одного файла используется типовой механизм построчного сравнения текстов. Данный механизм осуществляет анализ и сравнение объединяемых версий с базовой и создает список изменений. Основной единицей данных для этого механизма является строка, даже минимальное отличие делает сравниваемые строки различными.

Если наборы модифицированных строк по содержанию не пересекаются между собой, то они считаются совместимыми и автоматически соединяются. Если в сливаемых данных обнаруживаются модификации, затрагивающие одну и ту же строку файла, то это приводит к конфликту. Такие данные можно объединять только вручную. Файлы, которые являются бинарными не допускают автоматического слияния.

2.2.3 Сложности и их разрешение

В некоторых случаях при слиянии разных версий происходит пересечение произведенных в них модификаций. Данное стечение обстоятельств называют конфликтом. Система управления версий в данном случае не может произвести автоматическое слияние проектов, поэтому она требует вмешательства человека. Конфликтные ситуации могут возникать при

произведении фиксации изменений, обновлении или слиянии ветвей. В данных случаях текущая процедура приостанавливается в ожидании разрешения конфликта.

Система может предложить разработчику для устранения конфликта три варианта конфликтующих файлов: базовый, локальный и серверный. Конфликтующие модификации предоставляются для принятия решения прямо в объединенном файле со специальной отметкой либо отображаются в специальном окне для объединения изменений.

2.2.4 Блокировки

В некоторых случаях для внесения изменений в файл или группу файлов применяется блокировка, которая предоставляет одному из разработчиков их монопольное использование. При блокировке, файл остаётся доступным для всех остальных разработчиков только для чтения, и любая попытка произвести какие-либо его модификации отклоняется сервером. Блокировка может быть реализована различными способами.

В случае необходимости для произведения блокировки файлы могут быть помечены специальным флагом «заблокирован». Данная отметка может осуществляться системой автоматически при включении файла в проект. Если файл заблокирован, то при передаче рабочей копии с сервера на рабочую станцию он приобретает атрибут «только для чтения». Эта процедура препятствует случайным модификациям. Если разработчику необходимо изменить файл, он инициирует блокировку этого файла. После завершения необходимых модификаций и их фиксации блокировка автоматически снимается. В этот момент локальная копия снова получит атрибут «только для чтения», а с оригинального файла, размещенного на сервере, блокировка будет снята. Тем самым другие разработчики снова смогут получить возможность изменять данный файл.

В некоторых случаях применяют другой метод блокировки, когда для того, чтобы произвести какие-либо изменения необходимо заблокировать определенный набор файлов. Данная методика называется ещё стратегией «блокированного извлечения». Более ранние системы управления версиями, с целью предотвращения появления конфликтов, придерживались исключительно данной стратегии. В современных системах управления версиями данная методика не очень популярна. В настоящее время предпочтительным является способ, использующий неблокирующие извлечения, блокировки применяются только в очень крайних случаях.

Рассмотрим основные недостатки использования блокировок.

Во-первых, блокировки замедляют работу над проектом в группе. Это связано с тем, что блокировки вынуждают ожидать освобождения блокированных файлов остальными разработчиками. При этом в большинстве случаев, даже когда требуются совместные изменения одних и тех же файлов, но которые производятся в ходе разных по направлениям непересекающихся

работ, блокировок можно было избежать, а все изменения были бы объединены при автоматическом слиянии.

Во-вторых, возникновение конфликтов происходит достаточно редко и сложность их разрешения, как правило, не настолько велика, чтобы создать серьёзные затруднения. Серьёзные конфликты изменений могут возникать чаще всего по причине возникновения существенных расхождений во мнениях разных разработчиков относительно программного кода одного и того же фрагмента, либо как результат неправильной организации работы.

В-третьих, очень часто блокировки могут создавать проблемы администрирования. К примеру, кто-нибудь из разработчиков может забыть снять блокировку с занятых им файлов, и при этом уйдет в отпуск. Чтобы решить подобные проблемы будет необходимо прибегнуть к административным мерам, в том числе в системе должны быть предусмотрены специальные технические средства для сброса блокировок. Но даже при их наличии для приведения системы в рабочее состояние потребуется определенное время.

Но, не смотря на некоторые недостатки, использование блокировок в некоторых случаях может быть оправданно. Работа с бинарными файлами может послужить замечательным примером. Для бинарных файлов, как правило, нет инструментария для обеспечения процедуры слияния изменений. А порой такое слияние принципиально может быть невозможно. При отсутствии возможности автоматического слияния при обычном порядке работы к конфликту будет приводить любая параллельная модификация подобных файлов. Лучшим решением в данном случае будет применение блокировки файла с целью жесточайшего контроля последовательности внесения в него любых изменений.

2.2.5 Версии проекта, теги

Основной задачей системы управления версиями является организация учета и хранения всех существующих вариантов файлов, а также как следствие, всех вариантов проекта в целом, начиная с момента его разработки. Существует два различных способа трактовки понятия «версии» в различных системах.

В одних системах поддерживается версионность *файлов*. Этот способ предполагает присвоение любому файлу, появляющийся в проекте, собственного номера версии. Как правило, нумерация начинается с 1, пустой файл с тем же именем считается условной «нулевой» версией файла. Каждый раз, при завершении разработчиком каких-либо изменений, которые затрагивают файл, после применения фиксированных изменений к файлу, он получает новый номер версии, обычно следующий по порядку. В следствии того, что процедура модификации обычно производится только с частью файлов в репозитории, с течением времени версии файлов, постепенно расходятся. Сам проект, фактически не может иметь какого-либо «номера версии», так как он образован множеством файлов, которые имеют различные

номера версий. Примером подобного подхода при учете версий может послужить система управления версиями - CVS.

При втором способе, который использует другой тип систем, понятие «версия» применяется к *репозиторию* целиком, а не к отдельному файлу. При создании пустого репозитория, ему присваивается версия 1 или 0, после чего любая модификация приведет к увеличению этого номера. То есть даже если будет изменен хотя бы один байт одного из файлов проекта, то весь репозиторий получит новый номер версии. Примером такого способа трактовки номеров версий может послужить система Subversion. У отдельного файла здесь, фактически, не существует, номера версии. Таковым можно условно считать текущий номер версии репозитория. В данной системе, при каком-либо изменении, внесённом в репозиторий, номер версии изменяется для всех его файлов, даже для тех, которые не менялись. В таких системах, когда говорят о «версии файла», подразумевают ту версию репозитория, в которой последний раз был изменён какой-либо файл.

Как правило, в большинстве случаев имеет значение весь проект целиком, а не отдельный файл. Поэтому в системах, в которых ведется учет версий отдельных файлов, чтобы идентифицировать определённую версию проекта в некоторых случаях используют дату и время. В этом случае версия проекта будет образована из тех версий входящих в него файлов, которые имелись в репозитории на указанный момент времени. В свою очередь, когда поддерживается версионность репозитория, в качестве номера версии проекта может выступать номер версии репозитория. Но вследствие того, что ни дата, ни номер версии репозитория обычно не несут какой-либо информации о значимых изменениях в проекте, оба этих варианта не слишком удобны. В связи с этим, в системах управления версиями для организации более удобного учета версий проектов было введено понятие *тегов*.

Тег — это символическая метка, которой обозначается какая-либо определённая версия файла или каталога в репозитории. Заданная метка может быть присвоена всем файлам проекта или их части, которые отвечают определённым условиям. Если зафиксировать состояние проекта на некоторый желаемый момент времени, то можно идентифицировать его версию определенным образом: версия «XX.XXX.XXX» — это набор версий файлов репозитория, имеющих тег «XX.XXX.XXX». В связи с определенной гибкостью системы тегов, можно помечать одним тегом версии файлов и каталогов. Это дает возможность собирать «версию проекта» любым произвольным образом. Пометка тегами может иметь различный вид. В некоторых системах она формируется именно в виде пометки. В других системах (например, Subversion) тег может представлять собой некий отдельный каталог на файловом дереве репозитория, где размещают копии нужных версий файлов из ствола и ветвей проекта. В этом случае тег выглядит как сборник копий версий определенных файлов репозитория, вынесенный в отдельный каталог. Общепринято, что фиксация каких-либо изменений в

дереве каталогов, соответствующем тегу, запрещена. То есть версия проекта, которая представлена тегом, является неизменяемой.

2.2.6 Базовые принципы разработки программного обеспечения

В любой фирме или организации, занимающейся разработкой проектов, устанавливается определенный порядок использования системы управления версиями. В каждом конкретном случае это обусловлено техническими регламентами и правилами. Но, не смотря на то, что число разработок и систем управления версиями существует великое множество, общие принципы, определяющие регламент правильного применения систем управления версиями немногочисленны и едины.

Первый принцип – все виды версий проекта (рабочие, тестовые или демонстрационные) могут собираться только из репозитория системы. В некоторых случаях, с целью проведения промежуточного тестирования, разработчики могут делать так называемые «персональные» сборки, которые содержат какие-либо незафиксированные на данный момент изменения. Все это обеспечивает определенную возможность создания рабочей версии проекта в любой момент времени и гарантирует, что репозиторий содержит всё необходимое для данной процедуры.

Второй принцип – независимо от проведения модификаций, которые осуществляются в проекте, текущая версия главной ветви в любой момент времени должна быть корректна. Если какие-либо изменения являются неполными или не прошли хотя бы предварительное тестирование, ни в коем случае не допускается их фиксация в главной ветви проекта. Сборка проекта, произведённая из текущей версии в любой момент времени, должна быть гарантировано успешной.

Третий принцип – при проведении модификаций любое значимое изменение в проекте обязательно должно быть оформлено как отдельная ветвь. Все промежуточные результаты работы разработчика также фиксируются в эту ветвь в обязательном порядке. После окончания работ, связанных с модификациями проекта, выполняется процедура объединения данной ветви со стволом. Исключение могут составлять случаи, когда изменения достаточно меткие, и все работы проводятся в течение не более чем одного рабочего дня и, как правило, одним разработчиком.

