МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт Факультет приборостроения, информационных технологий и электроники	Кафедра «Информационная безопасность систем и технологий»
Специальность 10.05.03 –	Информационная безопасность автоматизированных систем
Специализация –	Защищенные автоматизированные системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ЭКСПЕРТНАЯ СИСТЕМА ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ПО УПРАВЛЕНИЮ ИНЦИДЕНТАМИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Студент _____ Первак Юлия Константиновна

Руководитель _____ Зефиров С.Л.

Нормоконтролёр _____ Фатеев А.Г.

Рецензент зам. ген. директора по специальным вопросам АО «ПНИЭИ» _____ Сомкин С.А.

Работа допущена к защите (протокол заседания кафедры от 21.06.19г. № 16)

Заведующий кафедрой _____ С.Л. Зефиров

Работа защищена с отметкой отлично (протокол заседания ГЭК от 25.06.19г. № 23)

Секретарь ГЭК _____ А.П. Иванов

Пенза 2019

Проверка (предзащита) 10 баллов. 21.06.19

Оглавление

Введение 4
1 Обзор процессов управления инцидентами информационной безопасности 6
2 Моделирование процессов управления инцидентами информационной безопасности в рамках экспертной системы 11
2.1 Требования к модели среды управления инцидентами информационной безопасности 11
2.2 Разработка модели среды управления инцидентами информационной безопасности 13
2.3 Разработка модели экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности 18
3 Разработка и проектирование архитектуры программного модуля, реализующего модель среды управления инцидентами информационной безопасности 25
4 Реализация модели среды управления инцидентами информационной безопасности в виде программного модуля 31
5 Разработка и проектирование архитектуры экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности 37
6 Реализация экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности в виде клиент-серверного приложения 47
6.1 Реализация серверного компонента 47
6.2 Реализация клиентского компонента 53
7 Вредные психофизиологические факторы, влияющие на органы зрения специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие 58
Заключение 61

Список используемых источников 63
Приложение А Исходный код класса ContextGen 64
Приложение Б Исходный код приложения ITestServer 83
Приложение В Исходный код приложения ITestClient 123

Введение

Основной целью обеспечения информационной безопасности (ИБ) организации является снижение рисков, действующих в отношении информационных ресурсов, и в конечном счете предотвращение или минимизация ущерба от возможных инцидентов ИБ. Управление инцидентами ИБ направлено на совершенствование подсистемы обеспечения безопасности компании.

Эффективное применение средств защиты информации основывается в том числе на выявлении, предотвращении, регистрации и устранении последствий инцидентов информационной безопасности или событий, нарушающих регламентированные процедуры защиты ИБ. События или инциденты ИБ должны быть выявлены и зарегистрированы, их последствия устранены, а на основании анализа причин их возникновения должны быть доработаны параметры процедур управления инцидентами ИБ.

При подготовке специалистов по информационной безопасности автоматизированных систем предусмотрено изучение управления инцидентами информационной безопасности защищенных автоматизированных систем, обеспечивающее формирование компетенции «способность обеспечить эффективное применение средств защиты информационно-технологических ресурсов автоматизированной системы и восстановление их работоспособности при возникновении нештатных ситуаций». Освоение этой компетенции предполагает овладение навыками обнаружения и идентификации инцидентов в процессе эксплуатации автоматизированной системы и навыками устранения инцидентов, возникших в процессе эксплуатации автоматизированной системы. При освоении указанной компетенции может быть использована экспертная система поддержки принятия решений по управлению инцидентами информационной безопасности.

Все процессы управления инцидентами информационной безопасности могут быть осуществлены построением ассоциаций между начальными данными

и действиями по управлению инцидентами, которые необходимо предпринять. Согласно Питеру Джексону, проблемы, которые могут быть решены перечислением ассоциативных связей между обозримыми совокупностями данных и классами событий, подходят для экспертных систем [1]. Это обосновывает возможность использования экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности.

Преимущество использования экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности в процессе обучения специалиста по защите информации в том, что она существенно упрощает процесс проверки остаточных знаний. Экспертная система позволяет с минимальными затратами человеческих ресурсов многократно повторять на практике процессы управления инцидентами информационной безопасности с различными исходными данными, что способствует их изучению и запоминанию правил их выполнения. Этим обуславливается актуальность выпускной квалификационной работы.

Таким образом, целью выпускной квалификационной работы является разработка экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности. При этом основное назначение такой экспертной системы заключается в ее использовании в учебном процессе при обучении специалистов по информационной безопасности управлению инцидентами ИБ.

1 Обзор процессов управления инцидентами информационной безопасности

Одним из основных понятий в сфере управления инцидентами информационной безопасности является событие информационной безопасности. Событие ИБ представляет собой идентифицированное возникновение состояния системы, услуги или сети, указывающее на возможное нарушение политики информационной безопасности, отказ защитных мер, а также возникновение ранее неизвестной ситуации, которая может быть связана с безопасностью [2].

При этом инцидент информационной безопасности определяется как Появление одного или нескольких нежелательных или неожиданных событий ИБ, с которыми связана значительная вероятность компрометации бизнес-операций и создания угрозы ИБ [3].

Всего определены пять видов инцидентов информационной безопасности:

— инциденты неавторизованного доступа;

— инциденты отказа в обслуживании;

— инциденты внедрения вредоносного кода;

— инциденты сбора информации;

— инциденты несоответствующего использования.

Управление инцидентами информационной безопасности осуществляется в три этапа:

— обнаружение событий информационной безопасности и оповещение о событиях информационной безопасности;

— анализ инцидента информационной безопасности;

— реагирование на инцидент информационной безопасности.

Обнаружение событий ИБ осуществляется с помощью программных средств, таких как средства обнаружения вторжений, антивирусное программное обеспечение, а также на основе записей системы мониторинга или при помощи сотрудников объекта на основе признаков инцидентов ИБ. Перечень различных

признаков инцидентов ИБ определяется заранее для каждого вида инцидентов ИБ.

Оповещение о событиях ИБ осуществляется сотрудниками объекта. Сотрудник, обнаруживший признаки инцидента ИБ, должен сообщить о случившемся оператору группы реагирования на инциденты ИБ в форме отчета о событии ИБ.

В отчете о событии ИБ должны содержаться сведения о дате и времени обнаружения события ИБ, дате и времени оповещения о событии ИБ, о лице, сообщившем об указанном событии, а также общее описание события: что произошло, как и почему. При достаточной компетентности сотрудника, обнаружившего признаки инцидента ИБ, он также может включить в отчет о событии ИБ перечень затронутых инцидентом ИБ активов и выявленных уязвимостей объекта. В отчете о событии ИБ сотрудник объекта должен изложить как можно больше информации, доступной ему в момент обнаружения события ИБ.

Этап анализа инцидента ИБ включает процессы первичной оценки события ИБ и вторичной оценки инцидента ИБ.

Первичная оценка события ИБ проводится с целью определения реальности инцидента ИБ на основании информации, полученной от сотрудника, обнаружившего признаки инцидента ИБ. Оператор группы реагирования на инциденты ИБ фиксирует получение от указанного сотрудника отчета о событии ИБ и осуществляет сбор дополнительной информации, которая может способствовать определению реальности инцидента ИБ. Затем оператор оценивает, является ли описанное событие ИБ инцидентом ИБ.

Если событие ИБ принимается оператором как отклонение от штатной ситуации или предупреждение о нем, не являющееся инцидентом ИБ, то оператором должен быть дополнен отчет о событии ИБ с указанием всех проведенных мероприятий и обоснованием того, что указанное событие не является инцидентом ИБ.

В случае, если событие ИБ определено оператором как инцидент ИБ, им должен быть составлен отчет об инциденте ИБ. Указанный отчет должен включать сведения о дате и времени обнаружения инцидента ИБ и оповещения о нем, о сотруднике, сообщившем об инциденте ИБ, а также общее описание инцидента ИБ: что произошло, как и почему. Также в отчет об инциденте ИБ должно быть включено описание реального или потенциального воздействия инцидента ИБ на деятельность объекта, затронутых инцидентом активов, предполагаемой значимости инцидента ИБ.

Вторичная оценка инцидента ИБ проводится аналитиками группы реагирования на инциденты ИБ с целью подтверждения реальности инцидента ИБ, а также определения действий по реагированию на него.

Аналитик группы реагирования на инциденты ИБ должен зафиксировать получение отчета об инциденте ИБ и проанализировать его содержание. Для проведения вторичной оценки аналитик осуществляет сбор дополнительной информации, способствующей проведению оценки, из всех доступных ему источников.

Если по результатам вторичного анализа реальность рассматриваемого инцидента ИБ не подтверждена, и он признан отклонением от штатной ситуации, не являющимся инцидентом ИБ, должен быть составлен новый отчет о событии ИБ.

В случае, если реальность рассматриваемого инцидента ИБ подтверждена аналитиком, для этого инцидента также должны быть определены его вид, уточнен перечень затронутых этих инцидентом активов, а также подготовлена дополнительная информация, способствующая определению последующих действий по реагированию на этот инцидент ИБ.

Реагирование на инцидент ИБ имеет своей целью ограничение возможного негативного воздействия инцидента на бизнес-процессы объекта, а также улучшение системы ИБ. Реагирование на инцидент в общем случае включает следующие процессы:

— сдерживание инцидента ИБ;

— устранение инцидента ИБ;

— восстановление после инцидента ИБ;

— формирование и хранение свидетельств инцидента ИБ.

Сдерживание инцидента ИБ осуществляется с целью предотвращения распространения инцидента на участки и компоненты системы, не затронутые им изначально, что способствует уменьшению ущерба, который будет нанесен инцидентом объекту.

При сдерживании инцидента ИБ группа реагирования на инциденты ИБ принимает решение о выборе стратегии сдерживания. Набор таких стратегий и процедур сдерживания должен быть заранее продуман и определен для каждого вида инцидентов.

По завершении сдерживания инцидента ИБ производится подтверждение факта нахождения инцидента под контролем. В случае, если принятые меры и процедуры сдерживания оказались достаточно эффективны и инцидент ИБ после их применения находится под контролем, необходимо переходить к следующему процессу – устранение инцидента ИБ.

В противном случае должны быть предприняты антикризисные действия. В частности, должен выполняться план обеспечения непрерывности работы объекта, учитывающий приоритеты деятельности объекта и ориентированный на полное восстановление его деятельности.

Процесс устранения инцидента ИБ заключается в устранении компонентов этого инцидента, удалении из системы объекта элементов, использованных инцидентом. В частности, в рамках устранения инцидента ИБ может быть осуществлено удаление вредоносного кода, блокирование скомпрометированных учетных записей, удаление нежелательных материалов.

Восстановление после инцидента осуществляется после завершения его устранения и заключается в устранении последствий инцидента ИБ и подготовке объекта и всех его компонентов к работе в штатном режиме. Работоспособность системы, в частности, может быть достигнута с помощью восстановления из резервных копий программного обеспечения или баз данных, запуска

компонентов системы из доверенных состояний, изменения аутентификационной информации учетных записей.

По завершении реагирования на инцидент ИБ должны быть сформированы свидетельства инцидента ИБ. В рамах формирования таких свидетельств информация в отчете об инциденте ИБ обновляется с учетом всех действий по реагированию на инцидент и их результатов, а также итоговой оценки реального воздействия инцидента на объект. Также должны быть собраны статистические данные об инциденте и его обработке, включая информацию о ресурсах, затраченных на его обработку. Указанные свидетельства должны быть сохранены в течение срока, определяемого на основе индивидуальных параметров инцидента и установленных в рамках объекта норм.

2 Моделирование процессов управления инцидентами информационной безопасности в рамках экспертной системы

2.1 Требования к модели среды управления инцидентами информационной безопасности

Среда управления инцидентами информационной безопасности включает в себя элементы структуры и процессов реальной системы объекта, в рамках которых выполняются процессы управления инцидентами.

Модель такой среды управления инцидентами ИБ должна в том или ином виде воспроизводить или имитировать операции, связанные с деятельностью объекта. Такие операции могут быть отражены посредством отображения результатов работы компонентов системы реального объекта, а в частности, программного обеспечения и средств защиты информации.

Деятельность объекта может быть представлена некоторым генератором случайных событий ИБ. При этом в том случае, если сам объект и его информационные процессы являются в достаточной степени изученными, набор таких возможных событий ИБ может быть составлен заранее.

События ИБ корректнее всего представлять в виде признаков инцидента ИБ, поскольку процесс обнаружения событий ИБ основан именно на наличии у события признаков инцидентов. Для моделирования оповещения о событии ИБ, осуществляемого сотрудниками объекта, некоторые события ИБ могут быть представлены в виде отчетных форм «Отчет о событии ИБ».

Среда управления инцидентами ИБ включает также совокупность данных об объекте, его характеристиках и особенностях его функционирования. Эти данные используются участниками группы реагирования на инциденты ИБ на этапе анализа инцидента ИБ при проведении первичной оценки события ИБ и вторичной оценки инцидента ИБ.

Модель среды управления инцидентами ИБ должна реализовывать эту особенность среды. Для этого при построении модели может быть использован набор заранее определенных фактов об объекте и его бизнес-процессах. В таком

случае при осуществлении анализа инцидента ИБ пользователю, исполняющему роль оператора или аналитика группы реагирования на инциденты ИБ, по его запросу должен быть предоставлен определенный набор сведений, который он сможет использовать для получения наиболее точных результатов первичной и вторичной оценки.

В процессе реагирования на инцидент ИБ должны быть приняты меры по его сдерживанию, устранению и восстановлению после него. Соответственно, модель среды управления инцидентами ИБ должна предоставлять возможность выбора защитных мер в каждом из указанных процессов реагирования на инцидент. Модель может включать фиксированный набор различных мер при условии достаточной обширности такого набора. При этом модель должна предусматривать возможность оценки эффективности выбранных мер сдерживания, устранения и восстановления.

Модель среды управления инцидентами ИБ должна поддерживать возможность реализации процесса формирования свидетельств инцидента. Для обеспечения поддержки такой возможностью может быть использовано подробное протоколирование всех сгенерированных компонентами модели обстоятельств и принятых пользователем решений за одну итерацию моделирования полной последовательности процессов управления инцидентами ИБ. В том числе пользователю может быть предоставлена возможность заполнения соответствующих отчетных форм отчета о событии ИБ и отчета об инциденте ИБ.

Таким образом, модель среды управления инцидентами ИБ должна соответствовать следующим требованиям:

— обеспечение возможности воспроизведения или имитации моделью операций, связанных с деятельностью объекта;

— обеспечение возможности получения дополнительных данных об объекте, его характеристиках и особенностях его функционирования;

— обеспечение возможности выбора защитных мер из заданного набора в рамках сдерживания инцидента, его устранения и восстановления после него;

— обеспечение возможности оценки эффективности выбранных мер сдерживания инцидента, его устранения и восстановления после него;

— обеспечение возможности формирования свидетельств инцидента.

2.2 Разработка модели среды управления инцидентами информационной безопасности

При разработке модели среды управления инцидентами информационной безопасности необходимо обеспечить выполнение всех требований к такой модели.

Для обеспечения возможности воспроизведения или имитации моделью операций, связанных с деятельностью объекта может быть использован эмулятор событий ИБ. Сами же события ИБ могут быть представлены в виде базы данных событий, в которой они представлены совокупностью признаков инцидентов, либо отчетами о событиях ИБ.

В этом случае главной задачей эмулятора событий ИБ будет корректное оповещение наблюдателя о событии ИБ, случайно выбранном из базы данных, то есть отображение заданных для него признаков или отчетных форм.

С целью обеспечения возможности получения дополнительных данных об объекте, его характеристиках и особенностях его функционирования, набор заранее определенных фактов об объекте и его бизнес-процессах может быть представлен в виде базы данных, в которой такие факты представлены в виде строго формализованных записей БД, либо же в форме произвольного описания на естественном языке. При этом при осуществлении анализа инцидента ИБ пользователю, исполняющему роль оператора или аналитика группы реагирования на инциденты ИБ, должен быть предоставлен перечень категорий

данных, которые он может получить в контексте рассматриваемого события или инцидента ИБ.

Для обеспечения возможности выбора защитных мер из заданного набора в рамках реагирования на инцидент ИБ может быть использован фиксированный набор различных мер, представленный в виде базы данных. При осуществлении процессов сдерживания инцидента, устранения и восстановления после него пользователю, исполняющему роль аналитика группы реагирования на инциденты ИБ, может быть предоставлен перечень всех возможных защитных мер, доступных для конкретного процесса в контексте рассматриваемого инцидента ИБ.

Для обеспечения возможности оценки эффективности выбранных мер сдерживания инцидента, его устранения и восстановления после него может быть использован эталонный набор мер для каждого из указанных процессов. Также может быть применена весовая оценка защитных мер, когда каждой из доступных защитных мер присваивается ее вес, соответствующей ее эффективности, а для каждого инцидента устанавливается порог суммы весов защитных мер, принятых в каждом процессе по отношению к нему, при достижении которого совокупность выбранных защитных мер может быть принята достаточно эффективной.

В целях обеспечения возможности формирования свидетельств инцидента может быть использовано журналирование всех действий, выполненных с использованием модели. Также могут быть представлены к заполнению пользователем отчетные формы отчета о событии ИБ и отчета об инциденте ИБ, а сформированные пользователями отчеты могут быть сохранены в базе данных.

При таком моделировании модель среды управления инцидентами ИБ может быть описана диаграммой вариантов использования, приведенной на рисунке 1.

В соответствии с этапами управления инцидентами ИБ для модели среды управления инцидентами ИБ можно выделить следующие режимы функционирования:

— режим «Обнаружение и первичная оценка события ИБ»;

— режим «Вторичная оценка инцидента ИБ»;

— режим «Реагирование на инцидент ИБ».

Рисунок 1 – Диаграмма вариантов использования модели среды управления инцидентами ИБ

В этом случае при группировке действий, доступных в каждом отдельном режиме и направленных на достижение единой цели, диаграмма вариантов использования может быть обобщена до представленной на рисунке 2.

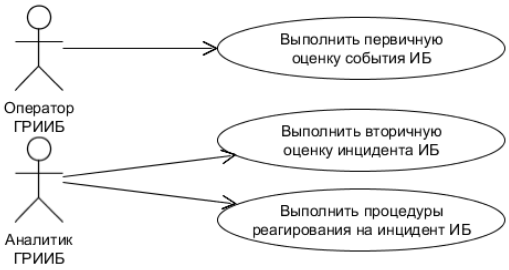

Рисунок 2 – Обобщенная диаграмма вариантов использования модели среды управления инцидентами ИБ

В режиме «Обнаружение и первичная оценка события ИБ» происходит моделирование процессов обнаружения события ИБ, оповещения о нем пользователя, исполняющего роль оператора группы реагирования на инциденты ИБ, а также осуществления этим пользователем первичной оценки события ИБ. Диаграмма вариантов использования для этого режима функционирования модели приведена на рисунке 3.

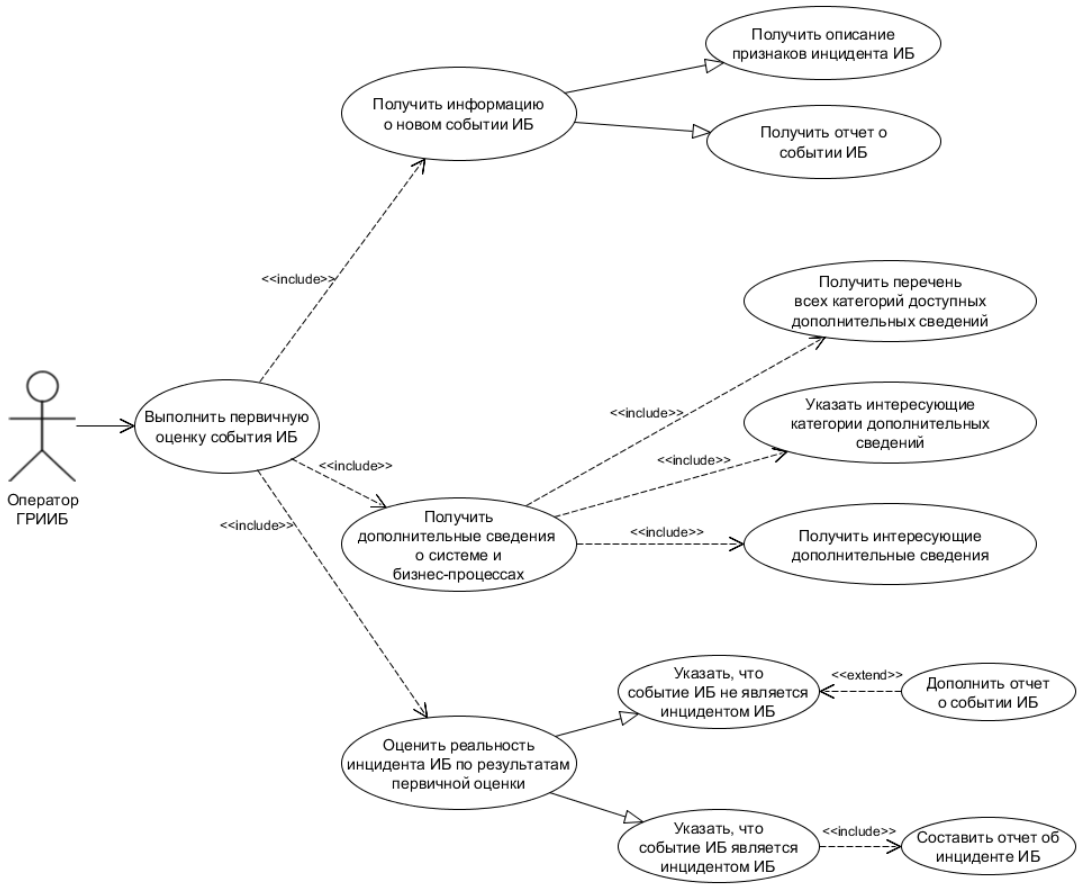

Рисунок 3 – Диаграмма вариантов использования модели среды управления инцидентами ИБ в режиме функционирования «Обнаружение и первичная оценка события ИБ»

В режиме «Вторичная оценка инцидента ИБ» происходит моделирование процесса вторичной оценки инцидента ИБ пользователем, исполняющим роль аналитика группы реагирования на инциденты ИБ. Диаграмма вариантов использования для этого режима функционирования модели приведена на рисунке 4.

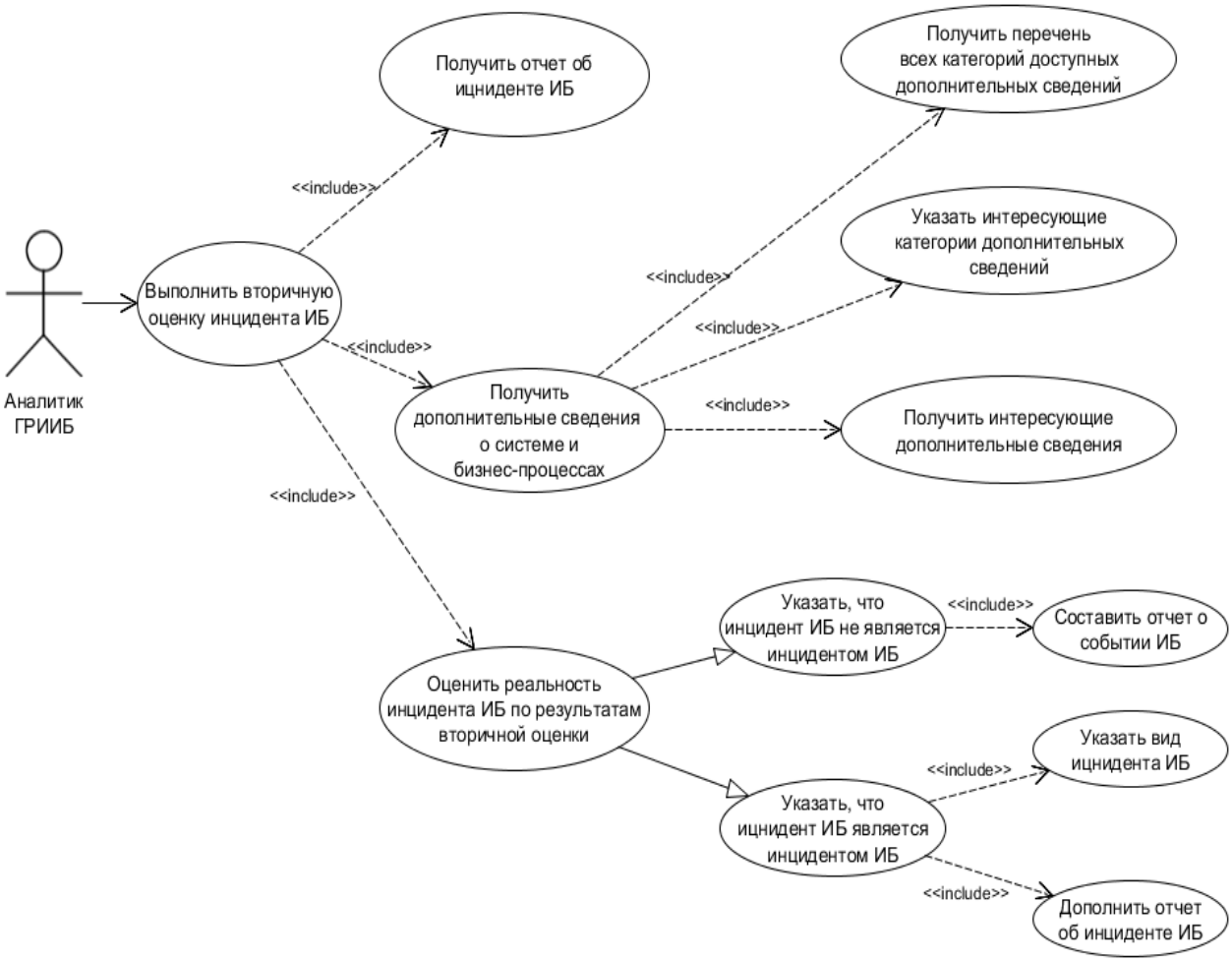

Рисунок 4 – Диаграмма вариантов использования модели среды управления инцидентами ИБ в режиме функционирования «Вторичная оценка инцидента ИБ»

В режиме «Реагирование на инцидент ИБ» происходит моделирование процессов сдерживания инцидента ИБ пользователем, исполняющим роль аналитика группы реагирования на инциденты ИБ, устранения инцидента и восстановления после него. Этот режим функционирования модели подразумевают выбор указанным пользователем защитных мер в каждом из

названных процессов. Диаграмма вариантов использования для этого режима функционирования модели приведена на рисунке 5.

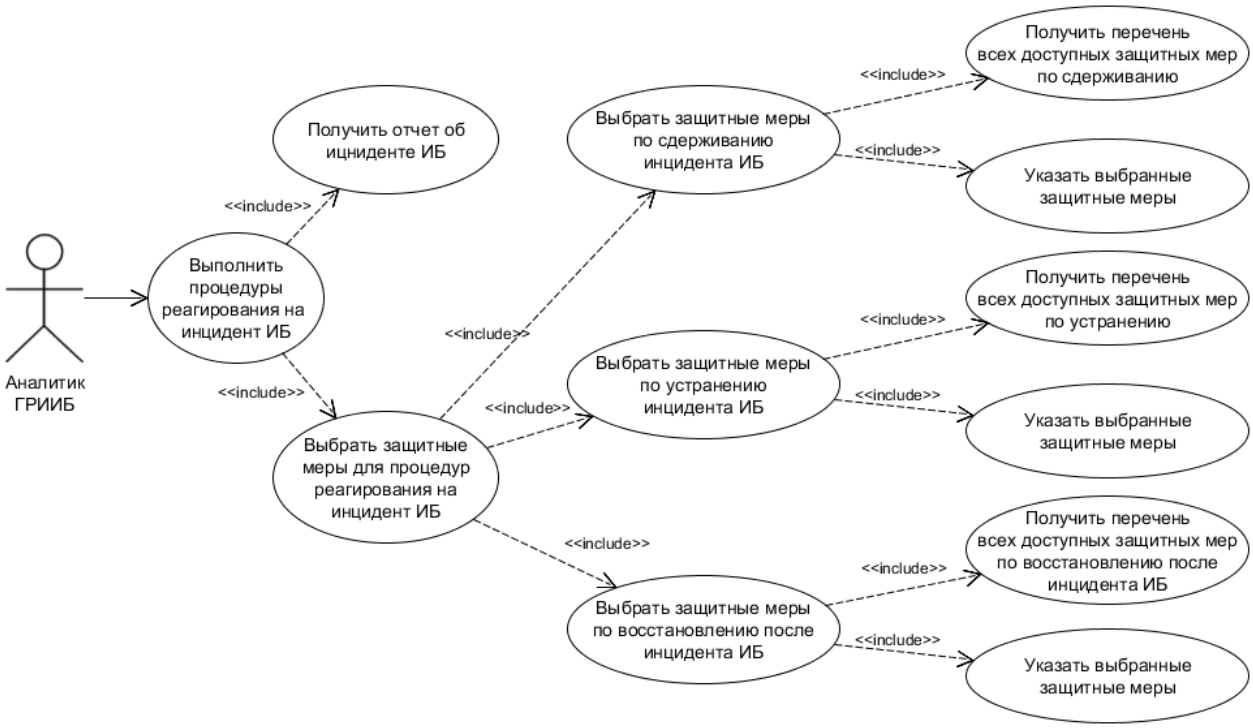

Рисунок 5 – Диаграмма вариантов использования модели среды управления инцидентами ИБ в режиме функционирования «Реагирование на инцидент ИБ»

2.3 Разработка модели экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности

Экспертные системы – это прикладные системы, в которых база знаний представляет собой формализованные эмпирические знания экспертов в какой-либо узкой предметной области. Экспертные системы предназначены для замены при решении задач экспертов в силу их недостаточного количества, недостаточной оперативности в решении задачи или в опасных для них условиях [4].

