Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования «Казанский (Приволжский) федеральный университет»
Институт вычислительной математики и информационных технологий
Кафедра системного анализа и информационных технологий

Направление подготовки: 09.04.03 — Прикладная информатика

Магистерская программа: Информационная безопасность экономических систем

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

РАЗРАБОТКА СИСТЕМЫ МОДЕЛИРОВАНИЯ УГРОЗ И ОЦЕНКИ РИСКОВ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Обучающийся 2 курса
группы 09-975 (Бобоназаров Р.Ч.)

Руководитель

д-р физ.-мат. наук, профессор (Ишмухаметов Ш.Т.)

Заведующий кафедрой системного анализа и информационных технологий

д-р техн. наук, профессор (Латыпов Р.Х.)

Казань – 2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Информационная безопасность (далее ИБ) в современном мире представляет собой широкую самостоятельную область, вопрос актуальности которой уже можно считать риторическим. Практически любая крупная компания, работающая с персональными данными, обладающая коммерческими тайнами, раскрытие которых может повлечь серьезные убытки, или тем более работающая с государственной тайной, владеет отдельными отделами или даже департаментами, отвечающими за обеспечение ИБ. Причем в случае работы с конфиденциальной информацией, такой как персональные данные, или при работе с гос. тайной, данные вопросы имеют жесткую регуляторную законодательную основу. В РФ в качестве примеров можно привести федеральные законы «Об информации, информационных технологиях и о защите информации», «О персональных данных», «О государственной тайне», а также ряд других (список приведен в приложении 1). Данные законы подразумевают соблюдение строгих требований и регламентов при хранении, обработке и передаче защищаемой информации. Кроме того, разработан ряд ГОСТов, основанных на международных стандартах оценки безопасности информационных систем, менеджмента рисков ИБ, построения систем управления ИБ и т.д. Следует упомянуть, что все ГОСТы, на которые даны ссылки в данной работе, являются действующими на момент написания.

В связи как с необходимостью соблюдать предъявляемые законом требования, так и естественной потребностью обеспечить защищенность информации, утечка которой грозит денежным и/или репутационным ущербом, предприятия сталкиваются с вопросом обеспечения информационной безопасности внутренних информационных систем. Для этого могут проводится как разовые мероприятия аудита ИБ и внедрения/ обновления системы защиты по его результатам, так и построение полноценной системы управления/менеджмента информационной безопасности, функционирующей на предприятии (СУИБ/СМИБ).

Любой аудит ИБ автоматизированных систем или же построение СМИБ неотъемлемо включают в себя два тесно связанных между собой этапа: моделирование угроз и оценка рисков, без которых они теряют смысл. Иногда этап моделирования угроз включается в этап оценки рисков, например, в [1].

Поэтому исследование методик моделирования угроз и оценки рисков и, соответственно, разработка системы, реализующей данные методики является актуальной задачей, которая значительно упростила бы как аудит ИБ, так и текущий менеджмент рисков ИБ. Особенно данная система будет полезна, если ориентировать её на современные российские стандарты, как будет сделано в данной работе. Это позволит применять её не только как вспомогательный инструмент, но даже как одно из основных средств менеджмента риска ИБ/ аудита ИБ/ построения СМИБ.

Целью данной работы является разработка автоматизированной информационной системы, позволяющего проводить идентификацию информационных активов, строить для заданных активов модель угроз и на основе построенной модели вычислять количественную/качественную оценку рисков информационной безопасности на основе различных методик.

На основе цели были сформулированы следующие задачи:

1) изучение российского законодательства, стандартов и руководящих документов в области, связанной с менеджментом информационной безопасности в целом и построением моделей угроз, менеджментом рисков информационной безопасности в частности;

2) изучение методик моделирования угроз и оценки рисков информационной безопасности;

3) исследование изученных методик и предложение возможных качественных улучшений;

4) исследование применения нечеткой логики к задаче оценки рисков информационной безопасности;

5) реализация программного модуля баз данных угроз и уязвимостей на основе банка данных угроз ФСТЭК России;

6) реализация программного модуля идентификации информационных активов;

7) реализация программного модуля моделирования угроз для активов;

8) реализация программного модуля оценки рисков для модели угроз на основе изученных методик, включая предложенные на этапе 3 улучшения. Также в данном модуле предполагается реализация оценки рисков на основе алгоритма нечеткой логики.

1. Анализ предметной области

Рассмотрим термины и определения, используемые в работе и необходимые для дальнейшего понимания.

Под риском информационной безопасности чаще всего подразумевается потенциальная возможность того, что некоторая угроза сможет использовать уязвимость актива/группы активов для реализации и впоследствии повлечет ущерб организации. Риск ИБ измеряется на основе вероятности реализации события и количественной оценке последствий этого события [1].

Под активом, согласно [2], следует понимать все, что имеет ценность для организации. Понятно, что с точки зрения именно информационной безопасности в качестве активов в первую очередь рассматриваются объекты, задействованные в информационной инфраструктуре организации, например, все объекты, задействованные при эксплуатации информационных автоматизированных систем.

Понятия угрозы и уязвимости представлены в ГОСТе [3]. Согласно ему, под угрозой безопасности информации понимается совокупность условий и факторов, приводящих к возникновению потенциальной или реально существующей опасности нарушения безопасности информации. Для уязвимости в том же документе дано следующее определение: уязвимость – это свойство/особенность информационной системы, предоставляющее возможность реализации угрозы информации, обрабатываемой в данной системе. Соответственно, если для угрозы существует некоторая уязвимость, которая может быть использована для её реализации, тогда существует риск. Зачастую условием для реализации угрозы БИ является наличие технологического недостатка, слабого места в информационной системе.

Модель угроз, согласно [3] – это представление в том или ином виде (физическом, описательном, математическом) виде характеристик и свойств угроз БИ.

С понятием риска связан ряд сопутствующих определений:

Идентификация риска – процесс нахождения (или деятельность по нахождению), составления списка элементов риска и их описания.

Анализ риска – обработка доступной информации с целью определения источников риска и количественной оценки риска.

Количественная оценка риска – процесс вычисления (или деятельность по вычислению) значения вероятности риска и его возможных последствий.

Оценивание риска – процесс сравнения количественной оценки риска с некоторыми заданными критериями риска с целью определения его значимости/приоритетности в рассмотрении.

Обработка риска – процесс выбора и реализации мер по модификации риска (данные меры можно классифицировать как направленные на избежание(предотвращение), снижение, перенос или сохранение риска).

Снижение риска – действия, направленные на уменьшение вероятности и/или негативных последствий (ущерба), связанных с риском.

Сохранение риска – принятие возможного бремени ущерба от риска.

Перенос риска – разделение с другой стороной возможного бремени ущерба от риска.

Предотвращение риска – решение по избеганию вовлеченности в рискованную ситуацию или некоторое действие, предупреждающее вовлечение в неё.

Остаточный риск – риск, остающийся после его обработки, т.е. риск после реализации мер по модификации риска.

Принятие риска – решение по принятию риска.

Коммуникация риска – обмен информацией о риске или совместное пользование ей основным лицом, ответственным за принятие решений о риске и сопричастными сторонами.

Менеджмент риска – скоординированные действия по руководству предприятием в отношении риска (в менеджмент риска включают оценку, обработку, принятие/предотвращение и коммуникацию риска) [1, 2].

2. Идентификация информационных активов

Первым необходимым шагом в построении модели угроз и оценке рисков ИБ является проведение полной идентификации имеющихся информационных активов, т.к. ни первое, ни второе невозможно провести в отрыве от идентификации активов организации.

Идентификация активов включает [4]:

1) формирование реестров активов,
2) инвентаризацию активов,
3) классификацию и категорирование активов.

Первый этап формирования реестров активов подразумевает определение возможных видов активов. Информационные активы, согласно [2] делятся на (но могут не ограничиваться указанным списком):

1) непосредственно информацию (базы и файлы данных, контракты и соглашения, системная и другая документация, научно-исследовательская информация, обучающие материалы и пр.);
2) материальные активы (техническое оборудование: ЭВМ, средства связи и т.п.);
3) программное обеспечение;
4) сервисы (обслуживающая инфраструктура);
5) нематериальные ресурсы (репутация компании);
6) людей (например, сотрудники, их квалификация и опыт) [5].

Этап инвентаризации заключается в составлении перечня всех ценных активов организации. Определяется, для каких из активов нарушение ИБ нанесет ущерб, и данные активы в дальнейшем считаются ценными. Подразумевается, что ценные активы должны иметь некий уровень гарантированной защиты, потому все они учитываются при анализе и оценке информационных рисков [6].

В большинстве случаев инвентаризация инф. активов и оценка их ценности проводится на «верхнем» уровне, не требуя точных деталей и

длительных процедур. Однако конечная степень детализации устанавливается, в первую очередь, исходя из целей безопасности.

В качестве основных характеристик информационных активов, подлежащих рассмотрению, рассматривают: ценность, чувствительность активов (уязвимости и их критичность), а также имеющиеся защитные меры.

Выделенные как ценные активы, категорируются по их критичности для бизнес-процессов организации. В качестве ценности или критичности актива, чаще всего, рассматривают ущерб, который может понести компания в случае его потери/кражи/нарушения функционирования. Ценность активов может быть определена из сопроводительных к ним документов, на основе их стоимости или на основе экспертных оценок их владельцев. В случае применения соответствующих методик оценки рисков, ценность может оцениваться раздельно по конфиденциальности, целостности и доступности. Поэтому активы также категорируются по типу возможной угрозы, что, впоследствии, упрощает проведение оценки рисков.

3. Моделирование угроз безопасности информации и их оценка

3.1. Методики моделирования угроз безопасности информации и оценки угроз безопасности ФСТЭК России

9 апреля 2020 года на сайте ФСТЭК России был опубликован проект методического документа «Методика моделирования угроз безопасности информации» [7], являющегося идейным продолжением документа 2015-ого года «Методика определения угроз информации в информационных системах», а 25 февраля 2021 года был опубликован и введен в действие окончательный вариант со слегка изменившимся названием «Методика оценки угроз безопасности информации» [8]. В связи с утверждением документа более не актуальны и не применяются методики «определения актуальных угроз безопасности персональных данных при их обработке в информационных системах персональных данных» (ФСТЭК России, 2008 г.) и методику «определения актуальных угроз безопасности информации в ключевых системах информационной инфраструктуры» (ФСТЭК России, 2007 г.), которые были изданы относительно давно. Данный документ должен использоваться для определения угроз безопасности информации при ее обработке с использованием любых объектов, требования по защите к которым утверждены ФСТЭК. К таким объектам относятся значимые объекты КИИ, ИСПДн, ГИС и иные типы информационных систем (например, ИС финансовых организаций).

Далее приведен краткий последовательный разбор методики, разработанной регулятором, а также некоторые особенности её проекта, не включенные в окончательный вариант, однако представляющие интерес с точки зрения интересного подхода к оценке рисков ИБ.

Общий процесс моделирования угроз безопасности информации в итоговой методике представлен на рисунке 1. Согласно ей, угроза БИ возможна, если существуют её источник, объект, на который направлено воздействие, способ(ы) (сценарии) реализации угрозы, а сама реализация повлечет за собой негативные последствия для государства или организации:

УБИ = [нарушитель (источник угрозы); объекты воздействия; способы реализации угроз; негативные последствия].

Рисунок 1 – Процесс моделирования угроз безопасности информации из утвержденной методики

Угроза принимается за актуальную в случае наличия хотя бы одного сценария реализации данной угрозы и включается в модель угроз. Очевидно, такое определение перекликается с понятием риска ИБ, данным во введении, а именно в том плане, что если угроза является актуальной, то существует риск ИБ. В соответствии со схемой на рисунке 1 можно сделать вывод, что определение актуальности угрозы безопасности проводится на 3 этапе на основе информации, полученной на этапах 1 и 2.

На рисунке 2 представлен процесс моделирования угроз безопасности информации из проекта методики. Видно, что несмотря на изменения в конечном варианте, общая схема процесса не претерпела значительных изменений, а сама схема стала даже менее информативной. Этапы 1 в финальной версии и в проекте совпадают, этап 2 был переименован и обобщен, а этапы 3-5 проекта были объединены в один, причем заключительный 5 этап был изменен с «оценки уровня опасности угроз БИ» на оценку «актуальности угрозы БИ». Уровень опасности рассчитывался для

актуальных угроз, т.е. 5-ый этап проекта включал в себя определение актуальности, поэтому можно сделать вывод, что методику просто сократили, убрав данный параметр.

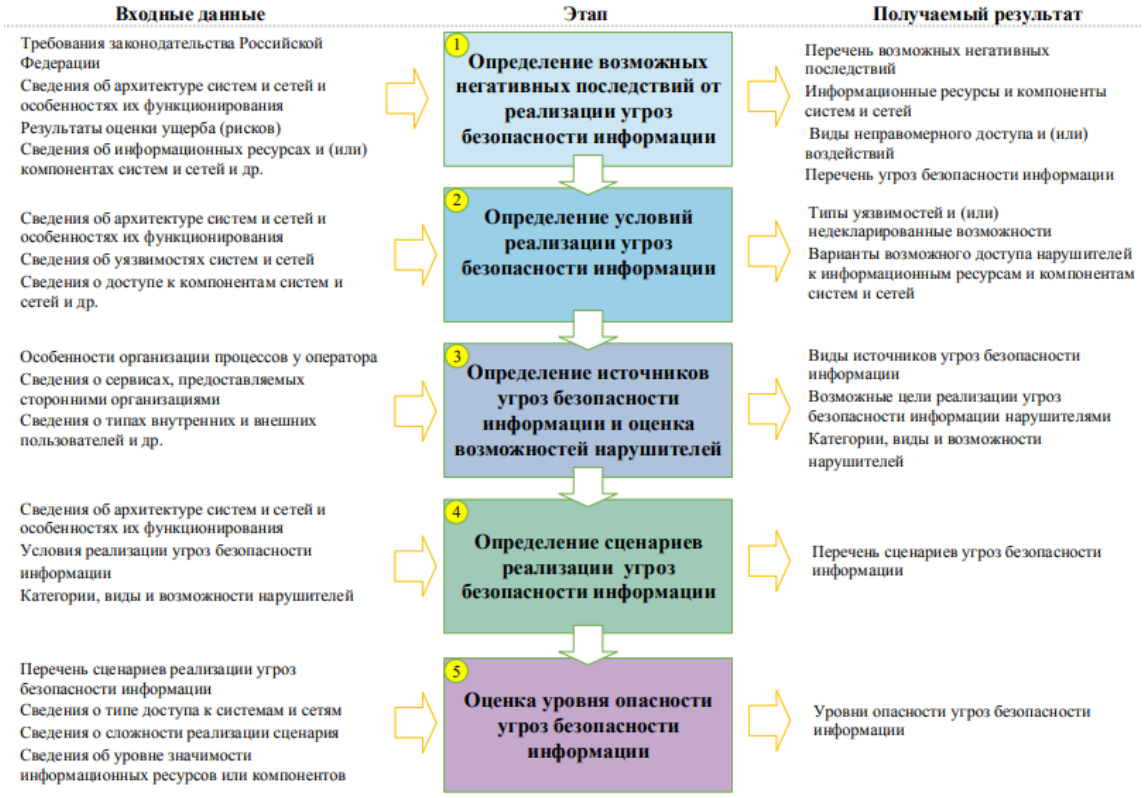


Рисунок 2 – Процесс моделирования угроз безопасности информации из проекта методики

Уровень угрозы можно было бы рассматривать как аналог оценки риска и, соответственно использовать в качестве критерия в управлении рисками. Например, с помощью данного параметра можно определять приоритетность устранения угроз. В качестве минуса можно выделить то, что дифференциация опасности по уровням посредственная: «низкий», «средний», «высокий» (приложение 2, таблица 5). Однако проводится она в соответствии с численным показателем, таким образом, можно проводить сравнения не по уровням, а по численной оценке. Почему параметр был исключен – вопрос открытый, т.к. в текущей методике исключительно определяется актуальность угроз, однако нет никаких рекомендаций по их приоритезации при введении контрмер, что можно считать упущением, т.к. трудовые и финансовые ресурсы любой организации ограничены, в связи с чем приоритезация работы по

устранению/противодействию угрозам является важной задачей. В данной работе предлагается вернуть данный параметр в заключительный этап методики, обоснование этого предложения приведено далее в описании итогового документирования модели угроз.