Четвертый принцип – все версии проекта помечаются определенными тегами. Если над версией уже произведена процедура выделения, и она помечена тегом, то такая версия больше никогда не изменяется [11].

2.2.7 Распределенные системы управления версиями

Некоторые системы управления версиями используют распределённую модель взамен традиционной клиент-серверной. Как правило, такие системы, не нуждаются в организации централизованного хранилища. При использовании распределенной модели вся история изменении документов хранится в локальных хранилищах, размещенных на персональных

компьютерах каждого участника проекта. В случае необходимости, отдельные фрагменты истории локального хранилища могут быть синхронизированы с аналогичным хранилищем, расположенном на компьютере другого разработчика. При таком построении системы, в некоторых случаях, локальное хранилище может располагаться непосредственно в каталогах рабочей копии.

Пользователь такой системы всегда работает со своей локальной копией хранилища, когда выполняет обычные действия, связанные с извлечением определённой версии документа либо созданием новой версии и тому подобное. С течением времени хранилища, которые принадлежат разным разработчикам по мере внесения изменений, начинают различаться. И рано или поздно возникает необходимость в синхронизации хранилищ этих пользователей. Как правило, такая синхронизация осуществляется путем формирования специальных патчей или так называемых наборов изменений, с последующим их обменом между пользователями системы.

Данная модель по своей логике очень близка процедуре создания отдельной ветки для каждого разработчика из классической системы управления версиями. Очень часто перед началом работы с локальным хранилищем в распределённых системах необходимо создавать новую ветвь. И основным отличием является то, что другие разработчики этой ветви не увидят вплоть до момента синхронизации системы. Работа разработчика не влияет на других участников проекта, пока он изменяет свою ветвь, и соответственно другие участники не могут оказывать влияния на его работу. Внесённые в ветви изменения, по завершении индивидуальной части работ, в обязательном порядке сливают с основной (общей) ветвью. Безусловно, что при синхронизации разных хранилищ, а также при слиянии ветвей возможно возникновение конфликтов версий. Во всех системах управления версиями для обнаружения и разрешения конфликтов слияния, как правило, предусмотрены те или иные методы.

Для обычного пользователя основное отличие распределённой системы заключается в необходимости создания локального репозитория, а также наличием двух дополнительных функций в языке команд:

- запрос для получения репозитория от удалённого компьютера;
- передача на удалённый компьютер своего репозитория.

Первая процедура осуществляет функцию слияния изменений удалённого и локального репозиториев, после чего результат размещается в локальном репозитории; вторая — также выполняет функцию слияния изменений двух репозиториев с тем лишь отличием, что результат размещается в удалённом репозитории. Как правило, при выполнении процедуры слияния в распределённых системах есть возможность выбора следующих параметров:

- какие наборы изменений необходимо передавать в другой репозиторий или извлекать из него;
- исправлять конфликты слияния непосредственно в ходе операции или после её неудачного завершения;
- возобновлять или повторять неоконченное слияние.

Обычно только при условии отсутствия конфликтов передача своих изменений в чужой репозиторий завершается успешно. Если конфликты все же возникают, прежде чем передавать свои изменения другим, пользователь должен сначала слить версии в своём репозитории.

Для удобства рекомендуется организовывать работу с системой таким образом, чтобы пользователи на постоянной основе выполняли слияние в своем репозитории. То есть, если в централизованных системах пользователи передают свои изменения на центральный сервер, когда сами считают нужным, то в распределённых системах, в отличие от этого — кому нужно получить слияние версий в данный момент (например, разработчик, управляющий сборочным сервером), тот чаще всего и инициирует его.

Основными преимуществами распределённых систем по сравнению с централизованными системами, безусловно, являются их гибкость и большая независимость отдельных рабочих мест. Фактически каждый компьютер разработчика в подобной системе является самостоятельным и полнофункциональным сервером. Достаточную гибкость обеспечивает то условие, что, задав желаемый порядок синхронизации, из таких компьютеров можно построить произвольные системы различные как по структуре, так и уровню сложности. При этом все разработчики могут осуществлять свою работу совершенно независимо. При выполнении изменений и сохранений промежуточных версий документов, все возможности системы, в том числе доступ к истории изменений, находятся в распоряжении пользователя даже в отсутствие сетевого соединения с сервером. Как правило, связь с сервером или другими разработчиками необходима только для проведения процедур синхронизации хранилищ. Обмен наборами изменений при этом может осуществляться разными методами и по различным схемам.

К сожалению, распределённые системы не лишены некоторых недостатков. К ним можно отнести некоторое увеличение требуемого объёма дисковой памяти на каждом компьютере пользователя. Это связано с необходимостью хранения полной истории версий в локальной системе, тогда как в централизованной системе на компьютере разработчика, как правило, хранятся лишь рабочая копия (срез репозитория на какой-то момент времени) и внесённые изменения. Весьма неприятным недостатком, хотя и менее очевидным, является то, что в распределённой системе, в отличие от централизованных систем, практически невозможно реализовать некоторые виды функциональности:

- не возможна функция блокировки файла или группы файлов, так как для хранения признака блокировки нужен центральный сервер, который всегда должен быть общедоступным и постоянно находящимся в онлайне;
- отсутствует какая-либо возможность слежения за определённым файлом или группой файлов в виду того, что изменения файлов, а также выполнение процедур слияния происходят локально; об изменениях может стать известно только после процедуры синхронизации;

- нет возможности единой сквозной нумерации версий системы или файлов, в которой номер версии монотонно возрастает, так как такая нумерация также требует наличия в системе центрального сервера, который предоставляет номера версий для всех остальных разработчиков. Как правило, в распределённых системах приходится обходиться механизмом локальных обозначений версий и применять назначение тегов, которые заранее определены корпоративными стандартами фирмы или соглашением между разработчиками;
- отсутствует возможность осуществления разработчиком локальных модификаций с отдельной, небольшой по объёму выборкой. При условии, что эта выборка произведена из хранилища, имеющего внушительные размеры, сложную внутреннюю структуру и расположенном на удалённом сервере.

В тоже время, в некоторых ситуациях, использование распределённых систем, позволяет добиться заметных преимуществ, по сравнению с централизованными системами:

- возможность периодической синхронизации сразу нескольких компьютеров, на которых может работать один разработчик (рабочий и домашний компьютер, ноутбук и так далее). В распределённых системах нет необходимости выделять в качестве центрального сервера один из компьютеров. Синхронизация может выполняться только по мере необходимости, например, в случаях, когда разработчик работает на разных устройствах и периодически «пересаживается» с одного на другое;
- достаточно простая организация совместной работы над одним проектом небольшой группы разработчиков, которая может быть территориально распределена, при этом нет необходимости для выделения общих информационных ресурсов. В этом случае также реализуется схема, когда в главном сервере нет необходимости. Актуальность репозиториев обеспечивается методом периодических синхронизаций по схеме, которую можно было характеризовать как «каждый с каждым»;
- возможность ведения крупных распределённых проектов, в которых все разработчики могут долгое время работать независимо каждый над своей частью, при этом у них нет никакой необходимости в постоянном подключении к единой сети.

Такой проект может для удобства использовать централизованный сервер, с которым будут синхронизироваться все его участники, но это не является обязательным условием. Возможны и другие более сложные варианты. К примеру, для работы над достаточно крупным проектом, могут быть созданы небольшие группы для организации работы по отдельным направлениям «ветвям». При этом для каждой группы может быть выделен отдельный «групповой» сервер для синхронизации работы группы разработчиков. В этом случае процесс завершения работы над проектом (окончательного слияния изменений) приобретает древовидный характер: сначала отдельные группы разработчики синхронизируют свои изменения на групповых серверах, а уже на следующем этапе обновлённые репозитории с

групповых серверов синхронизируются с центральным сервером. Использование «групповых» серверов для организации совместной работы совсем не обязательно, в этом случае разработчики одной группы должны предварительно произвести синхронизацию изменений между собой, после этого любой из них, к примеру, руководитель группы, может передать изменения на главный сервер.

При организации разработки проектов в рамках одного офиса, когда относительно небольшая группа разработчиков целиком находится в одном помещении. Когда работы ведутся с подключением к единой локальной компьютерной сети, с постоянно доступными серверами. В этом случае централизованная система из-за своей более жёсткой структуры и наличия функциональности, которая отсутствует в распределённых системах (в частности, уже упомянутая ранее блокировка), может оказаться лучшим выбором. В таких условиях возможность фиксации изменений без слияния модификаций в центральную ветвь можно легко реализовать путём выделения в отдельные ветви разработки незавершённых работ [9; 10].

2.2.8 Нумерация версий программного обеспечения

Как правило, жизненный цикл успешной реализации программного обеспечения может быть достаточно долгим. Изменения, которые производятся в программном обеспечении на протяжении его жизни, могут носить разный характер. Модификации могут осуществляться как для исправления какой-либо ошибки, так и для полного пересмотра концепции. При этом в большинстве случаев название программы не изменяется, а меняется именование модификации. Это именование модификации называется версией.

Порядок именования версий программного обеспечения может быть разным. Это могут быть целые числа (Corel Draw 11), дробные числа (Windows 3.11) или последовательность чисел (JDK 1.6.0). В качестве обозначения версий также может быть год (Windows 2000) или какой-либо текст (Embarcadero Delphi XE). Система присваивания версий, в большинстве случаев, выбирается по определенным критериям:

- поддержка со стороны программного обеспечения для разработки (компилятора, системы контроля версий и т. д.) той или иной системы учета версий;
- частота выхода новых версий и необходимость выпуска промежуточных тестовых версий.

В некоторых случаях, когда программное обеспечение очень сложное и выпуск релизов производится раз в несколько лет, то перед выпуском устойчивой версии проводится всеобъемлющее тестирование. Подобное программное обеспечение может именоваться как «Microsoft Word 97 SP2». Для программного обеспечения с частыми нестабильными выпусками может понадобится принятие более сложной нумерации версий.