Под экспертной системой поддержки принятия решений по управлению инцидентами информационной безопасности понимается программное средство, использующее экспертные знания для обеспечения высокоэффективного решения задач управления инцидентами информационной безопасности.

Ключевая особенность такой экспертной системы – возможность к накоплению и организации знаний. Основу экспертной системы составляет база знаний о предметной области, которая накапливается в процессе построения и эксплуатации экспертной системы. Эти знания представляют мнение квалифицированных экспертов по управления инцидентами ИБ и могут быть применены для решения проблем в этой области.

При реализации экспертной системы поддержки принятия решений по управлению инцидентами ИБ в нее может быть заложена возможность использования нескольких режимов функционирования. Основным режимом функционирования экспертной системы в общем случае является режим приобретения знаний.

Режим приобретения знаний необходим для пополнения базы знаний экспертной системы. В этом режиме общение с экспертной системой осуществляет эксперт. Эксперт описывает процессы управления инцидентами ИБ в виде совокупности фактов и правил. Эксперт вносит в систему правила первичной и вторичной оценки инцидента ИБ, а также правила выбора защитных мер по сдерживанию и устранению инцидента, а также восстановлению после него.

Экспертная система также может иметь режим решения задачи. В режиме решения задачи общение с экспертной системой осуществляет ее конечный пользователь. Задача экспертной системы поддержки принятия решений по управлению инцидентами ИБ, функционирующей в этом режиме – формирование решений по управлению инцидентами ИБ на основе накопленных ранее знаний.

Поскольку основой экспертной системы является совокупность информации, структурированная в целях упрощения процесса принятия решения [5], в основу ее модели может быть заложена модель среды управления инцидентами ИБ. Эта модель обеспечивает функционирование такой экспертной системы, создавая необходимый контекст управления инцидентами. Она содержит базы событий ИБ, базы возможных защитных мер по реагированию на

инциденты ИБ, а также предоставляет возможность моделировать процессы управления инцидентами ИБ с участием человека.

Таким образом в модели экспертной системы может быть реализован режим приобретения знаний экспертной системы. Поскольку модель среды управления инцидентами ИБ имеет три режима функционирования, функционирование модели экспертной системы в этом режиме будет осуществляться в три этапа.

На первом этапе моделирования процесса приобретения знаний экспертной системы происходит внесение в экспертную систему знаний о решениях в процессе первичной оценки события ИБ. При этом модель среды управления инцидентами ИБ функционирует в режиме «Обнаружение и первичная оценка события ИБ». Процесс представляется последовательным выполнением следующих действий:

— модель среды управления инцидентами ИБ генерирует начальные условия: описание события ИБ и характеристик объекта, в рамках которого осуществляется процесс управления инцидентами ИБ;

— эксперт выполняет первичную оценку события ИБ на основе сгенерированных данных и сообщает экспертной системе решение о реальности инцидента ИБ;

— экспертная система вносит описание начальных условий и экспертное решение в свою базу знаний с целью дальнейшего использования при решении задач с идентичными начальными условиями.

На втором этапе моделирования процесса приобретения знаний экспертной системы происходит внесение в экспертную систему знаний о решениях в процессе вторичной оценки инцидента ИБ. При этом модель среды управления инцидентами ИБ функционирует в режиме «Вторичная оценка инцидента ИБ». Процесс представляется последовательным выполнением следующих действий:

— модель среды управления инцидентами ИБ генерирует начальные условия: описание инцидента ИБ и характеристик объекта, в рамках которого осуществляется процесс управления инцидентами ИБ;

— эксперт выполняет вторичную оценку инцидента ИБ на основе сгенерированных данных и сообщает экспертной системе решение о реальности инцидента ИБ и его виде;

— экспертная система вносит описание начальных условий и экспертное решение в свою базу знаний с целью дальнейшего использования при решении задач с идентичными начальными условиями.

На третьем этапе моделирования процесса приобретения знаний экспертной системы происходит внесение в экспертную систему знаний о решениях в процессе реагирования на инцидент ИБ. При этом модель среды управления инцидентами ИБ функционирует в режиме «Реагирование на инцидент ИБ». Процесс представляется последовательным выполнением следующих действий:

— модель среды управления инцидентами ИБ генерирует начальные условия: описание инцидента ИБ и набора защитных мер, доступных для объекта, в рамках которого осуществляется процесс управления инцидентами ИБ;

— эксперт выполняет выбор защитных мер, необходимых для осуществления сдерживания инцидента ИБ на основе сгенерированных данных и сообщает экспертной системе решение в виде перечня выбранных защитных мер по сдерживанию инцидента;

— эксперт выполняет выбор защитных мер, необходимых для осуществления устранения инцидента ИБ на основе сгенерированных данных и сообщает экспертной системе решение в виде перечня выбранных защитных мер по устранению инцидента;

— эксперт выполняет выбор защитных мер, необходимых для осуществления восстановления после инцидента ИБ на основе сгенерированных данных и сообщает экспертной системе решение в виде перечня выбранных защитных мер по восстановлению после инцидента;

— экспертная система вносит описание начальных условий и экспертные решения в свою базу знаний с целью дальнейшего использования при решении задач с идентичными начальными условиями.

Общая схема моделирования процесса приобретения знаний экспертной системы приведена на рисунке 6.

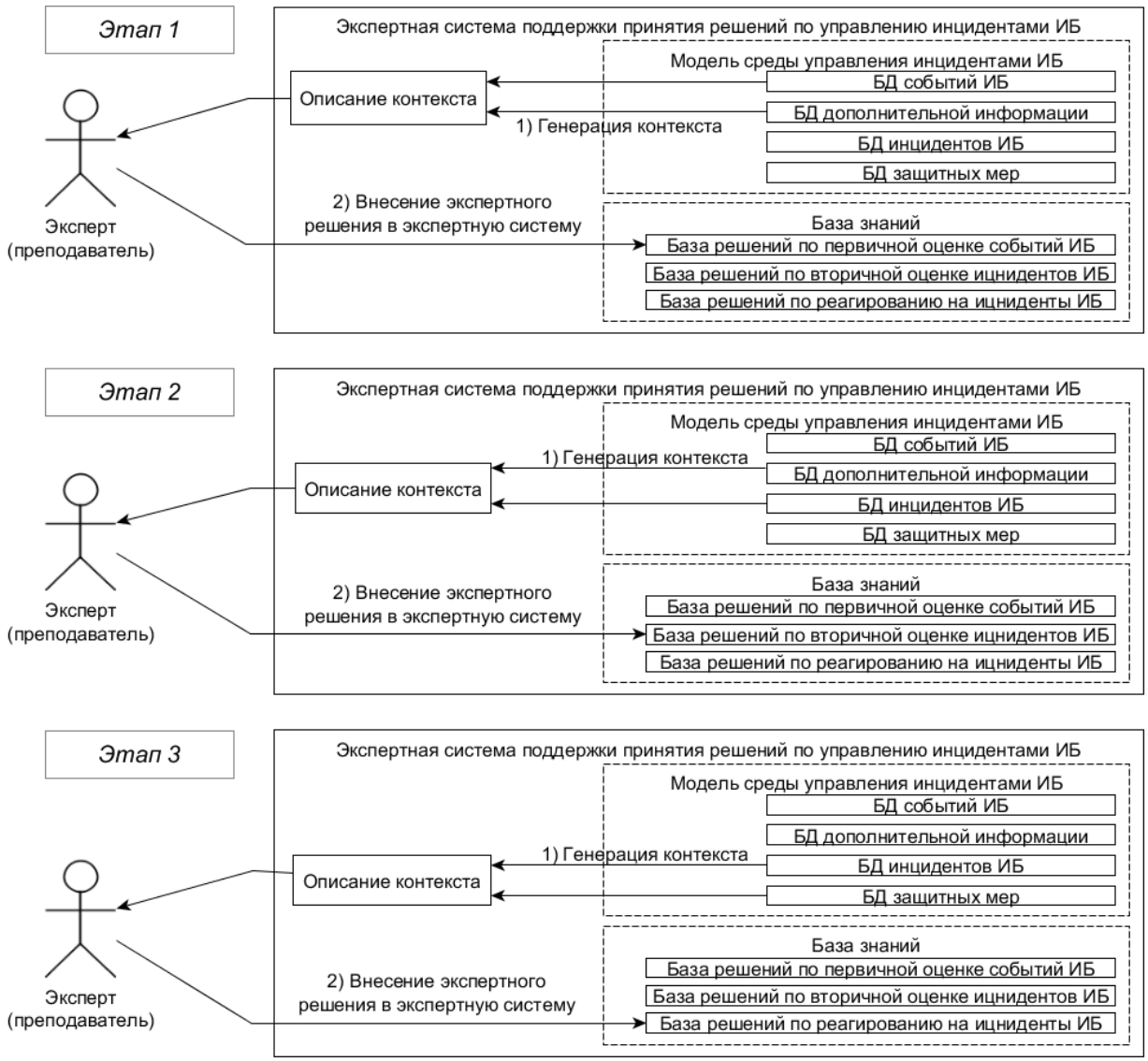

Рисунок 6 – Последовательность действий при функционировании экспертной системы в режиме приобретения знаний

В модель экспертной системы поддержки принятия решений по управлению инцидентами ИБ может быть добавлен еще один режим функционирования – режим сравнения решений. Предполагается, что моделирование в этом режиме происходит аналогично режиму приобретения знаний с разницей в том, что в общение с экспертной системой осуществляется

не компетентным экспертом, а обучаемым специалистом, а все принятые им решения сравниваются с решениями, уже заложенными в базу знаний экспертной системы.

Такой режим функционирования модели экспертной системы может быть использован для обучения и тренировки специалистов путем обеспечения их знаниями, заложенными компетентными экспертами в экспертную систему. При этом предполагается, что база знаний экспертной системы содержит только те решения, которые были признаны верными компетентным экспертом.

Общая схема процесса использования экспертной системы в режиме сравнения решений приведена на рисунке 7.

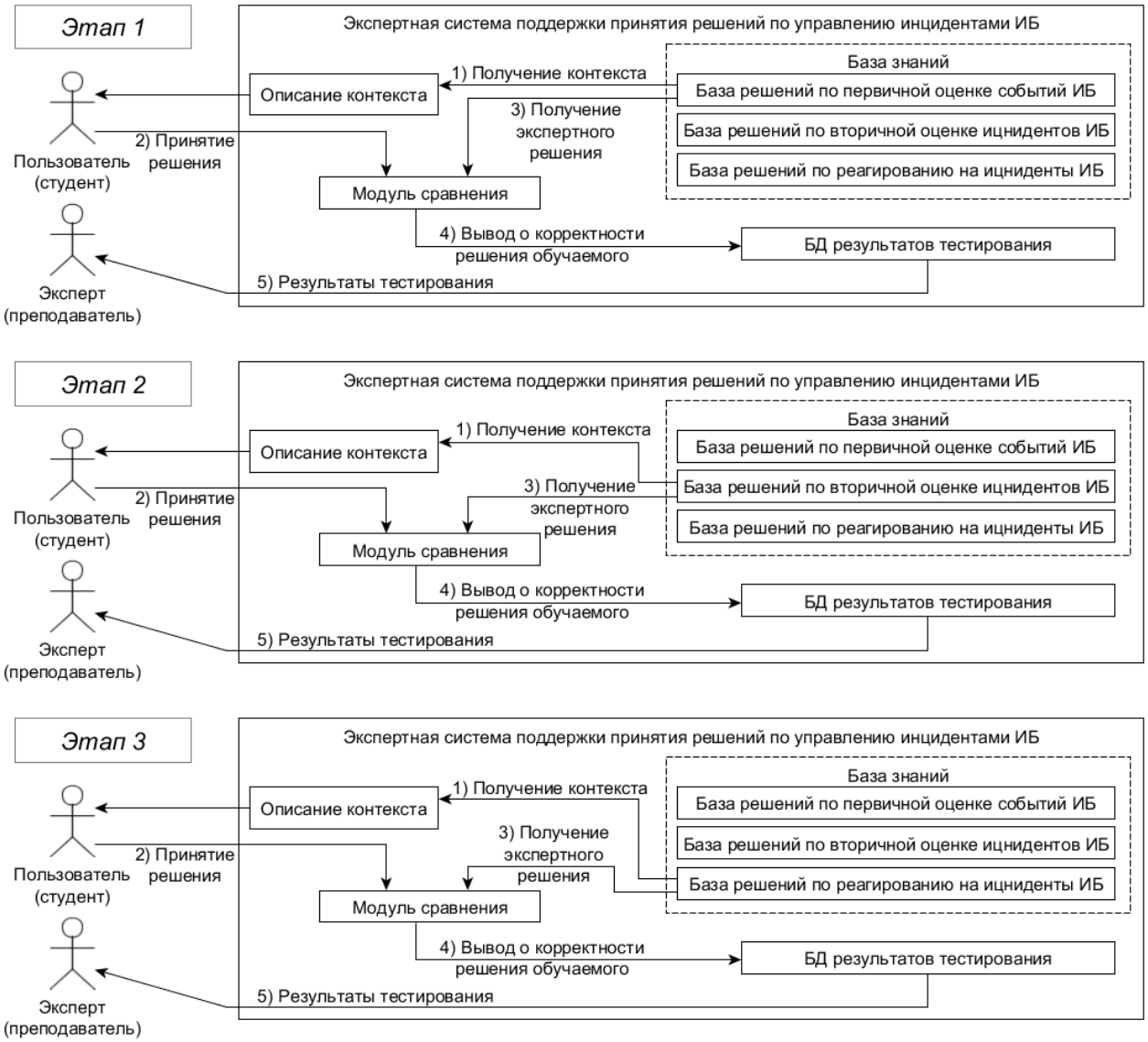

Рисунок 7 – Последовательность действий при функционировании экспертной системы в режиме сравнения решений

Модель экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности, построенная таким образом, может быть использована при подготовке специалистов по информационной безопасности.

3 Разработка и проектирование архитектуры программного модуля, реализующего модель среды управления инцидентами информационной безопасности

Было принято решение при реализации модели среды управления инцидентами ИБ представлять исходные данные для описания событий ИБ в рамках генерации контекста для первичной оценки событий ИБ в виде признаков инцидентов и отчетов о событиях ИБ. При этом вместе с отчетами о событиях ИБ должна быть представлена дополнительная информация.

На основе исходных данных должно быть сгенерировано описание события ИБ. Описание события было решено представлять в двух различных формах: либо в виде совокупности двух признаков инцидентов, либо в виде совокупности отчета о событии ИБ и дополнительных данных. Было принято решение, что из числа всех сгенерированных описаний событий ИБ 20% будут представлены в виде отчетов о событиях ИБ и соответствующей им дополнительной информации, а 80% - в виде пары признаков инцидентов.

Для этого было решено включить в программный модуль, реализующий модель среды управления инцидентами ИБ, метод генерации всех возможных пар признаков инцидентов. Исходными данными для такой генерации будет являться перечень признаков инцидентов. В свою очередь, описание событий ИБ в виде отчетов о событиях ИБ и дополнительных сведений было решено представить простой случайной выборкой из перечня всех отчетов ИБ и дополнительных сведений к ним.

Перечень всех сгенерированных описаний событий ИБ, перечень всех доступных признаков инцидентов, а также перечень всех доступных отчетов о событиях ИБ и соответствующих им дополнительных сведений было решено представлять в виде базы данных (БД) в трех различных таблицах. При этом запись БД с описанием признака инцидента ИБ должна включать непосредственно само описание признака, запись БД с описанием отчета о событии ИБ – содержание самого отчета о событии ИБ, а также набор

дополнительной информации, соответствующей данному отчету, а запись БД с описанием события ИБ должна содержать следующие поля:

— идентификатор;

— либо два признака инцидента ИБ, либо отчет о событии ИБ и дополнительные сведения к нему.

При эксплуатации программного модуля модели среды управления инцидентами ИБ в режиме «Первичная оценка события ИБ» было принято использовать алгоритм действий, приведенный на рисунке 8.

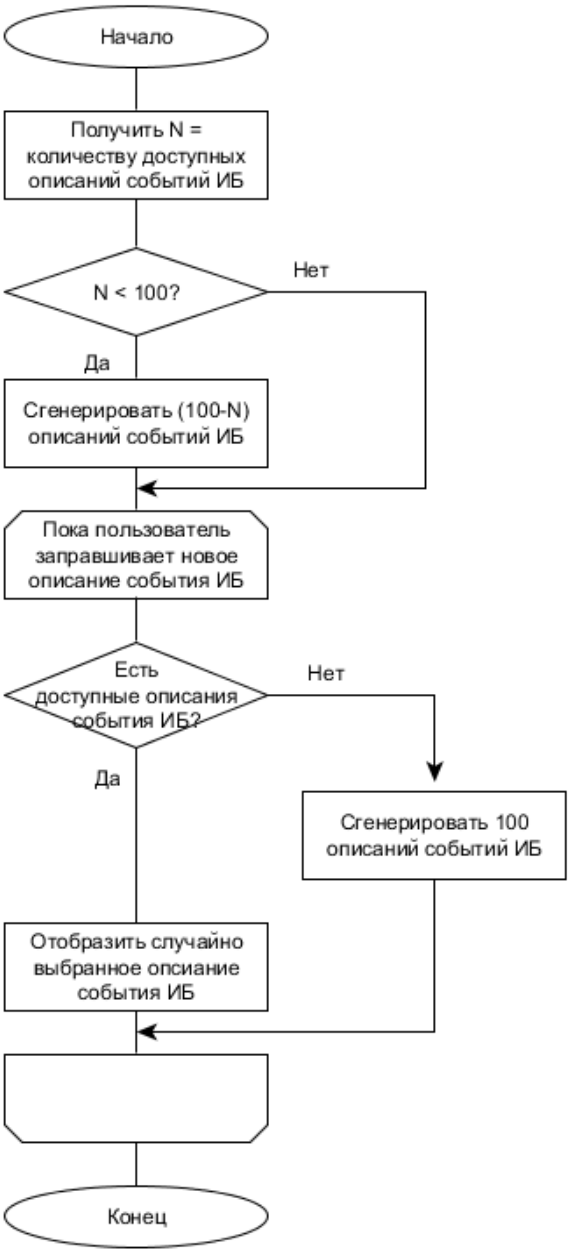

Рисунок 8 – Алгоритм функционирования программного модуля модели среды управления инцидентами ИБ в режиме «Первичная оценка события ИБ»

Было принято решение при реализации модели среды управления инцидентами ИБ представлять исходные данные для инцидентов ИБ в рамках генерации контекста для вторичного анализа инцидентов ИБ в виде отчетов об инцидентах ИБ и некоторой дополнительной информации.

На основе исходных данных должно быть сгенерировано описание инцидента ИБ. Описание инцидента было решено представлять в виде совокупности отчета об инциденте ИБ и комбинации из четырех блоков дополнительных сведений.

Для этого было решено включить в программный модуль, реализующий модель среды управления инцидентами ИБ, метод генерации всех возможных комбинаций из четырех блоков дополнительных сведений. Исходными данными для такой генерации будет являться перечень дополнительных сведений. Также было решено включить в указанный программный модуль метод генерации всех возможных комбинаций, состоящих из отчета об инциденте ИБ и набора из четырех блоков дополнительных сведений. Исходными данными для такой генерации будут являться сгенерированные ранее наборы дополнительных сведений, а также перечень всех доступных отчетов об инцидентах ИБ.

Перечень всех сгенерированных описаний инцидентов ИБ, перечень всех доступных блоков дополнительных сведений, а также перечень всех доступных отчетов об инцидентах ИБ было решено представлять в виде БД в трех различных таблицах. При этом запись БД с описанием набора дополнительных сведений должна включать непосредственно сами дополнительные сведения, запись БД с описанием отчета об инциденте ИБ – содержание самого отчета об инциденте ИБ, а запись БД с описанием инцидента ИБ должна содержать следующие поля:

— идентификатор;
— текст отчета об инциденте ИБ;
— текст, содержащий первый блок дополнительных сведений;
— текст, содержащий второй блок дополнительных сведений;
— текст, содержащий третий блок дополнительных сведений;

— текст, содержащий четвертый блок дополнительных сведений.

При эксплуатации программного модуля модели среды управления инцидентами ИБ в режиме «Вторичный анализ инцидента ИБ» было принято использовать алгоритм действий, приведенный на рисунке 9.

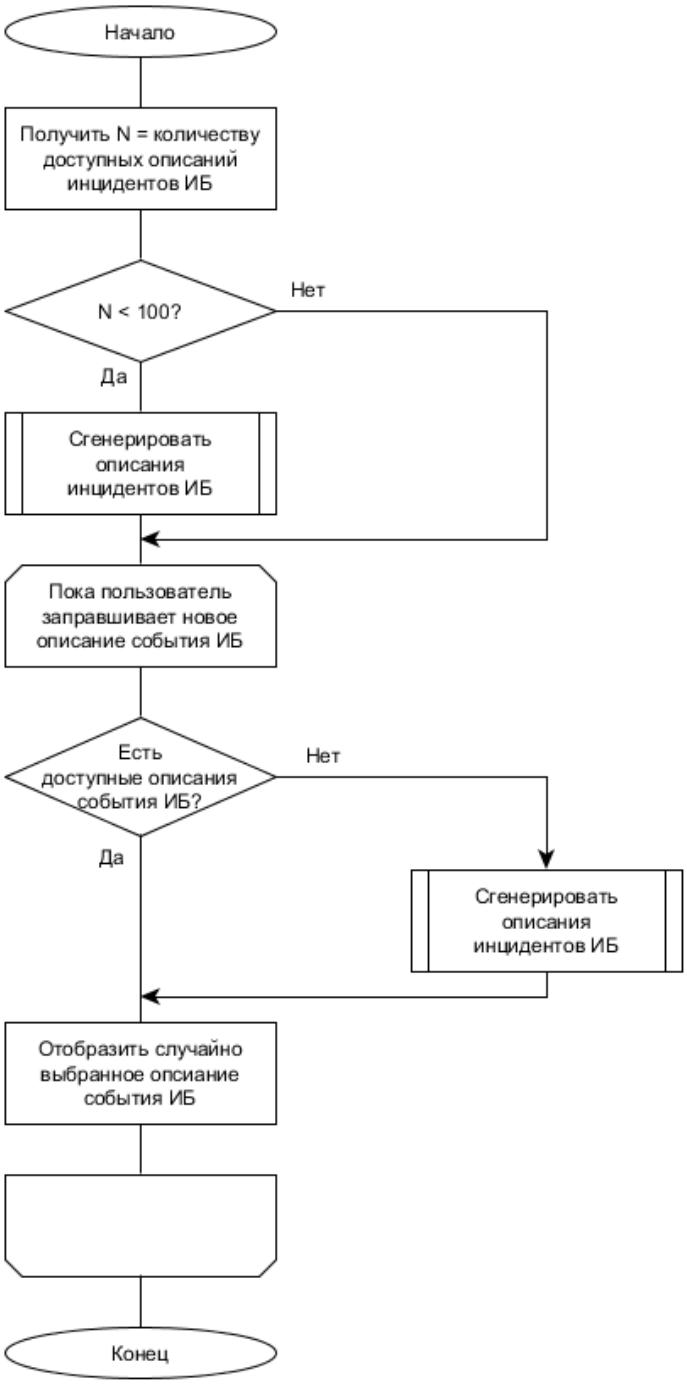

Рисунок 9 – Алгоритм функционирования программного модуля модели среды управления инцидентами ИБ в режиме «Первичная оценка события ИБ»

Было принято решение при реализации модели среды управления инцидентами ИБ представлять исходные данные для инцидентов ИБ в рамках

генерации контекста для реагирования на инциденты ИБ в виде отчетов об инцидентах ИБ и набора защитных мер по сдерживанию инцидента, устранению инцидента и восстановлению после него отдельно.

На основе исходных данных должно быть сгенерировано описание контекста выполняемой операции по реагированию на инцидент ИБ. Описание такого контекста было решено представлять в виде совокупности отчета об инциденте ИБ, указания конкретного этапа реагирования и перечня доступных на данном этапе защитных мер.

Для этого было решено включить в программный модуль, реализующий модель среды управления инцидентами ИБ, метод генерации всех возможных комбинаций из пяти защитных мер для каждого этапа реагирования. Исходными данными для такой генерации будет являться перечень всех доступных защитных мер. Также было решено включить в указанный программный модуль метод генерации всех возможных комбинаций, состоящих из отчета об инциденте ИБ и набора из пяти защитных мер для каждого этапа реагирования на инцидент ИБ. Исходными данными для такой генерации будут являться сгенерированные ранее наборы защитных мер для отдельных этапов реагирования, а также перечень всех доступных отчетов об инцидентах ИБ.

Перечень всех сгенерированных описаний контекстов выполняемых операций по реагированию на инцидент ИБ, перечень всех доступных защитных мер, а также перечень всех доступных отчетов об инцидентах ИБ было решено представлять в виде БД в пяти различных таблицах при условии расположения защитных мер для каждого из этапов реагирования в отдельных таблицах БД. При этом запись БД с описанием набора защитной меры должна включать непосредственно описание самой защитной меры, запись БД с описанием отчета об инциденте ИБ – содержание самого отчета об инциденте ИБ, а запись БД с описанием контекста выполняемой операции по реагированию должна содержать следующие поля:

— идентификатор;

— текст отчета об инциденте ИБ;

— указание одного из трех этапов реагирования;

— набор защитных мер, доступных для данного этапа реагирования.

При эксплуатации программного модуля модели среды управления инцидентами ИБ в режиме «Реагирование на инцидент ИБ» было принято использовать алгоритм действий, приведенный на рисунке 10.

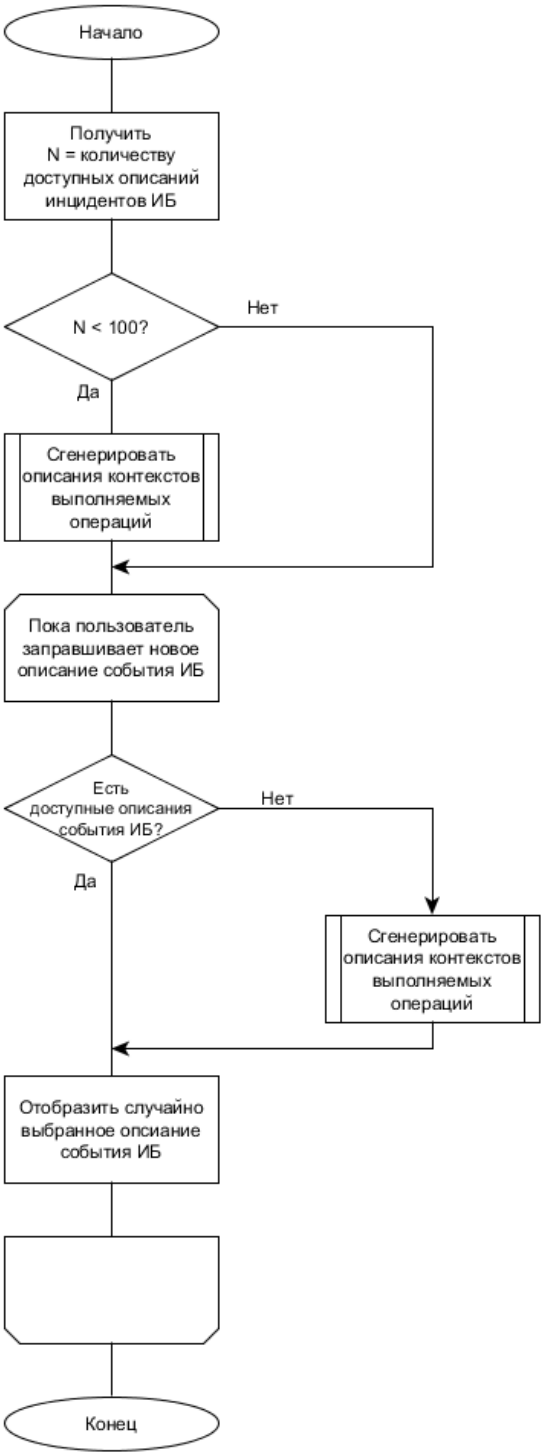

Рисунок 10 – Алгоритм функционирования программного модуля модели среды управления инцидентами ИБ в режиме «Реагирование на инцидент ИБ»

4 Реализация модели среды управления инцидентами информационной безопасности в виде программного модуля

Модель среды управления инцидентами информационной безопасности была реализована на языке C++ в виде класса ContextGen. Данный класс имеет набор статических методов, необходимых для генерации контекста управления инцидентами ИБ. Полный исходный код класса Context Gen приведен в приложении А.