В качестве источника информации об угрозах и уязвимостях при построении методики рассматривается база данных угроз ФСТЭК (на текущий момент содержит 217 угроз и 29934 уязвимости), а также базовые и типовые модели угроз безопасности.

Перейдем к более краткому рассмотрению каждого из этапов методики.

На первом этапе необходимо определить возможные негативные последствия на основе законодательства РФ или исходя из проведенной обладателем информации оценки ущерба от нарушения основных процессов и/или от нарушения безопасности обрабатываемой информации. Виды рисков и типовые негативные последствия приводятся в приложении 4 методики.

На втором этапе определяются объекты воздействия угроз безопасности информации. Пример определения объектов и видов воздействия на их приводится в приложении 5 методики. По сути, данный этап представляет собой инвентаризацию инф. ресурсов и объектов инф. инфраструктуры. Для всех выявленных ресурсов определяются виды неправомерного доступа и (или) воздействий, которые могут привести к наступлению негативных последствий, определенных в соответствии с пунктом 4.5 методики: утечка данных, несанкционированный доступ, отказ в обслуживании, нарушение целостности данных, несанкционированное использование вычислительных ресурсов, нарушение функционирования средств обработки информации.

Также на втором этапе оцениваются условия, которые определяют наличие возможности для реализации угроз безопасности. К этим условиям относят уязвимости, незадекларированные возможности в системах, наличие доступа к системам для реализации угроз. Описание уязвимостей, которые могут быть использованы для реализации угроз безопасности информации, содержится в ГОСТ Р 56546-2015 «Защита информации. Уязвимости

информационных систем. Классификация уязвимостей информационных систем».

На первой стадии третьего этапа определяются источники угроз безопасности: техногенные и антропогенные. Упор делается на антропогенные источники: «следует, в первую очередь, уделять внимание оценке антропогенных источников угроз, связанных с действиями нарушителей». Нарушители делятся на две категории:

- внешние - не имеющие полномочий доступа в системе;
- внутренние - обладающие полномочиями доступа в системе.

В методике определены также виды нарушителей, их возможные цели реализации угроз безопасности, а также уровни возможностей (высокий – Н4, средний – Н3, базовый повышенный – Н2, базовый – Н1). Отметим, что в БДУ ФСТЭК нарушители распределены всего по трем уровням (высокий, средний, низкий), поэтому неясно, как проводить корреляцию с уровнем «базовый повышенный» (следует ли отнести его к среднему или к низкому уровню по БДУ ФСТЭК).

По завершении первой стадии должны быть определены источники угроз БИ, возможные цели угроз БИ, категории и виды нарушителей, возможности каждого вида нарушителя.

На второй стадии определяются сценарии реализации угроз безопасности. Возможность реализации любой из угроз определяется существованием хотя бы одного сценария ее реализации. Определение сценария состоит в установлении всей совокупности тактик и техник, применяемых нарушителями. Все основные тактики и техники приводятся в методике в приложении 11.

Наконец, на третьей стадии определяются актуальные угрозы. Именно на этот этап предлагается вернуть расчет проектного параметра «уровень опасности угрозы».

В проекте методики отсутствуют строгие математические обозначения, поэтому автор предлагает ввести их самостоятельно. Для каждой угрозы

формируется перечень возможных сценариев. Пусть для i-той угрозы существует m_i сценариев. Обозначим тип доступа, используемый в сценарии (D) и сложность (P) j-ого сценария как D_{ij} и P_{ij} соответственно, $j = \overline{1, m_i}$. Кроме того, обозначим значимость ресурсов и компонентов, на которые направлена i-ая угроза как S_i. Способы определения значимости ресурсов и компонентов, типа доступа и сложности приведены в приложении 2 в таблицах 1, 2, 3 соответственно, а в таблице 4 приведено соответствие им численных значений. Пусть изначально у нас определено n угроз, тогда обозначим опасность i-ой угрозы при реализации j-ого сценария как Th_{ij}, $i = \overline{1, n}$. Согласно методике, она вычисляется как сумма всех трех параметров:

$$Th_{ij} = D_{ij} + P_{ij} + S_i \ .$$

Согласно проекту методики, уровень угрозы определяется с целью определения наиболее уязвимых информационных ресурсов и компонентов систем и сетей. Поэтому более точно будет определить формулу следующим образом:

$$Th_{ij} = D_{ij} + P_{ij} + max_{k=\overline{1,r}}(S_k) \ ,$$

где k-номер ресурса k, S_k-значимость k-ого ресурса, r - количество ресурсов, на которые направлена i-ая угроза. Т.е. общая значимость ресурсов и компонентов определяется как максимум из значимостей каждого из ресурсов. Со смысловой точки зрения это означает, что при расчете опасности угрозы учитывается максимальный ущерб, который может быть нанесен, в случае ее реализации (ущерб определяется качественно, а не количественно, поэтому предлагается использовать максимум, а не сумму, см. приложение 2, таблица 1).

Как уже упоминалось, уровень опасности угрозы со смысловой точки зрения похож на оценку риска. Уровень опасности вычисляется на основе того, что для некоторой угрозы существует сценарий её реализации, а реализация приводит к некоторому ущербу. Однако риск измеряется на основе критичности уязвимости, или в нашем случае сценария [5], в рассматриваемой

же методике критичность сценария опускается. Отметим данный момент, который будет разобран далее.

Кроме того, возникает также вопрос, как именно вычислять опасность отдельной угрозы в целом (по всем сценариям её реализации). Авторы проекта методики не дали ответа, предлагая рассматривать уровень угроз исключительно отдельно по каждому из сценариев. Однако агрегированный параметр необходим для рассмотрения как с точки зрения приоритезации устранения/обработки угроз, так и для расчета риска в целом по какому-либо ресурсу с целью определения наиболее уязвимых ресурсов (как, например, в методике [5]). Собственно, уровень угрозы для этого и предназначен, но сравнивать угрозы раздельно по каждому сценарию слишком трудозатратно. Поэтому в данной работе предлагается дополнить методику еще одним новым параметром - уровнем опасности угрозы по всем сценариям. Однако какой подход выбрать: суммировать по всем возможным сценариям или вычислять среднее/максимальное по сценариям? Рассмотрим эти варианты.

«Суммирование по сценариям», первый из возможных подходов, является неверным, что легко показать на примере. Допустим, угроза 1 реализуется единственным сценарием и ее опасность $Th_{11} = 10$, а угроза 2 имеет три сценария реализации с соответствующими опасностями $Th_{21} = 4, Th_{22} = 4, Th_{23} = 4$. В таком случае угроза 1 расценивается как менее опасная, нежели угроза 2 ($Th_1 = 10$ против $Th_2 = 12$), хотя уровень опасности угрозы 1 высокий, а у угрозы два по всем трем сценариям низкий.

«Среднее по сценариям». Если брать среднее значение, то может возникнуть такая ситуация, когда большинство сценариев угрозы не опасны, и средний по сценариям уровень низкий, однако при этом имеется сценарий, при котором возможно прекращение работоспособности всей системы при низкой сложности реализации. Пусть условно $Th_{1-3} = 4, Th_4 = 7$. Очевидно, злоумышленником в большинстве случаев скорее всего будет рассматриваться наиболее простой вариант с Th_4, и мы, соответственно недооцениваем

опасность угрозы (среднее значение $Th_{\text{среднее}} = 4{,}75$). В таком случае кажется разумным брать максимальное значение.

«Максимальное по сценариям» действительно является хорошей стратегией, исходя из того, что вычисление уровня угрозы производится на основе доступности и сложности. Собственно, чем доступнее и более прост в реализации сценарий угрозы, тем выше будет уровень опасности по нему и тем предпочтительнее сценарий для нарушителя. Конечно, можно подобрать еще один контрпример: допустим, сценарий, при котором угроза имеет опасность 8, крайне маловероятен (например, необходим физический доступ к хорошо охраняемому помещению), и с другой стороны, сценарий, имеющий опасность 7 является наиболее «удобным» при реализации угрозы. В данном случае мы переоцениваем угрозу. Однако в целом, такая ситуация является редкой, а сама ошибка не столь критична, поэтому вариант с максимумом является наилучшим среди рассмотренных и будет включен в разрабатываемую систему.

Здесь мы подходим к параметру, который упоминался чуть ранее и который авторы проекта методики решили не рассматривать: вероятность реализации угрозы заданным сценарием, или критичность сценария. Он был заменен комбинацией доступность + сложность. С одной стороны, это плюс, т.к. параметр вероятности реализации через конкретную уязвимость очень сложно оценить: зачастую нет никакой статистической информации о реализации угроз раздельно по каждой уязвимости в системе (особенно если она введена в эксплуатацию недавно), и обычно дается некоторая субъективная экспертная оценка, зачастую весьма приближенная. С другой стороны, становится проблемой вычислить агрегированную опасность угрозы по всем сценариям.

Автор данной работы предлагает ещё один подход в дополнение к варианту «максимальное по сценариям», который бы сохранил идею разработчиков методики, и при этом учел вероятность каждого сценария. В

основе лежит предположение о том, что доступность сценария и его сложность напрямую влияют на его выбор злоумышленником для реализации угрозы.

Введем обозначение Pr_{ij} для обозначения вероятности предпочтения злоумышленником для i-той угрозы j-того сценария.

Очевидным способом перевести тип доступа и сложность в вероятность является нормализация по всем сценариям для заданной угрозы:

$$Pr_{ij} = \frac{D_{ij} + P_{ij}}{\sum_l (D_{il} + P_{il})}\,. \qquad (1)$$

Однако в данном случае значение Pr_{ij} не будет полезным для вычисления результирующей опасности угрозы, т.к. в итоге при подсчете будет получено среднее арифметическое по всем Th_{ij} (поэтому далее формула (1) будет обозначаться как «подход среднего») и возникнет проблема, описанная при рассмотрении «Среднее по сценариям». Необходимо добавить коэффициент, который бы смещал вероятности в сторону сценариев i-той угрозы, имеющих большее значение $D_{ij} + P_{ij}$.

Введем вспомогательную функцию σ(x), определяемую как:

$$\sigma(x) = \begin{cases} 0, & x \le 0\,; \\ x, & x > 0\,. \end{cases}$$

Тогда предлагается использовать следующую формулу для расчета вероятности реализации угрозы i посредством сценария j:

$$Pr_{ij} = \frac{D_{ij} + P_{ij} + \sum_{l \neq j} \rho\sigma\left(D_{ij} + P_{ij} - (D_{il} + P_{il})\right)}{\sum_j \left(D_{ij} + P_{ij} + \sum_{l \neq j} \sigma\left(D_{ij} + P_{ij} - (D_{il} + P_{il})\right)\right)}, \qquad (2)$$

где ρ – некоторый коэффициент смещения вероятности в сторону сценариев, имеющих большее значение $D_{ij} + P_{ij}$. Чем больше ρ , тем больше смещение. Данный коэффициент предлагается для возможности более индивидуальной настройки методики специалистами, при работе с ней. По умолчанию при дальнейшем рассмотрении примем $\rho = 1$.

Описанная выше формула не носит случайного характера и у автора есть обоснование по ее построению.

Во-первых, D_{ij}, P_{ij} всегда положительны, а функция $\sigma(x)$ гарантирует неотрицательность дополнительного слагаемого $\sum_{l\neq j}\rho\sigma\,(D_{ij}+P_{ij}-(D_{il}+P_{il}))$. Таким образом, как числитель, так и соответственно, знаменатель, всегда являются положительными числами.

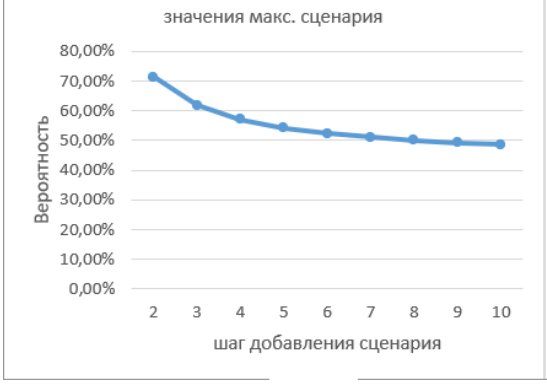
Во-вторых, $D_{ij}+P_{ij}-(D_{il}+P_{il})$ - параметр, напрямую отражающий взаимоотношение сценария, для которого вычисляется вероятность, со всеми остальными сценариями. Чем более прост и доступен сценарий по сравнению с остальными, тем больше слагаемое $\rho\sum_{l\neq j}\sigma\,(D_{ij}+P_{ij}-(D_{il}+P_{il}))$, и соответственно тем больше смещение вероятности в сторону данного сценария.

И, наконец, в-третьих, нормализация по сумме гарантирует, что сумма всех вероятностей сценариев заданной угрозы $\sum_j Pr_{ij}=1$, т.е. вероятностная интерпретация корректна.

Кроме того, была проведена серия вычислительных экспериментов по данной формуле для исследования её поведения при различных значениях $D_{ij}+P_{ij}$. Результаты представлены на рисунках 2, 3, 4.

Как видно из рисунка 3, в уже описанной ранее ситуации, когда один сценарий сильно превалирует над другими по показателю $D_{ij}+P_{ij}$, снижение вероятности выбора данного сценария уменьшается на каждом шаге добавления в расчет маловероятного сценария. Таким образом вероятность данного сценария в таком примере стремится к некоторой асимптоте согласно полученному графику.

значение $D_{ij}+P_{ij}$ / количество сценариев	7	4	4	4	4	4	4	4	4	4	сумма
2	71,43%	28,57%									100%
3	61,90%	19,05%	19,05%								100%
4	57,14%	14,29%	14,29%	14,29%							100%
5	54,29%	11,43%	11,43%	11,43%	11,43%						100%
6	52,38%	9,52%	9,52%	9,52%	9,52%	9,52%					100%
7	51,02%	8,16%	8,16%	8,16%	8,16%	8,16%	8,16%				100%
8	50,00%	7,14%	7,14%	7,14%	7,14%	7,14%	7,14%	7,14%			100%
9	49,21%	6,35%	6,35%	6,35%	6,35%	6,35%	6,35%	6,35%	6,35%		100%
10	48,57%	5,71%	5,71%	5,71%	5,71%	5,71%	5,71%	5,71%	5,71%	5,71%	100%
Подход среднего:	16,28%	9,30%	9,30%	9,30%	9,30%	9,30%	9,30%	9,30%	9,30%	9,30%	100%



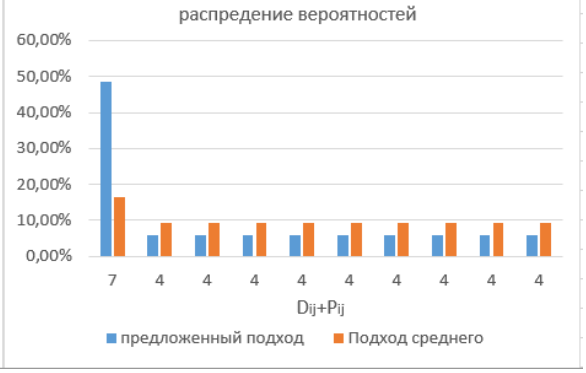
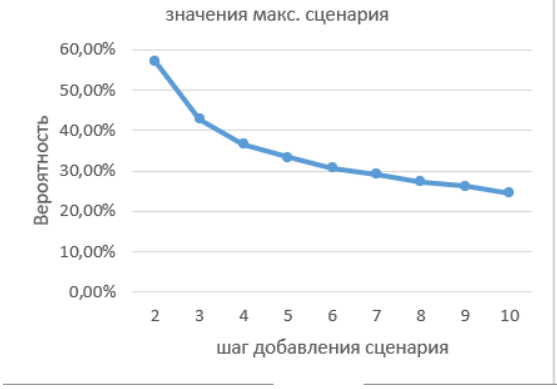


Рисунок 3 – Эксперимент 1 – исследование значения вероятности (по формуле (2)) сценария с наибольшим значением $D_{ij} + P_{ij}$ при добавлении сценариев со значительно меньшими значениями $D_{ij} + P_{ij}$ и сравнение с распределением вероятности при стандартном подходе по формуле (1)

На рисунке 4 продемонстрирована состоятельность формулы в примере, когда таких превалирующих сценариев несколько – сценарий с наибольшим показателем $D_{ij} + P_{ij}$ имеет наибольшую вероятность, следующие за ним два сценария также получают значительные вероятности, но со смещением вниз относительно первого, вероятности маловероятных сценариев дополнительно занижены, что хорошо видно на диаграмме распределения вероятностей в сравнении с подходом среднего.