В некоторых схемах нумерации последовательные идентификаторы используются для определения значимости перемен между стадиями

разработки: эти перемены классифицируются по уровням значимости. Решение о том, какую из последовательностей (в сложной нумерации версий) изменить между стадиями разработки, основано на значимости перемен на последней стадии разработки. Например, в схеме, с версией, состоящей из последовательности 4 чисел, первое число может быть прибавлено только тогда, когда код полностью переписан, в то время как четвертое число изменяется при незначительных переменах в интерфейсе или документации. Эта практика позволяет пользователям (или потенциальным последователям), оценить, насколько было протестировано в реальных условиях данное программное обеспечение. Если изменения сделаны, например, между 1.3rc4 (RC — release candidate, релиз-кандидат) и продукционным выпуском 1.3, то выпуск 1.3rc4 не предполагает уровень тестирования производственного класса и, на самом деле, содержит изменения, которые не обязательно были испытаны в реальных условиях. Такой подход обычно допускает применение третьего уровня нумерации («изменения»). Например: 1.3.1, 1.3.2, 1.3.3, 1.3.4 и т. д.

В более поздних релизах, главное число (major) увеличивается, когда происходят значительные переходы в функциональности, второстепенное число (minor) прибавляется только тогда, когда были добавлены незначительные функции или внесены исправления. Номер версии изменяется, если исправлены все мелкие неполадки. Для типичного программного продукта используются следующие номера: 0.9 (для бета-версии), 0.9.1, 1.0, 1.0.1, 1.1, 1.1.1, 2.1.2, 2.2, и т. д. Разработчики порой перескакивают, например от версии 5.0 сразу к 5.5, для того чтобы обозначить добавление нескольких значимых функций в программе, однако их не достаточно, чтобы изменить главный номер версии. Некоторые считают, что такие скачки неуместны.

Другой подход использования главных и второстепенных номеров версий заключается в добавлении буквенно-цифровой последовательности, определяя тем самым стадию разработки релиза: «альфа», «бета», «релиз-кандидат». Серия версий с использованием этого подхода может выглядеть следующим образом: если к версиям 0.5, 0.6, 0.7, 0.8, 0.9 добавляются новые функции их можно назвать — 1.0b1, 1.0b2, еще плюс новые функции — 1.0b3, затем версия становится 1.0rc1. Если версия 1.0rc1 достаточно стабильна, то она становится 1.0, однако если в 1.0rc1 обнаруживаются ошибки, которые необходимо исправить, она будет иметь номер 1.0rc2 и т. д. Важной характеристикой этого подхода является соблюдение идентичности стадий разработки версий. Нельзя вносить никаких изменений между последней бета-версией и первым или последним релизом-кандидатом и готовым продуктом. Если это сделано, необходимо выпустить другую версию на более низкой стадии разработки.

Иногда присутствие человеческого фактора в создании номеров версий приводит к ошибкам в изменении версий. Например, разработчики могут изменить последовательность между версиями, даже если ни одна строчка кода не была переписана, лишь для того чтобы создать ложное впечатление, что были внесены значительные изменения.

Другие схемы передают значение на отдельных последовательностях:

- major.minor[.build[.revision]];
- major.minor[.maintenance[.build]].

Опять же, в этих примерах отличия «существенных» от «незначительных» изменений, даются произвольно и по усмотрению автора, как то, что означает «build» или как «revision» отличается от «minor change». Схожие проблемы относительной значимости изменений и номенклатуры версий существуют в сфере книгоиздания, где номер издания или название может быть выбрано на основе различных критериев.

В большинстве проприетарного программного обеспечения, первая официальная версия программного продукта имеет версию 1.

Выводы по второму разделу

В современном мире компьютерные технологии приобретают все более важное значение. Персональные компьютеры наполнили все сферы жизни человека. Вычислительная техника широко применяется как в области начального образования, так и в исследованиях новейших технологий. Невозможно представить какое-либо современное предприятие без серьезного серверного оборудования и информационных систем. Программные комплексы обеспечивают работу сложнейших бизнес-процессов.

В условиях бурного развития информационных технологий программное обеспечение претерпевает постоянные модификации, с целью исправления каких-либо ошибок или совершенствования функциональных возможностей и функций. Каждая модификация программного продукта или компонента может сопровождаться необходимостью добавления новых файлов, а также замены имеющихся файлов файлами более новых версий. После того, как разработчик выявил и устранил ошибку в своем программном продукте и создал реализацию решения данной проблемы, требуется включить эту модификацию в соответствующее обновление, и предоставить новую реализацию продукта его пользователям. Разработчик весьма заинтересован в получении произведенных обновлений программного обеспечения всеми пользователями в кратчайшие сроки и с максимальной гарантией.

3 АНАЛИЗ ИНФОРМАЦИОННОЙ СИСТЕМЫ «РЕГИСТРАЦИОННЫЙ ПУНКТ ДОКУМЕНТИРОВАНИЯ И РЕГИСТРАЦИИ»

В декабре 1991 года был принят Закон о независимости Республики Казахстан. После провозглашения государственного суверенитета в Республике Казахстан в 1993-1994 годах были утверждены новые Положения о паспортной службе и образцах национального паспорта и удостоверения личности.

На основании указанных нормативных документов в городе Алматы был организован «Центр документирования населения» Министерства внутренних дел Республики Казахстан. Осенью 1994 года немецкой компанией «Giesecke & Devrient» на базе «Центра документирования населения» была запущена в эксплуатацию информационно-производственная система, которая положила начало автоматизации документирования населения в Республике Казахстан.

Основной переход на документы Казахстанского образца был завершен весной 1999 года. К этому моменту было документировано порядка 16 миллионов граждан, создана база данных документирования населения Республик Казахстан. Но за время эксплуатации существующая система успела морально и физически устареть. Повысились требования к надежности и информационной безопасности системы, появились необходимость развития поискового функционала и обработки существующей информации, получения различных статистических данных.

В связи со сложившейся ситуацией весной 2003 года был запущен проект по комплексной модернизации информационно-производственной системы документирования населения. Данная модернизация проводилась силами разработчиков «Центра документирования населения» Министерства внутренних дел Республики Казахстан. Была произведена полная реорганизация структуры хранения данных. Значительно расширена последовательность технологических операций для повышения качества и надежности ввода данных, обеспечения информационной безопасности информационно-производственной системы.

Разработанный программный продукт позволял осуществлять основные производственные операции, связанные с вводом и проверкой текстовой информации, а также производить персонализацию удостоверяющих личность документов Республики Казахстан: удостоверения личности гражданина, паспорта гражданина, а также вида на жительство для иностранцев и удостоверения лица без гражданства. В обновленных документах была организована цветная печать фотоизображения владельца.

Программное обеспечение расширило функционал производственной системы, связанный с проверкой правильности ввода текстовой и графической информации, что позволило повысить качество работы операторов. Была модифицирована структура базы данных с целью оптимизации хранения данных, что обеспечило возможность исключить хранение дублирующей информации. Это в свою очередь облегчило труд операторов, исключив

необходимость повторного ввода уже имеющихся сведений, повысило качество информации.

Была разработана система регистрации операторов и рабочих станций, с разграничением прав доступа к тем или иным производственным операциям. Все действия в системе стали регистрироваться с фиксацией даты и времени, а также со сведениями об операторе, совершившим данное действие. Это обеспечило прозрачность производимых операций, повысило надежность и безопасность системы.

Позднее (весной 2005 года) была модифицирована система ввода графической информации, а также модернизирован комплекс персонализации паспортов. Обработка графических данных посредством аналоговых камер при документировании была заменена процедурой автоматизированного сканирования фотоизображения и образца подписи заявителя.

Разработанная информационная система персонализации удостоверяющих личность документов на сегодняшний день полностью удовлетворяет требованиям информационной безопасности, а также технологическим параметрам производства удостоверяющих личность документов.

3.1 Производственная подсистема ввода и передачи данных для оформления заявлений на документирование

Процесс изготовления удостоверяющих личность документов связан с определенной спецификой: строгий учет и жесточайший контроль использования бланочной продукции и уникальных расходных материалов; дорогостоящее специализированное и высокотехнологичное оборудование. Данная специфика обуславливает централизованную организацию основного производства по персонализации удостоверяющих личность документов. Но в то же время оформление заявлений на документирование должно осуществляться в регионах, по месту жительства заявителей.

Первоначально при создании автоматизированной системы документирования населения за основу был принят традиционный способ оформления заявлений – на бумажном бланке (формуляре Ф-1). Форму заявления заполнял работник документирования в региональном подразделении, затем посредством службы специализированной доставки формуляр по инстанциям направлялся в «Центр документирования населения» Министерства внутренних дел Республики Казахстан. В Центре производился ввод данных с заполненного формуляра, далее осуществлялся процесс изготовления документов. Готовые документы вместе с заявлением (формуляром Ф-1) направлялись через службу специализированной доставки по месту оформления. От оформления заявления до момента получения готового документа требовалось около трех месяцев. В связи со сбоями в работе службы доставки сроки ожидания гражданами готовых документов значительно затягивались. Ситуация усложнялась в случаях ошибок при заполнении формы заявления, когда формуляр направлялся повторно к месту

заполнения для внесения исправлений. Немаловажным фактором риска была порча или даже утеря заявлений (формуляров Ф-1) при транспортировке.

Вполне очевидно, что данная технология передачи данных весьма устарела для современных технологий. Поэтому в 2005 году была пересмотрена общая технология сбора и обработки информации по линии документирования и регистрации граждан. В результате, в рамках проекта развития государственной программы построения «электронного» правительства Республики Казахстан была разработана новая концепция Республиканской информационной системы документирования и регистрации граждан.