Генерация описаний событий ИБ в режиме «Первичная оценка событий ИБ» реализуется методом UpdateExperienceComboMode1 класса ContextGen. В этом случае все возможные пары признаков инцидентов составляются методом перебора. Исходные данные для такой генерации хранятся в таблицах Reports и Signs базы данных PrimaryAnalysis.db.

Таблица Reports базы данных PrimaryAnalysis.db имеет следующие поля:

— id – идентификатор записи;

— Report – текст отчета о событии ИБ;

— Info1 – текст первого блока дополнительной информации;

— Info2 – текст второго блока дополнительной информации;

— Info3 – текст третьего блока дополнительной информации.

При этом поля Info2 и Info3 могут быть пустыми.

Таблица Signs базы данных PrimaryAnalysis.db имеет следующие поля:

— id – идентификатор записи;

— Sign – описание признака инцидента ИБ.

Метод UpdateExperienceComboMode1 имеет в качестве входного аргумента число, определяющее количество новых описаний событий ИБ, которые должны быть получены в результате генерации. В результате выполнения метода сгенерируются описания событий ИБ в форме пары признаков инцидентов в количестве, равном 80% от общего заданного числа новых описаний событий, и в форме комбинации отчетов о событиях ИБ и соответствующих им дополнительных сведений в количестве 20% от общего

заданного числа новых описаний событий. При расчете количества описаний в той и иной форме округление до целого выполняется в пользу формы представления в виде пары признаков инцидентов.

После непосредственной генерации описаний событий ИБ они сохраняются в таблицу MyExperience базы данных PrimaryAnalysis.db. Таблица MyExperience базы данных PrimaryAnalysis.db имеет следующие поля:

— id – идентификатор записи;

— Info1 – либо описание первого признака инцидента ИБ, либо текст отчета о событии ИБ;

— Info2 – либо описание второго признака инцидента ИБ, либо дополнительная информация к отчету о событии ИБ.

Генерация описаний событий ИБ в режиме «Вторичный анализ инцидентов ИБ» реализуется методом UpdateExperienceComboMode1 класса ContextGen. Алгоритм генерации описаний инцидентов ИБ, реализуемый методом UpdateExperienceComboMode2, приведен на рисунке 11. Исходные данные для такой генерации хранятся в таблицах Reports и Info базы данных SecondaryAnalysis.db.

Таблица Reports базы данных SecondaryAnalysis.db имеет следующие поля:

— id – идентификатор записи;

— Report – текст отчета об инциденте ИБ.

Таблица Info базы данных SecondaryAnalysis.db имеет следующие поля:

— id – идентификатор записи;

— Info – один блок дополнительной информации.

Метод UpdateExperienceComboMode2 имеет в качестве входного аргумента число, определяющее минимальное количество новых описаний событий ИБ, которые должны быть получены в результате генерации. В результате выполнения метода сгенерируются описания инцидентов ИБ в форме совокупности отчета об инциденте ИБ и четырех блоков дополнительной информации.

Рисунок 11 – Алгоритм генерации описаний инцидентов ИБ для режима «Вторичный анализ инцидентов ИБ»

Для генерации всех возможных наборов по четыре блока дополнительной информации используется метод GetAllInfoSets класса ContextGen. После получения таких наборов для каждого из отчетов об инцидентах ИБ случайным образом выбирается одинаковое число наборов дополнительной информации. Полученные комбинации из текста отчета об инциденте ИБ и набора дополнительной информации сохраняются в таблицу MyExperience базы данных

SecondaryAnalysis.db. Таблица MyExperience базы данных SecondaryAnalysis.db имеет следующие поля:

— id – идентификатор записи;
— Report – текст отчета об инциденте ИБ;
— Info1 – первый блок дополнительной информации;
— Info2 – второй блок дополнительной информации;
— Info3 – третий блок дополнительной информации;
— Info4 – четвертый блок дополнительной информации.

Генерация описаний событий ИБ в режиме «Реагирование на инциденты ИБ» реализуется методом UpdateExperienceComboMode3 класса ContextGen. Алгоритм генерации описаний контекстов выполняемых операций, реализуемый методом UpdateExperienceComboMode3, приведен на рисунке 12. Исходные данные для такой генерации хранятся в таблицах Reports, Control, Removal и Recovery базы данных ProtectiveMeasures.db.

Таблица Reports базы данных ProtectiveMeasures.db имеет следующие поля:

— id – идентификатор записи;
— Report – текст отчета об инциденте ИБ.

Таблица Control базы данных ProtectiveMeasures.db имеет следующие поля:

— id – идентификатор записи;
— Measure – описание меры по сдерживанию инцидента ИБ.

Таблица Removal базы данных ProtectiveMeasures.db имеет следующие поля:

— id – идентификатор записи;
— Measure – описание меры по устранению инцидента ИБ.

Таблица Recovery базы данных ProtectiveMeasures.db имеет следующие поля:

— id – идентификатор записи;
— Measure – описание меры по восстановлению после инцидента ИБ.

Рисунок 12 – Алгоритм генерации описаний контекста для режима «Реагирование на инцидент ИБ»

Метод UpdateExperienceComboMode3 имеет в качестве входного аргумента число, определяющее минимальное количество новых описаний событий ИБ, которые должны быть получены в результате генерации. В результате выполнения метода сгенерируются описания инцидентов ИБ в форме совокупности отчета об инциденте ИБ, информации об этапе реагирования и пяти защитных мер для указанного этапа реагирования.

Для генерации всех возможных наборов по пять защитных мер используется метод GetAllMeasuresSets класса ContextGen. Данный метод принимает в качестве входного аргумента название этапа реагирования, для которого нужно составить комбинации мер. В методе UpdateExperienceComboMode3 выполняется генерация наборов защитных мер для каждого из трех этапов реагирования на инцидент ИБ. Затем для каждого из отчетов об инцидентах ИБ одинаковое количество раз случайным образом выбирается один из этапов реагирования и случайный набор соответствующих ему защитных мер. Полученные комбинации из текста отчета об инциденте ИБ, этапа реагирования и набора защитных мер сохраняются в таблицу MyExperience базы данных ProtectiveMeasures.db. Таблица MyExperience базы данных ProtectiveMeasures.db имеет следующие поля:

— id – идентификатор записи;
— Report – текст отчета об инциденте ИБ;
— Stage – этап реагирования на инцидент;
— Measures – набор мер для заданного этапа реагирования.

5 Разработка и проектирование архитектуры экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности

При реализации экспертной системы поддержки принятия решений по управлению инцидентами ИБ было решено использовать фреймворк QT, поскольку он позволяет упростить работу с графическим интерфейсом, клиент-серверной архитектурой и базами данных.

Было принято решение реализовать режим приобретения знаний экспертной системы полностью на стороне серверного компонента экспертной системы, поскольку общение с экспертной системой в этом режиме может осуществлять только эксперт, а серверный компонент предполагается поставлять только экспертам.

Для обеспечения работы экспертной системы в режиме приобретения знаний в серверный компонент должен быть включен программный модуль, реализующий модель среды управления инцидентами ИБ. Поскольку фактически этот модуль используется только в режиме приобретения знаний экспертной системы, было принято решение не включать его в состав клиентского компонента.

В режиме приобретения знаний эксперт описывает процессы управления инцидентами ИБ в виде совокупности фактов и правил. Эксперт вносит в систему правила первичной и вторичной оценки инцидента ИБ, а также правила выбора защитных мер по сдерживанию и устранению инцидента, а также восстановлению после него.

Для обеспечения возможности получения и сохранения экспертных решений экспертной системой было решено разбить режим приобретения знаний экспертной системы на три разных режима в соответствии с моделируемым экспертной системой процессом управления инцидентами ИБ и типом принимаемых экспертом решений: режим приобретения знаний по первичному анализу событий ИБ, режим приобретения знаний по вторичному

анализу инцидентов ИБ и режим приобретения знаний по реагированию на инциденты ИБ.

В режиме приобретения знаний по первичному анализу событий ИБ происходит внесение в экспертную систему знаний о решениях в процессе первичной оценки события ИБ. Было принято решение представить этот процесс последовательностью следующих действий:

— программный модуль модели среды управления инцидентами ИБ генерирует описание события ИБ;

— из числа всех ранее сгенерированных выбирается случайное описание события ИБ, по которому эксперт никогда не выносил решения;

— выбранное описание события ИБ демонстрируется эксперту посредством графического интерфейса;

— эксперт выполняет первичную оценку события ИБ и выносит свое решение о реальности инцидента ИБ посредством графического интерфейса;

— экспертная система сохраняет для данного описания события ИБ решение эксперта.

Для обеспечения возможности выполнения указанных действий было принято решение реализовать отдельное диалоговое окно для общения с пользователем в режиме приобретения знаний по первичному анализу событий ИБ. Также было решено вносить решение эксперта в уже существующую запись БД с описанием события ИБ, а также вносить в нее признак того, что эксперт уже вынес решение по этому описанию события ИБ. Для обеспечения удобства использования экспертной системы также было решено добавить возможность эксперту пометить описание события ИБ как нежелательное для отображения. Признак такой отметки также было решено хранить в уже существующей записи БД с описанием события ИБ.

В режиме приобретения знаний по вторичному анализу инцидентов ИБ происходит внесение в экспертную систему знаний о решениях в процессе вторичной оценки инцидента ИБ. Было принято решение представить этот процесс последовательностью следующих действий:

— программный модуль модели среды управления инцидентами ИБ генерирует описание инцидента ИБ;

— из числа всех ранее сгенерированных выбирается случайное описание инцидента ИБ, по которому эксперт никогда не выносил решения;

— выбранное описание инцидента ИБ демонстрируется эксперту посредством графического интерфейса;

— эксперт выполняет вторичную оценку инцидента ИБ и выносит свое решение о реальности инцидента ИБ и его виде посредством графического интерфейса;

— экспертная система сохраняет для данного описания инцидента ИБ решение эксперта.

Для обеспечения возможности выполнения указанных действий было принято решение реализовать отдельное диалоговое окно для общения с пользователем в режиме приобретения знаний по вторичному анализу инцидентов ИБ. Также было решено вносить решение эксперта в уже существующую запись БД с описанием инцидента ИБ, а также вносить в нее признак того, что эксперт уже вынес решение по этому описанию инцидента ИБ. Для обеспечения удобства использования экспертной системы также было решено добавить возможность эксперту пометить описание инцидента ИБ как нежелательное для отображения. Признак такой отметки также было решено хранить в уже существующей записи БД с описанием инцидента ИБ.

В режиме приобретения знаний по реагированию на инциденты ИБ происходит внесение в экспертную систему знаний о решениях в процессе реагирования на инцидент ИБ. Было принято решение представить этот процесс последовательностью следующих действий:

— программный модуль модели среды управления инцидентами ИБ генерирует описание контекста выполняемой операции по реагированию на инцидент ИБ;

— из числа всех ранее сгенерированных выбирается случайное описание контекста, по которому эксперт никогда не выносил решения;

— выбранное описание контекста демонстрируется эксперту посредством графического интерфейса;

— эксперт выполняет выбор защитных мер, необходимых для осуществления сдерживания, устранения инцидента ИБ или восстановления после него в зависимости от описания контекста и выносит свое решение о выбранных мерах посредством графического интерфейса;

— экспертная система сохраняет для данного описания контекста решение эксперта.

Для обеспечения возможности выполнения указанных действий было принято решение реализовать отдельное диалоговое окно для общения с пользователем в режиме приобретения знаний по реагированию на инциденты ИБ. Также было решено вносить решение эксперта в уже существующую запись БД с описанием контекста выполняемой операции по реагированию на инцидент ИБ, а также вносить в нее признак того, что эксперт уже вынес решение по этому описанию контекста. Для обеспечения удобства использования экспертной системы также было решено добавить возможность эксперту пометить описание контекста как нежелательное для отображения. Признак такой отметки также было решено хранить в уже существующей записи БД с описанием контекста.

Поскольку основным назначением экспертной системы является ее использование в учебном процессе, было принято решение реализовать режим принятия решений таким образом, что принимаемое экспертной системой решение используется для проверки решения, принятого пользователем. При этом решение, принятое пользователем, не может повлиять на уже существующие знания экспертной системы.

Для обеспечения возможности экспертной системы функционировать в трех указанных режимах, клиентский компонент экспертной системы должен иметь соответствующий графический интерфейс, а также возможность получения контекста управления инцидентами ИБ, хранящегося на стороне серверного компонента. Таким образом, режимы принятия решений

реализуются с участием и клиентского, и серверного компонентов экспертной системы.

Было решено разбить режим принятия решений экспертной системы на три разных режима в соответствии с моделируемым экспертной системой процессом управления инцидентами ИБ и типом принимаемых пользователем решений: режим принятия решений по первичному анализу событий ИБ, режим принятия решений по вторичному анализу инцидентов ИБ и режим принятия решений по реагированию на инциденты ИБ.

В режиме принятия решений по первичному анализу событий ИБ пользователь выносит решения по процессу первичной оценки события ИБ. Было принято решение представить этот процесс последовательностью следующих действий:

— клиентский компонент экспертной системы запрашивает у серверного компонента некоторое количество описаний событий ИБ;

— на серверной стороне из числа всех ранее сгенерированных в заданном клиентской стороной количестве выбираются случайные описания события ИБ, по которым эксперт уже вынес решение;

— выбранные описания событий ИБ с экспертными решениями по ним передаются на клиентскую сторону;

— описания событий ИБ последовательно по одному демонстрируется пользователю посредством графического интерфейса;

— пользователь выполняет первичную оценку события ИБ и выносит свое решение о реальности инцидента ИБ посредством графического интерфейса;

— на клиентской стороне осуществляется сравнение пользовательского решения с экспертным, по результатам проверки изменяется число правильных решений пользователя;

— после вынесения пользователем решений по всем описаниям событий ИБ клиентский компонент экспертной системы передает серверу информацию о количестве правильных решений пользователя;

— серверный компонент сохраняет информацию о количестве правильных решений пользователя.

Для обеспечения возможности выполнения указанных действий было принято решение реализовать отдельное диалоговое окно для общения с пользователем в режиме принятия решений по первичному анализу событий ИБ.

В режиме принятия решений по вторичному анализу инцидентов ИБ происходит внесение в экспертную систему знаний о решениях в процессе вторичной оценки инцидента ИБ. Было принято решение представить этот процесс для пользователя последовательностью следующих действий:

— клиентский компонент экспертной системы запрашивает у серверного компонента некоторое количество описаний инцидентов ИБ;

— на серверной стороне из числа всех ранее сгенерированных в заданном клиентской стороной количестве выбираются случайные описания инцидентов ИБ, по которым эксперт уже вынес решение;

— выбранные описания инцидентов ИБ с экспертными решениями по ним передаются на клиентскую сторону;

— описания инцидентов ИБ последовательно по одному демонстрируется пользователю посредством графического интерфейса;

— пользователь выполняет вторичный анализ инцидента ИБ и выносит свое решение о реальности инцидента ИБ и его виде посредством графического интерфейса;

— на клиентской стороне осуществляется сравнение пользовательского решения с экспертным, по результатам проверки изменяется число правильных решений пользователя;

— после вынесения пользователем решений по всем описаниям инцидентов ИБ клиентский компонент экспертной системы передает серверу информацию о количестве правильных решений пользователя;

— серверный компонент сохраняет информацию о количестве правильных решений пользователя.

Для обеспечения возможности выполнения указанных действий было принято решение реализовать отдельное диалоговое окно для общения с

пользователем в режиме принятия решений по вторичному анализу инцидентов ИБ.

В режиме принятия решений по реагированию на инциденты ИБ происходит внесение в экспертную систему знаний о решениях в процессе реагирования на инцидент ИБ. Было принято решение представить этот процесс для пользователя последовательностью следующих действий:

— клиентский компонент экспертной системы запрашивает у серверного компонента некоторое количество описаний контекстов;

— на серверной стороне из числа всех ранее сгенерированных в заданном клиентской стороной количестве выбираются случайные описания контекстов, по которым эксперт уже вынес решение;

— выбранные описания контекстов с экспертными решениями по ним передаются на клиентскую сторону;

— описания контекстов последовательно по одному демонстрируется пользователю посредством графического интерфейса;

— пользователь выполняет мер по реагированию на инцидент ИБ в соответствии с заданным контекстом и выносит свое решение о выбранных мерах посредством графического интерфейса;

— на клиентской стороне осуществляется сравнение пользовательского решения с экспертным, по результатам проверки изменяется число правильных решений пользователя;

— после вынесения пользователем решений по всем описаниям инцидентов ИБ клиентский компонент экспертной системы передает серверу информацию о количестве правильных решений пользователя;

— серверный компонент сохраняет информацию о количестве правильных решений пользователя.

Для обеспечения возможности выполнения указанных действий было принято решение реализовать отдельное диалоговое окно для общения с пользователем в режиме принятия решений по реагированию на инциденты ИБ.

Результаты принятия пользователем решений было решено хранить на серверной стороне бессрочно в виде записей БД в трех различных таблицах БД,

соответствующих режиму принятия решений. При этом запись БД должна содержать следующую информацию:

— имя пользователя;

— дата тестирования;

— время начала тестирования;

— время окончания тестирования;

— количество правильных решений.

Было принято решение обеспечить возможность просмотра всех сохраненных в БД результатов в виде таблицы с использованием графического интерфейса серверного компонента.

Была составлена диаграмма вариантов использования экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности, описывающая функциональные возможности проектируемой системы. Указанная диаграмма приведена на рисунке 13.

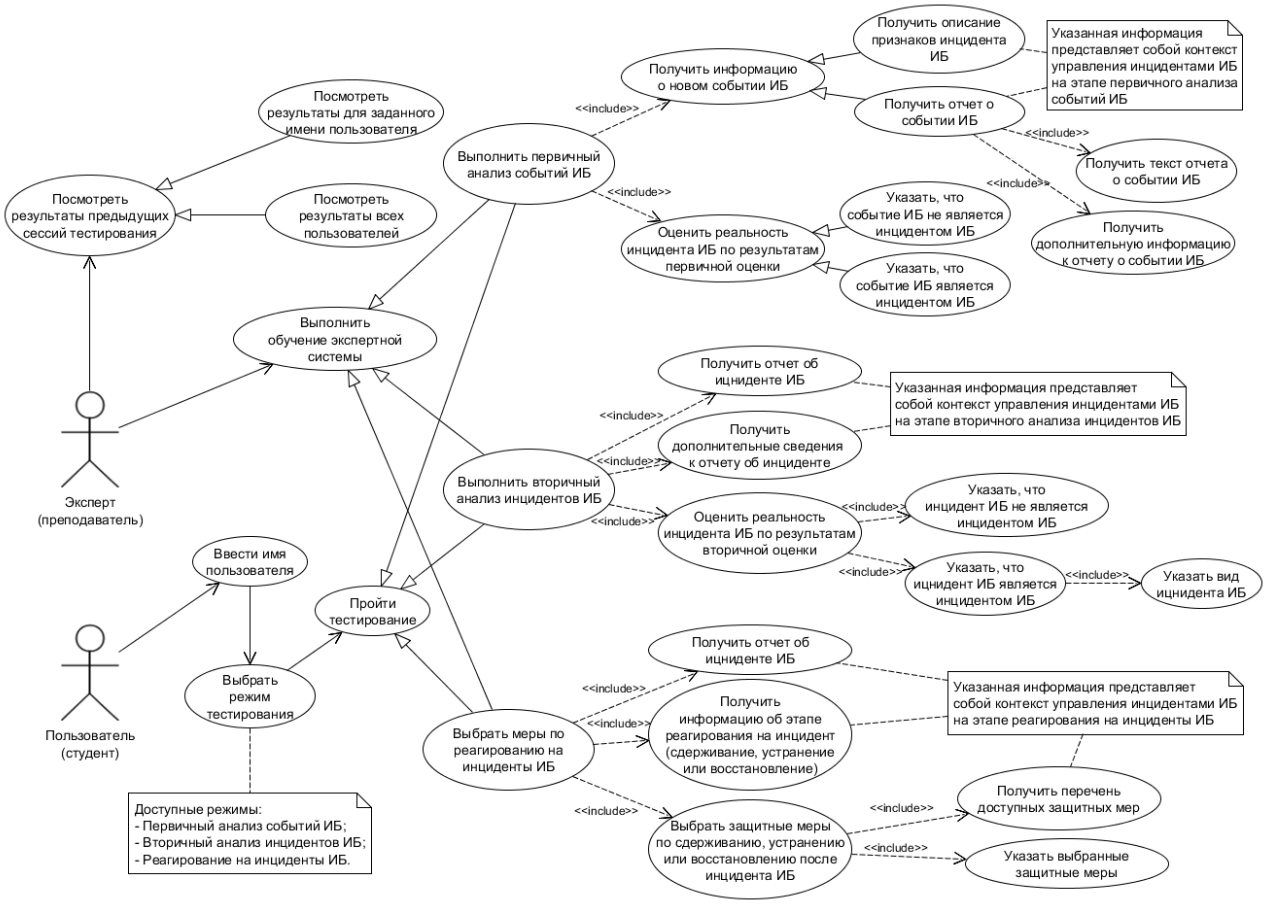

Рисунок 13 – Диаграмма вариантов использования экспертной системы

На основе принятых решений была составлена диаграмма классов серверного компонента экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасностью, приведенная на рисунке 14.

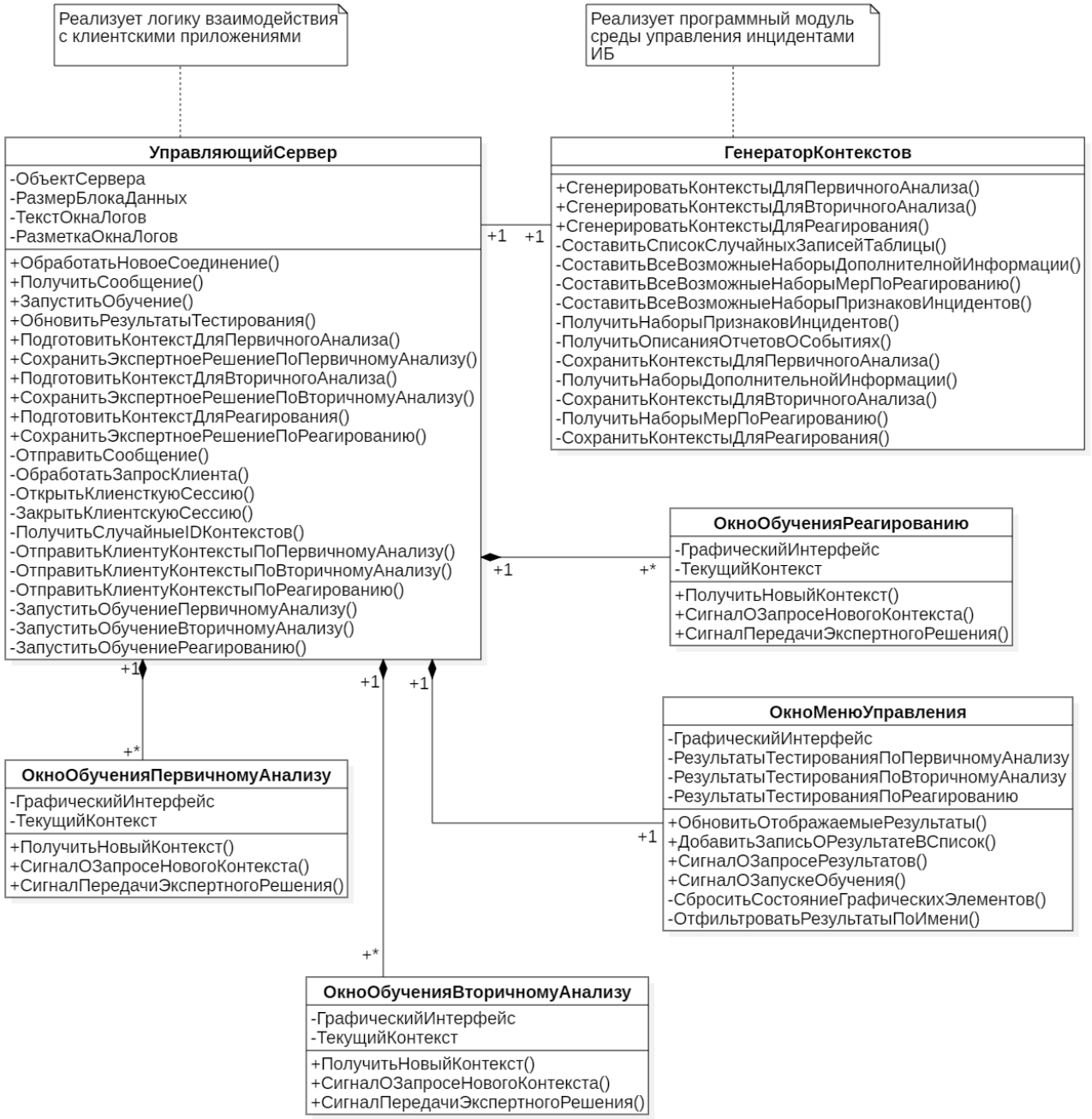

Рисунок 14 – Диаграмма классов серверного компонента экспертной системы

Также была составлена диаграмма классов клиентского компонента экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасностью, приведенная на рисунке 15.

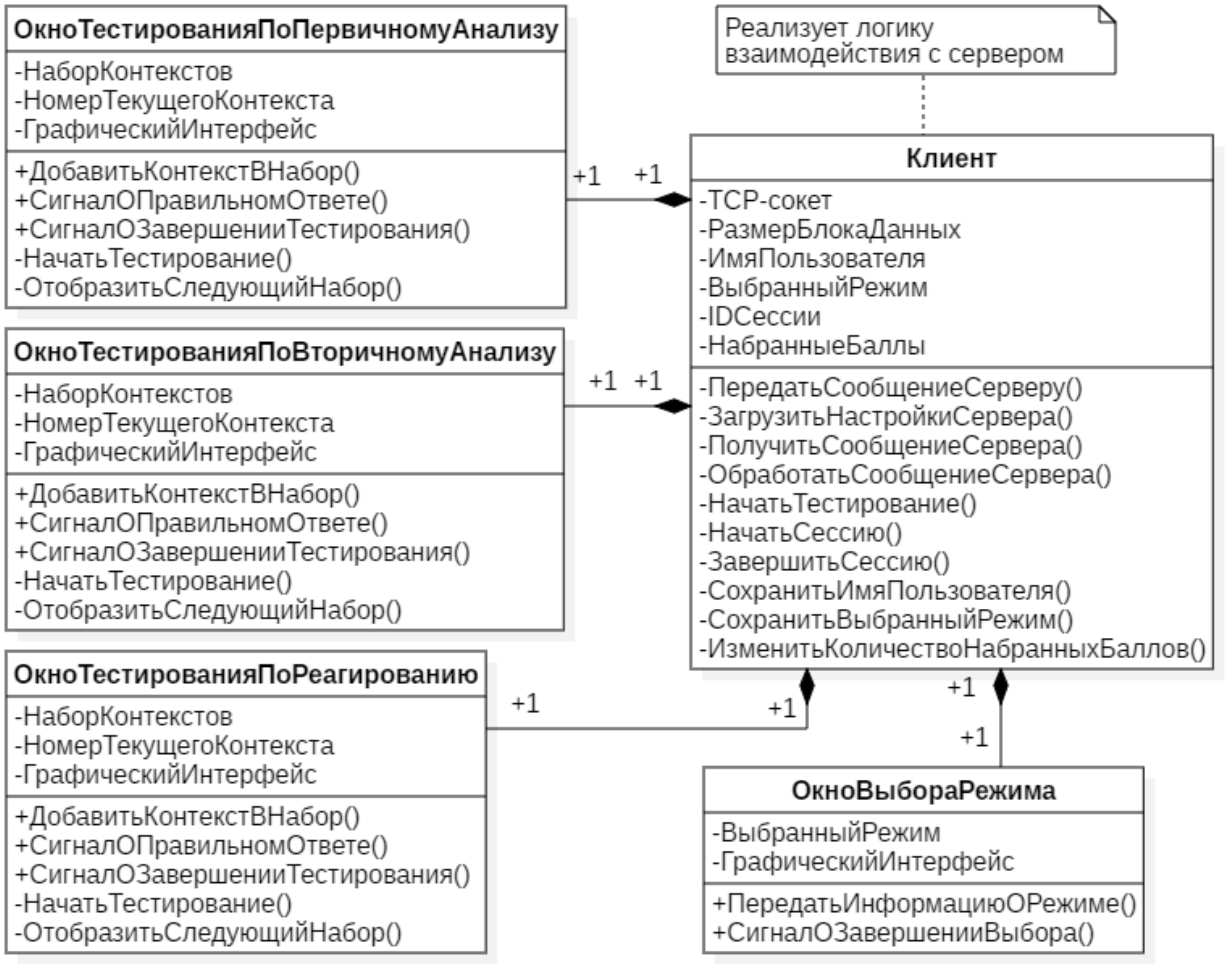

Рисунок 15 – Диаграмма классов клиентского компонента экспертной системы

6 Реализация экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности в виде клиент-серверного приложения

6.1 Реализация серверного компонента

Было реализовано приложение-сервер ITestServer экспертной системы, предназначенное для использования в среде операционных систем семейства Windows. Поскольку при создании серверного приложения был использован фреймворк QT, для поддержки функционирования приложения в среде Windows необходимо наличие соответствующих библиотек указанного фреймворка. Также для обеспечения корректной работы приложения в рабочем каталоге должны располагаться базы данных исходного набора сведений для генерации контекстов управления инцидентами информационной безопасности, а также база данных для хранения результатов тестирования.