значение $D_{ij}+P_{ij}$ / количество сценариев	7	6	6	5	5	4	4	3	3	2	сумма
2	57,14%	42,86%									100%
3	42,86%	28,57%	28,57%								100%
4	36,67%	23,33%	23,33%	16,67%							100%
5	33,33%	20,51%	20,51%	12,82%	12,82%						100%
6	30,77%	19,23%	19,23%	11,54%	11,54%	7,69%					100%
7	29,23%	18,46%	18,46%	10,77%	10,77%	6,15%	6,15%				100%
8	27,38%	17,86%	17,86%	10,71%	10,71%	5,95%	5,95%	3,57%			100%
9	26,21%	17,48%	17,48%	10,68%	10,68%	5,83%	5,83%	2,91%	2,91%		100%
10	24,62%	16,92%	16,92%	10,77%	10,77%	6,15%	6,15%	3,08%	3,08%	1,54%	100%
Подход среднего:	15,56%	13,33%	13,33%	11,11%	11,11%	8,89%	8,89%	6,67%	6,67%	4,44%	100%



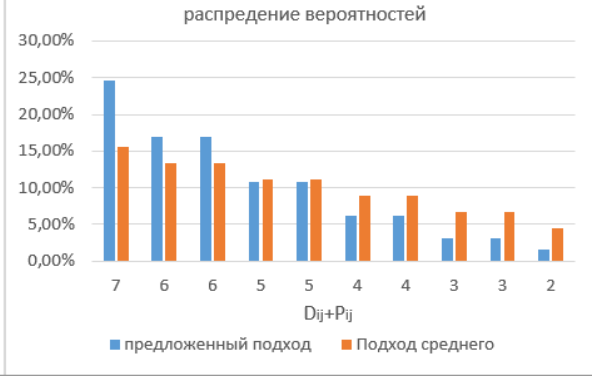
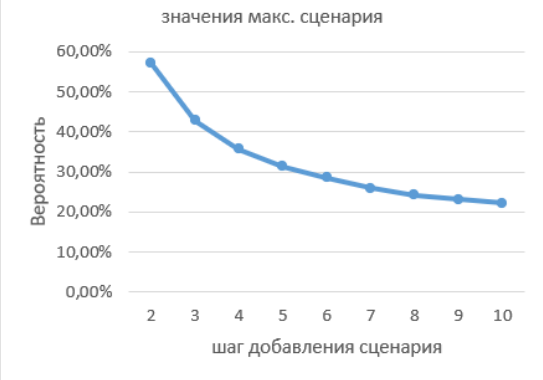


Рисунок 4 – Эксперимент 2 – исследование значения вероятности (по формуле (2)) сценария с наибольшим значением $D_{ij} + P_{ij}$ при добавлении сценариев, как со значительно меньшими значениями $D_{ij} + P_{ij}$, так и с близкими значениями $D_{ij} + P_{ij}$ и сравнение с распределением вероятности при стандартном подходе по формуле (1)

На рисунке 5 представлены результаты вычислений в случае, когда различие в $D_{ij} + P_{ij}$ между сценариями минимально. Видно, что на каждом шаге добавления сценария, разрыв между сценариями увеличивается, и, например, в наблюдаемом эксперименте, сценарий со значением $D_{ij} + P_{ij} = 7$ превосходит сценарий с $D_{ij} + P_{ij} = 6$ в 2 раза, а сценарий с $D_{ij} + P_{ij} = 5$ в 4 раза. В таких случаях можно задавать параметр ρ таким, чтобы он уменьшал смещение, однако для этого нужно вручную рассматривать каждый отдельный случай.

значение $D_{ij}+P_{ij}$ / количество сценариев	7	6	6	6	6	6	5	5	5	5	сумма
2	57,14%	42,86%									100%
3	42,86%	28,57%	28,57%								100%
4	35,71%	21,43%	21,43%	21,43%							100%
5	31,43%	17,14%	17,14%	17,14%	17,14%						100%
6	28,57%	14,29%	14,29%	14,29%	14,29%	14,29%					100%
7	25,93%	12,96%	12,96%	12,96%	12,96%	12,96%	9,26%				100%
8	24,24%	12,12%	12,12%	12,12%	12,12%	12,12%	7,58%	7,58%			100%
9	23,08%	11,54%	11,54%	11,54%	11,54%	11,54%	6,41%	6,41%	6,41%		100%
10	22,22%	11,11%	11,11%	11,11%	11,11%	11,11%	5,56%	5,56%	5,56%	5,56%	100%
Подход среднего:	12,28%	10,53%	10,53%	10,53%	10,53%	10,53%	8,77%	8,77%	8,77%	8,77%	100%



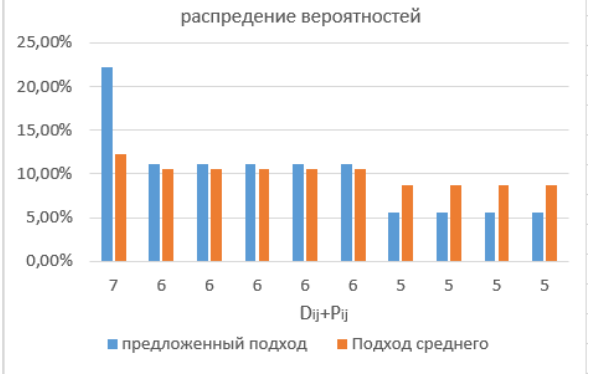


Рисунок 5 – Эксперимент 3 – исследование значения вероятности (по формуле (2)) сценария с наибольшим значением $D_{ij} + P_{ij}$ при добавлении сценариев с незначительно меньшими значениями $D_{ij} + P_{ij}$ и сравнение с распределением вероятности при стандартном подходе по формуле (1)

В качестве второго варианта формулы предлагается следующее решение:

$$Pr_{ij} = \frac{D_{ij} + P_{ij} + \rho * \sigma(\max\left(D_{ij} + P_{ij} - (D_{il} + P_{il})\right))}{\sum_j D_{ij} + P_{ij} + \rho * \sigma(\max\left(D_{ij} + P_{ij} - (D_{il} + P_{il})\right))}. \quad (3)$$

В отличие от формулы (2), эта лишена недостатка, продемонстрированного в эксперименте 3, т.к. дополнительное слагаемое в числителе не будет продолжать расти при добавлении каждого нового сценария с меньшим уровнем доступности и сложности, чем у текущего сценария, а вычисляется как максимум среди всех разностей по сценариям. Функция максимума выбрана, чтобы сохранить логику: чем больше показатель $D_{ij} + P_{ij}$ по сравнению с остальными сценариями, тем больше смещение вероятности. Обоснование формулы аналогично формуле (2).

Для формулы (3) были проведены все аналогичные вычисления, что и для (2). Результаты представлены на рисунках 5, 6, 7.

Как видно из рисунка 6, при сильном превалировании одного сценария над всеми остальными, оценка его вероятности смещается в большую сторону, и уменьшается все меньше с каждым шагом добавления маловероятного сценария, т.е. стремится к некоторой асимптоте, как и в случае с предыдущей формулой.

значение $D_{ij}+P_{ij}$ / количество сценариев	7	4	4	4	4	4	4	4	4	4	сумма
2	71,43%	28,57%									100%
3	55,56%	22,22%	22,22%								100%
4	45,45%	18,18%	18,18%	18,18%							100%
5	38,46%	15,38%	15,38%	15,38%	15,38%						100%
6	33,33%	13,33%	13,33%	13,33%	13,33%	13,33%					100%
7	29,41%	11,76%	11,76%	11,76%	11,76%	11,76%	11,76%				100%
8	26,32%	10,53%	10,53%	10,53%	10,53%	10,53%	10,53%	10,53%			100%
9	23,81%	9,52%	9,52%	9,52%	9,52%	9,52%	9,52%	9,52%	9,52%		100%
10	21,74%	8,70%	8,70%	8,70%	8,70%	8,70%	8,70%	8,70%	8,70%	8,70%	100%
Подход среднего:	16,28%	9,30%	9,30%	9,30%	9,30%	9,30%	9,30%	9,30%	9,30%	9,30%	100%

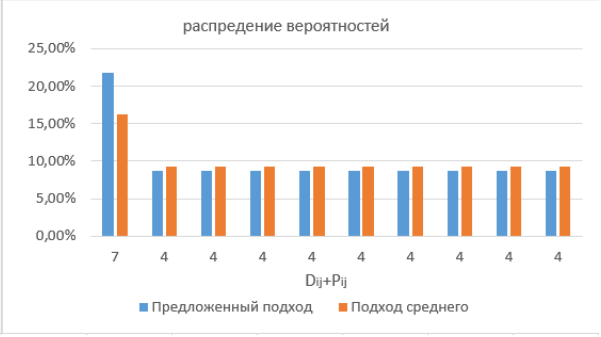
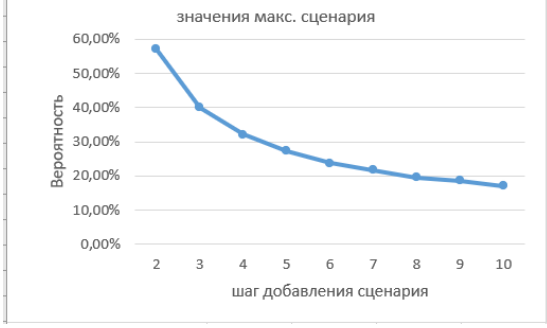


Рисунок 6 – Эксперимент 1 – исследование значения вероятности (по формуле (3)) сценария с наибольшим значением $D_{ij} + P_{ij}$ при добавлении сценариев со значительно меньшими значениями $D_{ij} + P_{ij}$ и сравнение с распределением вероятности при стандартном подходе по формуле (1)

На рисунке 7 представлены результаты вычислений для случая, когда доминирующих сценариев несколько. Если сравнивать с аналогичными вычислениями для формулы (2) (см. рисунок 3), то видно, что формула (3) ближе по значениям к подходу среднего. Впрочем, если необходимо более сильное смещение, достаточно увеличить значение коэффициента ρ.

значение $D_{ij}+P_{ij}$ / количество сценариев	7	6	6	5	5	4	4	3	3	2	сумма
2	57,14%	42,86%									100%
3	40,00%	30,00%	30,00%								100%
4	32,14%	25,00%	25,00%	17,86%							100%
5	27,27%	21,21%	21,21%	15,15%	15,15%						100%
6	23,81%	19,05%	19,05%	14,29%	14,29%	9,52%					100%
7	21,74%	17,39%	17,39%	13,04%	13,04%	8,70%	8,70%				100%
8	19,64%	16,07%	16,07%	12,50%	12,50%	8,93%	8,93%	5,36%			100%
9	18,64%	15,25%	15,25%	11,86%	11,86%	8,47%	8,47%	5,08%	5,08%		100%
10	17,14%	14,29%	14,29%	11,43%	11,43%	8,57%	8,57%	5,71%	5,71%	2,86%	100%
Подход среднего:	15,56%	13,33%	13,33%	11,11%	11,11%	8,89%	8,89%	6,67%	6,67%	4,44%	100%



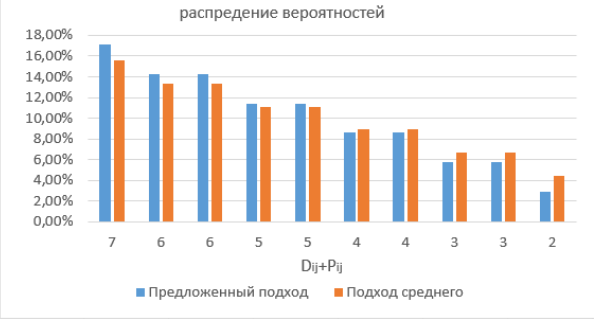


Рисунок 7 – Эксперимент 2 – исследование значения вероятности (по формуле (3)) сценария с наибольшим значением $D_{ij} + P_{ij}$ при добавлении сценариев с незначительно меньшими значениями $D_{ij} + P_{ij}$ и сравнение с распределением вероятности при стандартном подходе по формуле (1)

Наконец, на рисунке 8 представлен случай, когда значения $D_{ij} + P_{ij}$ сценариев «близко» расположены. Как видно, большое количество сценариев не приводит к тому, что вероятность наибольшего из них сильно возрастает над остальными. Более того, даже увеличение коэффициента ρ с 1 до 2 не приводит к резкому росту разницы вероятностей сценариев, что видно из рисунка 9.

Таким образом, предложено два способа вычисления вероятности сценария: с «резким» смещением вероятностей в сторону сценариев с большим значением $D_{ij} + P_{ij}$ по формуле (2), и с более плавным по формуле (3). В разрабатываемую систему предлагается включить оба варианта, однако в связи с описанными выше причинами, при выборе расчета по формуле (2) будет выводится предупреждение, что данный способ может искажать оценки при большом количестве сценариев, и в таких случаях предпочтительнее формула (3).

значение Dij+Pij / количество сценариев	7	6	6	6	6	6	5	5	5	5	сумма
2	57,14%	42,86%									100%
3	40,00%	30,00%	30,00%								100%
4	30,77%	23,08%	23,08%	23,08%							100%
5	25,00%	18,75%	18,75%	18,75%	18,75%						100%
6	21,05%	15,79%	15,79%	15,79%	15,79%	15,79%					100%
7	18,37%	14,29%	14,29%	14,29%	14,29%	14,29%	10,20%				100%
8	16,67%	12,96%	12,96%	12,96%	12,96%	12,96%	9,26%	9,26%			100%
9	15,25%	11,86%	11,86%	11,86%	11,86%	11,86%	8,47%	8,47%	8,47%		100%
10	14,06%	10,94%	10,94%	10,94%	10,94%	10,94%	7,81%	7,81%	7,81%	7,81%	100%
Подход среднего:	12,28%	10,53%	10,53%	10,53%	10,53%	10,53%	8,77%	8,77%	8,77%	8,77%	100%

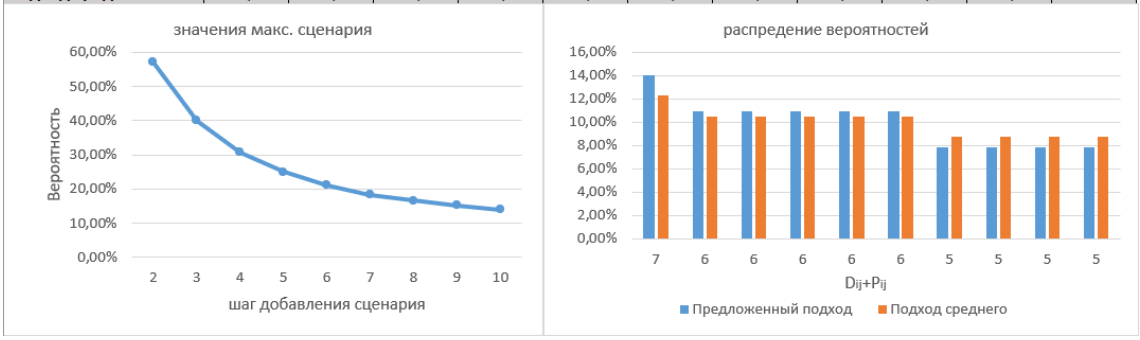


Рисунок 8 – Эксперимент 3 – исследование значения вероятности (по формуле (3)) сценария с наибольшим значением $D_{ij} + P_{ij}$ при добавлении сценариев с незначительно меньшими значениями $D_{ij} + P_{ij}$ и сравнение с распределением вероятности при стандартном подходе по формуле (1)

Теперь вернемся к изначальной цели, ради которой вводилась вероятность реализации сценария: вычисление среднего значения угрозы. Предлагается вычислять его по следующей формуле:

$$Th_i = \sum_j Pr_{ij} * Th_{ij} \ . \qquad (4)$$

Рассмотрим на примере с вероятностями сценариев из эксперимента 1 (рисунок 5 и 6). Пусть $S_i = 1$, тогда по формуле (2) $Th_i = 6{,}457 \ \ (\rho = 1)$, по формуле (3) $Th_i = 5{,}652 \ \ (\rho = 1)$ и $Th_i = 6{,}457$ с $(\rho = 9)$. Для сравнения, если вычислять по формуле среднего, $Th_i = 5{,}488$. Очевидно, оценки по предложенным формулам являются гораздо более приближенными к реальности, нежели подход среднего.