В рамках реализации новой концепции была разработана информационная система «Регистрационный пункт документирования и регистрации» (далее ИС «РП ДРН»), которая стала неотъемлемой частью автоматизированного программного комплекса документирования и регистрации населения. Данная система призвана автоматизировать в региональных подразделениях основные бизнес-процессы, связанные с документированием и регистрацией населения Республики Казахстан, а также позволила полностью отказаться от передачи сведений на бумажных носителях.

Внедрение ИС «РП ДРН» позволило значительно повысить качество обслуживания населения, сократить сроки изготовления удостоверяющих личность документов, облегчить труд работников документирования, позволило усилить контроль и персональную ответственность за принятие решений должностных лиц, а так же обеспечило прозрачность процесса документирования на всех ключевых этапах, в том числе и для самого заявителя.

3.2 Информационная система «Регистрационный пункт документирования и регистрации населения»

С февраля 2010 года по всей Республике Казахстан запущен в эксплуатацию проект электронного документирования и регистрации населения – ИС «РП ДРН». При оформлении заявок на документирование, а также при регистрации граждан данная информационная система обеспечивает обмен информацией региональных подразделений документирования с центральным банком данных в on-line режиме. Работники документирования на местах получили автоматизированный доступ к центральной архивной электронной картотеке документирования и регистрации населения.

ИС «РП ДРН» развернута на базе центров обслуживания населения (ЦОН) по всей Республике Казахстан. На данный момент Система обеспечивает обработку следующих бизнес-процессов:

- идентификация заявителя при регистрации различного вида заявок;
- регистрация заявок на документирование граждан (заказ удостоверения личности и паспорта гражданина Республики Казахстан);
- ввод текстовых сведений и графической информации (фотографирование и получение образца подписи);
- выдача готовых удостоверений личности и паспортов заявителям;

- регистрация изменений юридического адреса (адреса места жительства) граждан Республики Казахстан;
- генерация индивидуального идентификационного номера (ИИН) для персон не имеющих такового;
- различные сервисные функции для работы с архивными сведениями.

При оформлении заявок граждан сотруднику документирования из центрального банка данных предоставляется вся необходимая информация о предыдущем документировании заявителя, в том числе биометрические данные (фотография и образец подписи). Это позволяет улучшить качество идентификации персоны и ускорить принятие решения специалистом о возможности документирования или регистрации.

Ввод всей информации по новому заявлению, а также фотографирование и получение образца подписи заявителя осуществляется с помощью электронной системы. Процедура ввода текстовых данных снабжена подсистемой справочников, а также расширенной функцией форматно-логического контроля.

После оформления и проверки все данные в электронном виде передаются в центральную систему. Это позволило отказаться от бумажных носителей и значительно сократить сроки доставки заявлений до производственной системы. Регистрация выдачи готовых документов гражданам также автоматически фиксируется в системе.

Новая информационная система также автоматизирует и централизует регистрацию юридических адресов граждан на местах. Информация о новом юридическом адресе моментально попадает в центральный банк данных. А при наличии у гражданина удостоверения нового образца одновременно осуществляется запись нового юридического адреса в чип документа.

В ходе эксплуатации программного комплекса были выявлены некоторые технологические особенности, связанные с удаленным доступом к центральной системе и способом хранения данных на различных производственных этапах. Появилась необходимость в некоторых изменениях и доработках информационной системы.

С целью оптимизации и развития ИС «РП ДРН» была произведена комплексная модернизация программного обеспечения. Были пересмотрены основные бизнес-процессы обмена данными. Перераспределены контрольные функции, повышена персональная ответственность участников на всех этапах процедур документирования и регистрации населения. С марта 2013 года обновленная версия ИС «РП ДРН» была введена в эксплуатацию. В отличие от предыдущей версии системы данные полученные и измененные на всех этапах ИС «РП ДРН», начиная с регистрации заявки, сохраняются на центральном сервере. Данная процедура позволяет исключить какую-либо возможность потери важной информации из-за возможных сбоев серверного оборудования на местах.

В рамках ИС «РП ДРН» процедура документирования граждан производится работниками отделов документирования в районных

регистрационных пунктах, расположенных на территории Центров обслуживания населения. При обращении заявителя в ЦОН по месту жительства, работником отдела документирования производится первичная регистрация заявки с обязательной идентификацией персоны по центральному банку данных. При этом из центральной базы данных работник документирования получает всю необходимую информацию, в том числе и графическую, для соответствующих проверок и принятия решения о возможности документирования.

Далее, в рамках информационной системы, осуществляется ввод новых графических данных (фото и образец подписи), а также ввод новых или корректировка существующих текстовых данных, полученных из центральной системы. При этом правильность и обязательность заполнения всех необходимых полей контролируется системой. Ввод информации облегчается системой благодаря наличию справочников, что снижает вероятность ошибок заполнения. Если данные о персоне уже существуют в центральном банке, то при оформлении нового заявления достаточно только внести необходимые изменения.

После оформления и проверки электронной заявки в районном подразделении правильность данных визируется личной подписью заявителя и электронно-цифровой подписью работника документирования. Затем заявка попадает в ведение центральной системы для дальнейшей обработки и изготовления документов.

На республиканском уровне ответственными сотрудниками документирования, в рамках информационной системы осуществляется контроль правильности принятия решения о документировании в районном подразделении с возможностью получения исторических сведений из центрального архивного банка данных. После произведенных проверок электронные заявки на документирование становятся доступными для информационно-производственной системы изготовления документов.

В центральной информационно-производственной системе осуществляются процедуры связанные с изготовлением документов, удостоверяющих личность. Выполняется процесс персонализации документов – заполнения установочных данных на видимую часть документа и во встроенный чип. После завершения производства и проверки качества готовые документы направляются для выдачи заявителям в ЦОН, по месту оформления заявок.

Процедура выдачи готовых документов гражданам в регистрационных пунктах на базе ЦОН также регистрируется в системе, при этом в центральном банке данных фиксируется непосредственная дата получения документов заявителем.

3.3 Структура ИС «РП ДРН»

ИС «РП ДРН» имеет разветвленную иерархическую структуру. В её состав входят, расположенные в районных регистрационных пунктах

клиентские рабочие станции и коммуникационные сервера, а так же центральный сервер, являющийся частью центральной информационно-производственной системы, развернутой в РГП «ИПЦ» МВД Республики Казахстан. Через коммуникационные серверы осуществляется обмен необходимой информацией между клиентскими рабочими станциями и центральным сервером, как показано на рисунке 3.1.

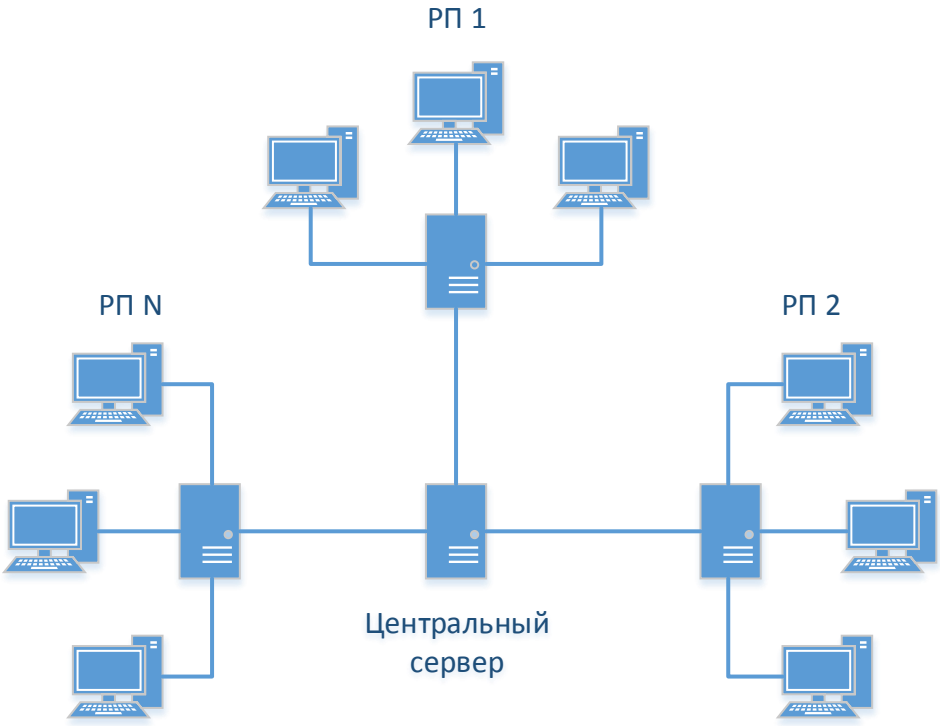

Рисунок 3.1 – Структура ИС «РП ДРН»

Неотъемлемой частью ИС «РП ДРН» является программное обеспечение, разработанное РГП «ИПЦ» и требующее обновления в случаях модернизации информационной системы, которое функционирует на клиентских рабочих станциях и коммуникационных серверах. Так же необходимо учесть, что в районных регистрационных пунктах клиентские рабочие станции функционируют только в рабочее время, в отличие от коммуникационного сервера, который работает круглосуточно.

ИС «РП ДРН» является сложным программно-техническим комплексом, использующим различное оборудование и технические средства для осуществления необходимых бизнес-процессов. В ИС «РП ДРН» используются различные считыватели документов и их чипов, фотоаппараты и специализированные камеры, сканеры изображений и устройства захвата подписи. Также к системе предъявляются повышенные требования безопасности. В виду невозможности в данных условиях применения «тонкого клиента», система построена по принципу использования «толстого клиента».

ИС «РП ДРН» включает в себя порядка 500 регистрационных пунктов. Каждый регистрационный пункт, в зависимости от размера и возможностей

ЦОН-а, оснащен 5-30 рабочими станциями. Общее количество рабочих станций в системе превышает 12 000. С наращиванием функционала и пунктов обслуживания населения эти цифры постоянно растут.