Структура проекта приложения ITestServer приведена на рисунке 16.

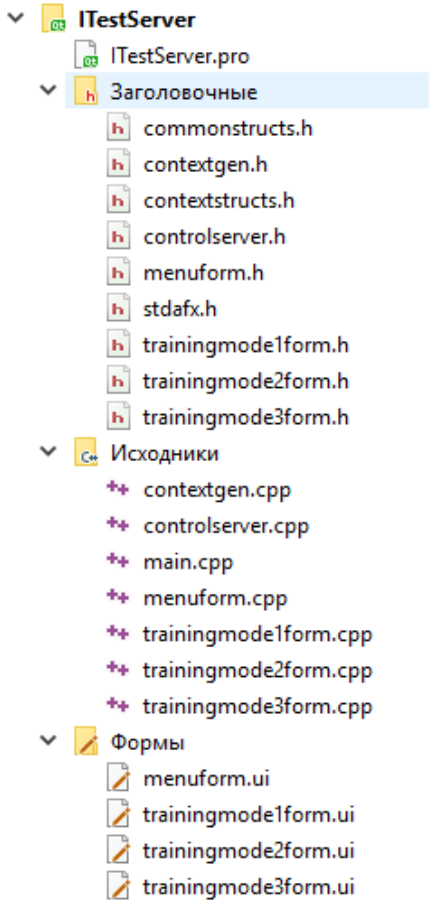

Рисунок 16 – Структура проекта приложения ITestServer

Полный исходный код серверного компонента ITestServer приведен в приложении Б.

Для реализации клиент-серверной архитектуры были использованы классы QTcpServer и QTcpSocket фреймворка QT. Эти классы используют TCP-соединение [6][7]. Для реализации взаимодействия с базами данных были использованы классы QSqlDatabase, QSqlQuery и QSqlRecord. Классы всех диалоговых окон были пронаследованы от класса QWidget.

Был реализован класс сервера ControlServer. Этот класс содержит следующие базовые методы для взаимодействия с клиентскими приложениями:

— слот slotNewConnection используется для установления нового соединения с клиентом;

— слот slotReadClient используется для получения данных от клиента;

— метод sendToClient реализует отправку данных клиенту;

— метод ProcessRequest реализует обработку запросов, полученных от клиента.

Класс сервера ControlServer также содержит следующие методы для обработки результатов тестирования:

— метод OpenNewSession реализует сохранение в БД данных о начале тестирования конкретным пользователем;

— метод CloseSession реализует сохранение в БД данных о завершении тестирования конкретным пользователем и его результатах;

— слот slotUpdateResults реализует подготовку сохраненных результатов для отображения в соответствующем диалоговом окне.

Также класс сервера содержит следующие методы для обработки запросов клиентских приложений:

— метод GetRandomExpIDs производит выбор набора контекстов управления инцидентами ИБ с экспертными решениями для передачи клиенту;

— метод SendTakeItToClientMode1 осуществляет передачу клиенту по его запросу заданного количества описаний событий ИБ с экспертными решениями;

— метод SendTakeItToClientMode2 осуществляет передачу клиенту по его запросу заданного количества описаний инцидентов ИБ с экспертными решениями;

— метод SendTakeItToClientMode3 осуществляет передачу клиенту по его запросу заданного количества описаний контекстов выполняемой операции по реагированию на инцидент ИБ с экспертными решениями.

Слот slotTraining запускает один из выбранных режимов приобретения знаний экспертной системы. Для обеспечения функционирования режима приобретения знаний экспертной системы по первичному анализу событий ИБ используются следующие методы класса ControlServer:

— метод StartTrainingMode1 производит начало работы в данном режиме, включая всю необходимую подготовку;

— метод slotNextRowTrainingMode1 реализует подготовку нового описания события ИБ для отображения эксперту;

— метод slotSetArgsTrainingMode1 реализует сохранение решение эксперта по данному описанию события ИБ.

Методы StartTrainingMode2, slotNextRowTrainingMode2, slotSetArgsTrainingMode2 выполняют аналогичные операции для режима приобретения знаний по вторичному анализу инцидентов ИБ, а методы StartTrainingMode3, slotNextRowTrainingMode3, slotSetArgsTrainingMode3 – для режима приобретения знаний по реагированию на инциденты ИБ. Указанные методы для всех трех режимов используют для генерации контекста класс ContextGen, представляющий модуль модели среды управления инцидентами ИБ.

Был реализован класс диалогового окна главного меню управления сервером MenuForm. Графический интерфейс этого окна приведен на рисунке 17.

Нажатие кнопки «Начать обучение» инициирует начало работы экспертной системы в режиме приобретения знаний в соответствии с выбранным элементом из перечня под кнопкой.

Нажатие кнопки «Показать результаты» вызывает метод инициирует отображение всех сохраненных в БД результатов тестирования, в том числе информацию о незавершенных сессиях. Если при этом в поле «Имя:» будет введено имя пользователя, то будут отображены только результаты пользователя с заданным именем. Результаты отображаются в виде таблицы в нижней части диалогового окна.

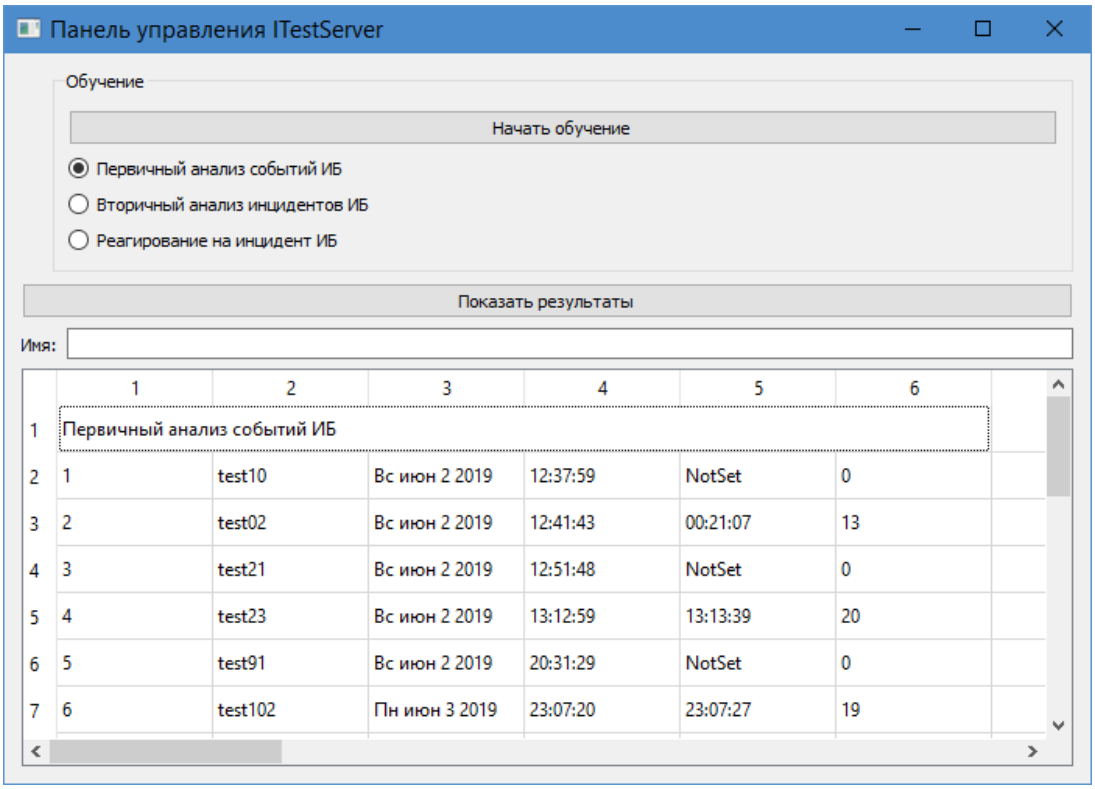

Рисунок 17 – Графический интерфейс главного меню управления сервером

Был реализован класс диалогового окна режима приобретения знаний по первичному анализу событий ИБ TrainingMode1Form. Графический интерфейс этого окна приведен на рисунке 18.

В верхней части окна отображается описание события ИБ, а в нижней расположены управляющие элементы. Кнопки «Инцидент» и «Не инцидент (событие ИБ)» инициируют передачу классу ControlServer решения эксперта для его дальнейшего сохранения. Кнопка «Пропустить этот набор» позволяет эксперту перейти к следующему описанию события ИБ без вынесения решения. Кнопка «Никогда не показывать этот набор» позволяет эксперту убрать описание события ИБ из числа отображаемых в этой и во всех следующих сессиях. Кнопка «Завершить обучение» закрывает диалоговое окно.

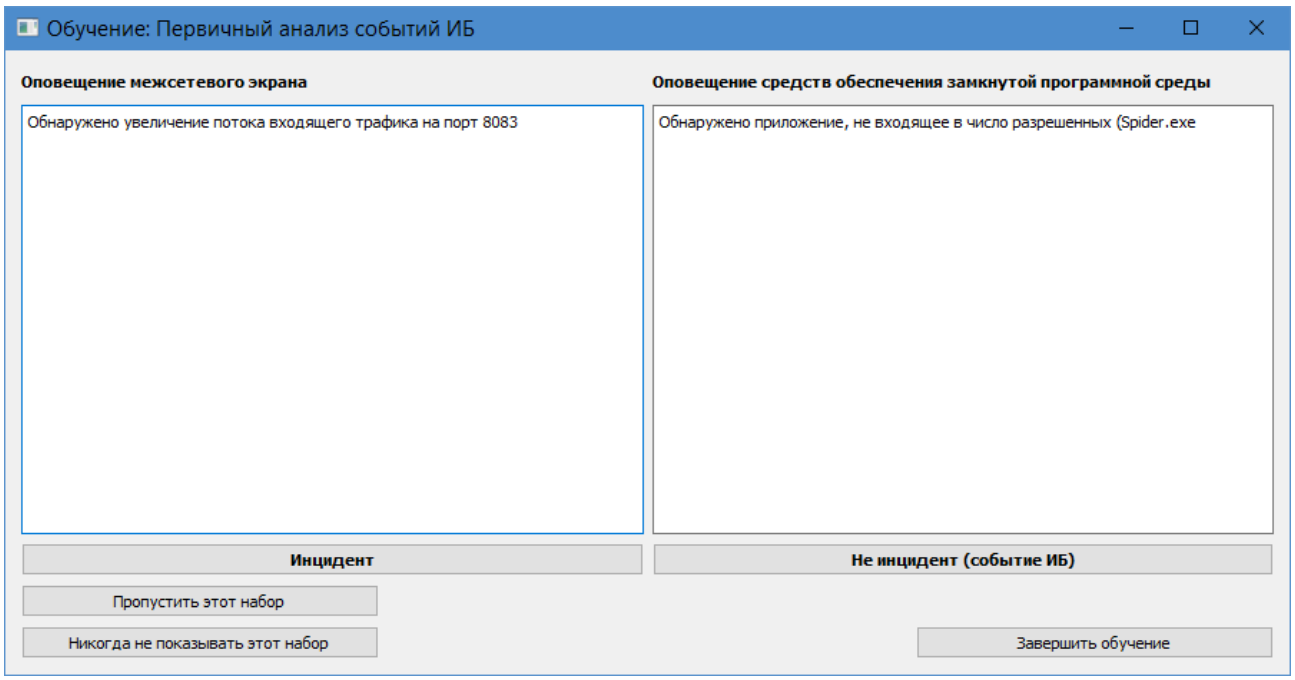

Рисунок 18 – Графический интерфейс режима приобретения знаний по первичному анализу событий ИБ

Был реализован класс диалогового окна режима приобретения знаний по вторичному анализу инцидентов ИБ TrainingMode2Form. Графический интерфейс этого окна приведен на рисунке 19.

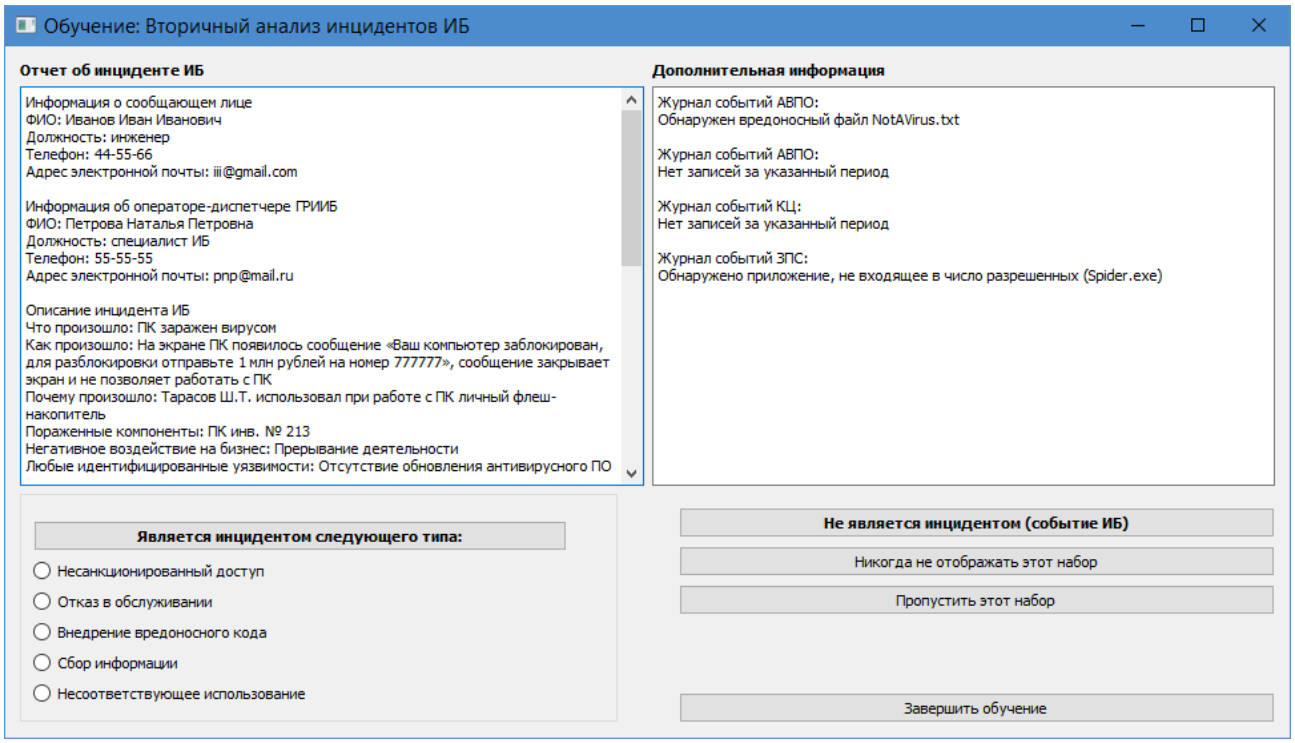

Рисунок 19 – Графический интерфейс режима приобретения знаний по вторичному анализу инцидентов ИБ

В верхней части окна отображается описание инцидента ИБ, а в нижней расположены управляющие элементы. Кнопки «Является инцидентом следующего типа:» и «Не является инцидентом (событие ИБ)» инициируют передачу классу ControlServer решения эксперта для его дальнейшего сохранения. Кнопка «Пропустить этот набор» позволяет эксперту перейти к следующему описанию события ИБ без вынесения решения. Кнопка «Никогда не показывать этот набор» позволяет эксперту убрать описание события ИБ из числа отображаемых в этой и во всех следующих сессиях. Кнопка «Завершить обучение» закрывает диалоговое окно.

Был реализован класс диалогового окна режима приобретения знаний по реагированию на инциденты ИБ TrainingMode3Form. Графический интерфейс этого окна приведен на рисунке 20.

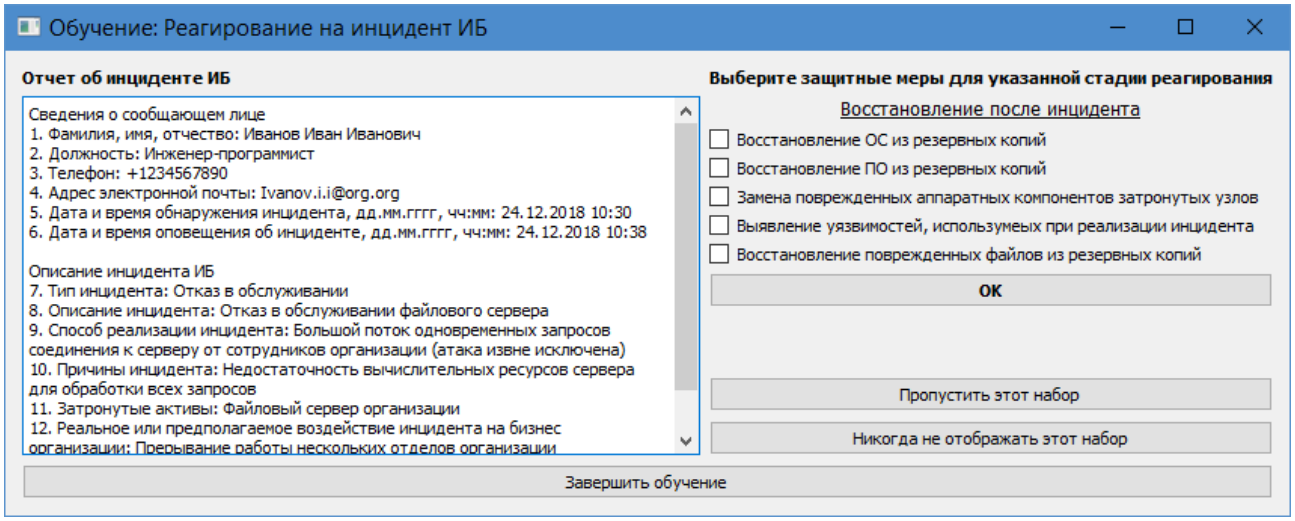

Рисунок 20 – Графический интерфейс режима приобретения знаний по реагированию на инциденты ИБ

В левой части окна отображается отчет об инциденте ИБ, а в правой представлен этап реагирования для данного контекста и набор мер. Кнопка «ОК» инициирует передачу классу ControlServer решения эксперта для его дальнейшего сохранения. Кнопка «Пропустить этот набор» позволяет эксперту перейти к следующему описанию события ИБ без вынесения решения. Кнопка «Никогда не показывать этот набор» позволяет эксперту убрать описание события ИБ из числа отображаемых в этой и во всех следующих сессиях. Кнопка «Завершить обучение» закрывает диалоговое окно.

6.2 Реализация клиентского компонента

Было реализовано приложение-клиент ITestClient экспертной системы, предназначенное для использования в среде операционных систем семейства Windows. Поскольку при создании клиентского приложения был использован фреймворк QT, для поддержки функционирования приложения в среде Windows необходимо наличие соответствующих библиотек указанного фреймворка. Также для обеспечения возможности установления соединения с сервером, в рабочем каталоге должен располагаться текстовый файл, содержащий настройки сервера, а именно: IP-адрес сервера и номер порта.

Структура проекта приложения ITestClient приведена на рисунке 21. Полный исходный код клиентского компонента ITestClient приведен в приложении В.

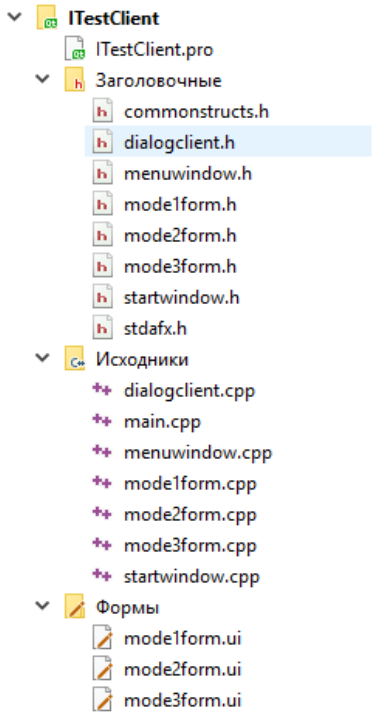

Рисунок 21 – Структура проекта приложения ITestClient

Для реализации клиент-серверной архитектуры был использован класс QTcpSocket фреймворка QT. Классы всех диалоговых окон были пронаследованы от класса QWidget.

Был реализован класс клиента DialogClient. Этот класс содержит следующие базовые методы для взаимодействия с серверным приложением:

— метод LoadSettings реализует загрузку и применение настроек сервера из текстового файла;

— метод SendToServer реализует отправку сообщения серверу;

— слот slotReadyRead реализует получение сообщения от сервера;

— метод ProcessRequest реализует обработку сообщений сервера;

— слот slotError реализует обработку ошибок соединения с сервером;

— слот slotConnected реализует подтверждение установления соединения с сервером.

Также класс DialogClient содержит следующие методы для работы с текущей сессией:

— слот slotNameRecieved реализует сохранение имени пользователя и инициирует запуск окна для выбора режима принятия решений;

— слот slotModeRecieved реализует сохранение информации о выбранном пользователем режима принятия решений;

— метод StartTesting реализует отправку серверу запроса на открытие новой сессии под заданным именем пользователя;

— метод StartSession реализует запуск режима принятия решений в соответствии с выбранным видом режима;

— слот slotAddPoint реализует увеличение количества правильных решений пользователя в течение сессии;

— слот slotCompleteSession реализует отправку серверу информации о завершении сессии и количестве правильных решений пользователя.

Было реализовано диалоговое окно начала работы StartWindow. Графический интерфейс этого окна приведен на рисунке 22.

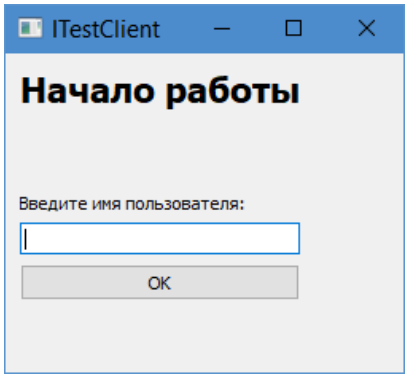

Рисунок 22 – Диалоговое окно начала работы

Нажатие кнопки «ОК» инициирует передачу имени пользователя, введенного в поле для ввода, классу DialogClient для сохранения и дальнейшего использования в рамках текущей сессии.

Было реализовано диалоговое окно выбора режима тестирования MenuWindow. Графический интерфейс этого окна приведен на рисунке 23.

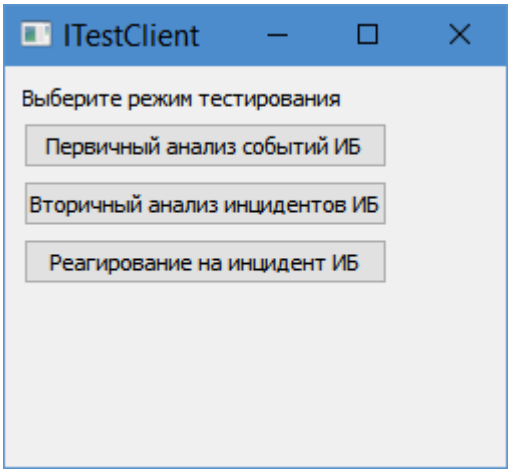

Рисунок 23 – Диалоговое окно выбора режима тестирования

Нажатие одной из трех представленных кнопок инициирует передачу информации о выбранном режиме принятия решений классу DialogClient для сохранения и дальнейшего использования в рамках текущей сессии.

Было реализовано диалоговое окно режима принятия решений по первичному анализу событий ИБ Mode1Form. Графический интерфейс этого окна приведен на рисунке 24.

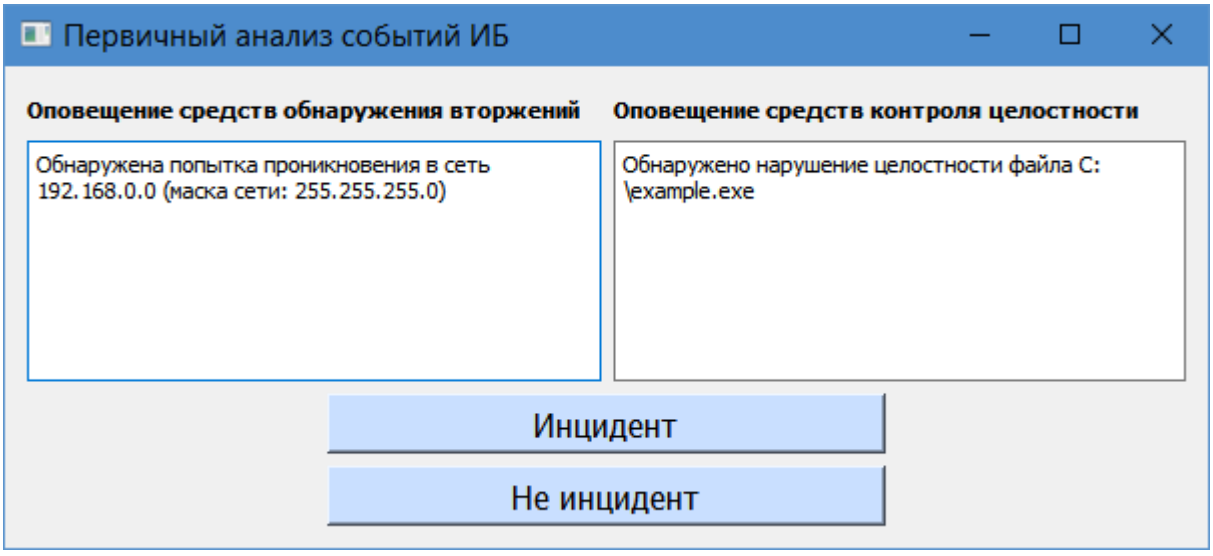

Рисунок 24 – Графический интерфейс режима принятия решений по первичному анализу событий ИБ

В верхней части окна отображается описание события ИБ, а в нижней расположены управляющие элементы. Кнопки «Инцидент» и «Не инцидент (событие ИБ)» инициируют сравнение решения пользователя с экспертным решением и, в случае идентичности этих решений, передачу классу DialogClient команды об увеличении числа правильных решений в текущей сессии. Пользователю предоставляется для оценки 20 описаний событий ИБ. Когда пользователем будут даны решения по всем событиям ИБ, диалоговое окно закроется и классу DialogClient будет передан сигнал о завершении сессии.

Был реализован класс диалогового окна режима принятия решений по вторичному анализу инцидентов ИБ Mode2Form. Графический интерфейс этого окна приведен на рисунке 25.