значение $D_{ij}+P_{ij}$ / количество сценариев	7	6	6	6	6	6	5	5	5	5	сумма
2	60,00%	40,00%									100%
3	42,86%	28,57%	28,57%								100%
4	33,33%	22,22%	22,22%	22,22%							100%
5	27,27%	18,18%	18,18%	18,18%	18,18%						100%
6	23,08%	15,38%	15,38%	15,38%	15,38%	15,38%					100%
7	19,64%	14,29%	14,29%	14,29%	14,29%	14,29%	8,93%				100%
8	18,03%	13,11%	13,11%	13,11%	13,11%	13,11%	8,20%	8,20%			100%
9	16,67%	12,12%	12,12%	12,12%	12,12%	12,12%	7,58%	7,58%	7,58%		100%
10	15,49%	11,27%	11,27%	11,27%	11,27%	11,27%	7,04%	7,04%	7,04%	7,04%	100%
Подход среднего:	12,28%	10,53%	10,53%	10,53%	10,53%	10,53%	8,77%	8,77%	8,77%	8,77%	100%

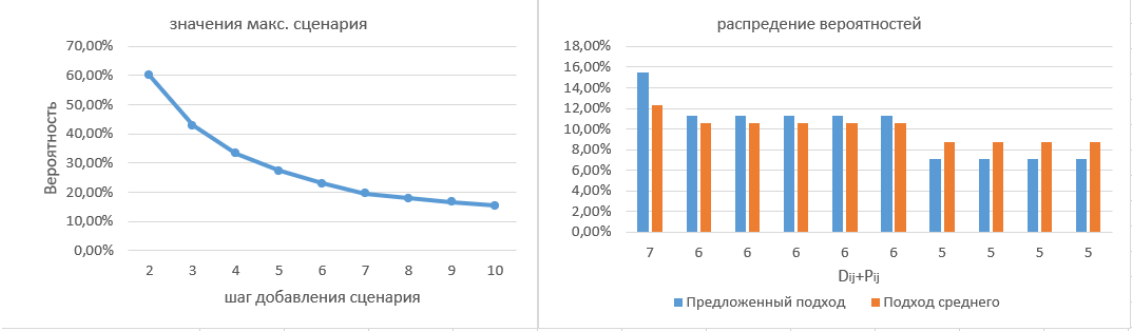


Рисунок 9 – Эксперимент 3 – исследование значения вероятности (по формуле (3)) сценария с наибольшим значением $D_{ij} + P_{ij}$ при добавлении сценариев с незначительно меньшими значениями $D_{ij} + P_{ij}$ и сравнение с распределением вероятности при стандартном подходе по формуле (1). Коэффициент $\rho = 2$

Проект методики предполагает распределение угроз по опасности на три уровня: низкий ($Th_i < 4$), средний ($4 \leq Th_i \leq 7$) и высокий ($8 \leq Th_i$). Однако, т.к. опасность угрозы, вычисленная через среднее значение по сценариям, зачастую не будет являться целочисленной, логичнее хранить её в численном формате. Плюсом будет и то, что при таком подходе будет мало угроз с одинаковым уровнем опасности (в отличие от стандартного, где область определения $Th_i = \{1,2,3,4 \dots 10,11\}$), поэтому их проще будет приоритезировать в очереди на устранение угроз.

Последнее, что нужно упомянуть – подсчет уровня угроз по ресурсу, т.к. кажется разумным рассматривать не просто угрозы в целом, но угрозы раздельно для каждого из активов. Очевидной формулой для расчета общего уровня угроз по отдельно взятому ресурсу является следующая:

$$Th^k = \sum_{i \in T^k} Th_i\,(S_k)\,, \qquad (5)$$

где T^k- множество номеров угроз, затрагивающих k-ый ресурс, $Th_i(S_k)$ – средний или максимальный по сценариям уровень опасности i-той угрозы для

k-ого ресурса (т.е. при вычислении Th_i используется S_k - ценность k-ого ресурса).

Угрозы безопасности информации, согласно методике [7], подлежат описанию в формате, представленном в таблице 1. Последние две строки таблицы «максимальный уровень опасности по сценариям» и «средний уровень опасности угрозы» предложены автором работы в качестве дополнительной информации об угрозе, позволяющей более полно оценить угрозу, расчет предлагается проводить на основе описанных выше методов.

Таблица 1 – Формат описания угрозы безопасности информации

Идентификатор	Указывается идентификатор, содержащийся в банке данных угроз безопасности или ином источнике
Наименование	Указывается наименование угрозы, содержащееся в банке данных угроз безопасности
Описание	Указывается описание угрозы, содержащееся в банке данных угроз безопасности или ином источнике
Сценарии	Описываются все возможные сценарии реализации угроз безопасности информации
Уровень опасности по сценариям	Указывается значение уровня опасности угрозы с конкретным сценарием ее реализации
Максимальный уровень опасности угрозы	Указывается максимальное значение уровня опасности угрозы среди всех сценариев ее реализации
Средний уровень опасности угрозы	Указывается среднее значение уровня опасности угрозы по всем сценариям ее реализации

Также предлагается расширить методику формой отчетности по ресурсу, которая бы включала агрегированные оценки угроз по ресурсу. Предлагаемый формат отчетности представлен в таблице 2. Данная форма позволяет рассмотреть угрозы со стороны информационных ресурсов и(или) компонентов и провести приоритезацию устранения угроз/введения контрмер по ним, что делает методику более гибкой в применении. Либо при построении отчетности по таблице 1, следует группировать таблицы описания угроз по ресурсу, на который они направлены.

Таблица 2 – Формат описания угроз безопасности информации, направленных на конкретный информационный ресурс и(или) компонент

Наименование информационного ресурса и(или) компонента	Указывается наименование согласно технической документации или экспертное наименование
Значимость указанного информационного ресурса и(или) компонента	Указывается значение значимости информационного ресурса согласно таблице 1 Приложения 2
Угрозы, направленные на указанный информационный ресурс и(или) компонент	Перечень угроз, направленных на указанный информационный ресурс и(или) компонент, совместно с максимальным и средним значением уровня опасности для каждой угрозы
Уровень угроз по информационному ресурсу и(или) компоненту	Значение уровня угроз, направленных на указанный информационный ресурс и(или) компонент, рассчитанное по формуле (5)

Для расчета параметра, аналогичного количественной оценке риска ИБ, (назовем его «риск по угрозе») на основе предложенного среднего уровня опасности угрозы необходимо наличие информации о вероятности реализации угрозы в целом (подобная информация гораздо доступнее, нежели информация по реализации угрозы отдельно по каждому сценарию). Это следует из того, что риск ИБ измеряется на основе вероятности реализации события и количественной оценке последствий этого события [1]. Количественная (по уровням) оценка последствий уже заложена в уровень опасности угрозы, не хватает только вероятности реализации угрозы. Таким образом риск по угрозе (обозначим RTh_i) можно вычислить как:

$$RTh_i = Th_i * PTh_i \ , \qquad (6)$$

где PTh_i – вероятность реализации i-той угрозы. Эта вероятность может вычисляться любым из способов для оценки вероятности события.

Риск по угрозе может записываться как дополнительная строка в таблице 1, как параметр, наиболее полно характеризующий угрозу.

Аналогично введем риск по ресурсу (RTh^k), за основу возьмем формулу (5):

$$RTh^k = \sum_{i \in T^k} RTh_i\,(S_k), \qquad (7)$$

где T^k- множество номеров угроз, затрагивающих k-ый ресурс, $RTh_i(\,S_k)$ – риск по i-той угрозе для k-ого ресурса (т.е. при вычислении RTh_i используется S_k - ценность k-ого ресурса). Данный параметр следует записывать отдельной строкой в таблице 2.

Документирование модели угроз.

Построенная на основе методики модель угроз, согласно [8], должна содержать следующие пункты:

1) Общие положения. Содержит назначение и область действия документа, наименование обладателя информации, ответственных за построение модели угроз, а также перечень нормативных правовых документах, использованных при построении модели.

2) Описание систем и сетей и их характеристика как объектов защиты.

3) Возможные негативные последствия от реализации(возникновения) угроз безопасности информации.

4) Возможные объекты воздействия угроз безопасности.

5) Источники угроз безопасности информации и результаты оценки возможностей нарушителя (модель нарушителя).

6) Способы реализации(возникновения) угроз безопасности информации (Сценарии реализации угроз безопасности информации).

7) Актуальные угрозы безопасности информации.

Заключительный раздел включает перечень возможных угроз, возможности злоумышленников, а также сценарии их реализации. В нем же делаются выводы об актуальности угроз. В методике сказано, что к модели угроз, может прилагаться схема с отображением их сценариев, поэтому документация угроз по таблице 1, предлагавшейся в проекте, не противоречит требованиям. Более того в методике в п. 1.6 сказано, что на основе методики

могут разрабатываться ведомственные и корпоративные методики, основным требованием является непротиворечивость положениям основной методике. Параметр опасности угрозы был разработан самим ФСТЭК, и не был включен в итоговый вариант методики, вероятно, лишь с целью её упрощения, что, однако, повлекло к потере возможности приоритизации угроз. Поэтому дополнение методики параметром уровня опасности угрозы, как и введение таблицы 2 в качестве дополнительного формата отчетности, удовлетворяет требованию непротиворечивости и ведет к качественному улучшению, позволяющему проводить приоритезацию по обработке угроз.

3.2. Методы расчета вероятности возникновения угрозы

Риск оценивается на основе вероятности реализации события и количественной оценке последствий. Вероятность реализации события – это вероятность того, что угроза будет успешно реализована посредством эксплуатации некоторой уязвимости в отношении какого-либо актива и повлечет за собой ущерб для организации [4]. В общем виде его расчет на основе этих двух параметров можно представить, как их произведение [7, 8, 9]:

$$\text{Величина риска} = \text{Вероятность события} * \text{размер ущерб.}$$

Вероятность события складывается из вероятности самой угрозы и количественной оценки критичности уязвимости. Вероятность угрозы – вероятность реализации угрозы в отношении какого-либо актива (вероятность наступления события угрозы определяется величиной критичности уязвимости). На практике зачастую применяется не математически рассчитанная вероятность (probability), а частота попыток реализации (likelihood) за заданный отрезок времени [4]. Именно по этой причине в [1] по этому поводу даже есть примечание: «В контексте национального стандарта применительно к количественной оценке риска вместо термина “возможность, вероятность” (probability) используется термин вероятность (likelihood)». Критичность уязвимости – вероятность того, что в случае попытки реализации угрозы посредством данной уязвимости эта попытка будет успешной и

повлечет за собой ущерб для организации. Вероятность события рассчитывается также произведением:

$$\text{Величина риска} = \text{Вероятность угрозы} \times \text{Критичность уязвимости}.$$

Таким образом, величина риска определяется в общем случае как:

$$\text{Величина риска} = \text{Вероятность угрозы} \times \text{Критичность уязвимости} \times \text{размер ущерб}.$$

Интересным подходом является представление событий угроз в виде случайного потока событий, а именно пуассоновского потока. Пуассоновский поток – это математическая модель, позволяющая описывать крайне случайное поведение статистически независимых величин с использованием теории вероятностей. Пуассоновский поток используют для описания многих реальных потоков, таких как несчастные случаи, отказы в обслуживании и т.д., в целом – тогда, когда наступление события зависит от различных источников, независимых друг от друга. Возникновение события угрозы вполне укладывается в такую модель, т.к. возникает вследствие эксплуатации различных уязвимостей различными злоумышленниками. Простейший случайный пуассоновский поток обладает следующими свойствами [9]:

- стационарность – частота возникновения событий $\lambda(t) = const(t)$, т.е. вероятность возникновения n событий зависит только от числа n и длительности промежутка времени t, и абсолютно не зависит от начала отсчета t;
- ординаторность - вероятность возникновения двух событий одновременно в таких потоках равна нулю, т.е. вероятностью наступления одномоментно двух событий за минимальный в системе промежуток времени можно пренебречь по сравнению с вероятностью наступления не более одного события за тот же промежуток времени;
- отсутствие последействия – вероятность возникновения события независима, т.е. не зависит от момента наступления предшествующих ему событий.

Для простейшего потока вероятность возникновения n событий за время t равна:

$$P_n(t) = \frac{(\lambda t)^n \cdot e^{-\lambda t}}{n!},$$

где λ – интенсивность потока (число событий за единицу времени). Обозначим $\lambda = N/t$, где N – среднее количество возникновений угрозы за время t (аналогично $\lambda = 1/T$, где T – среднее время возникновения угрозы).

В нашем случае проще подойти с точки зрения вероятности не возникновения угрозы за время t, ведь в таком случае $n = 0$:

$$P_0(t) = \frac{(\lambda t)^0 \cdot e^{-\lambda t}}{0!} = e^{-\lambda t}.$$

Тогда вероятность возникновения угрозы, т.е. обратного события:

$$P_{\text{у}}(t) = 1 - e^{-\lambda t}. \quad (8)$$

Имея статистику по инцидентам ИБ, можно вычислить вероятность возникновения угрозы. Пример расчета представлен в таблице 3, а на рисунке 10 соответствующий график вероятности возникновения угрозы в зависимости от статистики возникновения.

Таблица 3 – Пример расчета вероятности возникновения угрозы при рассмотрении потока событий возникновения угроз как простейшего пуассоновского потока

N – среднее количество возникновений угрозы, кол-во/год	0,25	1	2	3	4
λ – интенсивность потока угроз, кол-во/год	0,25	1	2	3	4
$P_{\text{у}}(t)$ – вероятность возникновения угрозы, $t = 1$ год	0,22	0,63	0,86	0,95	0,98
T – среднее время возникновения угрозы, год	4	1	0,5	0,33	0,25

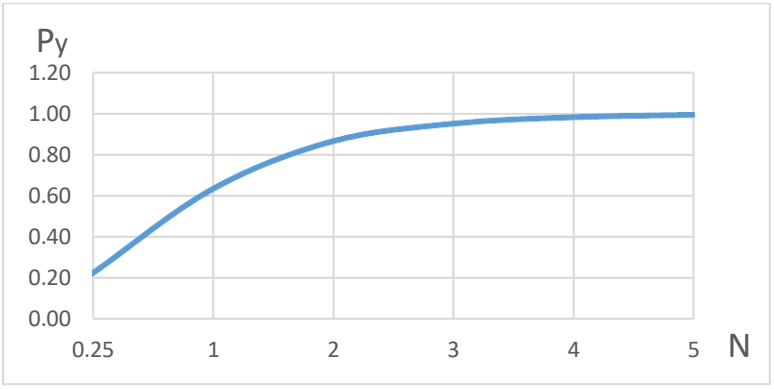


Рисунок 10 – Вероятность возникновения угрозы ($P_У$) за время t в зависимости от числа возникновений угроз за время t

К сожалению, иногда нет никакой информации о частоте инцидентов с угрозами информационной безопасности, однако рассчитать риски необходимо. Как поступить в таком случае? В ГОСТе [1] в приложении «Подходы к оценке риска ИБ» предлагается вариант вычисления степени вероятности на основе уровня угрозы и уровня уязвимости. Вычисление проводится на основе специальной таблицы (таблица 4). Информацию об уровнях угроз и уязвимостей предлагается собирать с помощью контрольных листов, заполняемых квалифицированным техническим персоналом, на основе данных обследований фактического месторасположения и проверки документации.