3.4 Функциональные возможности клиентского программного обеспечения

В клиентском приложении РП ДРН реализованы следующие возможности:

- идентификация физического лица посредством ИИН или на основании предъявленного документа, удостоверяющего личность;
- разделение процесса оформления на разные этапы, несколько способов сбора графической информации, которые подробно описаны в следующих разделах;
- перевод подразделения ДРН на единые форматы данных, систему электронного документирования и регистрации физических лиц;
- оптимизация регистрационных процессов в подразделениях ДРН в соответствии с требованиями технологии обработки информации в ИБД СИД, сокращение ручных операций, улучшение обслуживания населения;
- доступ к клиентскому приложению РП ДРН в зависимости от присвоенных прав на этапы оформления заявок;
- запись электронной цифровой подписи владельца удостоверения личности;
- процедура ввода заявителем персонального PIN-кода при выдаче готовых удостоверений личности.

В целях исключения предъявления заявителем подтверждающих бумажных документов реализованы взаимодействия РП ДРН с:

- автоматизированной информационной системой «Государственный центр по выплате пенсий» Министерства Республики Казахстан труда и социальной защиты населения;
- информационной системой «Записи актов гражданского состояния»;
- информационной системой Агентства Республики Казахстан по статистике «Статистический регистр населения»;
- государственной базой данных «Регистр недвижимости» и информационной системой «Адресный регистр».

Взаимодействие РП ДРН с автоматизированной информационной системой «Государственный центр по выплате пенсий» Министерства Республики Казахстан труда и социальной защиты населения предназначено для подтверждения факта освобождения заявителя от уплаты государственной пошлины.

В случае составления актовых записей в органах регистрации актов гражданского состояния, в результате которого производится изменение установочных данных, с ИС ЗАГС поступают измененные сведения на центральный узел РП ДРН.

В результате взаимодействия РП ДРН с информационной системой Агентства Республики Казахстан по статистике «Статистический регистр населения» сбор сведений о регистрации необходимой органам статистики будут производиться в клиентском приложении РП ДРН и направляться в автоматическом режиме.

Взаимодействие РП ДРН с государственной базой данных «Регистр недвижимости» позволяет получить сведения об объекте недвижимости и собственнике жилья.

Взаимодействие РП ДРН с информационной системой «Адресный регистр» организовано в целях унификации использования адресной информации в различных информационных системах и позволяет получить адресную информацию из ИС АР по коду адресного ресурса.

Первым этапом при работе с клиентским программным обеспечением является авторизация оператора как показано на рисунке 3.2.

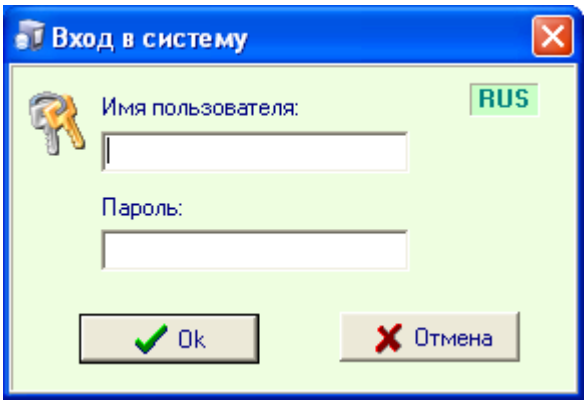

Рисунок 3.2 – Авторизация оператора

На следующем этапе осуществляется связь с центральным сервером системы, происходит идентификация пользователя, проверяется доступ для работы с системой, и на экране появится главное меню изображенное на рисунке 3.3.

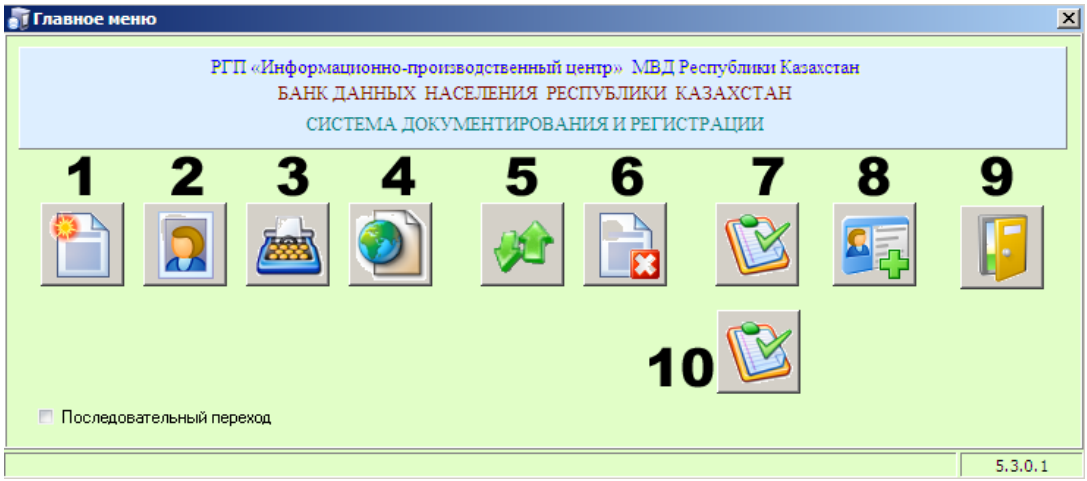

Рисунок 3.3 – Главное меню клиентского приложения

На рисунке изображено:

1. Регистрация заявок;
2. Ввод графической информации (фотографирование/сканирование);
3. Ввод текстовой информации (ввод/корректировка текстовых данных);
4. Регистрация смены юридического адреса, Временная регистрация места жительства;
5. Передача заявок на центральный узел;
6. Работа с возвратами;
7. Выдача готовых документов;
8. Изменение PIN-кода и подготовка ID-карты к записи электронно-цифровой подписи;
9. Выход из программы;
10. Регистрация выдачи документов лицам, находящимся в розыске.

Для осуществления приема заявки от гражданина на предоставление необходимой ему услуги вызывается соответствующая форма изображенная на рисунке 3.4.

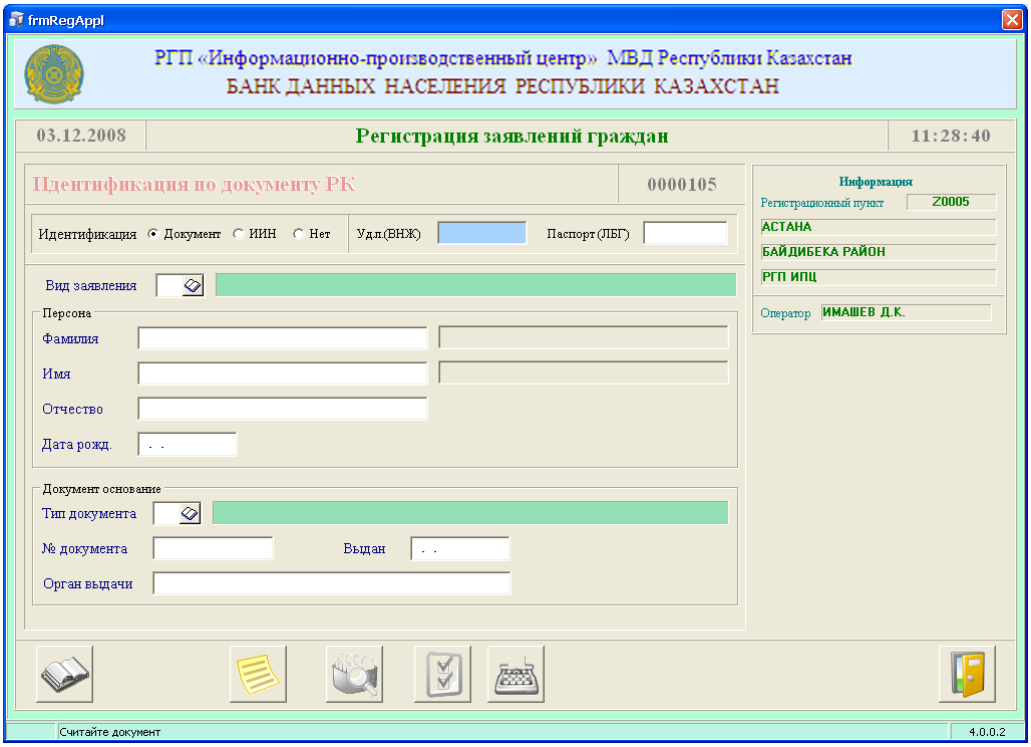

Рисунок 3.4 – Форма регистрации услуги

Работа с гражданами начинается с регистрации заявок. После проверки документов и подтверждения обоснованности и законности обращения, в ИС «РП «ДРН» регистрируется обращение гражданина. Если гражданин имеет

документы Республики Казахстан, то можно провести его идентификацию по номерам УДЛ или паспорта.

Выводы к третьему разделу

ИС «РП ДРН» находится в постоянной динамике. Это связано с развитием функционала, оптимизацией бизнес-процессов, внедрением новых технологий и т.д. При использовании технологии «толстого клиента» любое обновление системы требует обновления программного обеспечения непосредственно на персональных компьютерах клиентских рабочих мест. При комплексных обновлениях необходимо обновлять программное обеспечение практически одновременно по всей стране и как можно в более сжатые сроки.

Проведя анализ информационной системы, изучив её структуру и основные бизнес-процессы можно сказать, что одним из её основных недостатков является отсутствие системы автоматического обновления программного обеспечения.

4 РЕШЕНИЕ ПРОБЛЕМЫ АКТУАЛИЗАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИС «РП ДРН»

Сложность при обновлениях программного обеспечения ИС «РП ДРН» заключается в том, что система предназначена для обслуживания населения в реальном времени. Поэтому любые задержки с обновлением программного обеспечения приведут к остановке работы регистрационного пункта, и как следствие к огромным очередям и недовольству граждан.