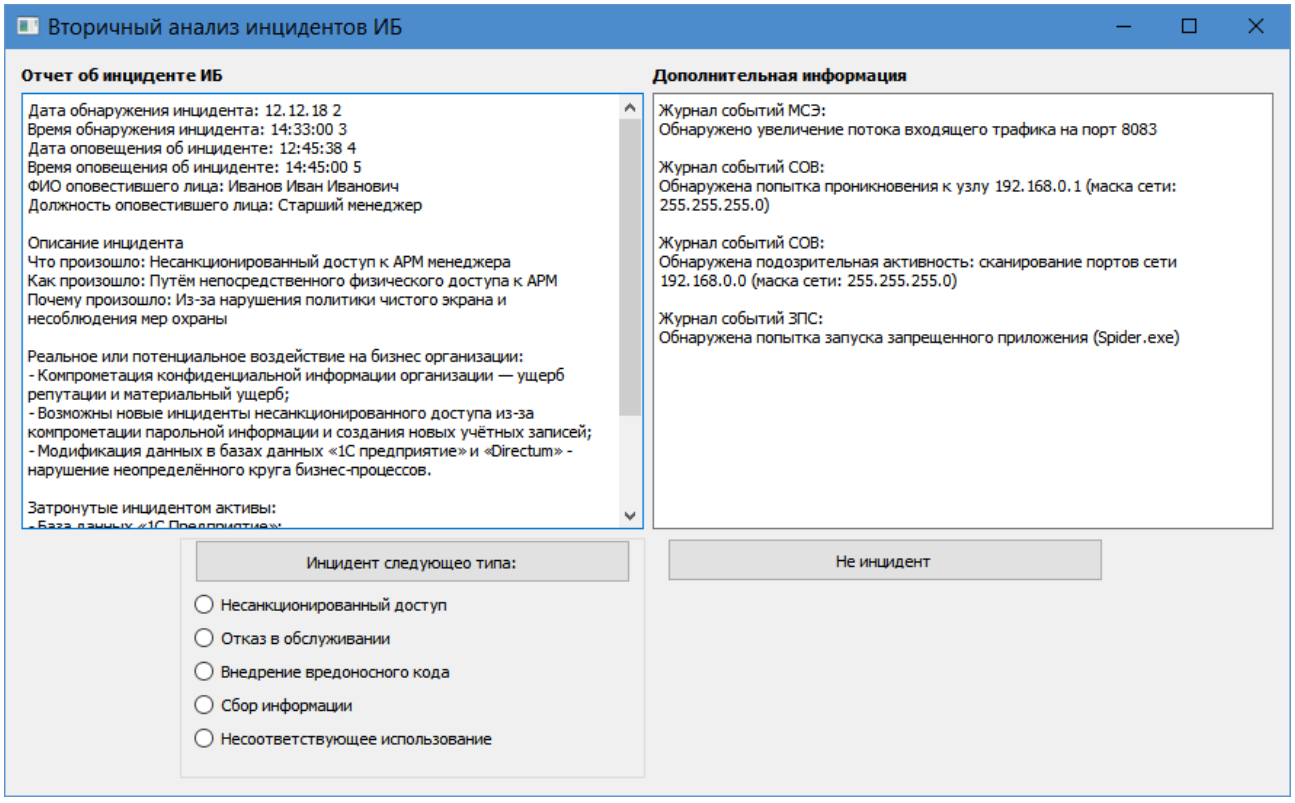

Рисунок 25 – Графический интерфейс режима принятия решений по вторичному анализу инцидентов ИБ

В верхней части окна отображается описание инцидента ИБ, а в нижней расположены управляющие элементы. Кнопки «Инцидент следующего типа:» и «Не инцидент» инициируют сравнение решения пользователя с экспертным решением и, в случае идентичности этих решений, передачу классу DialogClient

команды об увеличении числа правильных решений в текущей сессии. Пользователю предоставляется для оценки 20 описаний инцидентов ИБ. Когда пользователем будут даны решения по всем инцидентам ИБ, диалоговое окно закроется и классу DialogClient будет передан сигнал о завершении сессии.

Был реализован класс диалогового окна режима принятия решений по реагированию на инциденты ИБ Mode3Form. Графический интерфейс этого окна приведен на рисунке 26.

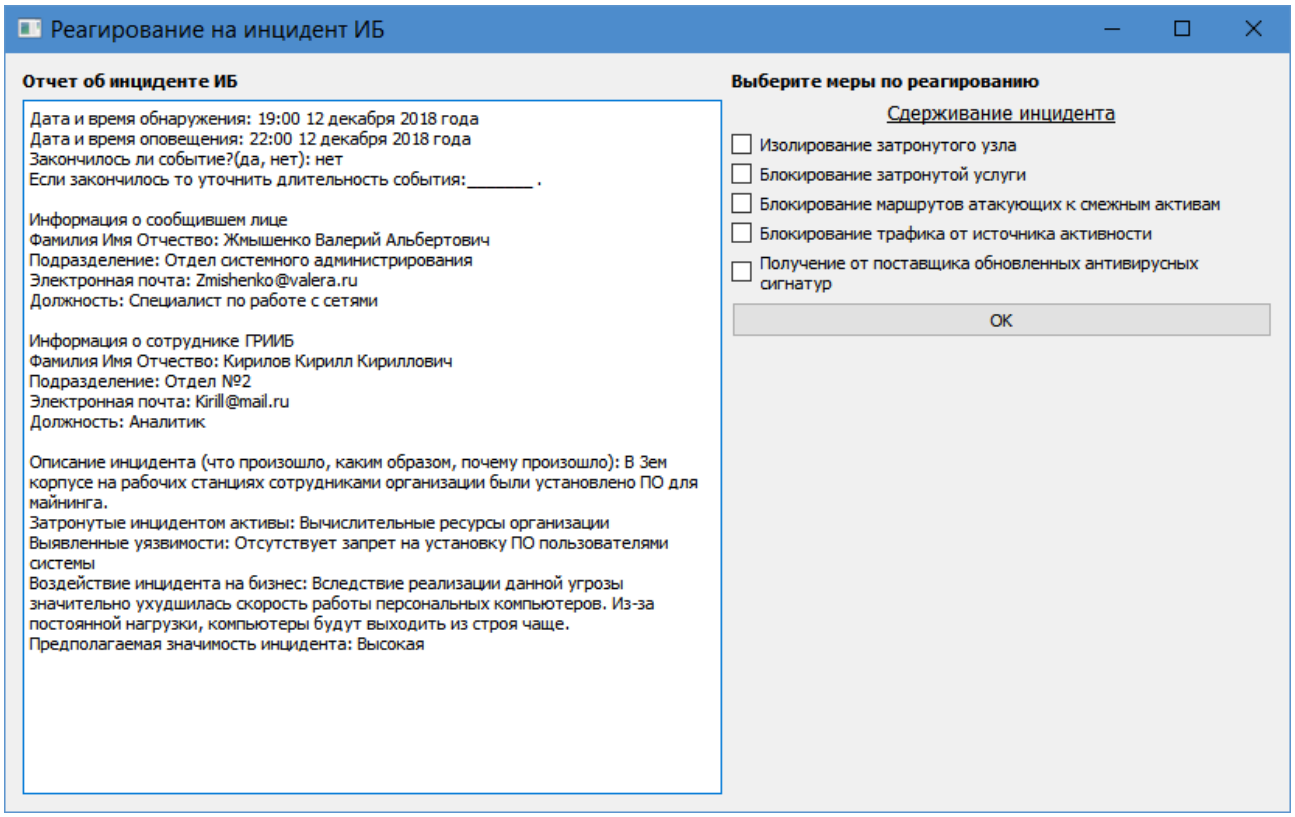

Рисунок 26 – Графический интерфейс режима приобретения знаний по реагированию на инциденты ИБ

В левой части окна отображается отчет об инциденте ИБ, а в правой представлен этап реагирования для данного контекста и набор мер. Кнопка «ОК» инициируют сравнение решения пользователя с экспертным решением и, в случае идентичности этих решений, передачу классу DialogClient команды об увеличении числа правильных решений в текущей сессии. Пользователю предоставляется для оценки 20 описаний инцидентов ИБ. Когда пользователем будут даны решения по всем инцидентам ИБ, диалоговое окно закроется и классу DialogClient будет передан сигнал о завершении сессии.

7 Вредные психофизиологические факторы, влияющие на органы зрения специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

При выполнении задач специалисты в области защиты информации подвержены воздействию опасных и вредных производственных факторов. Соответственно, специалисты должны быть осведомлены и проинструктированы о воздействии тех или иных негативных факторов.

Согласно ГОСТ 12.0.003-74 «ССБТ. Опасные и вредные производственные факторы. Классификация» [8] к вредным и опасным производственным факторам относятся, в том числе, следующие факторы:

- отсутствие или недостаток естественного света;
- недостаточная искусственная освещенность рабочей зоны;
- повышенная яркость света;
- повышенная контрастность;
- прямая и отраженная блескость;
- зрительное напряжение.

Поскольку деятельность специалиста в области защиты информации связана с обработкой больших объемов информации с использованием АРМ, реализацией необходимых решений в короткие сроки, а также работой с большим объемом документов в бумажном виде, все перечисленные факторы могут иметь место в рабочей обстановке специалиста и оказывать негативное воздействие на его органы зрения.

Продолжительная работа на АРМ (работа за монитором) способствует чрезмерной нагрузке на компоненты зрительного аппарата, а также способствует появлению усталости и переутомлению глазного анализатора, что является причиной большинства болезней органов зрения.

Такая работа вызывает заболевания глаз, а именно: боль и резь в глазах, их покраснения, возникновение пелены, раздвоение предметов, неприятное

мерцание, чрезмерную утомляемость и тому подобное. Это влечет за собой нарушение зрительных органов и снижение работоспособности специалиста.

Специалистам в области ИБ необходима регулярная проверка зрения, направленная на поддержание и улучшение здоровья глаз.

С целью снизить риски возникновения заболеваний глаз, важно соблюдать правила безопасности, также необходимо прибегать к методам, обеспечивающим расслабление глазного аппарата.

На основе выявленных опасных факторов, были выбраны методы и средства обеспечения безопасности работы на АРМ, описанные ниже.

Необходимо избегать излишнего напряжения глаз. Для их защиты не следует длительное время фокусироваться на одном и том же предмете. Рекомендуется, в качестве профилактики, время от времени отвлекаться и переводить взгляд на предметы вдали, что поспособствует четкости объектов на сетчатке, а также позволит сохранить упругость мышц глаз.

Успешно себя зарекомендовала методика У. Г. Бейтса Пальминга. Простая тренировка для глаз, направленная на расслабление мышц глаза и восстановление зрения. На глаза следует положить ладони обеих рук, таким образом, чтоб пальцы рук перекрещивались на лбу. Так необходимо находиться на протяжении 5 минут. Упражнение выполняется без усилий, чтобы не вызвать напряжения на глаза.

Моргание – простая и эффективная профилактика, обеспечивающая временный отдых для глаз. Слезная жидкость имеет дезинфицирующие и увлажняющие свойства:

— укрепление нервных волокон сетчатки;

— обеспечение расслабления глазных мышц;

— улучшение способности глаза фокусироваться на изображении.

Глазам необходим не только в периодический отдых и расслабление, но естественный свет. Чтобы предотвратить ухудшение зрения следует постоянно укреплять сетчатку. Каждый день следует начинать с принятия световых ванн (соляризации). Рекомендуется подставить закрытые глаза на несколько минут

солнцу, либо, при его отсутствии, другому источнику света. Это усилит кровообращение, нормализуются процессы в глазных мышцах, повысится вероятность сохранения зрения.

Значимый фактор, влияющий на остроту зрения и предотвращающий его ухудшение, это правильная освещенность. От искусственного освещения не должна чувствоваться резь в глазах, но оно должно быть ярким. Чтобы сохранить зрение при работе за письменным столом, рекомендуется использовать настольные лампы мощностью 60 Вт с плафонами. Источник искусственного освещения следует располагать слева (у правшей) в 50 см от документов.

Выполнение перечисленных упражнений для глаз поспособствует их расслаблению и тренировке, а также снизит воздействие вредных психофизиологических факторов, влияющих на работу.

Заключение

В ходе выполнения выпускной квалификационной были проанализированы 8 различных источников информации. Полученная из источников информация была структурирована и представлена в пояснительной записке, а также использована при выполнении технического задания на выпускную квалификационную работу.

Был осуществлен обзор основных понятий и процессов управления инцидентами информационной безопасности. Было составлено описание процессов обнаружения и оповещения о событиях ИБ, первичной оценки события ИБ и вторичной оценки инцидента ИБ, а также процессов сдерживания инцидентов ИБ, их устранения и восстановления после инцидентов ИБ. Для указанных процессов были изложены цели и особенности их осуществления.

Были рассмотрены особенности моделирования процессов управления инцидентами ИБ в рамках экспертной системы поддержки принятия решений. При этом были составлены требования к модели среды управления инцидентами ИБ. Требования составлялись с учетом целей и особенностей осуществления процессов управления инцидентами ИБ. Также была разработана модель среды управления инцидентами ИБ. Модель разрабатывалась с учетом требований к модели среды управления инцидентами ИБ. Затем была разработана модель экспертной системы поддержки принятия решений по управлению инцидентами ИБ.

Была спроектирована архитектура программного модуля, реализующего модель среды управления инцидентами информационной безопасности. Были построены алгоритмы процессов использования такой модели в различных ситуациях. Затем модель среды управления инцидентами информационной безопасности была реализована в виде класса на языке C++.

Была спроектирована архитектура экспертной системы поддержки принятия решений по управлению инцидентами информационной безопасности. Были составлено словесное описание алгоритмов функционирования такой

системы в различных режимах, а также диаграмма вариантов использования и диаграммы классов. Экспертная система поддержки принятия решений по управлению инцидентами информационной безопасности была реализована в виде клиент-серверного приложения на языке C++ с использованием фреймворка QT.

Результатом выпускной квалификационной работы является разработанная экспертная система поддержки принятия решений по управлению инцидентами информационной безопасности, ориентированная на ее использование в учебном процессе при подготовке специалистов по информационной безопасности для их обучения управлению инцидентами ИБ.

Таким образом, техническое задание на выпускную квалификационную работу было выполнено в полном объеме.

Список используемых источников

1 Джексон П. Введение в экспертные системы, 3-е издание / Питер Джексон. – М. : Вильямс, 2001. – 624 с.

2 ГОСТ Р ИСО/МЭК 27001-2006 Информационная технология (ИТ). Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Требования. [Текст]. – М. : Стандартинформ, 2008. – 64 с.

3 ГОСТ Р ИСО/МЭК ТО 18044-2007 Информационная технология (ИТ). Методы и средства обеспечения безопасности. Менеджмент инцидентов информационной безопасности. [Текст]. – М. : Стандартинформ, 2009. – 92 с.

4 Макаренко С. И. Интеллектуальные информационные системы: учебное пособие. / С. И. Макаренко – Ставрополь: СФ МГГУ им. М. А. Шолохова, 2009. – 206 с.

5 Интеллектуальные информационные системы. Краткое содержание курса. [Электронный ресурс] : Тема 1. ВВЕДЕНИЕ В ЭКСПЕРТНЫЕ СИСТЕМЫ. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ. – http://www.habarov.spb.ru/new_es/exp_sys/es01/es1.htm – режим доступа свободный — дата последнего обращения 18.06.2019.

6 Qt Documentation. [Электронный ресурс] : QTcpServer Class. – https://doc.qt.io/qt-5/qtcpserver.html – режим доступа свободный — дата последнего обращения 18.06.2019

7 Qt Documentation. [Электронный ресурс] : QTcpSocket Class. – https://doc.qt.io/qt-5/qtcpsocket.html – режим доступа свободный — дата последнего обращения 18.06.2019

8 ГОСТ 12.0.003-2015. Система стандартов по безопасности труда. Опасные и вредные производственные факторы. Классификация [Электронный ресурс] . — М.: Стандартинформ, 2016. - Режим доступа: http://www.internet-law.ru/gosts/gost/62075/. — Загл. с экрана.

Приложение А

(обязательное)

Исходный код класса ContextGen

Исходный код файла ContextGen.h:

```
#ifndef COTEXTGEN_H
#define COTEXTGEN_H

#include "stdafx.h"
#include "contextstructs.h"


struct SInfoSet{
    int info1;
    int info2;
    int info3;
    int info4;
};

struct SMeasuresSet{
    QString stage;
    int measure1;
    int measure2;
    int measure3;
    int measure4;
    int measure5;
};

struct SSignsSet{
    int info1;
    int info2;
};



class ContextGen
{
public:
    ContextGen();
    static void UpdateExperienceComboMode1(int amount);
    static void UpdateExperienceComboMode2(int amount);
    static void UpdateExperienceComboMode3(int amount);

private:
    static void GetRandomTableIDs(QVector<int>& vector, QString
tableName, int& amount);
    static void GetAllCombinations(QVector<int> IDs,
QVector<SInfoSet>& outVector);
    static void GetAllCombinations(QVector<int> IDs,
QVector<SMeasuresSet>& outVector);
```

```
    static void GetAllCombinations(QVector<int> IDs,
QVector<SSignsSet>& outVector);

    static QVector<SSignsSet> GetAllSignsSets();
    static QVector<SSignsSet> GetAllReportsSets();
    static void AddToDBAllSignsSets( QVector<SSignsSet>& allSets
);

    static QVector<SInfoSet> GetAllInfoSets();
    static void AddToDBAllInfoSetsForReport( QString& strReport,
QVector<SInfoSet>& allInfoSets );

    static QVector<SMeasuresSet> GetAllMeasuresSets(QString
stage);
    static void AddToDBAllMeasuresSetsForReport( QString&
strReport, QVector<SMeasuresSet>& measuresSets );

};

#endif // COTEXTGEN_H
```

Исходный код файла ContextGen.cpp:

```
#include "contextgen.h"

ContextGen::ContextGen()
{

}

void ContextGen::UpdateExperienceComboMode1(int amount)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_PRIMARY_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }
    QVector<SSignsSet> allSignsSets = GetAllSignsSets();
    if ( allSignsSets.size() == 0 )
    {
        QErrorMessage err;
        err.showMessage("Недостаточно признаков инцидентов в
базе");
         err.exec();
        return;
    }

    QVector<SSignsSet> allReports = GetAllReportsSets();
    if ( allReports.size() == 0 )
```

```
    {
        QErrorMessage err;
        err.showMessage("В базе нет ни одного отчета о событии
ИБ");
         err.exec();
        return;
    }

    int reportsAmount = amount / 5;
    int signsAmount = amount - reportsAmount;
    QVector<SSignsSet> fixed;
    for ( int i = 0; i < reportsAmount; i++ )
    {
        fixed.push_back( allReports[ rand() % allReports.size() ]
);
    }
    for ( int i = 0; i < signsAmount; i++ )
    {
        fixed.push_back( allSignsSets[ rand() %
allSignsSets.size() ] );
    }


     AddToDBAllSignsSets(fixed);
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ContextGen::UpdateExperienceComboMode2(int amount)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_SECONDARY_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }
    QVector<SInfoSet> allInfoSets = GetAllInfoSets();
    if ( allInfoSets.size() == 0 )
    {
        QErrorMessage err;
        err.showMessage("Недостаточно наборов дополнительных
сведений в базе");
         err.exec();
        return;
    }


    QSqlQuery query;
    if (!query.exec("SELECT id FROM Reports"))
    {
```

```
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }
    QVector<int> ReportsIDs;
    while (query.next())
    {
        ReportsIDs.push_back( query.value(0).toInt() );
    }

    if ( ReportsIDs.size() == 0 )
    {
        QErrorMessage err;
        err.showMessage("В базе нет ни одного отчета об инциденте
ИБ");
         err.exec();
        return;
    }

    if ( amount % ReportsIDs.size() )
    {
        amount += ( ReportsIDs.size() - (amount %
ReportsIDs.size()) );
    }
    int howMuchForReport = amount / ReportsIDs.size();

    for ( int reportsCount = 0; reportsCount < ReportsIDs.size();
reportsCount++ )
    {
        int currReport = ReportsIDs[reportsCount];

        if ( !query.exec("SELECT Report FROM Reports WHERE id=" +
QString::number(currReport)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
        QString strReport = query.value(0).toString();

        QVector<SInfoSet> cropped;
        for (int i = 0; i < howMuchForReport; i++)
        {
            cropped.push_back( allInfoSets[rand() %
allInfoSets.size()] );
        }

         AddToDBAllInfoSetsForReport(strReport, cropped);
```

```
    }
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ContextGen::UpdateExperienceComboMode3(int amount)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_PROTECTIVE_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }
    QSqlQuery query;
    if (!query.exec("SELECT id FROM Reports"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }
    QVector<int> ReportsIDs;
    while (query.next())
    {
        ReportsIDs.push_back( query.value(0).toInt() );
    }


    if ( ReportsIDs.size() == 0 )
    {
        QErrorMessage err;
        err.showMessage("В базе нет ни одного отчета об инциденте
ИБ");
         err.exec();
        return;
    }
    if ( amount % ReportsIDs.size() )
    {
        amount += ( ReportsIDs.size() - (amount %
ReportsIDs.size()) );
    }
    int howMuchForReport = amount / ReportsIDs.size();


    QVector<QString> stagesNames;
    stagesNames.push_back(STAGE_CTRL_NAME);
    stagesNames.push_back(STAGE_REM_NAME);
    stagesNames.push_back(STAGE_REC_NAME);


    QVector<SMeasuresSet> ctrlSets =
GetAllMeasuresSets(STAGE_CTRL_NAME);
```

```
    if ( ctrlSets.size() == 0 )
    {
        QErrorMessage err;
        err.showMessage("Недостаточно мер по сдерживанию в базе");
         err.exec();
        return;
    }
    QVector<SMeasuresSet> remSets =
GetAllMeasuresSets(STAGE_REM_NAME);
    if ( remSets.size() == 0 )
    {
        QErrorMessage err;
        err.showMessage("Недостаточно мер по устранению в базе");
         err.exec();
        return;
    }
    QVector<SMeasuresSet> recSets =
GetAllMeasuresSets(STAGE_REC_NAME);
    if ( recSets.size() == 0 )
    {
        QErrorMessage err;
        err.showMessage("Недостаточно мер по восстановлению в
базе");
         err.exec();
        return;
    }



    for ( int reportsCount = 0; reportsCount < ReportsIDs.size();
reportsCount++ )
    {
        int currReport = ReportsIDs[reportsCount];

        if ( !query.exec("SELECT Report FROM Reports WHERE id=" +
QString::number(currReport)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
        QString strReport = query.value(0).toString();

        QVector<SMeasuresSet> selected;
        for (int i = 0; i < howMuchForReport; i++)
        {
            QString stage = stagesNames[rand() %
stagesNames.size()];
```

```
            if ( stage == STAGE_CTRL_NAME )
            {
                selected.push_back( ctrlSets[ rand() %
ctrlSets.size() ] );
            }
            else if ( stage == STAGE_REM_NAME )
            {
                selected.push_back( remSets[ rand() %
remSets.size() ] );
            }
            else if ( stage == STAGE_REC_NAME )
            {
                selected.push_back( recSets[ rand() %
recSets.size() ] );
            }
        }
```

AddToDBAllMeasuresSetsForReport(*strReport*, *selected*);

```
    }
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ContextGen::GetRandomTableIDs(QVector<int>& vector, QString
tableName, int &amount)
{
    QSqlQuery query;
    if (!query.exec("SELECT id FROM " + tableName))
    {
        qDebug() << "Unable to execute select query: " <<
query.lastError().text();
        return;
    }

    int resNum = 0;
    while (query.next())
    {
        resNum++;
    }
    QSqlRecord rec = query.record();
    int idIndex = rec.indexOf("id");

    for (int i = 0; i < amount;)
    {
        int randNum = rand() % resNum;
        if (!query.seek( randNum ) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
            continue;
        }
```

```
        vector.push_back(query.value(idIndex).toInt());
        i++;
    }

    return;
}

void ContextGen::GetAllCombinations(QVector<int> IDs,
QVector<SInfoSet>& outVector)
{
    QVector<SInfoSet> doubleSets;
    for (int i = 0; i < IDs.size(); i++)
    {
        for ( int j = 0; j < IDs.size(); j++ )
        {
            if ( IDs[j] <= IDs[i] )
            {
                continue;
            }
            SInfoSet set;
            set.info1 = IDs[i];
            set.info2 = IDs[j];
            set.info3 = 0;
            set.info4 = 0;
            doubleSets.push_back(set);
        }
    }

    QVector<SInfoSet> tripleSets;
    for ( int i = 0; i < doubleSets.size(); i++ )
    {
        SInfoSet currSet = doubleSets[i];
        for ( int j = 0; j < IDs.size(); j++ )
        {
            if ( IDs[j] <= currSet.info2 )
            {
                continue;
            }
            SInfoSet newSet;
            newSet.info1 = currSet.info1;
            newSet.info2 = currSet.info2;
            newSet.info3 = IDs[j];
            newSet.info4 = 0;
            tripleSets.push_back(newSet);
        }
    }

    for ( int i = 0; i < tripleSets.size(); i++ )
    {
        SInfoSet currSet = tripleSets[i];
        for ( int j = 0; j < IDs.size(); j++ )
        {
```

```
            if ( IDs[j] <= currSet.info3 )
            {
                continue;
            }
            SInfoSet newSet;
            newSet.info1 = currSet.info1;
            newSet.info2 = currSet.info2;
            newSet.info3 = currSet.info3;
            newSet.info4 = IDs[j];
            outVector.push_back(newSet);
        }
    }
}

void ContextGen::GetAllCombinations(QVector<int> IDs,
QVector<SMeasuresSet> &outVector)
{
    QVector<SMeasuresSet> doubleSets;
    for (int i = 0; i < IDs.size(); i++)
    {
        for ( int j = 0; j < IDs.size(); j++ )
        {
            if ( IDs[j] <= IDs[i] )
            {
                continue;
            }
            SMeasuresSet set;
            set.measure1 = IDs[i];
            set.measure2 = IDs[j];
            set.measure3 = 0;
            set.measure4 = 0;
            set.measure5 = 0;
            doubleSets.push_back(set);
        }
    }

    QVector<SMeasuresSet> tripleSets;
    for ( int i = 0; i < doubleSets.size(); i++ )
    {
        SMeasuresSet currSet = doubleSets[i];
        for ( int j = 0; j < IDs.size(); j++ )
        {
            if ( IDs[j] <= currSet.measure2 )
            {
                continue;
            }
            SMeasuresSet newSet;
            newSet.measure1 = currSet.measure1;
            newSet.measure2 = currSet.measure2;
            newSet.measure3 = IDs[j];
            newSet.measure4 = 0;
            newSet.measure5 = 0;
            tripleSets.push_back(newSet);
```

```
        }
    }

    QVector<SMeasuresSet> quadSets;
    for ( int i = 0; i < tripleSets.size(); i++ )
    {
        SMeasuresSet currSet = tripleSets[i];
        for ( int j = 0; j < IDs.size(); j++ )
        {
            if ( IDs[j] <= currSet.measure3 )
            {
                continue;
            }
            SMeasuresSet newSet;
            newSet.measure1 = currSet.measure1;
            newSet.measure2 = currSet.measure2;
            newSet.measure3 = currSet.measure3;
            newSet.measure4 = IDs[j];
            newSet.measure5 = 0;
            quadSets.push_back(newSet);
        }
    }

    for ( int i = 0; i < quadSets.size(); i++ )
    {
        SMeasuresSet currSet = quadSets[i];
        for ( int j = 0; j < IDs.size(); j++ )
        {
            if ( IDs[j] <= currSet.measure4 )
            {
                continue;
            }
            SMeasuresSet newSet;
            newSet.measure1 = currSet.measure1;
            newSet.measure2 = currSet.measure2;
            newSet.measure3 = currSet.measure3;
            newSet.measure4 = currSet.measure4;
            newSet.measure5 = IDs[j];
            outVector.push_back(newSet);
        }
    }
}

void ContextGen::GetAllCombinations(QVector<int> IDs,
QVector<SSignsSet> &outVector)
{
    outVector.clear();
    for (int i = 0; i < IDs.size(); i++)
    {
        for ( int j = 0; j < IDs.size(); j++ )
        {
            if ( IDs[j] <= IDs[i] )
            {
```

```
                continue;
            }
            SSignsSet set;
            set.info1 = IDs[i];
            set.info2 = IDs[j];
            outVector.push_back(set);
        }
    }
}


QVector<SSignsSet> ContextGen::GetAllSignsSets()
{
    QVector<SSignsSet> sets;
    QSqlQuery query;
    if (!query.exec("SELECT id FROM Signs"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return sets;
    }
    QVector<int> InfoIDs;
    while (query.next())
    {
        InfoIDs.push_back( query.value(0).toInt() );
    }

    GetAllCombinations(InfoIDs, sets);
    return sets;
}

QVector<SSignsSet> ContextGen::GetAllReportsSets()
{
    QVector<SSignsSet> sets;
    QSqlQuery query;
    if (!query.exec("SELECT id FROM Reports"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return sets;
    }

    while (query.next())
    {
        int reportId = query.value(0).toInt();
        SSignsSet set;
        set.info1 = reportId;
        set.info2 = reportId;
        sets.push_back(set);
    }
    return sets;
}
```

```
void ContextGen::AddToDBAllSignsSets(QVector<SSignsSet> &allSets)
{
    QSqlQuery query;
    for ( int count = 0; count < allSets.size(); count++ )
    {
        QString strInfo1;
        QString strInfo2;
        SSignsSet signsSet = allSets[count];
        if ( signsSet.info1 == signsSet.info2 )
        {
            if ( !query.exec("SELECT Report, Info1, Info2, Info3
FROM Reports WHERE id=" + QString::number(signsSet.info1)) )
            {
                qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
                return;
            }
            if ( !query.seek(0) )
            {
                qDebug() << "Unable to seek result: " <<
query.lastError().text();
            }
            strInfo1 = "Отчет о событии ИБ$" +
query.value(0).toString();
            strInfo2 = query.value(1).toString() + '$' +
query.value(2).toString() + '$' + query.value(3).toString();
        }
        else
        {
            if ( !query.exec("SELECT Sign FROM Signs WHERE id=" +
QString::number(signsSet.info1)) )
            {
                qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
                return;
            }
            if ( !query.seek(0) )
            {
                qDebug() << "Unable to seek result: " <<
query.lastError().text();
            }
            strInfo1 = query.value(0).toString();

            if ( !query.exec("SELECT Sign FROM Signs WHERE id=" +
QString::number(signsSet.info2)) )
            {
                qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
                return;
            }
            if ( !query.seek(0) )
            {
```

```
                qDebug() << "Unable to seek result: " <<
query.lastError().text();
            }
            strInfo2 = query.value(0).toString();
        }


        QString queryStr = "SELECT * FROM MyExperience WHERE
Info1='" + strInfo1 + "' AND Info2='" + strInfo2 + "'";
        if ( !query.exec( ) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( query.seek(0))
        {
            if ( (query.value(0).toString() == strInfo1) &&
                 (query.value(1).toString() == strInfo2) )
            {
                continue;
            }
        }

        queryStr = "INSERT INTO MyExperience (Info1, Info2)
VALUES('" + strInfo1 + "', '" + strInfo2 +"')";
        if (!query.exec(queryStr))
        {
            qDebug() << "Unable to do insert opeation: " <<
query.lastError().text();
        }
    }
    query.clear();
}


QVector<SInfoSet> ContextGen::GetAllInfoSets()
{
    QVector<SInfoSet> sets;
    QSqlQuery query;
    if (!query.exec("SELECT id FROM Info"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return sets;
    }
    QVector<int> InfoIDs;
    while (query.next())
    {
        InfoIDs.push_back( query.value(0).toInt() );
    }

    GetAllCombinations(InfoIDs, sets);
```

```
    return sets;
}

void ContextGen::AddToDBAllInfoSetsForReport(QString &strReport,
QVector<SInfoSet> &allInfoSets)
{
    QSqlQuery query;

    for ( int infoCount = 0; infoCount < allInfoSets.size();
infoCount++ )
    {
        SInfoSet infoSet = allInfoSets[infoCount];

        if ( !query.exec("SELECT Info FROM Info WHERE id=" +
QString::number(infoSet.info1)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
        QString strInfo1 = query.value(0).toString();

        if ( !query.exec("SELECT Info FROM Info WHERE id=" +
QString::number(infoSet.info2)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
        QString strInfo2 = query.value(0).toString();

        if ( !query.exec("SELECT Info FROM Info WHERE id=" +
QString::number(infoSet.info3)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
```

```
        QString strInfo3 = query.value(0).toString();

        if ( !query.exec("SELECT Info FROM Info WHERE id=" +
QString::number(infoSet.info4)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
        QString strInfo4 = query.value(0).toString();

        QString queryStr = "SELECT * FROM MyExperience WHERE
Report='" + strReport + "' AND ";
        queryStr += "Info1='" + strInfo1 + "' AND Info2='" +
strInfo2 + "' AND ";
        queryStr += "Info3='" + strInfo3 + "' AND Info4='" +
strInfo4 +"'";
        if ( !query.exec( ) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( query.seek(0))
        {
            if ( (query.value(1).toString() == strReport) &&
                 (query.value(2).toString() == strInfo1) &&
                 (query.value(3).toString() == strInfo2) &&
                 (query.value(4).toString() == strInfo3) &&
                 (query.value(4).toString() == strInfo4) )
            {
                continue;
            }
        }

        QString strF =
                  "INSERT INTO MyExperience (Report, Info1, Info2,
Info3, Info4) "
                  "VALUES('%1', '%2', '%3', '%4', '%5')";
        queryStr = strF.arg(strReport)
                       .arg(strInfo1)
                       .arg(strInfo2)
                       .arg(strInfo3)
                       .arg(strInfo4);
        if (!query.exec(queryStr))
        {
            qDebug() << "Unable to do insert opeation: " <<
query.lastError().text();
```

```
        }
    }
    query.clear();
}

QVector<SMeasuresSet> ContextGen::GetAllMeasuresSets( QString
stage )
{
    QVector<SMeasuresSet> sets;
    QSqlQuery query;

    QString strQuery = "SELECT id FROM ";
    if ( stage == STAGE_CTRL_NAME )
    {
        strQuery += "Control";
    }
    else if (stage == STAGE_REM_NAME)
    {
        strQuery += "Removal";
    }
    else if (stage == STAGE_REC_NAME)
    {
        strQuery += "Recovery";
    }
    if (!query.exec(strQuery))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return sets;
    }


    QVector<int> measuresIDs;
    while (query.next())
    {
        measuresIDs.push_back( query.value(0).toInt() );
    }

    GetAllCombinations(measuresIDs, sets);
    for ( int i = 0; i < sets.size(); i++ )
    {
        sets[i].stage = stage;
    }
    return sets;
}

void ContextGen::AddToDBAllMeasuresSetsForReport(QString
&strReport, QVector<SMeasuresSet> &measuresSets)
{
    QSqlQuery query;
    for ( int measureCount = 0; measureCount <
measuresSets.size(); measureCount++ )
    {
```

```
        SMeasuresSet infoSet = measuresSets[measureCount];
        QString tableName;
        if ( infoSet.stage == STAGE_CTRL_NAME )
        {
            tableName = "Control";
        }
        else if ( infoSet.stage == STAGE_REM_NAME )
        {
            tableName = "Removal";
        }
        else if ( infoSet.stage == STAGE_REC_NAME )
        {
            tableName = "Recovery";
        }

        if ( !query.exec("SELECT Measure FROM " + tableName + "
WHERE id=" + QString::number(infoSet.measure1)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
        QString strInfo1 = query.value(0).toString();

        if ( !query.exec("SELECT Measure FROM " + tableName + "
WHERE id=" + QString::number(infoSet.measure2)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
        QString strInfo2 = query.value(0).toString();

        if ( !query.exec("SELECT Measure FROM " + tableName + "
WHERE id=" + QString::number(infoSet.measure3)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
```

```
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
        QString strInfo3 = query.value(0).toString();

        if ( !query.exec("SELECT Measure FROM " + tableName + "
WHERE id=" + QString::number(infoSet.measure4)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
        QString strInfo4 = query.value(0).toString();

        if ( !query.exec("SELECT Measure FROM " + tableName + "
WHERE id=" + QString::number(infoSet.measure5)) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }
        QString strInfo5 = query.value(0).toString();

        QString allMeasuresStr = strInfo1 + '$' + strInfo2 + '$' +
strInfo3;
        allMeasuresStr += ( '$' + strInfo4 + '$' + strInfo5 );


        QString queryStr = "SELECT * FROM MyExperience WHERE
Report='" + strReport + "' AND ";
        queryStr += "Stage='" + infoSet.stage + "' AND Measures='"
+ allMeasuresStr + "'";
        if ( !query.exec( ) )
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        if ( query.seek(0))
        {
            if ( (query.value(1).toString() == strReport) &&
                 (query.value(2).toString() == infoSet.stage) &&
                 (query.value(3).toString() == allMeasuresStr) )
```

```
            {
                continue;
            }
        }

        QString strF =
                 "INSERT INTO MyExperience (Report, Stage,
Measures) VALUES('%1', '%2', '%3')";
        queryStr = strF.arg(strReport)
                     .arg(infoSet.stage)
                     .arg(allMeasuresStr);
        if (!query.exec(queryStr))
        {
            qDebug() << "Unable to do insert opeation: " <<
query.lastError().text();
        }
    }
    query.clear();
}
```