Таблица 4 – Вероятность угрозы (осуществления сценария инцидента ИБ) в зависимости от уровня угрозы и уровня уязвимости

Уровни угрозы	Низкая			Средняя			Высокая		
Уровни уязвимости	Н	С	В	Н	С	В	Н	С	В
Степень вероятности угрозы	0	1	2	1	2	3	2	3	4

Например, для среднего уровня угрозы и среднего уровня уязвимости степень вероятности угрозы равна 2. При использовании в методиках, подразумевающих значение вероятности [0,1], рассчитанное с помощью таблицы значение, обозначим $P_{\text{таб}}$, может быть нормировано:

$$P_{\text{нормир}} = P_{\text{таб}}/4,$$

где $P_{\text{нормир}}$ – нормированное на [0,1] значение вероятности угрозы (представляет собой множество $\{0.25, 0.5, 0.75, 1\}$.

4. Оценка рисков информационной безопасности

4.1. Методики оценки рисков информационной безопасности

В данном разделе будут рассмотрено несколько методик оценки рисков. Условно все методики оценки рисков ИБ можно условно разделить на три группы в зависимости от идей, лежащих в их основе:

- качественные методики, в которых риск связывается с субъективными качественными (лингвистическими) оценками, например, риск может оцениваться по шкале: «очень высокий», «высокий», «средний», «низкий», «несущественный»;
- количественные методики, в которых риск оценивается с помощью математического аппарата на основе вероятности события, от размеров потерь, вероятности события, вероятности угрозы т.п.;
- смешанные методики, сочетающие в себе как качественные, так и количественные оценки риска [10].

Качественные методики предназначены к использованию, в первую очередь, в тех случаях, когда оценить точно вероятность реализации угрозы или размер потенциального ущерба затруднительно или невозможно. В таком случае обычно задаются приблизительные диапазоны, которым задается качественное определение, например, для ущерба: низкий – [100, 300] тыс. руб., средний - [300, 1000] тыс. руб., высокий – от 1 млн. руб.

Когда все необходимые для вычислений данные имеются в точном виде (стоимость активов, статистика по инцидентам ИБ для расчета вероятности возникновения угрозы, критичности уязвимостей), тогда рекомендуется применять количественные методики, т.к. в них будет представлена более точная оценка риска в виде некоторого численного параметра.

Смешанные методики могут сочетать в себе как качественные, так и количественные оценки. Обычно от качественных методик они отличаются тем, что подразумевают меньшую «размытость» оценок входных параметров, а входные данные преобразуются в качественные не вследствие отсутствия

информации, а намеренно в связи с особенностями методик. Чаще всего на выходе такие методики подразумевают некоторое численное значение риска.

В таблице 5 перечислены представители каждой группы методик.

Таблица 5 – Примеры качественных, количественных и смешанных методик

Качественные методики	Количественные методики	Смешанные методики
FRAP, OCTAVE	ГРИФ 2005, RiskWatch	CRAMM, методика Microsoft

4.2. Методика ГРИФ 2005

Хорошим примером количественной методики является методика ГРИФ 2005 [11] из состава Digital Security Office. Её суть заключается в расчете рисков по типам угроз информационной безопасности на основе модели угроз и уязвимостей.

Для того, чтобы оценить риски, анализируются все возможные угрозы, воздействующие на информационную систему и уязвимости, с помощью которых данные угрозы могут быть реализованы.

С точки зрения применимости в методике, угрозу и уязвимость можно трактовать согласно определениям, данным во введении. Кроме них следует ввести еще несколько понятий, прежде чем переходить к описанию методики.

Напомним, что базовые угрозы в области ИБ подразумевают нарушение конфиденциальности, целостности или доступности (отказ в обслуживании). Критичность реализации угрозы - степень воздействия реализованной угрозы на ресурс, т.е. как сильно реализация угрозы отразится на ресурсе. В рамках ИБ критичность реализации угрозы делят на критичности по конфиденциальности, целостности и доступности.

Уязвимость, через которые угроза может быть реализована, называются свойством угрозы. В свою очередь, вероятность реализации угрозы и

критичность реализации этой угрозы через конкретную уязвимость называют свойствами уязвимости.

На базе понятий, перечисленных выше, определяется следующая модель количественной оценки рисков.

В рамках первого этапа рассчитывается так называемый уровень угрозы по уязвимости с помощью свойств уязвимости- критичности и вероятности реализации угрозы через конкретную уязвимость:

$$Th = \frac{ER}{100} * \frac{P(V)}{100},$$

где ER −критичность реализации угрозы в %, $P(V)$ −вероятность реализации угрозы через конкретную уязвимость V в течение года в %. Уровень угрозы по уязвимости может вычисляться отдельно по конфиденциальности, целостности и доступности, тогда он обозначается специальным индексом (Th_c - конфиденциальность, Th_{in} – целостность, Th_a – доступность). Понятно, что все последующие параметры в таком случае также будут делится на эти 3 типа.

Далее рассчитывается уровень угрозы по всем уязвимостям, с помощью которых может быть реализована данная угроза (допустим, их n штук):

$$CTh = 1 - \prod_{i=1}^{n}(1 - Th_i),$$

где Th_i- уровень угрозы по i-ой уязвимости.

Общий уровень угроз по некоторому ресурсу (допустим для ресурса действует m различных угроз) рассчитывается по следующей формуле:

$$CThR = 1 - \prod_{j=1}^{m}\left(1 - CTh_j\right),$$

где CTh_j- уровень j-ой угрозы по всем её уязвимостям.

Комплексный риск по некоторому ресурсу можно вычислить как:

$$R = CThR * D,$$

где D- критичность ресурса.

D может задаваться по-разному, например, в денежном эквиваленте или в уровнях. В таблице 6 представлено равномерное распределение D по четырем уровням.

Таблица 6 – Пример задания критичности ресурса в уровнях

Уровень	Оценка уровня, в %
1	25
2	50
3	75
4	100

В случае угрозы доступности критичность ресурса вычисляется на основе максимального критичного времени простоя ресурса в год (пороговое значение простоя ресурса, например, серверов информационной системы, который будет считаться критичным для организации). Оно обозначается как T_{max}. Критичность ресурса будет вычисляться как:

$$D_{a/\text{год}} = D_{a/\text{час}} * T_{max},$$

где $D_{a/\text{год}}$ – критичность ресурса по угрозе доступности в течение года, $D_{a/\text{час}}$ – аналогично, критичность ресурса по угрозе доступности в течение часа, T_{max} – максимальное критичное время простоя в часах.

Для угроз конфиденциальности и целостности критичность ресурса задается просто в год.

В режиме работы с тремя типами угроз комплексный риск по всем типам угроз вычисляется по формуле:

$$R_{\sum} = \left(1 - \left(1 - \frac{R_c}{100}\right)\left(1 - \frac{R_{in}}{100}\right)\left(1 - \frac{R_c}{100}\right)\right) * 100 \ .$$

Если D задавалось в деньгах, то риск по всей системе можно вычислить как сумму рисков для всех ресурсов системы (пусть всего их N):

$$CR = \sum_{l=1}^{N} R_l,$$

где $R_l -$ риск для l-ого ресурса.

Аналогично, в случае режима работы с 3-мя типами угроз риск по системе будет вычисляться как:

$$CR_{\sum} = \sum_{l=1}^{N} R_{a,l} + \sum_{l=1}^{N} R_{in,l} + \sum_{l=1}^{N} R_{c,l},$$

где $R_{a,l}$ – риск угроз доступности для l-ого ресурса, $R_{in,l}$ – риск угроз целостности для l-ого ресурса, $R_{c,l}$ – риск угроз конфиденциальности для l-ого ресурса.

Если же D задавался в уровнях, то для вычисления риска по всей системе используется следующая формула:

$$CR = \left(1 - \prod_{l=1}^{N}\left(1 - \frac{R_l}{100}\right)\right) * 100\ ,$$

где R_l – риск для l-ого ресурса.

Риск по всей системе может вычисляться раздельно по типам угроз и обозначаться как CR_c, CR_{in}, CR_a (по конфиденциальности, целостности и доступности соответственно).

Аналогично, в случае режима работы с 3-мя типами угроз риск по системе будет вычисляться как:

$$R_{\Sigma} = \left(1 - \left(1 - \frac{CR_c}{100}\right)\left(1 - \frac{CR_{in}}{100}\right)\left(1 - \frac{R_a}{100}\right)\right) * 100\ .$$

В результате применения методики должны быть получены следующие данные:

- риск реализации по трем типам базовых угроз (или по одной суммарной угрозе) для ресурса;
- риск реализации суммарно по всем угрозам для ресурса;
- риск реализации по трем типам базовых угроз (или по одной суммарной угрозе) для информационной системы;
- риск реализации по всем угрозам для информационной системы.

4.3. Алгоритм нечеткого логического вывода оценки рисков ИБ

Важным вопросом для оценки рисков ИБ является не только выбор методики расчета, но также и состоятельность данной методики, точность получаемой на выходе оценки.

В качественных и смешанных методиках оценки рисков значения многих критериев, такие как вероятность реализации угрозы или величина возможного ущерба определяются экспертной оценкой по вербальным

шкалам, например, «часто, редко, крайне редко» или «высокий, средний, низкий» соответственно. Далее на основе некоторой совокупности правил методики выводится такая же качественная оценка риска. Плюсом данных методик является факт их применимости в тех случаях, когда для расчета каких-то необходимых критериев нет возможности использовать теоретико-информационный численный подход, например, нет накопленной статистики по реализации угроз для вычисления частоты инцидентов, а также возможность комбинировать численный и качественных подход (для смешанных методик) [12].

Более того, даже когда для представления информации о вероятности реализации угрозы, имеющихся уязвимостях и стоимости ресурсов используется численное представление, часто оно имеет «неточный», приближенный характер, например, при оценке экспертами на основе опыта или при использовании методов, вычисляющих «среднее» значение.

Наличие некоторой неопределенности в изучаемой системе или качественной информации о ней становится базовой предпосылкой, актуализирующей применение математического аппарата нечёткой логики (Fuzzy Logic) [13]. К преимуществам нечеткой логики относят её универсальность: она позволяет построить модель заданного объекта лишь на основе эвристической / эмпирической информации, предоставленной экспертами или полученной в результате проведения экспериментов. К минусам следует отнести сильную зависимость скорости работы от количества управляющих (продукционных) правил в системе.

Аппарат нечеткой логики предоставляет два важных механизма: фаззификацию и дефаззификацию. Фаззификация – переход от исходных данных, представленных в числовом виде, к некоторой нечеткой интерпретации на основе лингвистических переменных; дефаззификация – соответственно, обратный переход от лингвистических переменных к выходным четким значениям. Между фаззификацией и дефаззификацией

находится этап вывода системой продукционных правил нечеткой логики [14]. Общая схема процесса представлена на рисунке 11.

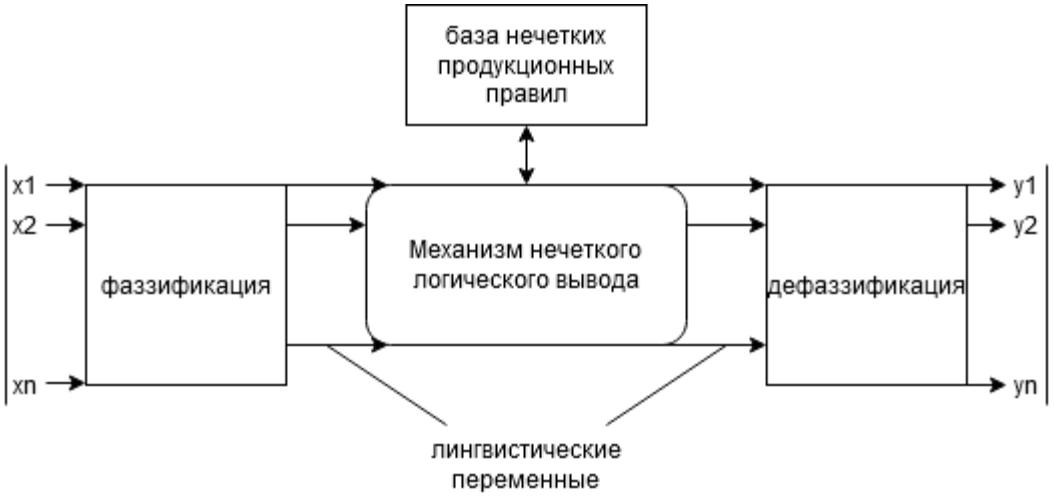


Рисунок 11 – Схема процесса вывода на основе нечеткой логики

Возможность получать относительно достоверное решение в обстоятельствах субъективности и неточности экспертной оценки – главное преимущество применения нечеткой логики при решении задачи оценки риска, вызывающее все больший интерес в последние годы.

В настоящее время предложено уже много вариантов алгоритмов нечеткого вывода, например, Ларсена, Мамдани, Сугено. В работе [15] приводится их упрощенное обобщение, а в работе [12] также предлагается обобщенный алгоритм нечеткого вывода конкретно для задачи оценки риска ИБ. Если объединить результаты обеих работ, получим наиболее полное описание возможного алгоритма:

1) постановка и формализация задачи на вербальном уровне;

2) определение набора входных и, соответственно, выходных переменных;

3) построение набора функций-принадлежности терм-множеств;

4) формирование базы нечетких продукционных правил;

5) реализация механизма фаззификации входных переменных;

6) реализация механизма нечеткого логического вывода: агрегирование подусловий в нечетких правилах продукций, активизация

подзаключений в нечетких правилах продукций, аккумуляция заключений в нечетких правилах продукций;

7) реализация механизма дефаззификации выходных переменных;

8) проведение тестирования и отладки полученного алгоритма нечеткой логики для оценки рисков.

Разберем подробно реализацию этапов описанного алгоритма на примере методических рекомендаций из ГОСТа [1], данных в приложении Е.2.1.

Этап 1. Постановка и формализация задачи на вербальном уровне.

Предложенная матричная методика при расчете риска опирается на качественные оценки, которые предлагается получать, например, на основе опросных листов (в источнике описаны подробные принципы оценки). Информация для вычисления риска по методике представлена в таблице 7. В исходной таблице значения риска распределены от 0 до 8, однако в методике есть рекомендательная шкала, согласно которой таблица была упрощена (0-2 – низкий уровень риска, 3-5 – средний, 6-8 – высокий).

Таблица 7 – Табличная методика оценки рисков

степень вероятности возникновения угрозы		низкая			средняя			высокая		
простота использования		низкая	средняя	высокая	низкая	средняя	высокая	низкая	средняя	высокая
ценность активов	0	Н	Н	Н	Н	Н	С	Н	С	С
	1	Н	Н	С	Н	С	С	С	С	С
	2	Н	С	С	С	С	С	С	С	В
	3	С	С	С	С	С	В	С	В	В
	4	С	С	В	С	В	В	В	В	В

Методика содержит качественные критерии, включает в себя все классические понятия, такие как вероятность угрозы, простота использования (уровень) уязвимости, ценность актива, и поэтому хорошо подходит для полного раскрытия применения нечеткого логического вывода к задаче оценки риска ИБ. Отметим, что несмотря на, казалось бы, численное

представление ценности актива, если он оценивается качественно, то данную оценку нельзя рассматривать как числовую в дальнейшем.

Этап 2. Определение набора входных и выходных переменных.

Для всех перечисленных выше понятий или критериев введем соответствующие им входные переменные. Для оценки риска R_i по всем угрозам $T_i, i = \overline{1,m}$, предлагается ввести:

1) $x_1^{(i)}$– вероятность возникновения i-й угрозы ($P_{\text{угр}}$);

2) $x_2^{(i)}$– простота использования (уровень) уязвимости для реализации i-й угрозы, $P_{\text{уязв}}$;

3) $x_3^{(i)}$– максимально возможный ущерб инф. активу при воздействии на него i-й угрозы (ценность (критичность) информационного ресурса по отношению к i-й угрозе).