Т. к. рассматриваемая далее система автоматического обновления программного обеспечения должна стать частью ИС «РП ДРН», она должна учитывать все особенности её архитектуры. Обновление должно происходить как можно незаметнее для пользователя одновременно на всех клиентских рабочих станциях и максимально быстро, что бы свести к минимуму задержки в работе системы. Учитывая архитектуру информационной системы, очевидно, что для решения поставленной задачи обновление программного обеспечения должно происходить централизованно.

Разработанная система включает в себя набор разработанного программного обеспечения и функций для обновления коммуникационных серверов и клиентских рабочих станций а так же элементов базы данных.

На центральном сервере организован репозиторий для хранения архива различных версий программного обеспечения, а так же были созданы таблицы, хранящие различную служебную информацию о файлах и приложениях.

Программное обеспечение, функционирующее на коммуникационных серверах и клиентских рабочих станциях будет обновляться с помощью разработанных вспомогательных программ.

Архитектура разработанной системы автоматического программного обеспечения показана на рисунке 4.1.

Рисунок 4.1 – Архитектура системы автоматического обновления ПО

4.1 Архив версий программного обеспечения

Репозиторий на центральном сервере представляет собой файловое хранилище в виде дерева каталогов в файловой системе Windows. Файловая система берет на себя задачи:

- хранение информации;
- структурирование файлов;
- обеспечение прав доступа к данным;
- защита данных.

В хранилище содержатся версии различного программного обеспечения. Для организации данных необходимо, чтобы каждая программа имела свой уникальный идентификатор, позволяющий отличать один программный продукт от другого. При этом программное обеспечение может включать в себя несколько отдельных файлов, которые так же должны иметь свой уникальный идентификатор.

Каждая программа храниться в отдельном каталоге, имя которого состоит из уникального идентификатора. Внутри каталога, принадлежащего программному обеспечению, предусмотрены подкаталоги с версиями файлов с именами, которые образуются из уникального идентификатора файла и его версии, как изображено на рисунке 4.2.

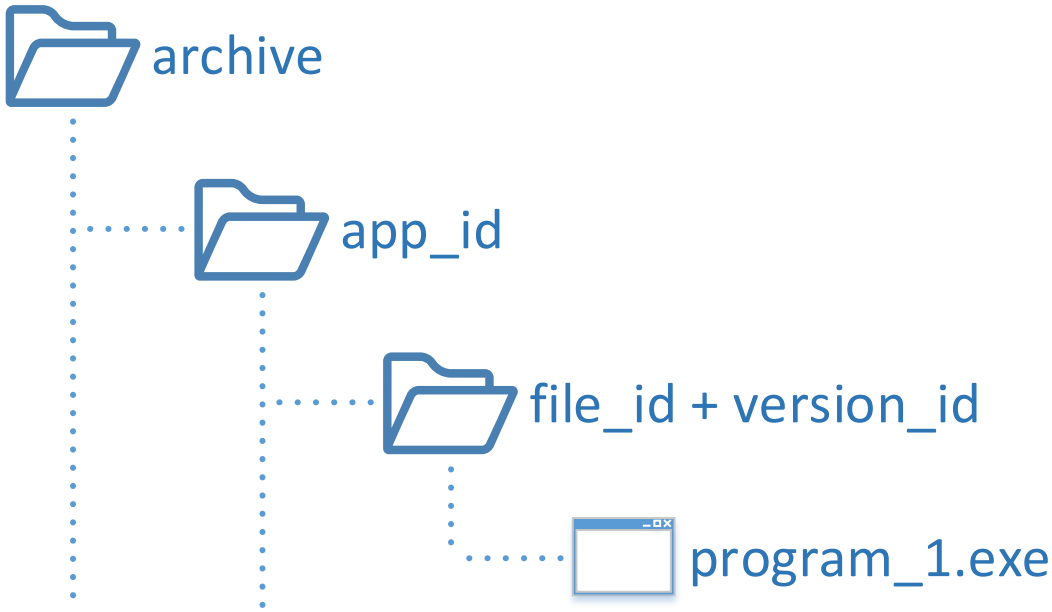

Рисунок 4.2 – Структура файлового хранилища на центральном сервере

Подобная иерархия позволит хранить в файловой системе файлы с одинаковыми именами, принадлежащие всевозможным программам, и при этом имеющие разные версии.

Так же были созданы таблицы необходимые для хранения информации о файлах, их описания, версий и т.д., как показано на рисунке 4.3.

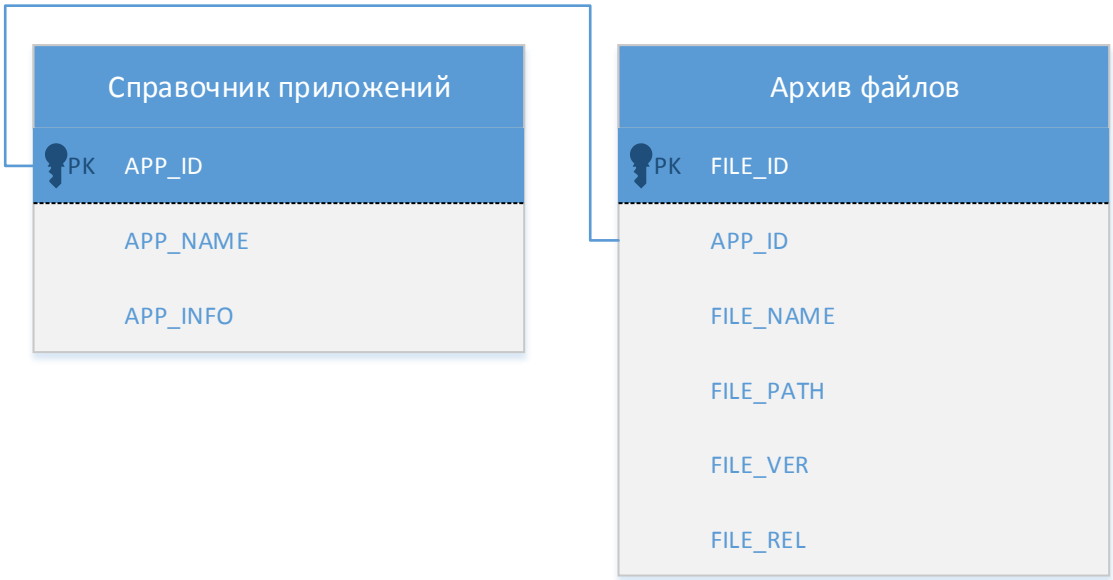

Рисунок 4.3 – Сема БД версий программного обеспечения

Таблица «Справочник приложений» служит для хранения информации о различных приложениях и имеет поля:

- APP_ID – уникальный идентификатор приложения;
- APP_NAME – имя программного обеспечения;
- APP_INFO – информация о программном обеспечении.

Таблица «Архив файлов» необходима для хранения информации о файлах разных версий и имеет поля:

- FILE_ID – уникальный идентификатор файла;
- APP_ID – уникальный идентификатор приложения, которому принадлежит этот файл;
- FILE_NAME – имя файла;
- FILE_PATH – относительный путь к файлу;
- FILE_VER – версия файла;
- FILE_REL – актуальность файла.

Чтобы получить доступ к файлам, находящимся на файловом хранилище, центральному серверу необходимо знать абсолютные (полные) пути к ним. В семействе операционных систем Microsoft Windows абсолютный путь к файлу имеет вид [буква диска:]\[относительный путь к файлу]\[имя файла]. Чтобы получить доступ к файлу центральный сервер конструирует абсолютный путь к файлу следующим образом. Центральный сервер считывает из своих настроек конфигурации букву диска, на котором хранится архив версий программного обеспечения. Зная уникальный идентификатор файла, сервер узнает из таблицы «Архив файлов» относительный путь и его имя. После чего происходит конкатенация полученных данных. В результате получиться путь, например D:\archive\001\001_3511\program1.exe.

4.2 Механизм обновления программного обеспечения

На коммуникационном сервере для хранения актуальных версий будет организован репозиторий, который в отличие от хранилища на центральном сервере способен содержать только актуальные версии программного обеспечения клиентских рабочих станций. Структура файлового хранилища на коммуникационном сервере изображена на рисунке 4.4.

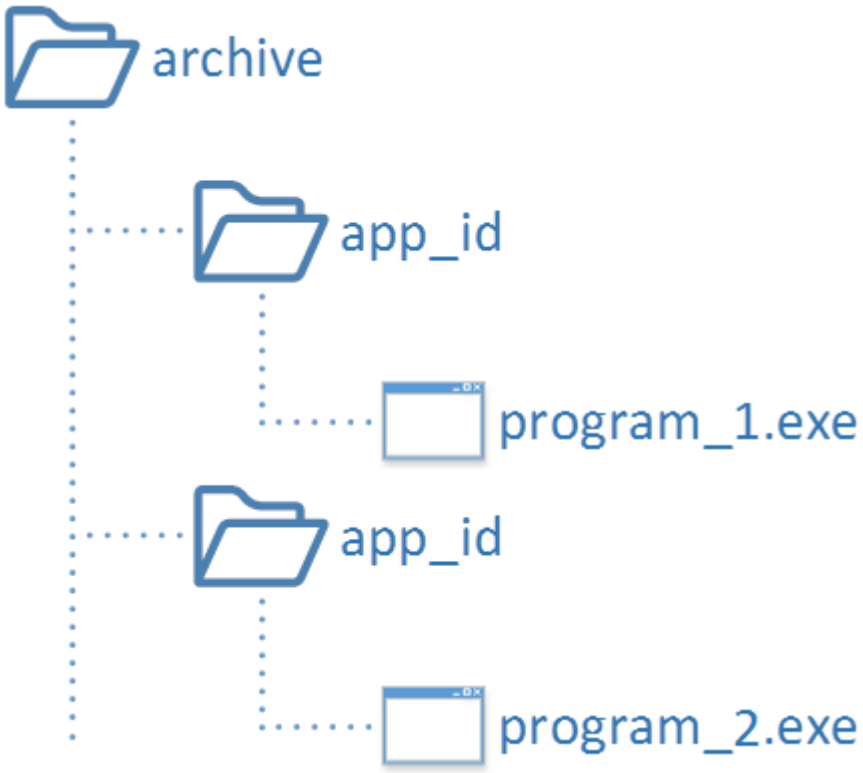

Рисунок 4.4 – Структура файлового хранилища на коммуникационном сервере

Подобная иерархия позволит хранить в файловой системе файлы, принадлежащие конкретной программе в своей отдельной папке.