Приложение Б

(обязательное)

Исходный код приложения ITestServer

Исходный код файла main.cpp:

```
#include "controlserver.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    ControlServer server(65035);
    server.show();

    return a.exec();
}
```

Исходный код файла stdafx.h:

```
#include <QWidget>
#include <QTcpServer>
#include <QTcpSocket>
#include <QMessageBox>
#include <QTextEdit>
#include <QVBoxLayout>
#include <QLabel>
#include <QTime>
#include <QDate>
#include <QtSql>
#include <QtSql/QSqlDatabase>
#include <QSqlQuery>
#include <QStandardItemModel>
#include <QErrorMessage>
```

Исходный код файла commonstructs.h:

```
#ifndef COMMONSTRUCTS_H
#define COMMONSTRUCTS_H

#include "stdafx.h"

typedef struct SPacket
{
    quint16 opCode;
    QString message;
}*PSPacket;

typedef enum EOpCode {
    EOp_Ping = 0x00,
    EOp_RegistrateMode1,
```

```
    EOp_RegistrareMode2,
    EOp_RegistrateMode3,
    EOp_SessionOpened,

    EOp_GiveMeMode1,
    EOp_TakeItMode1,
    EOp_GiveMeMode2,
    EOp_TakeItMode2,
    EOp_GiveMeMode3,
    EOp_TakeItMode3,

    EOp_FinishMode1,
    EOp_FinishMode2,
    EOp_FinishMode3
}OpCode;


#endif // COMMONSTRUCTS_H
```

Исходный код файла contextstructs.h:

```
#ifndef CONTEXTSTRUCTS_H
#define CONTEXTSTRUCTS_H

#define NOT_INC_NUM     0
#define NSD_TYPE_NUM    1
#define OO_TYPE_NUM     2
#define VVK_TYPE_NUM    3
#define SI_TYPE_NUM     4
#define NI_TYPE_NUM     5

#define MIN_UNCHECKED_RECORDS 100

#define DB_RESULTS_NAME     "Sources\\Results.db"
#define DB_PRIMARY_NAME     "Sources\\PrimaryAnalysis.db"
#define DB_SECONDARY_NAME   "Sources\\SecondaryAnalysis.db"
#define DB_PROTECTIVE_NAME  "Sources\\ProtectiveMeasures.db"

#define GET_MEASURE1_FLAG(flags) (flags & 0x1)
#define GET_MEASURE2_FLAG(flags) (flags & 0x2)
#define GET_MEASURE3_FLAG(flags) (flags & 0x4)
#define GET_MEASURE4_FLAG(flags) (flags & 0x8)
#define GET_MEASURE5_FLAG(flags) (flags & 0x10)

#define SET_MEASURE1_FLAG(flags) (flags |= 0x1)
#define SET_MEASURE2_FLAG(flags) (flags |= 0x2)
#define SET_MEASURE3_FLAG(flags) (flags |= 0x4)
#define SET_MEASURE4_FLAG(flags) (flags |= 0x8)
#define SET_MEASURE5_FLAG(flags) (flags |= 0x10)

#define STAGE_CTRL_NAME "Сдерживание инцидента"
#define STAGE_REM_NAME  "Устранение инцидента"
#define STAGE_REC_NAME  "Восстановление после инцидента"
```

```
#endif // CONTEXTSTRUCTS_H
```

Исходный код файла controlserver.h:

```
#include "stdafx.h"
#include "commonstructs.h"
#include "menuform.h"
#include "trainingmode2form.h"
#include "trainingmode3form.h"
#include "trainingmode1form.h"
#include "contextgen.h"

#define MAX_REPORTS_IN_MODE1    5
#define MIN_REPORTS_IN_MODE1    3



class ControlServer : public QWidget
{
    Q_OBJECT
public:
    ControlServer(quint16 nPort);
    ~ControlServer() {}

public slots:
    virtual void slotNewConnection();
    void slotReadClient();
    void slotUpdateResults();
    void slotTraining(int mode);
    void slotNextRowTrainingMode1();
    void slotSetArgsTrainingMode1(int id, int isIncident, int
show);
    void slotNextRowTrainingMode2();
    void slotSetArgsTrainingMode2(int id, int incidentType, int
show);
    void slotNextRowTrainingMode3();
    void slotSetArgsTrainingMode3(int id, int flags, int show);


private:
    void sendToClient(QTcpSocket* pSocket, PSPacket psPacket);
    void ProcessRequest(QTcpSocket* pSocket, EOpCode opcode,
QString& message);
    int OpenNewSession(QString& username, QString tableName);
    void CloseSession(int id, int points, QString tableName);

    void GetRandomExpIDs(QVector<int>& vector, int& amount);

    void SendTakeItToClientMode1(QTcpSocket* pSocket, int amount);
    void SendTakeItToClientMode2(QTcpSocket* pSocket, int amount);
```

```
    void SendTakeItToClientMode3(QTcpSocket* pSocket, int amount);

    void StartTrainingMode1();
    void StartTrainingMode2();
    void StartTrainingMode3();


private:
    QTcpServer m_server;
    quint16 m_nNextBlockSize;
    QTextEdit* m_pTxt;
    QVBoxLayout* m_pLayout;
    MenuForm* m_pMenuWindow;
    TrainingMode1Form* m_pMode1TrainingWindow;
    TrainingMode2Form* m_pMode2TrainingWindow;
    TrainingMode3Form* m_pMode3TrainingWindow;
};
```

Исходный код файла controlserver.cpp:

```
#include "controlserver.h"

ControlServer::ControlServer(quint16 nPort)
    : m_server(this)
{
    m_nNextBlockSize = 0;
    if ( !m_server.listen(QHostAddress::Any, nPort))
    {
        QMessageBox::critical(
                    nullptr,
                    "Server error",
                    "Failed to start the server: " +
m_server.errorString());
        m_server.close();
        return;
    }

    connect(&m_server,
            SIGNAL(newConnection()),
            this,
            SLOT(slotNewConnection()));

    m_pTxt = new QTextEdit;
    m_pTxt->setReadOnly(true);

    m_pLayout = new QVBoxLayout;
    m_pLayout->addWidget(new QLabel("<H1>Server<H1>"));
    m_pLayout->addWidget(m_pTxt);
    setLayout(m_pLayout);

    m_pMenuWindow = new MenuForm;
    qDebug() << connect(
        m_pMenuWindow,
```

```
        SIGNAL(signalNeedResults()),
        this,
        SLOT(slotUpdateResults()));
    qDebug() << connect(
        m_pMenuWindow,
        SIGNAL(signalTraining(int)),
        this,
        SLOT(slotTraining(int)));
    m_pMenuWindow->show();
}

void ControlServer::slotNewConnection()
{
     m_pTxt->append("Connection established");
     QTcpSocket* pClientSocket
=m_server.nextPendingConnection();

    connect(
        pClientSocket,
        SIGNAL(disconnected()),
        pClientSocket,
        SLOT(deleteLater()));

    connect(
        pClientSocket,
        SIGNAL(readyRead()),
        this,
        SLOT(slotReadClient()));
}

void ControlServer::slotReadClient()
{
    QTcpSocket* pClientSocket =
static_cast<QTcpSocket*>(sender());
    QDataStream in(pClientSocket);

    while (true)
    {
        if ( !m_nNextBlockSize )
        {
              if ( pClientSocket->bytesAvailable() <
sizeof(quint16) )
            {
                break;
            }
            in >> m_nNextBlockSize;
        }

         if ( pClientSocket->bytesAvailable() <
m_nNextBlockSize )
        {
            break;
```

```
        }

        QTime time;
        QString str;
        quint16 opcode;
        QByteArray dataBlock;
        quint8 byte[100];
        unsigned char uchTemp;

        for ( quint16 count = 0; count < m_nNextBlockSize; count++
)
        {
            in >> byte[count];
            uchTemp = byte[count];
            dataBlock.append((char)uchTemp);
        }

        QDataStream dataIn(&dataBlock, QIODevice::ReadOnly);
        dataIn >> time >> opcode >> str;
        m_pTxt->append("Получено: OpCode = " +
QString::number(opcode) + ", Message = \'" + str + "\'");
         ProcessRequest(pClientSocket, (EOpCode)opcode,
str);

        m_nNextBlockSize = 0;
    }

    return;
}

void ControlServer::slotUpdateResults()
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_RESULTS_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }

    QSqlQuery query;
    if (!query.exec("SELECT * FROM PrimaryAnalysis"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }
    QSqlRecord rec = query.record();

    SResultRow sRow;
    while (query.next())
    {
```

```
        sRow.id = query.value(rec.indexOf("id")).toInt();
        sRow.username =
query.value(rec.indexOf("Username")).toString();
        sRow.date = query.value(rec.indexOf("Date")).toString();
        sRow.startTime =
query.value(rec.indexOf("StartTime")).toString();
        sRow.completionTime =
query.value(rec.indexOf("CompletionTime")).toString();
        sRow.points =
query.value(rec.indexOf("ResultPoints")).toInt();
        m_pMenuWindow->AddResultRow(&sRow, 1);
    }


    if (!query.exec("SELECT * FROM SecondaryAnalysis"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }
    rec = query.record();

    while (query.next())
    {
        sRow.id = query.value(rec.indexOf("id")).toInt();
        sRow.username =
query.value(rec.indexOf("Username")).toString();
        sRow.date = query.value(rec.indexOf("Date")).toString();
        sRow.startTime =
query.value(rec.indexOf("StartTime")).toString();
        sRow.completionTime =
query.value(rec.indexOf("CompletionTime")).toString();
        sRow.points =
query.value(rec.indexOf("ResultPoints")).toInt();
        m_pMenuWindow->AddResultRow(&sRow, 2);
    }

    if (!query.exec("SELECT * FROM ProtectiveMeasures"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }
    rec = query.record();

    while (query.next())
    {
        sRow.id = query.value(rec.indexOf("id")).toInt();
        sRow.username =
query.value(rec.indexOf("Username")).toString();
        sRow.date = query.value(rec.indexOf("Date")).toString();
        sRow.startTime =
query.value(rec.indexOf("StartTime")).toString();
```

```
        sRow.completionTime =
query.value(rec.indexOf("CompletionTime")).toString();
        sRow.points =
query.value(rec.indexOf("ResultPoints")).toInt();
        m_pMenuWindow->AddResultRow(&sRow, 3);
    }

    m_pMenuWindow->ShowResults();
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ControlServer::slotTraining(int mode)
{
    switch (mode)
    {
    case 1:
        StartTrainingMode1();
        break;
    case 2:
        StartTrainingMode2();
        break;
    case 3:
        StartTrainingMode3();
        break;
    default:
        break;
    }
}

void ControlServer::slotNextRowTrainingMode1()
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_PRIMARY_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }

    QSqlQuery query;
    if (!query.exec("SELECT id FROM MyExperience WHERE Show=1 AND
Checked=0"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }

    int amount = 0;
    while ( query.next() )
    {
```

```
            amount++;
        }

        if (!amount)
        {
            query.clear();
            db.close();
            QSqlDatabase::removeDatabase("QSQLITE");

ContextGen::UpdateExperienceComboMode1(MIN_UNCHECKED_RECORDS);

            db = QSqlDatabase::addDatabase("QSQLITE");
            db.setDatabaseName(DB_PRIMARY_NAME);
            if (!db.open())
            {
                qDebug() << "Unable to open DB";
                return;
            }
            if (!query.exec("SELECT id FROM MyExperience WHERE Show=1
AND Checked=0"))
            {
                qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
                return;
            }
            amount = 0;
            while ( query.next() )
            {
                amount++;
            }

        }

        while ( true )
        {
            if ( !query.seek(rand() % amount) )
            {
                qDebug() << "Unable to seek result: " <<
query.lastError().text();
                continue;
            }
            int id = query.value(0).toInt();
            if (!query.exec("SELECT * FROM MyExperience WHERE id=" +
QString::number(id)))
            {
                qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
                continue;
            }

            break;
        }
```

```
    SExpRecordMode1 sRecord;
    QSqlRecord rec = query.record();
    if ( !query.seek(0) )
    {
        qDebug() << "Unable to seek result: " <<
query.lastError().text();
    }
    sRecord.id = query.value(rec.indexOf("id")).toInt();
    sRecord.info1 = query.value(rec.indexOf("Info1")).toString();
    sRecord.info2 = query.value(rec.indexOf("Info2")).toString();
    m_pMode1TrainingWindow->NewRecordInfo(&sRecord);
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ControlServer::slotSetArgsTrainingMode1(int id, int
isIncident, int show)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_PRIMARY_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }

    QString strF =
            "UPDATE MyExperience SET IsIncident=%1, Checked=1,
Show=%2 "
            "WHERE id=" + QString::number(id);
    QString queryStr = strF.arg(isIncident)
                .arg(show);
    QSqlQuery query;
    if (!query.exec(queryStr))
    {
        qDebug() << "Unable to do update opeation: " <<
query.lastError().text();
        return;
    }
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ControlServer::slotNextRowTrainingMode2()
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_SECONDARY_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
```

```
    }

    QSqlQuery query;
    if (!query.exec("SELECT id FROM MyExperience WHERE Show=1 AND
Checked=0"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }

    int amount = 0;
    while ( query.next() )
    {
        amount++;
    }

    if (!amount)
    {
        query.clear();
        db.close();
        QSqlDatabase::removeDatabase("QSQLITE");

ContextGen::UpdateExperienceComboMode2(MIN_UNCHECKED_RECORDS);
        db = QSqlDatabase::addDatabase("QSQLITE");
        db.setDatabaseName(DB_SECONDARY_NAME);
        if (!db.open())
        {
            qDebug() << "Unable to open DB";
            return;
        }
        if (!query.exec("SELECT id FROM MyExperience WHERE Show=1
AND Checked=0"))
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            return;
        }
        amount = 0;
        while ( query.next() )
        {
            amount++;
        }
    }

    while ( true )
    {
        if ( !query.seek(rand() % amount) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
            continue;
        }
```

```
        int id = query.value(0).toInt();
        if (!query.exec("SELECT * FROM MyExperience WHERE id=" +
QString::number(id)))
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            continue;
        }

        break;
    }

    SExpRecordMode2 sRecord;
    QSqlRecord rec = query.record();
    if ( !query.seek(0) )
    {
        qDebug() << "Unable to seek result: " <<
query.lastError().text();
    }
    sRecord.id = query.value(rec.indexOf("id")).toInt();
    sRecord.report =
query.value(rec.indexOf("Report")).toString();
    sRecord.info1 = query.value(rec.indexOf("Info1")).toString();
    sRecord.info2 = query.value(rec.indexOf("Info2")).toString();
    sRecord.info3 = query.value(rec.indexOf("Info3")).toString();
    sRecord.info4 = query.value(rec.indexOf("Info4")).toString();
    m_pMode2TrainingWindow->NewRecordInfo(&sRecord);

    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ControlServer::slotSetArgsTrainingMode2(int id, int
incidentType, int show)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_SECONDARY_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }

    QString strF =
            "UPDATE MyExperience SET IncidentType=%1, Checked=1,
Show=%2 "
            "WHERE id=" + QString::number(id);
    QString queryStr = strF.arg(incidentType)
                .arg(show);
    QSqlQuery query;
    if (!query.exec(queryStr))
    {
```

```
        qDebug() << "Unable to do update opeation: " <<
query.lastError().text();
        return;
    }
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ControlServer::slotNextRowTrainingMode3()
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_PROTECTIVE_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }

    QSqlQuery query;
    if (!query.exec("SELECT id FROM MyExperience WHERE Show=1 AND
Checked=0"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }

    int amount = 0;
    while ( query.next() )
    {
        amount++;
    }

    if (!amount)
    {
        query.clear();
        db.close();
        QSqlDatabase::removeDatabase("QSQLITE");

ContextGen::UpdateExperienceComboMode3(MIN_UNCHECKED_RECORDS);
        db = QSqlDatabase::addDatabase("QSQLITE");
        db.setDatabaseName(DB_PROTECTIVE_NAME);
        if (!db.open())
        {
            qDebug() << "Unable to open DB";
            return;
        }
        if (!query.exec("SELECT id FROM MyExperience WHERE Show=1
AND Checked=0"))
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
```

```
            return;
        }
        amount = 0;
        while ( query.next() )
        {
            amount++;
        }
    }

    while ( true )
    {
        if ( !query.seek(rand() % amount) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
            continue;
        }
        int id = query.value(0).toInt();
        if (!query.exec("SELECT * FROM MyExperience WHERE id=" +
QString::number(id)))
        {
            qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
            continue;
        }

        break;
    }

    SExpRecordMode3 sRecord;
    QSqlRecord rec = query.record();
    if ( !query.seek(0) )
    {
        qDebug() << "Unable to seek result: " <<
query.lastError().text();
    }
    sRecord.id = query.value(rec.indexOf("id")).toInt();
    sRecord.report =
query.value(rec.indexOf("Report")).toString();
    sRecord.stage = query.value(rec.indexOf("Stage")).toString();

    QString allMeasures =
query.value(rec.indexOf("Measures")).toString();
    QStringList mesList = allMeasures.split('$');
    if (mesList.size() < 5 )
    {
        qDebug() << "Something is wrong with Iteration message.
Size: " << mesList.size();
    }
    sRecord.measure1 = mesList.at(0);
    sRecord.measure2 = mesList.at(1);
    sRecord.measure3 = mesList.at(2);
    sRecord.measure4 = mesList.at(3);
```

```
    sRecord.measure5 = mesList.at(4);

    m_pMode3TrainingWindow->NewRecordInfo(&sRecord);
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ControlServer::slotSetArgsTrainingMode3(int id, int flags,
int show)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_PROTECTIVE_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }

    QString queryStr =
            "UPDATE MyExperience SET Flags=" +
QString::number(flags) + ", Checked=1, Show=" +
QString::number(show);
    queryStr += " WHERE id=" + QString::number(id);
    QSqlQuery query;
    if (!query.exec(queryStr))
    {
        qDebug() << "Unable to do update opeation: " <<
query.lastError().text();
        return;
    }
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ControlServer::sendToClient(QTcpSocket *pSocket, PSPacket
psPacket)
{
    QByteArray dataBlock;
    QDataStream out(&dataBlock, QIODevice::WriteOnly);
    out << quint16(0) << QTime::currentTime() << quint16(psPacket-
>opCode) << psPacket->message;
     out.device()->seek(0);
    out << quint16(dataBlock.size() - sizeof(quint16));

    if (pSocket->write(dataBlock) != qint64(dataBlock.size()))
    {
        m_pTxt->append("Ошибка передачи сообщения клиенту");
        return;
    }
    m_pTxt->append("Передано: OpCode = " +
QString::number(psPacket->opCode) + ", Message = \'" + psPacket-
>message + "\'");
```

```
}

void ControlServer::ProcessRequest(QTcpSocket* pSocket, EOpCode
opcode, QString &message)
{
    PSPacket psPacket = new SPacket;
    QStringList args;
    switch ( opcode )
    {
    case EOp_RegistrateMode1:
        psPacket->opCode = EOp_SessionOpened;
         psPacket->message =
QString::number(OpenNewSession(message,
"PrimaryAnalysis"));
        sendToClient(pSocket, psPacket);
        break;

    case EOp_RegistrareMode2:
        psPacket->opCode = EOp_SessionOpened;
         psPacket->message =
QString::number(OpenNewSession(message,
"SecondaryAnalysis"));
        sendToClient(pSocket, psPacket);
        break;

    case EOp_RegistrateMode3:
        psPacket->opCode = EOp_SessionOpened;
         psPacket->message =
QString::number(OpenNewSession(message,
"ProtectiveMeasures"));
        sendToClient(pSocket, psPacket);
        break;

    case EOp_GiveMeMode1:
        SendTakeItToClientMode1(pSocket, message.toInt());
        break;

    case EOp_GiveMeMode2:
        SendTakeItToClientMode2(pSocket, message.toInt());
        break;

    case EOp_GiveMeMode3:
        SendTakeItToClientMode3(pSocket, message.toInt());
        break;

    case EOp_FinishMode1:
        args.clear();
        args = message.split('|');
        if (args.size() < 2 )
        {
            qDebug() << "Something is wrong with Iteration
message";
```

```
        }
        CloseSession(args.at(0).toInt(), args.at(1).toInt(),
"PrimaryAnalysis");
        break;

    case EOp_FinishMode2:
        args.clear();
        args = message.split('|');
        if (args.size() < 2 )
        {
            qDebug() << "Something is wrong with Iteration
message";
        }
        CloseSession(args.at(0).toInt(), args.at(1).toInt(),
"SecondaryAnalysis");
        break;

    case EOp_FinishMode3:
        args.clear();
        args = message.split('|');
        if (args.size() < 2 )
        {
            qDebug() << "Something is wrong with Iteration
message";
        }
        CloseSession(args.at(0).toInt(), args.at(1).toInt(),
"ProtectiveMeasures");
        break;

    default:
        break;
    }
    return;
}

int ControlServer::OpenNewSession(QString &username, QString
tableName)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_RESULTS_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return INT_MAX;
    }

    QString strF =
              "INSERT INTO " + tableName + " (Username, Date,
StartTime) "
              "VALUES('%1', '%2', '%3');";
    QString queryStr = strF.arg(username)
                .arg(QDate::currentDate().toString())
```

```
                .arg(QTime::currentTime().toString());
    QSqlQuery query;
    if (!query.exec(queryStr))
    {
        qDebug() << "Unable to do insert opeation: " <<
query.lastError().text();
        return INT_MAX;
    }

    if (!query.exec("SELECT id FROM " + tableName + " WHERE
Username=\'" + username + "\' AND CompletionTime='NotSet'"))
    {
        qDebug() << "Unable to execute select query: " <<
query.lastError().text();
        return INT_MAX;
    }

    QSqlRecord rec = query.record();
    if ( !query.seek(0) )
    {
        qDebug() << "Unable to seek result: " <<
query.lastError().text();
    }
    int id = query.value(rec.indexOf("id")).toInt();
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
    return id;
}

void ControlServer::CloseSession(int id, int points, QString
tableName)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_RESULTS_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }

    QString strF =
              "UPDATE " + tableName + " SET CompletionTime='%1',
ResultPoints=%2 "
              "WHERE id=" + QString::number(id);
    QString queryStr = strF.arg(QTime::currentTime().toString())
                .arg(QString::number(points));
    QSqlQuery query;
    if (!query.exec(queryStr))
    {
        qDebug() << "Unable to do update opeation: " <<
query.lastError().text();
        return;
```

```
    }
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}



void ControlServer::SendTakeItToClientMode1(QTcpSocket *pSocket,
int amount)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_PRIMARY_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }


    QVector<int> recordsIDs;
     GetRandomExpIDs(recordsIDs, amount );
    if ( recordsIDs.size() < amount )
    {
        qDebug() << "Wrong vector size";
        return;
    }

    for (int i = 0; i < amount; i++)
    {
        QSqlQuery query;
        if (!query.exec("SELECT * FROM MyExperience WHERE id=" +
QString::number(recordsIDs[i])))
        {
            qDebug() << "Unable to execute select query: " <<
query.lastError().text();
            return;
        }

        QSqlRecord rec = query.record();
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }

        QString msgStr =
query.value(rec.indexOf("Info1")).toString();
        msgStr += '|';
        msgStr += query.value(rec.indexOf("Info2")).toString();
        msgStr += '|';
        msgStr += QString::number(
query.value(rec.indexOf("IsIncident")).toInt() );
```

```
        SPacket sPacket;
        sPacket.opCode = EOp_TakeItMode1;
        sPacket.message = msgStr;
        sendToClient(pSocket, &sPacket);
    }

    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ControlServer::SendTakeItToClientMode2(QTcpSocket *pSocket,
int amount)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_SECONDARY_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }

    QVector<int> recordsIDs;
    GetRandomExpIDs(recordsIDs, amount );
    if ( recordsIDs.size() < amount )
    {
        qDebug() << "Wrong vector size";
        return;
    }

    for (int i = 0; i < amount; i++)
    {
        QSqlQuery query;
        if (!query.exec("SELECT * FROM MyExperience WHERE id=" +
QString::number(recordsIDs[i])))
        {
            qDebug() << "Unable to execute select query: " <<
query.lastError().text();
            return;
        }

        QSqlRecord rec = query.record();
        if ( !query.seek(0) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }

        QString msgStr =
query.value(rec.indexOf("Report")).toString();
        msgStr += '|';
        msgStr += query.value(rec.indexOf("Info1")).toString();
```

```
        msgStr += '|';
        msgStr += query.value(rec.indexOf("Info2")).toString();
        msgStr += '|';
        msgStr += query.value(rec.indexOf("Info3")).toString();
        msgStr += '|';
        msgStr += query.value(rec.indexOf("Info4")).toString();
        msgStr += '|';
        msgStr += QString::number(
query.value(rec.indexOf("IncidentType")).toInt() );

        SPacket sPacket;
        sPacket.opCode = EOp_TakeItMode2;
        sPacket.message = msgStr;
        sendToClient(pSocket, &sPacket);
    }

    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ControlServer::SendTakeItToClientMode3(QTcpSocket *pSocket,
int amount)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_PROTECTIVE_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }

    QVector<int> recordsIDs;
    GetRandomExpIDs(recordsIDs, amount );
    if ( recordsIDs.size() < amount )
    {
        qDebug() << "Wrong vector size";
        return;
    }

    for (int i = 0; i < amount; i++)
    {
        QSqlQuery query;
        if (!query.exec("SELECT * FROM MyExperience WHERE id=" +
QString::number(recordsIDs[i])))
        {
            qDebug() << "Unable to execute select query: " <<
query.lastError().text();
            return;
        }

        QSqlRecord rec = query.record();
        if ( !query.seek(0) )
```

```
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
        }

        QString msgStr =
query.value(rec.indexOf("Report")).toString();
        msgStr += '|';
        msgStr += query.value(rec.indexOf("Stage")).toString();
        msgStr += '|';
        msgStr += query.value(rec.indexOf("Measures")).toString();
        msgStr += '|';
        msgStr += query.value(rec.indexOf("Flags")).toString();

        SPacket sPacket;
        sPacket.opCode = EOp_TakeItMode3;
        sPacket.message = msgStr;
        sendToClient(pSocket, &sPacket);
    }

    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");
}

void ControlServer::StartTrainingMode1()
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_PRIMARY_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }
    QSqlQuery query;
    if (!query.exec("SELECT id FROM MyExperience WHERE Checked=0
AND Show=1"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }

    int amount = 0;
    while ( query.next() )
    {
        amount++;
    }
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");

    if ( amount < MIN_UNCHECKED_RECORDS )
    {
```

```
ContextGen::UpdateExperienceComboMode1(MIN_UNCHECKED_RECORDS -
amount);
    }

     m_pMode1TrainingWindow = new TrainingMode1Form;
     qDebug() << connect(
         m_pMode1TrainingWindow,
         SIGNAL(signalSetArgs(int, int, int)),
         this,
         SLOT(slotSetArgsTrainingMode1(int, int, int)));
     qDebug() << connect(
         m_pMode1TrainingWindow,
         SIGNAL(signalGetNext()),
         this,
         SLOT(slotNextRowTrainingMode1()));
     m_pMode1TrainingWindow->show();


     slotNextRowTrainingMode1();
}


void ControlServer::StartTrainingMode2()
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_SECONDARY_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }
    QSqlQuery query;
    if (!query.exec("SELECT id FROM MyExperience WHERE Checked=0
AND Show=1"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }

    int amount = 0;
    while ( query.next() )
    {
        amount++;
    }
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");

    if ( amount < MIN_UNCHECKED_RECORDS )
    {
```

```
ContextGen::UpdateExperienceComboMode2(MIN_UNCHECKED_RECORDS -
amount);
    }

     m_pMode2TrainingWindow = new TrainingMode2Form;
     qDebug() << connect(
         m_pMode2TrainingWindow,
         SIGNAL(signalSetArgs(int, int, int)),
         this,
         SLOT(slotSetArgsTrainingMode2(int, int, int)));
     qDebug() << connect(
         m_pMode2TrainingWindow,
         SIGNAL(signalGetNext()),
         this,
         SLOT(slotNextRowTrainingMode2()));
     m_pMode2TrainingWindow->show();

     slotNextRowTrainingMode2();
}

void ControlServer::StartTrainingMode3()
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(DB_PROTECTIVE_NAME);
    if (!db.open())
    {
        qDebug() << "Unable to open DB";
        return;
    }
    QSqlQuery query;
    if (!query.exec("SELECT id FROM MyExperience WHERE Checked=0
AND Show=1"))
    {
        qDebug() << "Unable to do select opeation: " <<
query.lastError().text();
        return;
    }

    int amount = 0;
    while ( query.next() )
    {
        amount++;
    }
    db.close();
    QSqlDatabase::removeDatabase("QSQLITE");

    if ( amount < MIN_UNCHECKED_RECORDS )
    {

ContextGen::UpdateExperienceComboMode3(MIN_UNCHECKED_RECORDS -
amount);
```

```
    }

     m_pMode3TrainingWindow = new TrainingMode3Form;
     qDebug() << connect(
         m_pMode3TrainingWindow,
         SIGNAL(signalSetArgs(int, int, int)),
         this,
         SLOT(slotSetArgsTrainingMode3(int, int, int)));
     qDebug() << connect(
         m_pMode3TrainingWindow,
         SIGNAL(signalGetNext()),
         this,
         SLOT(slotNextRowTrainingMode3()));
     m_pMode3TrainingWindow->show();


     slotNextRowTrainingMode3();
}

void ControlServer::GetRandomExpIDs(QVector<int> &vector, int
&amount)
{
    QSqlQuery query;
    if (!query.exec("SELECT id FROM MyExperience WHERE Checked=1
AND Show=1"))
    {
        qDebug() << "Unable to execute select query: " <<
query.lastError().text();
        return;
    }

    int resNum = 0;
    while (query.next())
    {
        resNum++;
    }
    QSqlRecord rec = query.record();
    int idIndex = rec.indexOf("id");

    for (int i = 0; i < amount;)
    {
        int randNum = rand() % resNum;
        if (!query.seek( randNum ) )
        {
            qDebug() << "Unable to seek result: " <<
query.lastError().text();
            continue;
        }

        vector.push_back(query.value(idIndex).toInt());
        i++;
    }
```

```
    return;
}
```