Заметим, что вероятность возникновения угрозы может вычисляться не только методом опроса, но и другими способами, предложенными в разделе «Методы расчета вероятности возникновения угрозы», и, например, нормироваться по трем уровням для соответствия таблице 7. Уровень уязвимости также может быть заменен на вероятность использования уязвимости, если имеется информация для её расчета. В случае, если используется все-таки простота использования из таблицы 7, оценку по уровням «высокий», «средний», «низкий» следует нормализовать на отрезок [0,1], например, равномерно как значения 0.99, 0.66, 0.33 соответственно. Аналогично, для ценности активов оценки $\{0, 1, 2, 3, 4\}$ можно нормализовать на [0,1] как $\{0,\ 0.2,\ 0.4, 0.6, 0.8,\ 1\}$.

Таким образом все входные переменные принимают вещественные значения на отрезке [0,1].

В качестве выходных переменных $y_j, j = \overline{1,n}$ можно рассматривать значения риска R_j по всем угрозам $T_i, i = \overline{1,m}$ для j-ого информационного актива.

Этап 3. Построение набора функций-принадлежности терм-множеств.

Сначала задаются сами терм-множества для каждой из входных и выходных переменных. Их элементами являются вербальные (качественные) значения лингвистических переменных: вероятность возникновения угрозы, простота использования (уровень) уязвимости и ценность активов. Тогда, согласно таблице 7, можно выделить терм-множества следующим образом:

$$\text{Вероятность\_угрозы } X_1^{(i)} = \{\text{Низкая}, \text{Средняя}, \text{Высокая}\};$$

$$\text{Уровень\_уязвимости } X_2^{(i)} = \{\text{Низкий}, \text{Средний}, \text{Высокий}\};$$

$$\text{Ценность\_актива } X_3^{(i)} = \{0, 1, 2, 3, 4\};$$

$$\text{Риск } R_j = \{\text{Низкий}, \text{Средний}, \text{Высокий}\};$$

$$i = \overline{1, m}, j = \overline{1, n}.$$

Обозначим заданные термы как: Низкий – Н, Средний – С, Высокий – В.

Функция принадлежности – функция, определяющая степень принадлежности элемента пространства рассуждения к данному нечеткому множеству, является обобщением характеристической функции для классических множеств. Функции принадлежности бывают различных видов: столбчатая, кусочно-линейные, Z- и S-образные, П-образные. Выбор вида функции зависит от неопределенности, заложенной в задаче. В нашем случае неопределенность заключается в экспертной оценке, которая «приблизительно равна» объективной, может находиться, как «среднее». В таком случае, согласно [13], применимы кусочно-линейные функции.

К кусочно-линейным функциям принадлежности относят треугольные, трапециевидные.

В общем случае треугольная функция принадлежности задается как:

$$\mu(x_i, a, b, c) = \begin{cases} 0, \text{для } 0 \leq x_i \leq a; \\ (x_i - a)/(b - a), \text{для } a \leq x_i \leq b; \\ (c - x_i)/(c - b), \text{для } b \leq x_i \leq c; \\ 0, \text{для } c \leq x_i \leq 1. \end{cases}$$

Трапециевидная функция принадлежности задается как:

$$\mu(x_i, a, b, c, d) = \begin{cases} 0, \text{для } 0 \leq x_i \leq a; \\ (x_i - a)/(b - a), \text{для } a \leq x_i \leq b; \\ 1, \text{для } b \leq x_i \leq c; \\ (d - x_i)/(d - c), \text{для } c \leq x_i \leq d; \\ 0, \text{для } c \leq x_i \leq 1. \end{cases}$$

Для дефаззификации удобно использовать наиболее простой вид функции принадлежности – столбчатый, благодаря которому процедура значительно упрощается [12]. Однако всегда можно выбрать любую другую, сравнив, какая на практике дает наиболее объективную оценку на конечных этапах алгоритма.

Столбчатая функция принадлежности определяется всего одним аргументом:

$$\mu(x_i, a) = \begin{cases} 1, \text{для } x_i = a; \\ 0, \text{в остальных случаях.} \end{cases}$$

Параметры a, b, c, d для заданных функций ограничены следующим образом:$0 \leq a \leq b \leq c \leq d \leq 1$.

По умолчанию предлагается задать функции принадлежности как показано в таблице 8.

Таблица 8 – Пример задания функций принадлежности для терм-множеств переменных

переменная	**терм-множество**	**вид функции принадлежности**	**Значения параметров функции принадлежности**			
			a	**b**	**c**	**d**
Вероятность_угрозы $X_1^{(i)}$	Низкая	треугольная	0	0	0.5	-
	Средняя	треугольная	0	0.5	1	-
	Высокая	треугольная	0.5	1	1	-
Уровень_уязвимости $X_2^{(i)}$	Низкий	трапециевидная	0	0	0.2	0.5
	Средний	треугольная	0.2	0.5	0.8	-
	Высокий	трапециевидная	0.5	0.8	1	1
Ценность_актива $X_3^{(i)}$	0	трапециевидная	0	0	0.1	0.25
	1	треугольная	0.1	0.3	0.5	-
	2	треугольная	0.25	0.5	0.25	-
	3	треугольная	0.5	0.7	0.9	-
	4	трапециевидная	0.75	0.9	1	1
Риск R_j	Низкий	столбчатая	0.2	-	-	-
	Средний	столбчатая	0.6	-	-	-
	Высокий	столбчатая	1	-	-	-

На рисунке 12 продемонстрированы графики функций принадлежности, заданные в таблице 8.

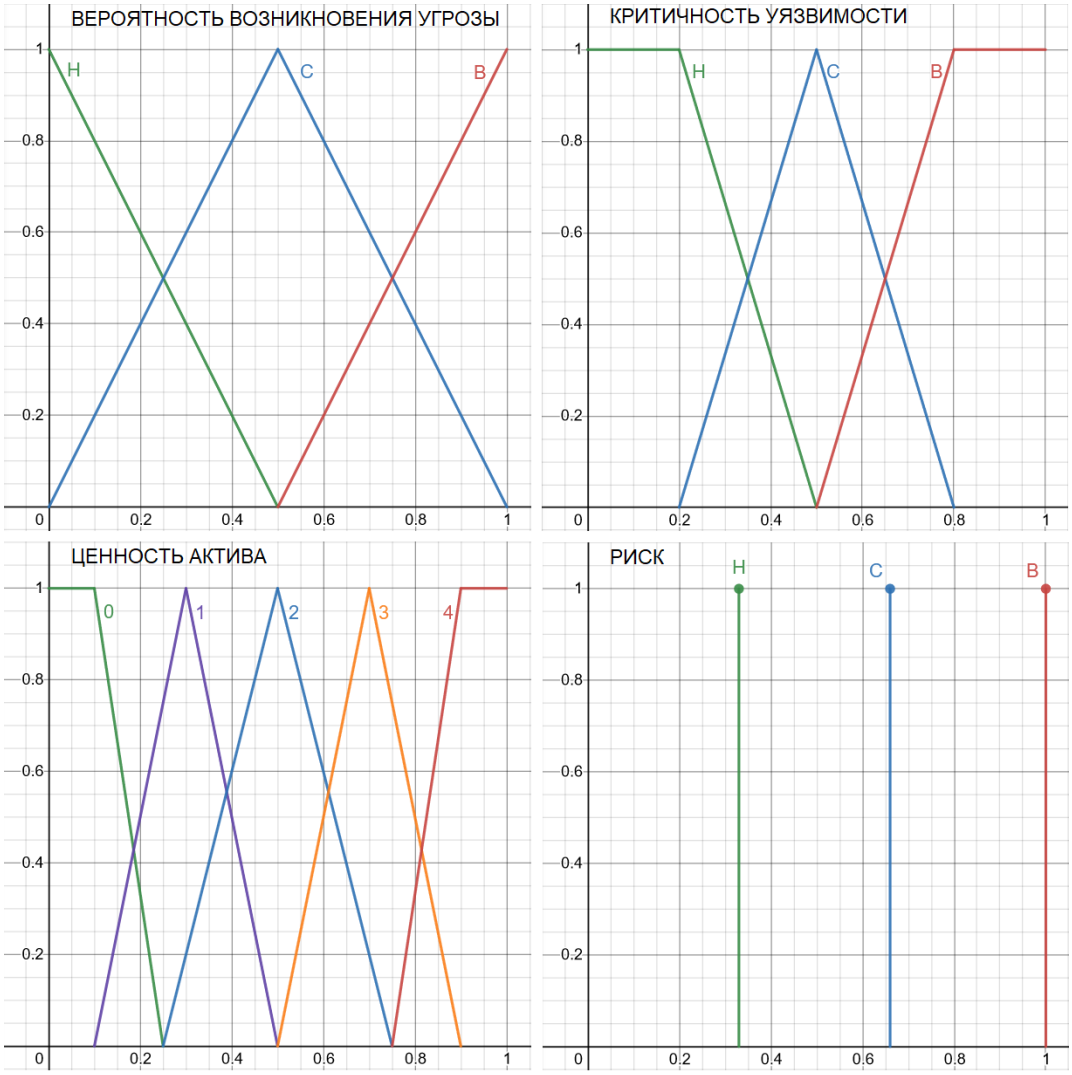


Рисунок 12 – Функции принадлежности терм-множеств, заданных в таблице 7

Этап 4. Формирование базы нечетких продукционных правил.

Для трех входных переменных x_1, x_2, x_3, имеющих терм-множества из трех, трех и пяти термов соответственно можно составить $3^2 \cdot 5 = 45$ правил, соответствующих всем возможным комбинациям термов. Наиболее удобной формой для представления этих правил можно считать табличную, всего для задания всех правил необходимо три таблицы, которые представлены на рисунке 13.

X1=		H				
X2=	H	H	H	H	C	C
	C	H	H	C	C	C
	B	H	C	C	C	B
X3=		0	1	2	3	4

X1=		C				
X2=	H	H	H	C	C	C
	C	H	C	C	C	B
	B	C	C	C	B	B
X3=		0	1	2	3	4

X1=		B				
X2=	H	H	C	C	C	B
	C	C	C	C	B	B
	B	C	C	B	B	B
X3=		0	1	2	3	4

Рисунок 13 – Представление правил для оценки риска в табличной форме

Правила в «классической» форме представлены в таблице 9.

Таблица 9 - Набор нечетких продукционных правил для методики, представленной в таблице 7

Антецедент	Консеквент
(x1=H ∧ x2=H ∧ x3=0) ∨ (x1=H ∧ x2=H ∧ x3=1) ∨ (x1=H ∧ x2=H ∧ x3=2) ∨ (x1=H ∧ x2=C ∧ x3=0) ∨ (x1=H ∧ x2=C ∧ x3=1) ∨ (x1=H ∧ x2=B ∧ x3=0) ∨ (x1=C ∧ x2=H ∧ x3=0) ∨ (x1=C ∧ x2=H ∧ x3=1) ∨ (x1=C ∧ x2=C ∧ x3=0) ∨ (x1=B ∧ x2=H ∧ x3=0)	*Риск = H*
(x1=H ∧ x2=H ∧ x3=3) ∨ (x1=H ∧ x2=H ∧ x3=4) ∨ (x1=H ∧ x2=C ∧ x3=2) ∨ (x1=H ∧ x2=C ∧ x3=3) ∨ (x1=H ∧ x2=C ∧ x3=4) ∨ (x1=H ∧ x2=B ∧ x3=1) ∨ (x1=H ∧ x2=B ∧ x3=2) ∨ (x1=H ∧ x2=B ∧ x3=3) ∨ (x1=C ∧ x2=H ∧ x3=2) ∨ (x1=C ∧ x2=H ∧ x3=3) ∨ (x1=C ∧ x2=H ∧ x3=4) ∨ (x1=C ∧ x2=C ∧ x3=1) ∨ (x1=C ∧ x2=C ∧ x3=2) ∨ (x1=C ∧ x2=C ∧ x3=3) ∨ (x1=C ∧ x2=B ∧ x3=0) ∨ (x1=C ∧ x2=B ∧ x3=1) ∨ (x1=C ∧ x2=B ∧ x3=2) ∨ (x1=B ∧ x2=H ∧ x3=1) ∨ (x1=B ∧ x2=H ∧ x3=2) ∨ (x1=B ∧ x2=H ∧ x3=3) ∨ (x1=B ∧ x2=C ∧ x3=0) ∨ (x1=B ∧ x2=C ∧ x3=1) ∨ (x1=B ∧ x2=C ∧ x3=2) ∨ (x1=B ∧ x2=B ∧ x3=0) ∨ (x1=B ∧ x2=B ∧ x3=1)	*Риск = C*
(x1=H ∧ x2=B ∧ x3=4) ∨ (x1=C ∧ x2=C ∧ x3=4) ∨ (x1=C ∧ x2=B ∧ x3=3) ∨ (x1=C ∧ x2=B ∧ x3=4) ∨ (x1=B ∧ x2=H ∧ x3=4) ∨ (x1=B ∧ x2=C ∧ x3=3) ∨ (x1=B ∧ x2=C ∧ x3=4) ∨ (x1=B ∧ x2=B ∧ x3=2) ∨ (x1=B ∧ x2=H ∧ x3=3) ∨ (x1=B ∧ x2=B ∧ x3=4)	*Риск = B*

Естественно, табличная форма представления гораздо компактнее и удобнее, однако представление в виде продукционных правил необходимо для дальнейшего понимания механизма нечеткого вывода.

Этап 5. Фаззификация входных переменных.

Данный этап, как и все последующие, проще всего продемонстрировать на конкретном примере.

Допустим, в результате экспертной оценки, были заданы следующие значения входных переменных: $x_1^* = 0.2$, $x_2^* = 0.3$, $x_3^* = 0.4$.

Тогда в соответствии с заданными в таблице 7.1 ФП получаем:

$$\alpha_1^{\text{Н}} = \mu_{\text{Н}}(X_1)|_{X_1^*=0.2} = 0.6\,;\alpha_1^{\text{С}} = \mu_{\text{С}}(X_1)|_{X_1^*=0.2} = 0.4\,;\alpha_1^{\text{В}} = 0;$$

$$\alpha_2^{\text{Н}} = \mu_{\text{Н}}(X_2)|_{X_2^*=0.3} = 0.67\,;\alpha_2^{\text{С}} = \mu_{\text{С}}(X_2)|_{X_2^*=0.3} = 0.33\,;\alpha_2^{\text{В}} = 0;$$

$$\alpha_3^0 = 0\,;\alpha_3^1 = \mu_1(X_3)|_{X_3^*=0.4} = 0.5\,;\alpha_3^2 = \mu_2(X_3)|_{X_3^*=0.4} = 0.6\,;\alpha_3^3 = 0;\alpha_3^4 = 0.$$

Таким образом x_1, x_2 принимают только значения (термы) «Н» и «В», а x_3 значения 1 и 2.

Этап 6. Реализация механизма нечеткого логического вывода.

Т.к. каждая из переменных x_1, x_2, x_3 принимают значения только двух термов, то активизируются только $2^3 = 8$ правил, а именно:

1)ЕСЛИ x_1=Н И x_2=Н И x_3=1, ТО R=Н;

2)ЕСЛИ x_1=Н И x_2=Н x_3=2, ТО R=Н;

3)ЕСЛИ x_1=Н И x_2=С И x_3=1, ТО R=Н;

4)ЕСЛИ x_1=Н И x_2=С И x_3=2, ТО R=С;

5)ЕСЛИ x_1=С И x_2=Н И x_3=1, ТО R=Н;

6)ЕСЛИ x_1=С И x_2=Н И x_3=2, ТО R=С;

7)ЕСЛИ x_1=С И x_2=С И x_3=1, ТО R=С;

8)ЕСЛИ x_1=С И x_2=С И x_3=2, ТО R=С.