В обязанности программного обеспечения на коммуникационном сервере входит не только обновление самого себя, но так же и актуализация всех файлов в своем репозитории, которые необходимы для работы программ на клиентских рабочих станциях.

Процедура обновления на коммуникационном сервере в штатном режиме будет выполняться в течение не рабочего времени по регламенту. В случае неудавшейся попытки обновления приложение предпримет еще несколько попыток обновления. Если программное обеспечение так и не сможет обновиться, совершать процедуру обновления придется системному администратору вручную после того, как будут решены проблемы связи с центральным сервером. Время, когда именно программное обеспечение будет совершать попытку обновления и количество его попыток указано в его настройках.

Функция обновления программного обеспечения коммуникационного сервера основывается на списках и инструкциях. В рабочей папке программного обеспечения коммуникационного сервера храниться список в виде отдельного файла. Список содержит информацию о файлах необходимых для работы программного обеспечения коммуникационного сервера и клиентских рабочих станций. Структура списка файлов изображена на рисунке 4.4.

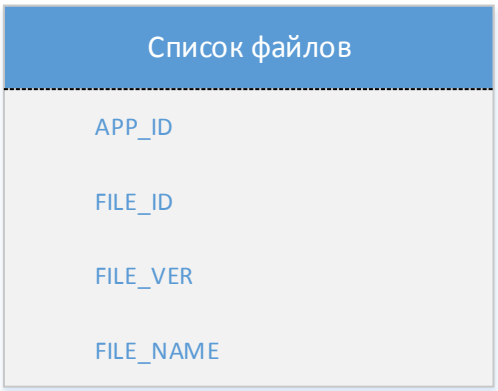

Рисунок 4.5 – Структура списка файлов

Структура списка файлов состоит из полей данных:
- APP_ID – уникальный идентификатор программы;
- FILE_ID – уникальный идентификатор файла;
- FILE_VER – версия файла;
- FILE_NAME – имя файла.

При совершении попытки обновления приложение посылает центральному серверу имеющийся список, содержащий уникальные идентификаторы файлов file_id и их версий. Центральный сервер, зная уникальный идентификатор программы, выполняет запрос к базе данных для получения актуального списка файлов, которые необходимы для работы программного обеспечения. Затем сервер сравнивает между собой присланный список файлов и актуальный. На основании сравнения списков, центральный сервер формирует новый список, содержащий информацию о файлах и инструкциях к ним. Структура списка файлов с инструкциями показано на рисунке 4.6.

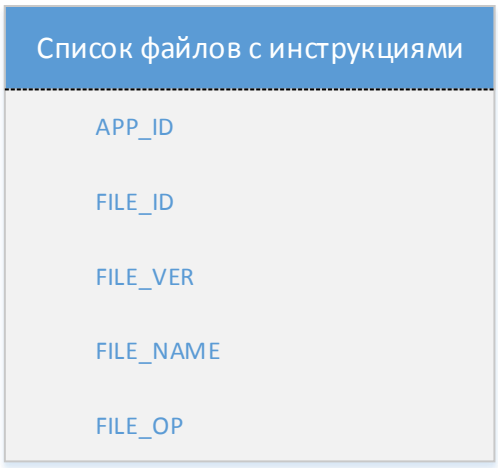

Рисунок 4.6 – Структура списка файлов с инструкциями

Структура списка файлов с инструкциями состоит из полей данных:

- APP_ID – уникальный идентификатор программы;
- FILE_ID – уникальный идентификатор файла;
- FILE_VER – версия файла;
- FILE_NAME – имя файла;
- FILE_OP – тип операции, совершаемой над файлом.

Инструкции к файлам формируются следующим образом:

- если при сравнении списков был обнаружен лишний файл, то в инструкции будет указано о необходимости его удаления;
- если при сравнении списков был обнаружен файл неактуальной версии, то в инструкции будет указано о необходимости его перезаписи;
- если при сравнении списков был обнаружен новый файл, то в инструкции будет указано о необходимости его создания.

В случае успешного формирования списка файлов центральный сервер отправляет его обратно коммуникационному серверу. В свою очередь, приложение на коммуникационном сервере начинает по очереди загружать с центрального сервера файлы, которые были указаны в полученном списке файлов инструкциями, отправляя ему, уникальный идентификатор file_id. Следующие шаги зависят от того файлы какого именно программного обеспечения были указаны в списке с инструкциями.

Программное обеспечение коммуникационного сервера всегда знает свой уникальный идентификатор. Благодаря этому оно способно отличать свои файлы от файлов клиентского программного обеспечения.

Если в списке встретился файл, относящийся к клиентскому программному обеспечению, тогда над ним будет совершена соответствующая операция, которая была указана в инструкции.

Если в списке встретился файл для самого приложения коммуникационного сервера, тогда он будет загружаться не на файловое хранилище актуального программного обеспечения, а в другую отдельную временную папку, расположенную в рабочем каталоге. После того, как были

загружены все его файлы, будет совершена процедура его самообновления, принцип которой заключается в следующем.

Программное обеспечение коммуникационного сервера на основании списка с инструкциями создает другой список в виде структурированного файла. В нем будет храниться информация, говорящая о том какой именно файл необходимо создать, перезаписать или изменить в рабочей папке приложения коммуникационного сервера.

Список инструкций необходим для самообновления с помощью разработанной вспомогательной программы (апдейтера). Принцип самообновления программного обеспечения показан на рисунке 4.7.

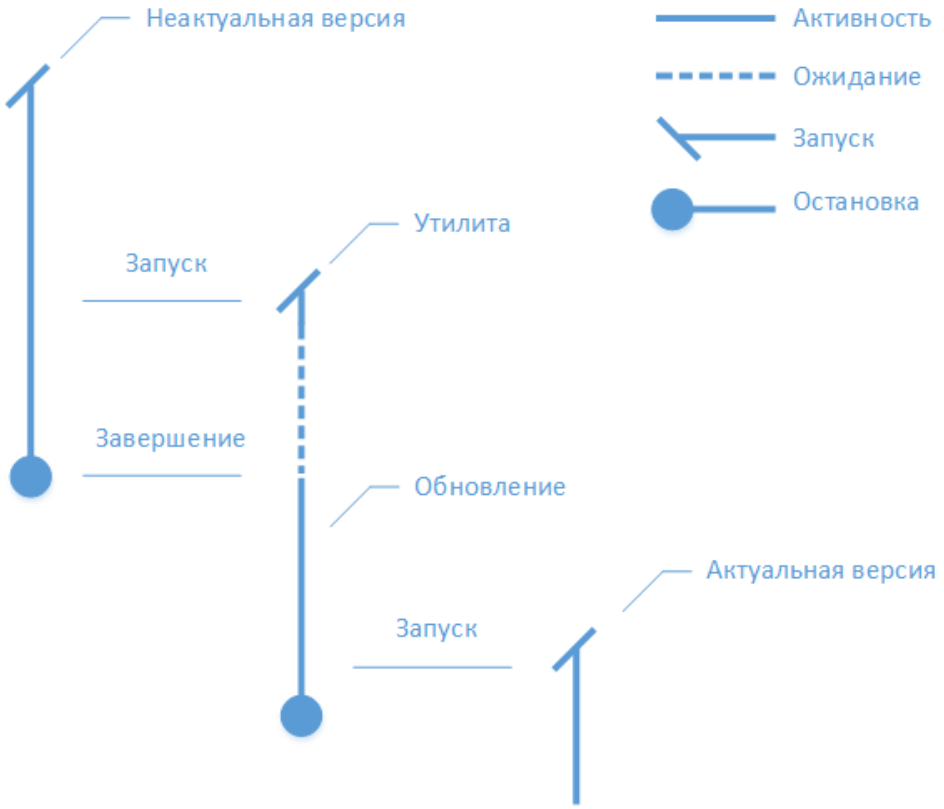

Рисунок 4.7 – Диаграмма состояния потоков

Основная техническая проблема, которую должна обходить процедура самообновления заключается в том, что невозможно каким-либо образом удалить, изменить, перезаписать исполняемый файл уже запущенной программы.

Разработанный вариант процедуры самообновления опирается на принципы межпроцессного взаимодействия и работает по следующему алгоритму. После того как был сформирован список инструкций, программное обеспечение запускает апдейтер, передавая ему в качестве параметра уникальный идентификатор своего процесса (PID) и после чего закрывается. Утилита, получив идентификатор процесса, с помощью функций синхронизации ожидает его завершения. После чего, на основании списка

инструкций она заменяет неактуальные файлы программного обеспечения на актуальные. После замены файлов утилита обновляет список файлов на коммуникационном сервере и запускает уже актуальное программное обеспечение и закрывается. Таким образом, будет выполнено самообновление программного обеспечения.

Алгоритм процедуры обновления программного обеспечения на коммуникационном сервере изображен в виде блок-схемы на рисунке 4.8.