Исходный код файла menuform.h:

```
#ifndef MENUFORM_H
#define MENUFORM_H

#include "stdafx.h"

struct SResultRow{
    int id;
    QString username;
    QString date;
    QString startTime;
    QString completionTime;
    int points;
};

namespace Ui {
class MenuForm;
}

class MenuForm : public QWidget
{
    Q_OBJECT

public:
    explicit MenuForm(QWidget *parent = nullptr);
    ~MenuForm();
    void ShowResults();
    void AddResultRow(SResultRow* psRow, quint8 modeNum);

signals:
    void signalNeedResults();
    void signalTraining(int mode);

private slots:
    void on_btnShowResults_clicked();
    void on_btnTraining_clicked();

private:
    void ResetControls();
    void CropResultsForName(QString username);

private:
    Ui::MenuForm *ui;
    QVector<SResultRow> m_mode1Results;
    QVector<SResultRow> m_mode2Results;
    QVector<SResultRow> m_mode3Results;
};

#endif // MENUFORM_H
```

Исходный код файла menuform.cpp:

```
#include "menuform.h"
#include "ui_menuform.h"

MenuForm::MenuForm(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::MenuForm)
{
    ui->setupUi(this);
    ResetControls();
}

MenuForm::~MenuForm()
{
    delete ui;
}

void MenuForm::ShowResults()
{
    static bool isFirst = true;
    if ( !isFirst )
    {
         ui->resultTable->model()->removeRows(0, ui-
>resultTable->model()->rowCount() );
    }

    if ( ui->lineUsername->text() != "" )
    {
        CropResultsForName( ui->lineUsername->text() );
    }

    int rows = 2*3;
    rows += (m_mode1Results.size()? m_mode1Results.size() : 1);
    rows += (m_mode2Results.size()? m_mode2Results.size() : 1);
    rows += (m_mode3Results.size()? m_mode3Results.size() : 1);

    int rowCount = 0;
    QVector<int> spanRows;
    QStandardItemModel *model = new QStandardItemModel(6, rows,
ui->resultTable);


    QStandardItem*item = new QStandardItem("Первичный анализ
событий ИБ");
    model->setItem(rowCount, 0, item);
    spanRows.push_back(rowCount);
    rowCount++;

    if ( !m_mode1Results.size() )
    {
```

```
        item = new QStandardItem("Нет результатов");
        model->setItem(rowCount, 0, item);
        spanRows.push_back(rowCount);
        rowCount++;
    }
    else
    {
        for ( int i = 0; i < m_mode1Results.size(); i++)
        {
            item = new
QStandardItem(QString::number(m_mode1Results[i].id));
            model->setItem(rowCount, 0, item);
            item = new QStandardItem(m_mode1Results[i].username);
            model->setItem(rowCount, 1, item);
            item = new QStandardItem(m_mode1Results[i].date);
            model->setItem(rowCount, 2, item);
            item = new QStandardItem(m_mode1Results[i].startTime);
            model->setItem(rowCount, 3, item);
            item = new
QStandardItem(m_mode1Results[i].completionTime);
            model->setItem(rowCount, 4, item);
            item = new
QStandardItem(QString::number(m_mode1Results[i].points));
            model->setItem(rowCount, 5, item);
            rowCount++;
        }
    }

    item = new QStandardItem("Вторичный анализ инцидентов ИБ");
    model->setItem(rowCount, 0, item);
    spanRows.push_back(rowCount);
    rowCount++;

    if ( !m_mode2Results.size() )
    {
        item = new QStandardItem("Нет результатов");
        model->setItem(rowCount, 0, item);
        spanRows.push_back(rowCount);
        rowCount++;
    }
    else
    {
        for ( int i = 0; i < m_mode2Results.size(); i++)
        {
            item = new
QStandardItem(QString::number(m_mode2Results[i].id));
            model->setItem(rowCount, 0, item);
            item = new QStandardItem(m_mode2Results[i].username);
            model->setItem(rowCount, 1, item);
            item = new QStandardItem(m_mode2Results[i].date);
            model->setItem(rowCount, 2, item);
            item = new QStandardItem(m_mode2Results[i].startTime);
            model->setItem(rowCount, 3, item);
```

```
            item = new
QStandardItem(m_mode2Results[i].completionTime);
            model->setItem(rowCount, 4, item);
            item = new
QStandardItem(QString::number(m_mode2Results[i].points));
            model->setItem(rowCount, 5, item);
            rowCount++;
        }
    }

    item = new QStandardItem("Выбор защитных мер");
    model->setItem(rowCount, 0, item);
    spanRows.push_back(rowCount);
    rowCount++;

    if ( !m_mode3Results.size() )
    {
        item = new QStandardItem("Нет результатов");
        model->setItem(rowCount, 0, item);
        spanRows.push_back(rowCount);
        rowCount++;
    }
    else
    {
        for ( int i = 0; i < m_mode3Results.size(); i++)
        {
            item = new
QStandardItem(QString::number(m_mode3Results[i].id));
            model->setItem(rowCount, 0, item);
            item = new QStandardItem(m_mode3Results[i].username);
            model->setItem(rowCount, 1, item);
            item = new QStandardItem(m_mode3Results[i].date);
            model->setItem(rowCount, 2, item);
            item = new QStandardItem(m_mode3Results[i].startTime);
            model->setItem(rowCount, 3, item);
            item = new
QStandardItem(m_mode3Results[i].completionTime);
            model->setItem(rowCount, 4, item);
            item = new
QStandardItem(QString::number(m_mode3Results[i].points));
            model->setItem(rowCount, 5, item);
            rowCount++;
        }
    }

    ui->resultTable->setModel(model);
    isFirst = false;
    for ( int i = 0; i < spanRows.size(); i++ )
    {
        ui->resultTable->setSpan(spanRows[i], 0, 1, 6);
    }

    ui->resultTable->update();
```

```
}

void MenuForm::AddResultRow(SResultRow *psRow, quint8 modeNum)
{
    switch (modeNum) {
    case 1:
        m_mode1Results.push_back(*psRow);
        break;
    case 2:
        m_mode2Results.push_back(*psRow);
        break;
    case 3:
        m_mode3Results.push_back(*psRow);
        break;
    default:
        QErrorMessage err(this);
        err.showMessage("Ошибка обработки результатов (Код
0x501");
         err.exec();
        return;
    }
}

void MenuForm::ResetControls()
{
    ui->lineUsername->clear();
}

void MenuForm::CropResultsForName(QString username)
{
    QVector<int> tmp;
    for ( int i = 0; i < m_mode1Results.size(); i++ )
    {
        if ( m_mode1Results[i].username != username )
        {
            tmp.push_back(i);
        }
    }
    for ( int i = (tmp.size() - 1); i >= 0; i-- )
    {
        m_mode1Results.remove( tmp[i] );
    }


    tmp.clear();
    for ( int i = 0; i < m_mode2Results.size(); i++ )
    {
        if ( m_mode2Results[i].username != username )
        {
            tmp.push_back(i);
        }
    }
    for ( int i = (tmp.size() - 1); i >= 0; i-- )
```

```
    {
        m_mode2Results.remove( tmp[i] );
    }


    tmp.clear();
    for ( int i = 0; i < m_mode3Results.size(); i++ )
    {
        if ( m_mode3Results[i].username != username )
        {
            tmp.push_back(i);
        }
    }
    for ( int i = (tmp.size() - 1); i >= 0; i-- )
    {
        m_mode3Results.remove( tmp[i] );
    }
}

void MenuForm::on_btnShowResults_clicked()
{
    m_mode1Results.clear();
    m_mode2Results.clear();
    m_mode3Results.clear();
    emit signalNeedResults();
}

void MenuForm::on_btnTraining_clicked()
{
    int mode = 0;
    if ( ui->radioMode1->isChecked() )
    {
        mode = 1;
    }
    else if ( ui->radioMode2->isChecked() )
    {
        mode = 2;
    }
    else if ( ui->radioMode3->isChecked() )
    {
        mode = 3;
    }
    else {
        QErrorMessage err(this);
        err.showMessage("Не выбран режим обучения");
         err.exec();
        return;
    }

    emit signalTraining(mode);
}
```

Исходный код файла trainingmode1form.h:

```
#ifndef TRAININGMODE1FORM_H
#define TRAININGMODE1FORM_H

#include "stdafx.h"
#include "commonstructs.h"
#include "contextstructs.h"

struct SExpRecordMode1 {
    int id;
    QString info1;
    QString info2;
};

namespace Ui {
class TrainingMode1Form;
}

class TrainingMode1Form : public QWidget
{
    Q_OBJECT

public:
    explicit TrainingMode1Form(QWidget *parent = nullptr);
    ~TrainingMode1Form();
    void NewRecordInfo(SExpRecordMode1* psRecord);

signals:
    void signalGetNext();
    void signalSetArgs(int id, int isIncident, int show);

private slots:
    void on_btnInc_clicked();
    void on_btnNotInc_clicked();
    void on_btnSkip_clicked();
    void on_btnNotShow_clicked();
    void on_btnExit_clicked();

private:
    Ui::TrainingMode1Form *ui;
    SExpRecordMode1 currentRecord;
};

#endif // TRAININGMODE1FORM_H
```

Исходный код файла trainingmode1form.cpp:

```
#include "trainingmode1form.h"
#include "ui_trainingmode1form.h"

TrainingMode1Form::TrainingMode1Form(QWidget *parent) :
    QWidget(parent),
```

```
    ui(new Ui::TrainingMode1Form)
{
    ui->setupUi(this);
}

TrainingMode1Form::~TrainingMode1Form()
{
    delete ui;
}

void TrainingMode1Form::NewRecordInfo(SExpRecordMode1 *psRecord)
{
    currentRecord = *psRecord;

    QStringList info1List = currentRecord.info1.split('$');
    if ( info1List.size() < 2 )
    {
        qDebug() << "Something is wrong with info1Liat. Size: " <<
info1List.size();
    }
    ui->lblLeft->setText(info1List.at(0));
    ui->txtLeft->setText(info1List.at(1));

    QStringList info2List = currentRecord.info2.split('$');
    if ( info2List.size() < 2 )
    {
        qDebug() << "Something is wrong with info1Liat. Size: " <<
info1List.size();
    }

    if ( info2List.size() == 2 )
    {
        ui->lblRight->setText(info2List.at(0));
        ui->txtRight->setText(info2List.at(1));
    }
    else
    {
        ui->lblRight->setText("Дополнительная информация");
        ui->txtRight->setText(info2List.at(0) + "\n\n" +
info2List.at(1) + "\n\n" +info2List.at(2));
    }
}

void TrainingMode1Form::on_btnInc_clicked()
{
    emit signalSetArgs(currentRecord.id, 1, 1);
    emit signalGetNext();
}

void TrainingMode1Form::on_btnNotInc_clicked()
{
    emit signalSetArgs(currentRecord.id, 0, 1);
    emit signalGetNext();
```

```
}

void TrainingMode1Form::on_btnSkip_clicked()
{
    emit signalGetNext();
}

void TrainingMode1Form::on_btnNotShow_clicked()
{
    emit signalSetArgs(currentRecord.id, 0, 0);
    emit signalGetNext();
}

void TrainingMode1Form::on_btnExit_clicked()
{
    close();
}
```

Исходный код файла trainingmode2form.h:

```
#ifndef TRAININGMODE2FORM_H
#define TRAININGMODE2FORM_H

#include "stdafx.h"
#include "commonstructs.h"
#include "contextstructs.h"

struct SExpRecordMode2 {
    int id;
    QString report;
    QString info1;
    QString info2;
    QString info3;
    QString info4;
};

namespace Ui {
class TrainingMode2Form;
}

class TrainingMode2Form : public QWidget
{
    Q_OBJECT

public:
    explicit TrainingMode2Form(QWidget *parent = nullptr);
    ~TrainingMode2Form();
    void NewRecordInfo(SExpRecordMode2* psRecord);

signals:
    void signalGetNext();
    void signalSetArgs(int id, int incidentType, int show);
```

```
private slots:
    void on_btnIncident_clicked();
    void on_bntNotIncident_clicked();
    void on_btnNotShow_clicked();
    void on_btnSkip_clicked();
    void on_btnExit_clicked();

private:
    Ui::TrainingMode2Form *ui;
    SExpRecordMode2 currentRecord;
};

#endif // TRAININGMODE2FORM_H
```

Исходный код файла trainingmode2form.cpp:

```
#include "trainingmode2form.h"
#include "ui_trainingmode2form.h"

TrainingMode2Form::TrainingMode2Form(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::TrainingMode2Form)
{
    ui->setupUi(this);
}

TrainingMode2Form::~TrainingMode2Form()
{
    delete ui;
}

void TrainingMode2Form::NewRecordInfo(SExpRecordMode2 *psRecord)
{
    currentRecord = *psRecord;

    ui->txtLeft->setText(currentRecord.report);
    ui->txtRight->clear();

    QStringList strList = currentRecord.info1.split('$');
    if ( strList.size() < 2 )
    {
        qDebug() << "Something is wrong with info string: "<<
strList.at(0);
        return;
    }
    ui->txtRight->append(strList.at(0) + ":");
    ui->txtRight->append(strList.at(1) + "\n");

    strList = currentRecord.info2.split('$');
    if ( strList.size() < 2 )
    {
```

```
        qDebug() << "Something is wrong with info string: "<<
strList.at(0);
        return;
    }
    ui->txtRight->append(strList.at(0) + ":");
    ui->txtRight->append(strList.at(1) + "\n");

    strList = currentRecord.info3.split('$');
    if ( strList.size() < 2 )
    {
        qDebug() << "Something is wrong with info string: "<<
strList.at(0);
        return;
    }
    ui->txtRight->append(strList.at(0) + ":");
    ui->txtRight->append(strList.at(1) + "\n");

    strList = currentRecord.info4.split('$');
    if ( strList.size() < 2 )
    {
        qDebug() << "Something is wrong with info string: "<<
strList.at(0);
        return;
    }
    ui->txtRight->append(strList.at(0) + ":");
    ui->txtRight->append(strList.at(1) + "\n");


    ui->txtLeft->update();
    ui->txtRight->update();
}

void TrainingMode2Form::on_btnIncident_clicked()
{
    int incidentType;
    if ( ui->radioNSD->isChecked() )
    {
        incidentType = NSD_TYPE_NUM;
    }
    else if ( ui->radioOO->isChecked() )
    {
        incidentType = OO_TYPE_NUM;
    }
    else if ( ui->radioVVK->isChecked() )
    {
        incidentType = VVK_TYPE_NUM;
    }
    else if ( ui->radioSI->isChecked() )
    {
        incidentType = SI_TYPE_NUM;
    }
    else if ( ui->radioNI->isChecked() )
    {
```

```
        incidentType = NI_TYPE_NUM;
    }
    else
    {
        QErrorMessage err(this);
        err.showMessage("Не выбран тип инцидента");
        err.exec();
        return;
    }

    emit signalSetArgs(currentRecord.id, incidentType, 1);
    emit signalGetNext();
}

void TrainingMode2Form::on_bntNotIncident_clicked()
{
    emit signalSetArgs(currentRecord.id, NOT_INC_NUM, 1);
    emit signalGetNext();
}

void TrainingMode2Form::on_btnNotShow_clicked()
{
    emit signalSetArgs(currentRecord.id, NOT_INC_NUM, 0);
    emit signalGetNext();
}

void TrainingMode2Form::on_btnSkip_clicked()
{
    emit signalGetNext();
}

void TrainingMode2Form::on_btnExit_clicked()
{
    close();
}
```

Исходный код файла trainingmode3form.h:

```
#ifndef TRAININGMODE3FORM_H
#define TRAININGMODE3FORM_H

#include "stdafx.h"
#include "contextstructs.h"

struct SExpRecordMode3 {
    int id;
    QString report;
    QString stage;
    QString measure1;
    QString measure2;
    QString measure3;
    QString measure4;
```

```
    QString measure5;
};


namespace Ui {
class TrainingMode3Form;
}

class TrainingMode3Form : public QWidget
{
    Q_OBJECT

public:
    explicit TrainingMode3Form(QWidget *parent = nullptr);
    ~TrainingMode3Form();
    void NewRecordInfo(SExpRecordMode3* psRecord);

signals:
    void signalGetNext();
    void signalSetArgs(int id, int flags, int show);

private slots:
    void on_btnOk_clicked();
    void on_btnSkip_clicked();
    void on_btnNotShow_clicked();
    void on_btnExit_clicked();

private:
    Ui::TrainingMode3Form *ui;
    SExpRecordMode3 currentRecord;
};

#endif // TRAININGMODE3FORM_H
```

Исходный код файла trainingmode3form.cpp:

```
#include "trainingmode3form.h"
#include "ui_trainingmode3form.h"

TrainingMode3Form::TrainingMode3Form(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::TrainingMode3Form)
{
    ui->setupUi(this);
}

TrainingMode3Form::~TrainingMode3Form()
{
    delete ui;
}

void TrainingMode3Form::NewRecordInfo(SExpRecordMode3 *psRecord)
```

```
{
    currentRecord = *psRecord;

    ui->lblStage->setText(currentRecord.stage);

    ui->txtReport->setText(currentRecord.report);
    ui->txtReport->update();

    ui->lbl1->setText(currentRecord.measure1);
    ui->checkBox_1->setCheckState(Qt::CheckState::Unchecked);
    ui->lbl2->setText(currentRecord.measure2);
    ui->checkBox_2->setCheckState(Qt::CheckState::Unchecked);
    ui->lbl3->setText(currentRecord.measure3);
    ui->checkBox_3->setCheckState(Qt::CheckState::Unchecked);
    ui->lbl4->setText(currentRecord.measure4);
    ui->checkBox_4->setCheckState(Qt::CheckState::Unchecked);
    ui->lbl5->setText(currentRecord.measure5);
    ui->checkBox_5->setCheckState(Qt::CheckState::Unchecked);
}

void TrainingMode3Form::on_btnOk_clicked()
{
    int flags = 0;
    if ( ui->checkBox_1->isChecked() )
    {
        SET_MEASURE1_FLAG(flags);
    }
    if ( ui->checkBox_2->isChecked() )
    {
        SET_MEASURE2_FLAG(flags);
    }
    if ( ui->checkBox_3->isChecked() )
    {
        SET_MEASURE3_FLAG(flags);
    }
    if ( ui->checkBox_4->isChecked() )
    {
        SET_MEASURE4_FLAG(flags);
    }
    if ( ui->checkBox_5->isChecked() )
    {
        SET_MEASURE5_FLAG(flags);
    }

    emit signalSetArgs(currentRecord.id, flags, 1);
    emit signalGetNext();
}

void TrainingMode3Form::on_btnSkip_clicked()
{
    emit signalGetNext();
}
```

```
void TrainingMode3Form::on_btnNotShow_clicked()
{
    emit signalSetArgs(currentRecord.id, 0, 0);
    emit signalGetNext();
}

void TrainingMode3Form::on_btnExit_clicked()
{
    close();
}
```

Приложение В

(обязательное)

Исходный код приложения ITestClient

Исходный код файла main.cpp:

```
#include <QApplication>
#include "dialogclient.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    DialogClient client;
    client.show();

    return a.exec();
}
```

Исходный код файла stdafx.h:

```
#ifndef STDAFX_H
#define STDAFX_H

#endif // STDAFX_H

#include <QWidget>
#include <QTcpServer>
#include <QTcpSocket>
#include <QMessageBox>
#include <QTextEdit>
#include <QVBoxLayout>
#include <QLabel>
#include <QTime>
#include <QLineEdit>
#include <QPushButton>
#include <QFile>
#include <QTextCodec>
#include <QErrorMessage>

#define ITERATIONS_COUNT        20
```

Исходный код файла commonstructs.h:

```
#ifndef COMMONSTRUCTS_H
#define COMMONSTRUCTS_H

#include "stdafx.h"

typedef enum EModeType {
    EMT_PrimaryAnalysis = 0,
```

```
    EMT_SecondaryAnalysis,
    EMT_ProtectiveMeasures,
    EMT_Undefined
}ModeType;

typedef struct SPacket
{
    quint16 opCode;
    QString message;
}*PSPacket;

typedef enum EOpCode {
    EOp_Ping = 0x00,
    EOp_RegistrateMode1,
    EOp_RegistrareMode2,
    EOp_RegistrateMode3,
    EOp_SessionOpened,

    EOp_GiveMeMode1,
    EOp_TakeItMode1,
    EOp_GiveMeMode2,
    EOp_TakeItMode2,
    EOp_GiveMeMode3,
    EOp_TakeItMode3,

    EOp_FinishMode1,
    EOp_FinishMode2,
    EOp_FinishMode3
}OpCode;

#define NOT_INC_NUM     0
#define NSD_TYPE_NUM    1
#define OO_TYPE_NUM     2
#define VVK_TYPE_NUM    3
#define SI_TYPE_NUM     4
#define NI_TYPE_NUM     5

#endif // COMMONSTRUCTS_H
```