Согласно определению, под пересечением (связка «И») нечетких множеств, обозначим их $A_1, A_2, \dots, A_n$, понимается нечеткое множество со следующей функцией принадлежности [13]:

$$\mu_{A_1 \cap A_2 \cap \dots \cap A_n}(x) = \min\{\mu_{A_1}(x), \mu_{A_2}(x), \dots, \mu_{A_n}(x)\} \;\forall\, x \in X.$$

Вычислим посредством этого min-оператора (так называют пересечение нечетких множеств) по каждому из правил уровень его активности исходя из вычисленных значений термов для переменных:

$$\alpha_1 = \min\{\alpha_1^{\text{Н}}; \alpha_2^{\text{Н}}; \alpha_3^1\} = \{0.6, 0.67, 0.5\} = 0.5\ ;$$

$$\alpha_2 = \min\{\alpha_1^{\text{Н}}; \alpha_2^{\text{Н}}; \alpha_3^2\} = \{0.6, 0.67, 0.6\} = 0.6\ ;$$

$$\alpha_3 = \min\{\alpha_1^{\text{Н}}; \alpha_2^{\text{С}}; \alpha_3^1\} = \{0.6, 0.33, 0.5\} = 0.33\ ;$$

$$\alpha_4 = \min\{\alpha_1^{\text{Н}}; \alpha_2^{\text{С}}; \alpha_3^2\} = \{0.6, 0.33, 0.6\} = 0.33;$$

$$\alpha_5 = \min\{\alpha_1^{\text{С}}; \alpha_2^{\text{Н}}; \alpha_3^1\} = \{0.4, 0.67, 0.5\} = 0.4\ ;$$

$$\alpha_6 = \min\{\alpha_1^{\text{С}}; \alpha_2^{\text{Н}}; \alpha_3^2\} = \{0.4, 0.67, 0.6\} = 0.4;$$

$$\alpha_7 = \min\{\alpha_1^{\text{С}}; \alpha_2^{\text{С}}; \alpha_3^1\} = \{0.4, 0.33, 0.5\} = 0.4\ ;$$

$$\alpha_8 = \min\{\alpha_1^{\text{С}}; \alpha_2^{\text{С}}; \alpha_3^2\} = \{0.4, 0.33, 0.6\ \} = 0.33.$$

Затем необходимо объединить указанные правила логической связкой «ИЛИ», чтобы найти значения функций принадлежности для терм-множеств выходной переменной – риска R. Для этого введем еще один оператор объединения нечетких множеств $A_1, A_2, \dots, A_n$, также называемый max-оператором, задающий нечеткое множество с функцией принадлежности вида [44]:

$$\mu_{A_1 \cup A_2 \cup \dots \cup A_n}(x) = \max\{\mu_{A_1}(x), \mu_{A_2}(x), \dots, \mu_{A_n}(x)\}\ \forall\ x \in X.$$

Вычислим с его помощью значения функций-принадлежности для риска:

$$\beta_1 = \mu_{\text{Н}}(R) = \max\{\alpha_1, \alpha_2, \alpha_3, \alpha_5\} = 0.6;$$

$$\beta_2 = \mu_{\text{С}}(R) = \max\{\alpha_4, \alpha_6, \alpha_7, \alpha_8\} = 0.4;$$

$$\beta_3 = \mu_{\text{В}}(R) = 0.$$

Этап 7. Деффазификация входных переменных.

Для дефаззификации применяются различные методы, используем один из них, называемый методом центра тяжести, получивший наибольшее распространение на практике. В нем выходное «четкое» значение вычисляется как:

$$x = \frac{\sum_{r=1}^{n} x_r \cdot \mu_{A_i}(x_r)}{\sum_{r=1}^{n} \mu_{A_i}(x_r)}.$$

Вычислим с его помощью итоговое «четкое» значение величины риска R^* для заданных в примере значений входных переменных X_1^* , X_2^* , X_3^* [44]:

$$R^* = \frac{\beta_1 \cdot R_{\text{Н}} + \beta_2 \cdot R_{\text{С}} + \beta_3 \cdot R_{\text{В}}}{\beta_1 + \beta_2 + \beta_3} = \frac{0.6 \cdot 0.2 + 0.4 \cdot 0.6}{0.6 + 0.4} = 0.36 .$$

Таким образом риск составляет 36% относительно указанной оценки ценности активов для заданных вероятности реализации и уровня критичности уязвимости.

8. Тестирование и отладка нечеткого алгоритма оценки рисков.

При наличии реальных значений оценок ущерба можно подстроить значения параметров $a,\ b,\ c,\ d$ так, чтобы оценка риска на выходе алгоритма имела минимальное расхождение с реальными значениями ущерба. Подстройка параметров, по сути, является задачей оптимизации вида:

$$\sum_{j}^{c} \sum_{i=1}^{n} (R_i^{(j)} - \hat{R}_i^{(j)})^2 \ \rightarrow min,$$

где $R_i^{(j)}$ –вычисленный на j-том обучающем примере показатель i-того риска, а $\hat{R}_i^{(j)}$ – реальный показатель риска в обучающем примере, c – количество примеров для задачи подстройки параметров.

В заключение можно сказать, что алгоритм оценки риска на основе нечеткого вывода применительно к качественным методикам преобразует их в смешанные, когда на основе «размытых» оценок входных параметров на выходе получаем точную количественную оценку риска.

5. Разработка системы моделирования угроз и оценки рисков

Для разработки был определен соответствующий стек технологий. Программный комплекс решено было реализовать как десктопное приложение под семейство операционных систем Windows, т.к. оно является наиболее распространенным среди настольных ОС (более 77% рынка)[16], в связи с чем были выбраны Windows Presentation Foundation (WPF – графическая презентационная система построения клиентских приложений Windows, использующая язык разметки XAML) и язык программирования C#, потому что он является WPF-совместимым и обладает обширными возможностями, в том числе работы с СУБД. В качестве СУБД для хранения информации, связанной с приложением, была выбрана SQLite как наиболее легковесная и простая в использовании, не требующая разработки серверной части, но при этом обладающая практически полными возможностями SQL. В качестве среды разработки используется Microsoft Visual Studio Community 2019 с версией Microsoft .NET Framework 4.8.0.

Был создан проект приложения WPF (.NET Framework). Далее будет вестись описание реализации с точки зрения интерфейса и с точки зрения логики работы приложения – язык программирования. Так как программный код разработанной системы достаточно обширный (более 8000 строк), в приложениях к работе приведены лишь отдельные его части.

5.1. Модуль работы с базами данных угроз и уязвимостей

На первом этапе разработки был реализован модуль взаимодействия с базами данных угроз и уязвимостей, за основу были взяты данные из банка угроз ФСТЭК России. Данные хранятся в соответствующих таблицах в базе приложения. Взаимодействие с базой данных происходит при помощи специального класса коннектора SQLLiteDB_Connector, программный код которого частично представлен в приложении 3. В самом приложении угрозы и уязвимости обрабатываются как объекты. Для этого были заведены класс уязвимости IS_Vulnerability и класс угрозы IS_Threat. Пример программного

кода для класса под некоторый объект на примере IS_Threat приведен в приложении 4.

Для угрозы задаются следующие поля: идентификатор, наименование, описание, объекты воздействия, список источников угрозы, флаги типа угрозы (по конфиденциальности, целостности и доступности). Для уязвимости: идентификатор, наименование, описание, вендор ПО, название ПО, версия ПО, тип ПО, наименование ОС и типа аппаратной платформы, класс уязвимости, дата выявления, уровень опасности уязвимости, меры по устранению, статус, информация об устранении, ссылка на источники и прочая информация.

Для взаимодействия пользователя с базами были реализованы соответствующие оконные интерфейсы, представленные на рисунках 14 и 15. Пользователь может добавлять, изменять, удалять угрозы и уязвимости, а также осуществлять поиск по любому из полей.

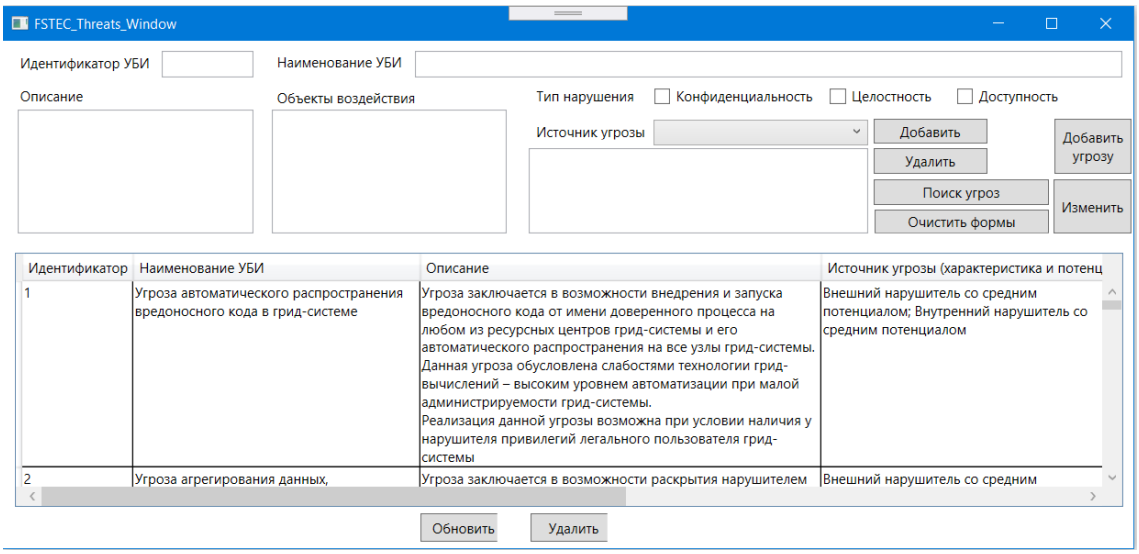


Рисунок 14 – Окно пользовательского интерфейса взаимодействия с базой данных угроз

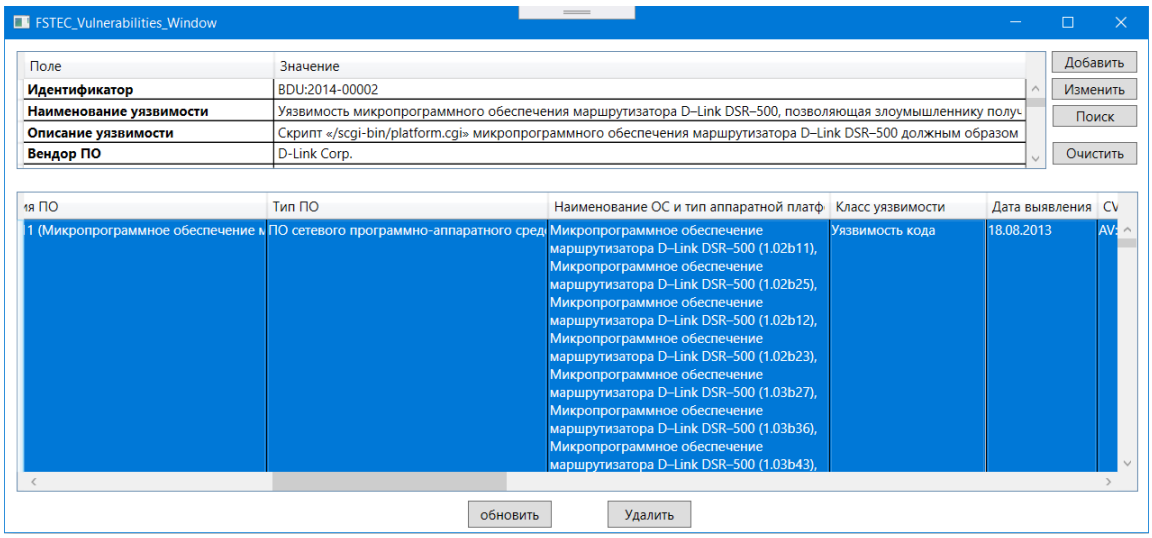


Рисунок 15 – Окно пользовательского интерфейса взаимодействия с базой данных уязвимостей

5.2. Модуль идентификации информационных активов

Следующим был реализован модуль идентификации информационных активов, описанных в главе 2, однако было решено исключить из перечня нематериальные активы ввиду трудности их идентификации и описания в общем виде и сервисы, которые можно включить в материальные активы.

Было решено реализовать иерархичную структуру, где на высшем уровне находятся сотрудники, на среднем материальные активы и на низшем – информационные ресурсы и ПО. Со смысловой точки зрения это означает, что материальные ресурсы закрепляются за сотрудниками, а информационные ресурсы и ПО за материальными активами.

С визуальной точки зрения иерархичная структура представлена деревом (класс TreeView). Узлом дерева является объект специально созданного класса Node (приложение 5), который может хранить в себе информацию об активе любого вида. По умолчанию реализованы две версии: с отображением и без отображения сотрудников (т.к. некоторым специалистам может показаться включение сотрудников в схему активов излишним). Реализованная иерархичная структура активов представлена на рисунке 16. Как видно, можно добавлять и удалять узлы (только если они не верхнего уровня – такие узлы добавляются автоматически и недоступны для удаления).

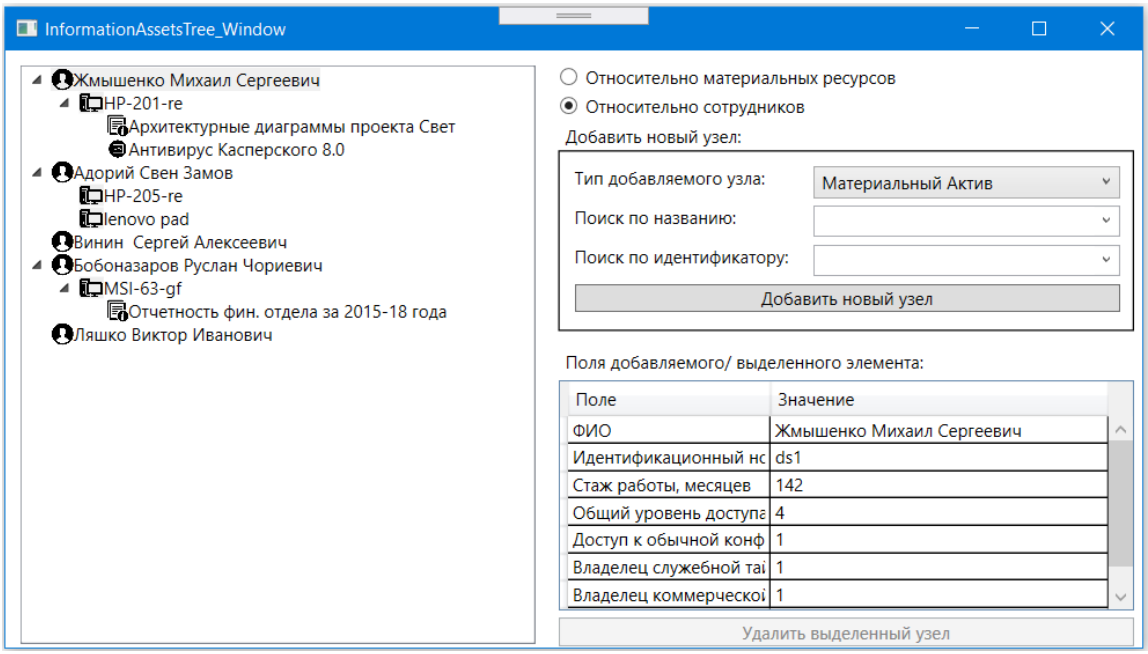


Рисунок 16 – Окно пользовательского интерфейса составления дерева информационных активов

Перейдем к рассмотрению идентификации активов по типам в порядке места в иерархии. Для каждого типа заведена отдельная таблица в базе приложения. Кроме того, сразу стоит упомянуть, что для всех активов доступны добавление, удаление, изменение, и осуществление поиска по какому-либо полю.

На самом верхнем уровне расположены сотрудники. Для обработки действий с ними создан класс Employee. Для сотрудников доступны следующие поля: ФИО, идентификатор, стаж работы (указывается в месяцах), общий уровень доступа от 0 до 5 (где 0 – низкий, а 5 - высокий), и четыре флага, отображающие статус по доступу сотрудника к просто конфиденциальной информации (например, персональные данные), к коммерческой, служебной и государственной тайнам. Окно составления реестра сотрудников представлено на рисунке 17.