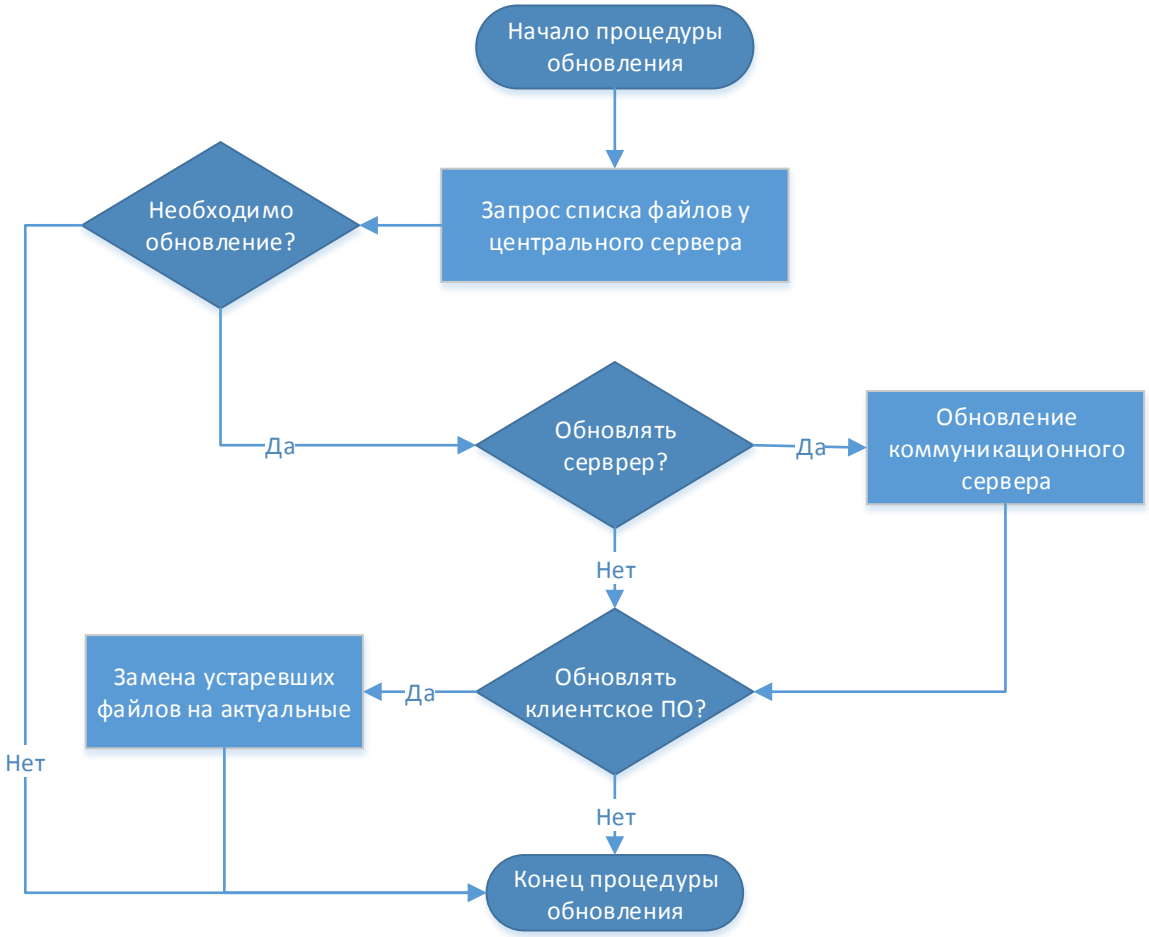

Рисунок 4.8 – Блок схема алгоритма обновления коммуникационного сервера

Программное обеспечение клиентских рабочих станциях обновляется в начале рабочего дня с помощью вспомогательной программы (лаунчера), которая запускается при старте ОС. Программа имеет информацию обо всех установленных программах на клиентских рабочих станциях, являющихся частью ИС «РП ДРН». Информация записана в виде отдельного файла в рабочей папке лаунчера и имеет структуру, которая показана на рисунке 4.9.

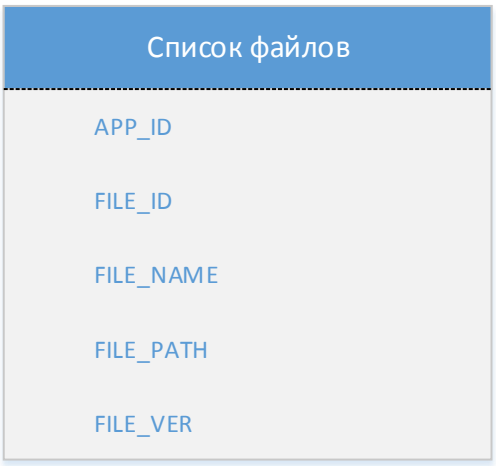

Рисунок 4.9 – Структура списка файлов на клиентских рабочих станциях

Структура списка файлов на клиентских рабочих станциях состоит из полей данных:

- APP_ID – уникальный идентификатор программы;
- FILE_ID – уникальный идентификатор файла;
- FILE_VER – версия файла;
- FILE_PATH – путь к файлу;
- FILE_NAME – имя файла.

Механизм обновления клиентского программного обеспечения схож с коммуникационным сервером. Блок-схема алгоритма обновления клиентской рабочей станции изображена на рисунке 4.10.

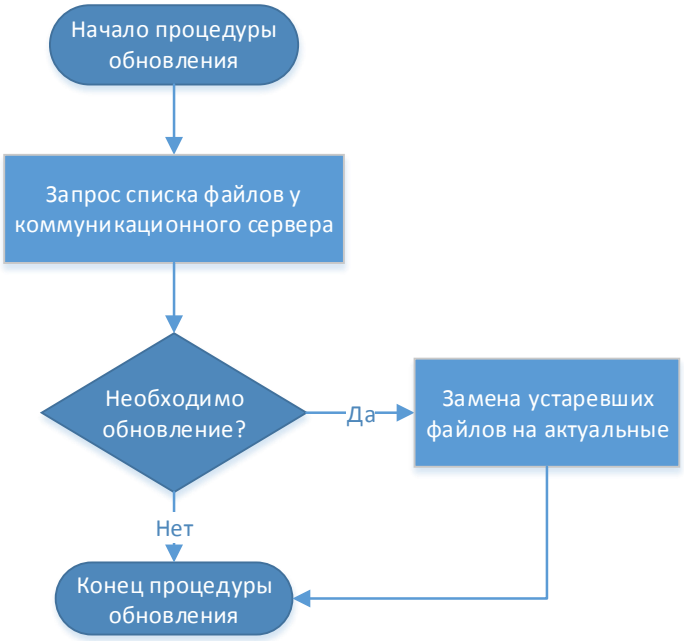

Рисунок 4.10 – Блок-схема алгоритма обновления клиентской рабочей станции.

При обновлении лаунчер посылает коммуникационному серверу свой список файлов и затем получает новый с инструкциями. Затем лаунчер, зная путь, куда записать файл по file_path, копирует необходимые файлы по локальной сети, обновляет свой список файлов и запускает уже обновленное программное обеспечение.

Таким образом, будет выполнено обновление программного обеспечения на клиентских рабочих станциях и коммуникационных серверах.

Выводы к четвертому разделу

Описанная система автоматизированного обновления программного обеспечения разработана в рамках модернизации ИС «РП ДРН». Процедура обновления программного обеспечения будет выполняться централизовано и учитывает ключевые особенности архитектуры информационной системы.

Автоматизация процессов обновления позволит снизить человеческий фактор и повысить гибкость и легкость в дальнейшем развитии и модернизациях, что, в свою очередь, позволит снизить очереди и недовольство граждан в ЦОН-ах.

ЗАКЛЮЧЕНИЕ

В диссертации решена задача, связанная с разработкой системы автоматического обновления программного обеспечения ИС «РП ДРН». Проведено исследование особенностей распределенных информационных систем, методов их построения, а так же проблем обновления ПО в распределенных системах использующих архитектуру «толстый клиент» на примере ИС «РП ДРН». И конечно была предложено возможное решение проблемы обновления ПО в ИС «РП ДРН».

Краткие выводы по результатам диссертационного исследования заключаются в следующем:

1. Проведено исследование особенностей распределенных информационных систем, методов их построения. Было дано определение понятию информационных систем и рассмотрены основные процессы, обеспечивающие работу информационной системы.
2. Были рассмотрены основные концепции, касающиеся технологии «клиент-сервер». Изучены основные методы построения информационных систем на базе технологии «клиент-сервер», в частности, были рассмотрены двухуровневые архитектуры «тонкий клиент» и «толстый клиент».
3. Проведен сравнительный анализ различных моделей технологии «клиент-сервер», такие как FS-модель, RDA-модель, DBS-модель и AS-модель.
4. Исследованы способы решения проблем актуализации программного обеспечения в информационных системах.
5. Проведен анализ информационной системы «Регистрационный пункт документирования и регистрации населения»
6. Разработана система автоматического обновления программного обеспечения, которая может решить проблему актуализации в информационной системе «Регистрационный пункт документирования и регистрации населения»

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Громов Г.Р. Очерки информационной технологии. – Москва: ИнфоАрт, 1992.
2 Емельянова Н.З. Основы построения автоматизированных информационных систем. – Москва: ИНФРА-М, 2005.
3 Мезенцев К. Н. Автоматизированные информационные системы. – Москва: Академия, 2010.
4 Голицына О.Л., Максимов Н.В. Информационные системы – Москва: ММИЭИФП, 2004.
5 Алешин Л.И., Максимов Н.В. Информационные технологии – Москва: ММИЭИФП, 2004.
6 Когаловский М. Р. Перспективные технологии информационных систем. — Москва: ДМК Пресс; Компания АйТи, 2003.
7 Васкевич Д. Стратегии клиент/сервер. – Киев: Диалектика, 1997.
8 Информационные технологии: учебник/под редакцией проф. В.В.Трофимова. – М.: Юрайт, 2011.
9 Коржов В. Многоуровневые системы клиент-сервер – Сети/network world № 06, 1997.
10 Популярный IT портал http://habrahabr.ru/.
11 Новостной сайт http://www.infoq.com/articles/dvcs-guide.
12 Блог Lucas Ward http://www.lucasward.net/2010/02/maturity-model-for-source-control-scmm.html.