Исходный код файла dialogclient.h:

```
#include "stdafx.h"
#include "startwindow.h"
#include "menuwindow.h"
#include "commonstructs.h"
#include "mode1form.h"
#include "mode2form.h"
#include "mode3form.h"

#define SERVER_SETTINGS_FILE    "servprops.txt"
#define SERVER_NAME_DEFAULT     "localhost"
#define PORT_DEFAULT            65035
```

```
class DialogClient : public QWidget
{
    Q_OBJECT
public:
    DialogClient();

private:
    /* Server functions */
    bool SendToServer(PSPacket psPacket);
    bool LoadSettings(QString* pStrHost, quint16* pnPort);
    void ProcessRequest(EOpCode opcode, QString& message);

    /* Logical functions */
    bool IsModeSelected()
    { return ( m_currentMode != EMT_Undefined ); }
    void StartTesting();
    void StartSession();



private slots:
    /* Server actions */
    void slotReadyRead();
    void slotError(QAbstractSocket::SocketError err);
    void slotConnected();

    /* Windows actions */
    void slotNameRecieved();
    void slotModeRecieved();
    void slotAddPoint();
    void slotCompleteSession();

private:
    /* Server memebers */
    QTcpSocket* m_pSocket;
    quint16 m_nNextBlockSize;

    /* Current state members */
    QString m_username;
    EModeType m_currentMode;
    int m_SessionID;
    int m_Points;

    /* Accessory windows */
    StartWindow* m_pStartWindow;
    MenuWindow* m_pMenuWindow;
    Mode1Form* m_pMode1Window;
    Mode2Form* m_pMode2Window;
    Mode3Form* m_pMode3Window;
};
```

Исходный код файла dialogclient.cpp:

```
#include "dialogclient.h"

DialogClient::DialogClient()
    : m_nNextBlockSize(0),
      m_currentMode(EMT_Undefined),
      m_SessionID(INT_MAX),
      m_Points(0)
{
    m_pSocket = new QTcpSocket(this);
    setMinimumSize(250,200);

    QString strHost;
    quint16 nPort;
    if ( !LoadSettings(&strHost, &nPort) )
    {
        strHost = SERVER_NAME_DEFAULT;
        nPort = PORT_DEFAULT;
    }

     m_pSocket->connectToHost(strHost, nPort);
    connect(
        m_pSocket,
        SIGNAL(connected()),
        SLOT(slotConnected()));
    connect(
        m_pSocket,
        SIGNAL(readyRead()),
        SLOT(slotReadyRead()));
    connect(
        m_pSocket,
        SIGNAL(error(QAbstractSocket::SocketError)),
        this,
        SLOT(slotError(QAbstractSocket::SocketError)));

    m_pStartWindow = new StartWindow(this);
    qDebug() << connect(
        m_pStartWindow,
        SIGNAL(signalIHaveName()),
        this,
        SLOT(slotNameRecieved()));
    m_pStartWindow->show();
}

bool DialogClient::LoadSettings(QString* pStrHost, quint16*
pnPort)
{
    *pStrHost = SERVER_NAME_DEFAULT;
    *pnPort = PORT_DEFAULT;



    QFile settsFile(SERVER_SETTINGS_FILE);
    if ( settsFile.exists() )
```

```
    {
         if ( !settsFile.open(QIODevice::ReadOnly) )
        {
            qDebug() << settsFile.error() <<
settsFile.errorString();
            return false;
        }

        QByteArray dataBlock = settsFile.readAll();
        QString tmp = QString::fromLatin1(dataBlock);
        QStringList list = tmp.split('\'');
        if (list.size() < 4 )
        {
            return false;
        }

        *pStrHost = list.at(1);
        *pnPort = list.at(3).toUShort();
    }
    else
    {
         if ( !settsFile.open(QIODevice::WriteOnly) )
        {
            qDebug() << settsFile.error() <<
settsFile.errorString();
            return false;
        }

        QByteArray dataBlock;
        QDataStream out(&dataBlock, QIODevice::WriteOnly);
        out << "Server = \'" << *pStrHost << "\'\n";
        out << "Port = \'" << *pnPort<< "\'";

        settsFile.write(dataBlock);
    }
    return true;
}

void DialogClient::ProcessRequest(EOpCode opcode, QString
&message)
{
    switch (opcode) {
    case EOp_SessionOpened:
        m_SessionID = message.toInt();
        if ( m_SessionID == INT_MAX )
        {
            QErrorMessage err(this);
            err.showMessage("Ошибка: Сервер не смог открыть
сессию");
              err.exec();
            close();
        }
        StartSession();
```

```
        break;

    case EOp_TakeItMode1:
        if ( !m_pMode1Window )
        {
            QErrorMessage err(this);
            err.showMessage("Внутренняя ошибка (Код 0x105)");
              err.exec();
            close();
        }
         m_pMode1Window->AddIteration(message);
        break;

    case EOp_TakeItMode2:
        if ( !m_pMode2Window )
        {
            QErrorMessage err(this);
            err.showMessage("Внутренняя ошибка (Код 0x105)");
              err.exec();
            close();
        }
         m_pMode2Window->AddIteration(message);
        break;

    case EOp_TakeItMode3:
        if ( !m_pMode3Window )
        {
            QErrorMessage err(this);
            err.showMessage("Внутренняя ошибка (Код 0x105)");
              err.exec();
            close();
        }
         m_pMode3Window->AddIteration(message);
        break;

    default:
        break;
    }
}

void DialogClient::StartTesting()
{
    PSPacket psPacket = new SPacket;

    switch (m_currentMode) {
    case EMT_PrimaryAnalysis:
        psPacket->opCode = EOp_RegistrateMode1;
        break;
    case EMT_SecondaryAnalysis:
        psPacket->opCode = EOp_RegistrareMode2;
        break;
    case EMT_ProtectiveMeasures:
```

```
            psPacket->opCode = EOp_RegistrateMode3;
            break;
        default:
            QErrorMessage err(this);
            err.showMessage("Внутренняя ошибка (Код 0x102)");
             err.exec();
            close();
        }

        psPacket->message = m_username;
        if (!SendToServer(psPacket))
        {
            close();
        }
    }

    void DialogClient::StartSession()
    {
        PSPacket psPacket = new SPacket;

        switch (m_currentMode) {

        case EMT_PrimaryAnalysis:
            m_pMode1Window = new Mode1Form;
            qDebug() << connect(
                m_pMode1Window,
                SIGNAL(signalTrueAnswer()),
                this,
                SLOT(slotAddPoint()));
            qDebug() << connect(
                m_pMode1Window,
                SIGNAL(signalFinishSession()),
                this,
                SLOT(slotCompleteSession()));
            psPacket->opCode = EOp_GiveMeMode1;
            psPacket->message = QString::number(ITERATIONS_COUNT);
            break;

        case EMT_SecondaryAnalysis:
            m_pMode2Window = new Mode2Form;
            qDebug() << connect(
                m_pMode2Window,
                SIGNAL(signalTrueAnswer()),
                this,
                SLOT(slotAddPoint()));
            qDebug() << connect(
                m_pMode2Window,
                SIGNAL(signalFinishSession()),
                this,
                SLOT(slotCompleteSession()));
            psPacket->opCode = EOp_GiveMeMode2;
            psPacket->message = QString::number(ITERATIONS_COUNT);
            break;
```

```
    case EMT_ProtectiveMeasures:
        m_pMode3Window = new Mode3Form;
        qDebug() << connect(
            m_pMode3Window,
            SIGNAL(signalTrueAnswer()),
            this,
            SLOT(slotAddPoint()));
        qDebug() << connect(
            m_pMode3Window,
            SIGNAL(signalFinishSession()),
            this,
            SLOT(slotCompleteSession()));
        psPacket->opCode = EOp_GiveMeMode3;
        psPacket->message = QString::number(ITERATIONS_COUNT);
        break;

    default:
        return;
    }

    if (!SendToServer(psPacket))
    {
        close();
    }
}

void DialogClient::slotReadyRead()
{
    QDataStream in(m_pSocket);

    while (true)
    {
        if ( !m_nNextBlockSize )
        {
              if (m_pSocket->bytesAvailable() < sizeof
(m_nNextBlockSize))
            {
                break;
            }
            in >> m_nNextBlockSize;
        }

         if( m_pSocket->bytesAvailable() <
m_nNextBlockSize )
        {
            break;
        }

        QTime time;
        QString str;
        quint16 opcode;
```

```
        QByteArray dataBlock;
        quint8 byte;
        unsigned char uchTemp;

        for ( quint16 count = 0; count < m_nNextBlockSize; count++
)
        {
            in >> byte;
            uchTemp = byte;
            dataBlock.append((char)uchTemp);
        }

        QDataStream dataIn(&dataBlock, QIODevice::ReadOnly);
        dataIn >> time >> opcode >> str;
         ProcessRequest((EOpCode)opcode, str);


        m_nNextBlockSize = 0;
    }
}

void DialogClient::slotError(QAbstractSocket::SocketError err)
{
    QString strError = "Error: ";
    switch (err) {
    case QAbstractSocket::HostNotFoundError:
        strError += "The host was not found";
        break;
    case QAbstractSocket::RemoteHostClosedError:
        strError += "The remote host is closed";
        break;
    case QAbstractSocket::ConnectionRefusedError:
        strError += "The connection was refused";
        break;
    default:
        strError += m_pSocket->errorString();
    }
    //m_pTxtInfo->append(strError);
}

bool DialogClient::SendToServer(PSPacket psPacket)
{
    QByteArray dataBlock;
    QDataStream out(&dataBlock, QIODevice::WriteOnly);
    out << quint16(0) << QTime::currentTime() << quint16(psPacket-
>opCode) << psPacket->message;
     out.device()->seek(0);
    out << quint16(dataBlock.size() - sizeof(quint16));

    if (m_pSocket->write(dataBlock) != qint64(dataBlock.size()))
    {
        QErrorMessage err(this);
        err.showMessage("Не удалось связаться с сервером");
```

```
            err.exec();
        return false;
    }
    return true;
}

void DialogClient::slotConnected()
{
    //m_pTxtInfo->append("Connection established");
}

void DialogClient::slotNameRecieved()
{
    m_username = m_pStartWindow->GetEnteredName();
    m_pStartWindow->close();

    m_pMenuWindow = new MenuWindow(this);
    qDebug() << connect(
        m_pMenuWindow,
        SIGNAL(signalIHaveMode()),
        this,
        SLOT(slotModeRecieved()));
    m_pMenuWindow->show();

}

void DialogClient::slotModeRecieved()
{
    m_currentMode = m_pMenuWindow->GetSelectedMode();
    m_pMenuWindow->close();
    if ( IsModeSelected() )
    {
        StartTesting();
    }
    else
    {
        m_pMenuWindow = new MenuWindow(this);
        qDebug() << connect(
            m_pMenuWindow,
            SIGNAL(signalIHaveMode()),
            this,
            SLOT(slotModeRecieved()));
        m_pMenuWindow->show();
    }
}

void DialogClient::slotAddPoint()
{
    m_Points++;
}

void DialogClient::slotCompleteSession()
{
```

```
    PSPacket psPacket = new SPacket;
    switch (m_currentMode) {
    case EMT_PrimaryAnalysis:
        psPacket->opCode = EOp_FinishMode1;
        break;
    case EMT_SecondaryAnalysis:
        psPacket->opCode = EOp_FinishMode2;
        break;
    case EMT_ProtectiveMeasures:
        psPacket->opCode = EOp_FinishMode3;
        break;
    default:
        break;
    }

    psPacket->message = QString::number(m_SessionID) + "|" +
QString::number(m_Points);
    if (!SendToServer(psPacket))
    {
        close();
    }
    QMessageBox::information(this, "Информация", "Тестирование
завершено");
    close();
}
```

Исходный код файла startwindow.h:

```
#include "stdafx.h"

class StartWindow : public QWidget
{
    Q_OBJECT
public:
    explicit StartWindow(QWidget *parent = nullptr);
    QString GetEnteredName();

signals:
    void signalIHaveName();

private slots:
    void slotOk();

private:
    QLabel* m_pInfo;
    QLineEdit* m_pLineEdit;
    QPushButton* m_pBtnOk;
    QString m_name;


};
```

Исходный код файла startwindow.cpp:

```
#include "startwindow.h"

StartWindow::StartWindow(QWidget *parent)
    : QWidget(parent)
{
    m_name = "Unknown";

    m_pInfo = new QLabel("Введите имя пользователя:");
    m_pLineEdit = new QLineEdit;
    m_pBtnOk = new QPushButton("&OK");

    qDebug() << connect(
        m_pBtnOk,
        SIGNAL(clicked()),
        this,
        SLOT(slotOk()));
    qDebug() << connect(
        m_pLineEdit,
        SIGNAL(returnPressed()),
        this,
        SLOT(slotOk()));

    QVBoxLayout* pLayout = new QVBoxLayout;
    pLayout->addWidget(new QLabel("<H1>Начало работы<H1>"));
    pLayout->addWidget(m_pInfo);
    pLayout->addWidget(m_pLineEdit);
    pLayout->addWidget(m_pBtnOk);
    pLayout->setAlignment(Qt::AlignHCenter);
    setLayout(pLayout);

    setAttribute(Qt::WA_DeleteOnClose, true);
}

QString StartWindow::GetEnteredName()
{
    return m_name;
}

void StartWindow::slotOk()
{
    if ( "" == m_pLineEdit->text() )
    {
        QErrorMessage err(this);
        err.showMessage("Введите корректное имя пользователя");
        err.exec();

        m_pLineEdit->setText("");
        return;
    }
```

```
    m_name = m_pLineEdit->text();
    emit signalIHaveName();
}
```

Исходный код файла menuwindow.h:

```
#include "stdafx.h"
#include "commonstructs.h"

class MenuWindow : public QWidget
{
    Q_OBJECT
public:
    explicit MenuWindow(QWidget *parent = nullptr);
    EModeType GetSelectedMode();

signals:
    void signalIHaveMode();

private slots:
    void slotSelectMode1();
    void slotSelectMode2();
    void slotSelectMode3();

private:
    QLabel* m_pInfo;
    QPushButton* m_pBtnMode1;
    QPushButton* m_pBtnMode2;
    QPushButton* m_pBtnMode3;
    EModeType m_mode;
};
```

Исходный код файла menuwindow.cpp:

```
#include "menuwindow.h"

MenuWindow::MenuWindow(QWidget *parent)
    : QWidget(parent),
      m_mode(EMT_Undefined)
{
    m_pInfo = new QLabel("Выберите режим тестирования");
    m_pBtnMode1 = new QPushButton("&Первичный анализ ИБ");
    qDebug() << connect(
        m_pBtnMode1,
        SIGNAL(clicked()),
        this,
        SLOT(slotSelectMode1()));
    m_pBtnMode2 = new QPushButton("&Вторичный анализ ИБ");
    qDebug() << connect(
        m_pBtnMode2,
        SIGNAL(clicked()),
```

```
        this,
        SLOT(slotSelectMode2()));
    m_pBtnMode3 = new QPushButton("&Реагирование на инцидент ИБ");
    qDebug() << connect(
        m_pBtnMode3,
        SIGNAL(clicked()),
        this,
        SLOT(slotSelectMode3()));

    QVBoxLayout* pLayout = new QVBoxLayout;
    pLayout->addWidget(m_pInfo);
    pLayout->addWidget(m_pBtnMode1);
    pLayout->addWidget(m_pBtnMode2);
    pLayout->addWidget(m_pBtnMode3);
    pLayout->setAlignment(Qt::AlignHCenter);
    setLayout(pLayout);

    setAttribute(Qt::WA_DeleteOnClose, true);
}

EModeType MenuWindow::GetSelectedMode()
{
    return m_mode;
}

void MenuWindow::slotSelectMode1()
{
    m_mode = EMT_PrimaryAnalysis;
    emit signalIHaveMode();
}

void MenuWindow::slotSelectMode2()
{
    m_mode = EMT_SecondaryAnalysis;
    emit signalIHaveMode();
}

void MenuWindow::slotSelectMode3()
{
    m_mode = EMT_ProtectiveMeasures;
    emit signalIHaveMode();
}
```

Исходный код файла mode1form.h:

```
#ifndef MODE1FORM_H
#define MODE1FORM_H

#include "stdafx.h"

struct SIterationMode1{
    QString signDescription;
    QString extraDescription;
```

```
    bool IsIncident;
};

namespace Ui {
class Mode1Form;
}

class Mode1Form : public QWidget
{
    Q_OBJECT

public:
    explicit Mode1Form(QWidget *parent = nullptr);
    ~Mode1Form();
    void AddIteration(QString& message);

signals:
    void signalTrueAnswer();
    void signalFinishSession();

private slots:
    void on_btnInc_clicked();
    void on_btnNotInc_clicked();

private:
    void Start();
    void ShowNextIter();
    void ResetControls();

private:
    Ui::Mode1Form *ui;

    QVector<SIterationMode1> m_iterations;
    quint8 m_currIter;
};

#endif // MODE1FORM_H
```

Исходный код файла mode1form.cpp:

```
#include "mode1form.h"
#include "ui_mode1form.h"

Mode1Form::Mode1Form(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Mode1Form),
    m_currIter(0)
{
    ui->setupUi(this);
}

Mode1Form::~Mode1Form()
```

```
{
    delete ui;
}

void Mode1Form::AddIteration(QString &message)
{
    if ( m_iterations.size() >= ITERATIONS_COUNT )
    {
        return;
    }

    QStringList paramsList = message.split('|');
    if (paramsList.size() < 2 )
    {
        qDebug() << "Something is wrong with Iteration message";
    }

    SIterationMode1 sIter;
    sIter.signDescription = paramsList.at(0);
    sIter.extraDescription = paramsList.at(1);
    sIter.IsIncident = paramsList.at(2).toInt();

    m_iterations.push_back(sIter);
    if ( m_iterations.size() == ITERATIONS_COUNT )
    {
        Start();
    }
}

void Mode1Form::Start()
{
    ResetControls();
    ShowNextIter();
    show();
}

void Mode1Form::ShowNextIter()
{
    if ( m_currIter == ITERATIONS_COUNT )
    {
        emit signalFinishSession();
        close();
        return;
    }

    SIterationMode1 sIter = m_iterations[m_currIter];

    QStringList info1List = sIter.signDescription.split('$');
    if ( info1List.size() < 2 )
    {
        qDebug() << "Something is wrong with info1Liat. Size: " <<
info1List.size();
    }
```

```
    ui->lblLeft->setText(info1List.at(0));
    ui->txtLeft->setText(info1List.at(1));

    QStringList info2List = sIter.extraDescription.split('$');
    if ( info2List.size() < 2 )
    {
        qDebug() << "Something is wrong with info1Liat. Size: " <<
info1List.size();
    }

    if ( info2List.size() == 2 )
    {
        ui->lblRight->setText(info2List.at(0));
        ui->txtRight->setText(info2List.at(1));
    }
    else
    {
        ui->lblRight->setText("Дополнительная информация");
        ui->txtRight->setText(info2List.at(0) + "\n\n" +
info2List.at(1) + "\n\n" +info2List.at(2));
    }
}

void Mode1Form::ResetControls()
{
    ui->lblLeft->clear();
    ui->lblRight->clear();
    ui->txtLeft->clear();
    ui->txtLeft->setReadOnly(true);
    ui->txtRight->clear();
    ui->txtRight->setReadOnly(true);
}

void Mode1Form::on_btnInc_clicked()
{
    if ( m_iterations[m_currIter].IsIncident )
    {
        emit signalTrueAnswer();
    }
    m_currIter++;
    ShowNextIter();
    update();
}

void Mode1Form::on_btnNotInc_clicked()
{
    if ( !m_iterations[m_currIter].IsIncident )
    {
        emit signalTrueAnswer();
    }
    m_currIter++;
    ShowNextIter();
    update();
```

```
}
```

Исходный код файла mode2form.h:

```
#ifndef MODE2FORM_H
#define MODE2FORM_H

#include "stdafx.h"
#include "commonstructs.h"

struct SIterationMode2 {
    QString report;
    QString info1;
    QString info2;
    QString info3;
    QString info4;
    int incidentType;
};

namespace Ui {
class Mode2Form;
}

class Mode2Form : public QWidget
{
    Q_OBJECT

public:
    explicit Mode2Form(QWidget *parent = nullptr);
    ~Mode2Form();
    void AddIteration(QString& message);

signals:
    void signalTrueAnswer();
    void signalFinishSession();

private slots:
    void on_btnIncident_clicked();
    void on_btnNotIncident_clicked();

private:
    void Start();
    void ShowNextIter();

private:
    Ui::Mode2Form *ui;
    QVector<SIterationMode2> m_iterations;
    quint8 m_currIter;
};

#endif // MODE2FORM_H
```

Исходный код файла mode2form.cpp:

```
#include "mode2form.h"
#include "ui_mode2form.h"

Mode2Form::Mode2Form(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Mode2Form),
    m_currIter(0)
{
    ui->setupUi(this);
}

Mode2Form::~Mode2Form()
{
    delete ui;
}

void Mode2Form::AddIteration(QString &message)
{
    if ( m_iterations.size() >= ITERATIONS_COUNT )
    {
        return;
    }

    QStringList paramsList = message.split('|');
    if (paramsList.size() < 6 )
    {
        qDebug() << "Something is wrong with Iteration message";
    }

    SIterationMode2 sIter;
    sIter.report = paramsList.at(0);
    sIter.info1 = paramsList.at(1);
    sIter.info2 = paramsList.at(2);
    sIter.info3 = paramsList.at(3);
    sIter.info4 = paramsList.at(4);
    sIter.incidentType = paramsList.at(5).toInt();

    m_iterations.push_back(sIter);
    if ( m_iterations.size() == ITERATIONS_COUNT )
    {
        Start();
    }
}

void Mode2Form::on_btnIncident_clicked()
{
    int incidentType;
    if ( ui->radioNSD->isChecked() )
    {
        incidentType = NSD_TYPE_NUM;
    }
```

```
    else if ( ui->radioOO->isChecked() )
    {
        incidentType = OO_TYPE_NUM;
    }
    else if ( ui->radioVVK->isChecked() )
    {
        incidentType = VVK_TYPE_NUM;
    }
    else if ( ui->radioSI->isChecked() )
    {
        incidentType = SI_TYPE_NUM;
    }
    else if ( ui->radioNI->isChecked() )
    {
        incidentType = NI_TYPE_NUM;
    }
    else
    {
        QErrorMessage err(this);
        err.showMessage("Не выбран тип инцидента");
        err.exec();
        return;
    }


    if ( m_iterations[m_currIter].incidentType == incidentType )
    {
        emit signalTrueAnswer();
    }

    m_currIter++;
    ShowNextIter();
}

void Mode2Form::Start()
{
    ShowNextIter();
    show();
}

void Mode2Form::ShowNextIter()
{
    if ( m_currIter == ITERATIONS_COUNT )
    {
        emit signalFinishSession();
        close();
        return;
    }

    SIterationMode2 sCurrIteration = m_iterations[m_currIter];
    ui->txtLeft->setText(sCurrIteration.report);
    ui->txtRight->clear();
```

```
    QStringList strList = sCurrIteration.info1.split('$');
    if ( strList.size() < 2 )
    {
        qDebug() << "Something is wrong with info string: "<<
strList.at(0);
        return;
    }
    ui->txtRight->append(strList.at(0) + ":");
    ui->txtRight->append(strList.at(1) + "\n");

    strList = sCurrIteration.info2.split('$');
    if ( strList.size() < 2 )
    {
        qDebug() << "Something is wrong with info string: "<<
strList.at(0);
        return;
    }
    ui->txtRight->append(strList.at(0) + ":");
    ui->txtRight->append(strList.at(1) + "\n");

    strList = sCurrIteration.info3.split('$');
    if ( strList.size() < 2 )
    {
        qDebug() << "Something is wrong with info string: "<<
strList.at(0);
        return;
    }
    ui->txtRight->append(strList.at(0) + ":");
    ui->txtRight->append(strList.at(1) + "\n");

    strList = sCurrIteration.info4.split('$');
    if ( strList.size() < 2 )
    {
        qDebug() << "Something is wrong with info string: "<<
strList.at(0);
        return;
    }
    ui->txtRight->append(strList.at(0) + ":");
    ui->txtRight->append(strList.at(1) + "\n");


    ui->txtLeft->update();
    ui->txtRight->update();
}

void Mode2Form::on_btnNotIncident_clicked()
{
    if ( m_iterations[m_currIter].incidentType == NOT_INC_NUM )
    {
        emit signalTrueAnswer();
    }

    m_currIter++;
```

```
    ShowNextIter();
}
```

Исходный код файла mode3form.h:

```
#ifndef MODE3FORM_H
#define MODE3FORM_H

#include "stdafx.h"
#include "commonstructs.h"

struct SIterationMode3 {
    QString report;
    QString stage;
    QString measure1;
    QString measure2;
    QString measure3;
    QString measure4;
    QString measure5;
    int flags;
};

#define GET_MEASURE1_FLAG(flags) (flags & 0x1)
#define GET_MEASURE2_FLAG(flags) (flags & 0x2)
#define GET_MEASURE3_FLAG(flags) (flags & 0x4)
#define GET_MEASURE4_FLAG(flags) (flags & 0x8)
#define GET_MEASURE5_FLAG(flags) (flags & 0x10)

#define SET_MEASURE1_FLAG(flags) (flags |= 0x1)
#define SET_MEASURE2_FLAG(flags) (flags |= 0x2)
#define SET_MEASURE3_FLAG(flags) (flags |= 0x4)
#define SET_MEASURE4_FLAG(flags) (flags |= 0x8)
#define SET_MEASURE5_FLAG(flags) (flags |= 0x10)

namespace Ui {
class Mode3Form;
}

class Mode3Form : public QWidget
{
    Q_OBJECT

public:
    explicit Mode3Form(QWidget *parent = nullptr);
    ~Mode3Form();
    void AddIteration(QString& message);

signals:
    void signalTrueAnswer();
    void signalFinishSession();

private slots:
    void on_btnOk_clicked();
```

```
private:
    void Start();
    void ShowNextIter();

private:
    Ui::Mode3Form *ui;
    QVector<SIterationMode3> m_iterations;
    quint8 m_currIter;
};

#endif // MODE3FORM_H
```

Исходный код файла mode3form.cpp:

```
#include "mode3form.h"
#include "ui_mode3form.h"

Mode3Form::Mode3Form(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Mode3Form),
    m_currIter(0)
{
    ui->setupUi(this);
}

Mode3Form::~Mode3Form()
{
    delete ui;
}

void Mode3Form::AddIteration(QString &message)
{
    if ( m_iterations.size() >= ITERATIONS_COUNT )
    {
        return;
    }

    QStringList paramsList = message.split('|');
    if (paramsList.size() < 4 )
    {
        qDebug() << "Something is wrong with Iteration message.
Size: " << paramsList.size();
    }

    SIterationMode3 sIter;
    sIter.report = paramsList.at(0);
    sIter.stage = paramsList.at(1);
    QString strMeasures = paramsList.at(2);
    sIter.flags = paramsList.at(3).toInt();

    QStringList measures = strMeasures.split('$');
    if (measures.size() < 5 )
```

```
    {
        qDebug() << "Something is wrong with Measures list. Size:
" << measures.size();
    }
    sIter.measure1 = measures.at(0);
    sIter.measure2 = measures.at(1);
    sIter.measure3 = measures.at(2);
    sIter.measure4 = measures.at(3);
    sIter.measure5 = measures.at(4);

    m_iterations.push_back(sIter);
    if ( m_iterations.size() == ITERATIONS_COUNT )
    {
        Start();
    }
}

void Mode3Form::Start()
{
    ShowNextIter();
    show();
}

void Mode3Form::ShowNextIter()
{
    if ( m_currIter == ITERATIONS_COUNT )
    {
        emit signalFinishSession();
        close();
        return;
    }

    SIterationMode3 sCurrIteration = m_iterations[m_currIter];
    ui->lblStage->setText(sCurrIteration.stage);

    ui->txtReport->setText(sCurrIteration.report);
    ui->txtReport->update();

    ui->lbl1->setText(sCurrIteration.measure1);
    ui->checkBox_1->setCheckState(Qt::CheckState::Unchecked);
    ui->lbl2->setText(sCurrIteration.measure2);
    ui->checkBox_2->setCheckState(Qt::CheckState::Unchecked);
    ui->lbl3->setText(sCurrIteration.measure3);
    ui->checkBox_3->setCheckState(Qt::CheckState::Unchecked);
    ui->lbl4->setText(sCurrIteration.measure4);
    ui->checkBox_4->setCheckState(Qt::CheckState::Unchecked);
    ui->lbl5->setText(sCurrIteration.measure5);
    ui->checkBox_5->setCheckState(Qt::CheckState::Unchecked);
}

void Mode3Form::on_btnOk_clicked()
{
    int flags = 0;
```

```
    if ( ui->checkBox_1->isChecked() )
    {
        SET_MEASURE1_FLAG(flags);
    }
    if ( ui->checkBox_2->isChecked() )
    {
        SET_MEASURE2_FLAG(flags);
    }
    if ( ui->checkBox_3->isChecked() )
    {
        SET_MEASURE3_FLAG(flags);
    }
    if ( ui->checkBox_4->isChecked() )
    {
        SET_MEASURE4_FLAG(flags);
    }
    if ( ui->checkBox_5->isChecked() )
    {
        SET_MEASURE5_FLAG(flags);
    }

    if ( flags == m_iterations[m_currIter].flags )
    {
        emit signalTrueAnswer();
    }
    m_currIter++;
    ShowNextIter();
}
```