Далее следуют материальные активы, для которых создан класс MaterialResource. Для материальных ресурсов заданы поля: название, идентификатор ресурса, идентификатор владельца (указывается идентификатор сотрудника из реестра сотрудников и впоследствии влияет на автоматическую сборку дерева активов), количество, цена, сложность

восстановления, категория материального ресурса (например, ноутбук, телефон и т.д.), критичность, максимальный уровень тайны данных, хранящихся на ресурсе. Окно материальных ресурсов представлено на рисунке 18.

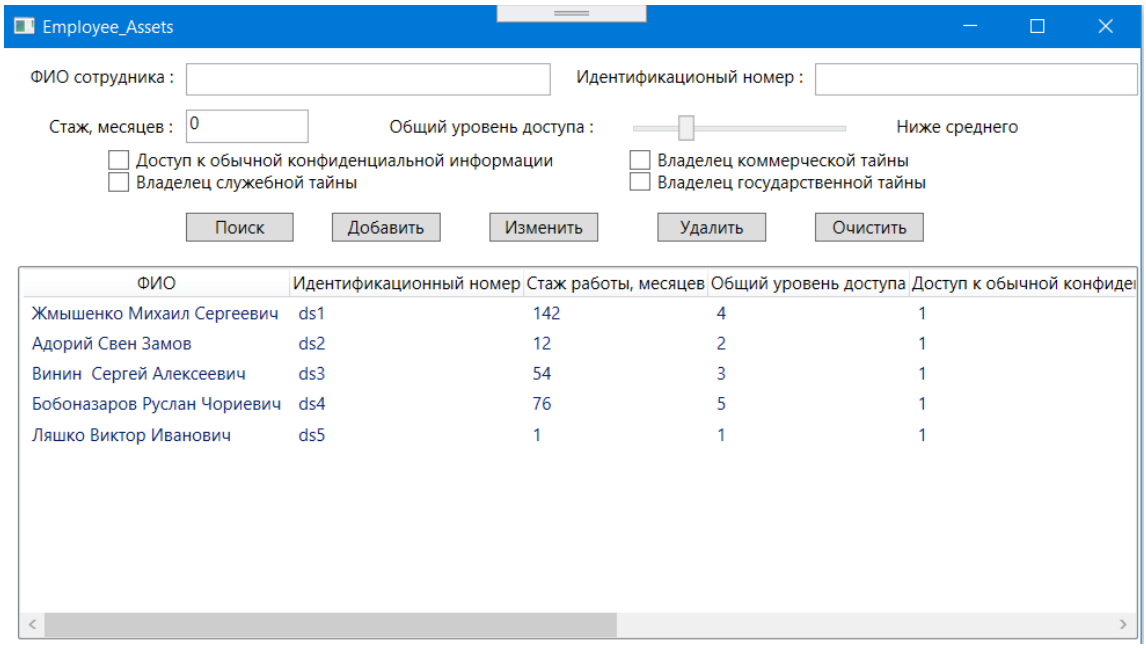


Рисунок 17 – Окно пользовательского интерфейса проведения идентификации сотрудников

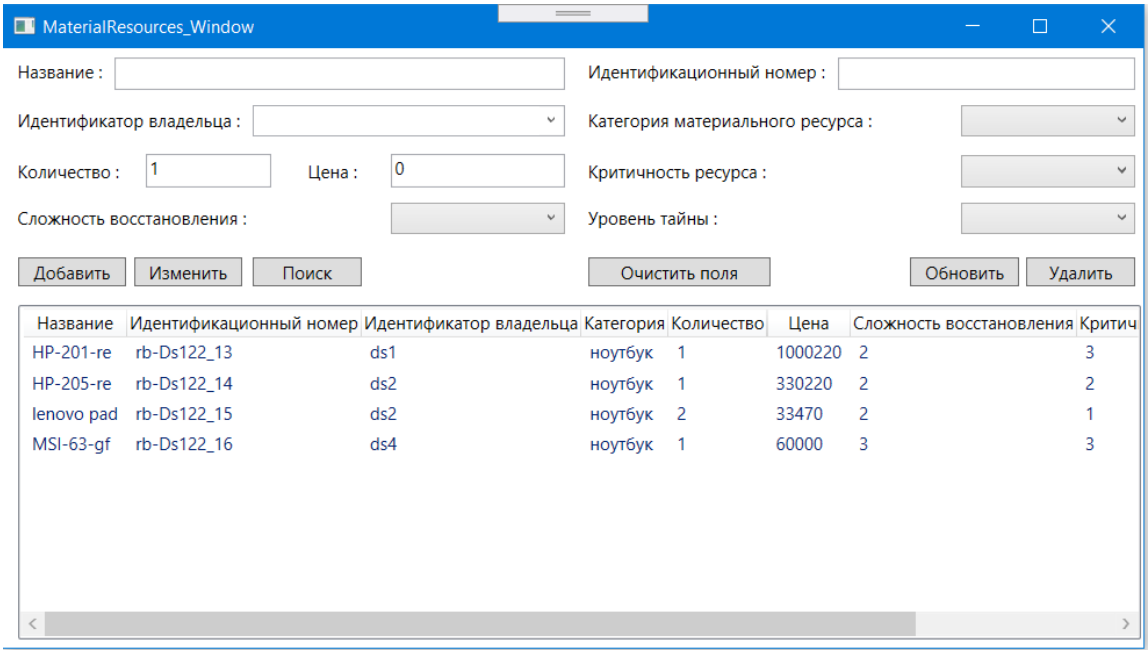


Рисунок 18 – Окно пользовательского интерфейса проведения идентификации материальных ресурсов

На нижнем уровне иерархии находятся ПО (класс Software) и информационные ресурсы (класс InformationResource). Для ПО доступны поля: название ПО и вендор ПО (возможен поиск по ПО из базы уязвимостей), версии ПО, количество копий, общая стоимость, тип ПО (например, графический редактор, антивирус и т.д.), класс стабильности (как часто подвергается изменениям), класс критичности, тип лицензии, требуемый уровень защиты, уровень надежности ПО и идентификатор, а также флаги открытого исходного кода и информации о взломе. Для информационных ресурсов задаются название, идентификатор, сложность восстановления данных, цена, ценность (если цена неопределенна), категория, критичность и уровень тайны. Оконные интерфейсы идентификации указанных типов активов представлены на рисунках 19 и 20.

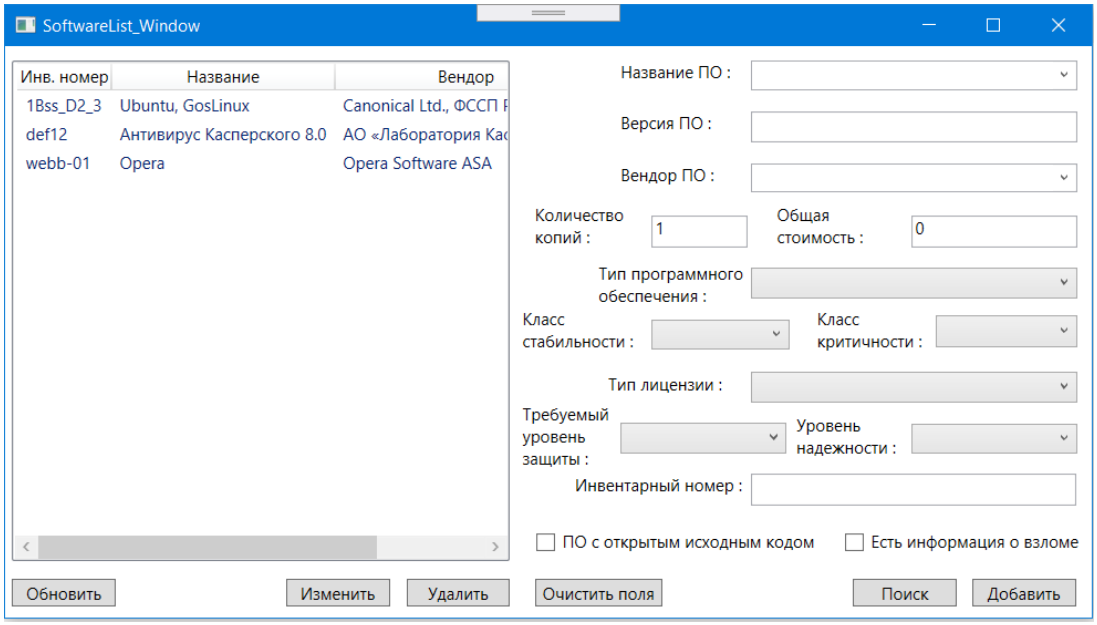


Рисунок 19 – Окно пользовательского интерфейса проведения идентификации программного обеспечения

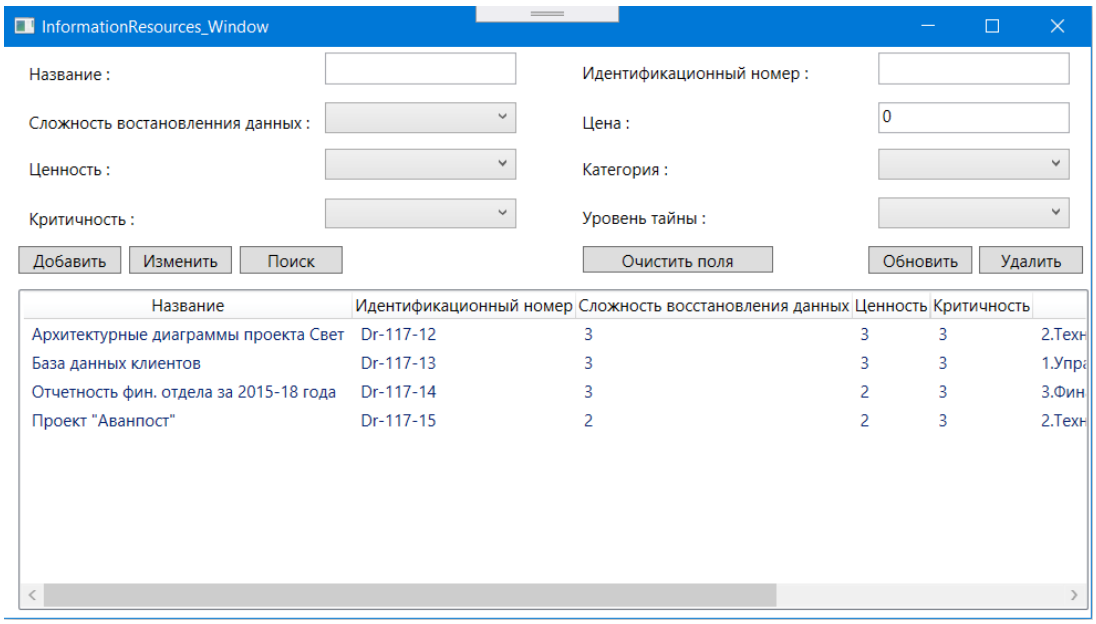


Рисунок 20 – Окно пользовательского интерфейса проведения идентификации информационных ресурсов

5.3. Модуль моделирования угроз и их оценки параметром уровня опасности угрозы

Первым из важных с точки зрения ценности всей проделанной работы является модуль моделирования угроз и их оценки параметром уровня опасности угрозы.

В оконном интерфейсе данного модуля, представленном на рисунке 21, также отображается дерево иерархии, составленное на этапе идентификации информационных активов. При выборе актива нужно указать значимость актива уже согласно методике оценки угроз безопасности информации ФСТЭК России. При выборе актива в верхней правой части окна, отведенной под моделирование угроз, указываются все угрозы для актива. Добавление угрозы происходит по нажатию кнопки добавить в новом окне, представленном на рисунке 22. Для связки «информационный актив-угроза» создана отдельная таблица в базе, а также новый класс Asset_Threat. При добавлении угрозы она находится через поля поиска по названию или идентификатору, а затем выбирается тип ущерба согласно методике, нарушители (список типов и категорий определяются автоматически по базе угроз), и возможные последствия (набор зависит от выбранного типа ущерба).

Кроме того, можно указать среднее количество инцидентов, связанных с данной угрозой, в год, если имеется таковая статистика, и далее, на основе указанного значения будет вычислена вероятность возникновения угрозы в течение года согласно способу из 5 главы.

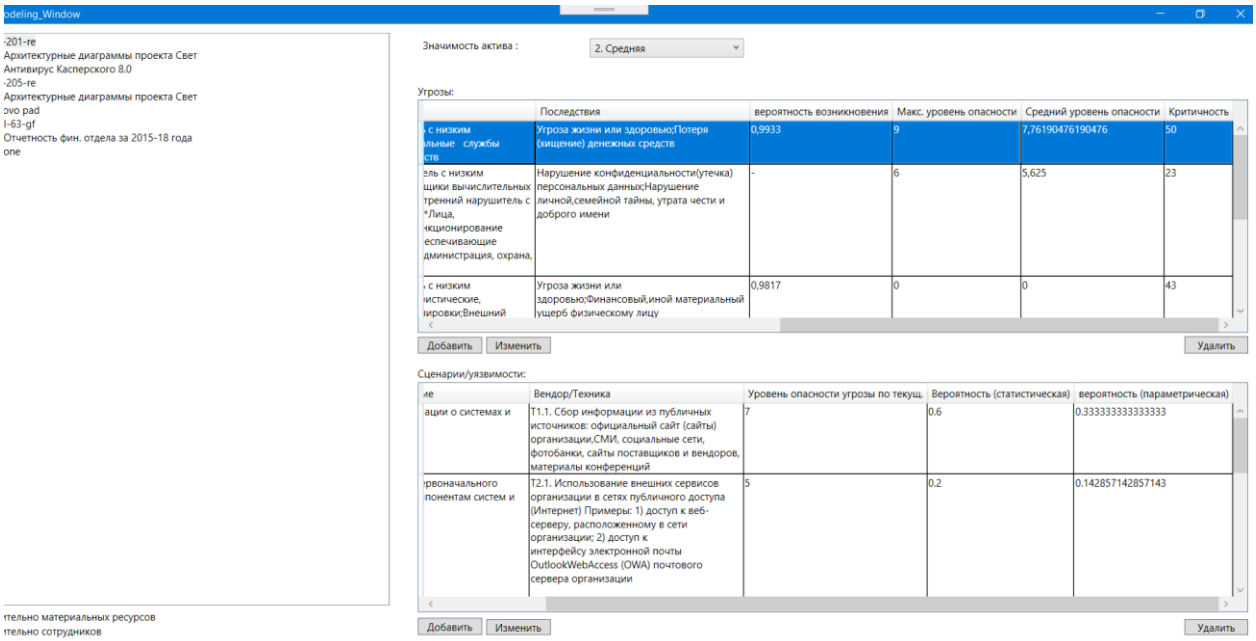


Рисунок 21 – Окно пользовательского интерфейса моделирования угроз и оценки рисков

Для каждой угрозы далее составляется список уязвимостей (в случае ПО) и/или сценариев ее реализации, под который отведена область в левой нижней части основного окна моделирования. Для связки «инф.актив-угроза-сценарий/уязвимость» добавлена отдельная таблица в базе приложения и реализован класс Asset_Threat_VulnScenario (приложение 6). При нажатии кнопки добавить открывается окно добавления уязвимости или сценария, представленное на рисунке 23 (что добавлять можно выбрать по нажатию флага, для ПО по умолчанию указывается уязвимость, для всего остального – сценарий реализации). Сначала осуществляется поиск по полям идентификатора и названия тактики и техники в случае сценария, или по идентификатору и названию уязвимости и вендору ПО в случае уязвимости. Далее указывается тип доступа, необходимый для реализации и сложность использования/осуществления, а также, сколько раз конкретная

уязвимость/сценарий участвовала в реализации угрозы при наличии такой статистики. После добавления угрозы происходит расчет вероятности её выбора на основе статистики как отношение общего числа реализаций угрозы к числу реализаций непосредственно указанным сценарием/уязвимостью. Также при добавлении происходит расчет параметра уровня опасности угрозы по сценарию/уязвимости и расчет параметрической вероятности по формуле (3). При этом для самой угрозы обновляется значение максимального уровня опасности и среднего уровня опасности по сценариям/уязвимостям, которые и служат параметрами оценки риска в представленной работе.

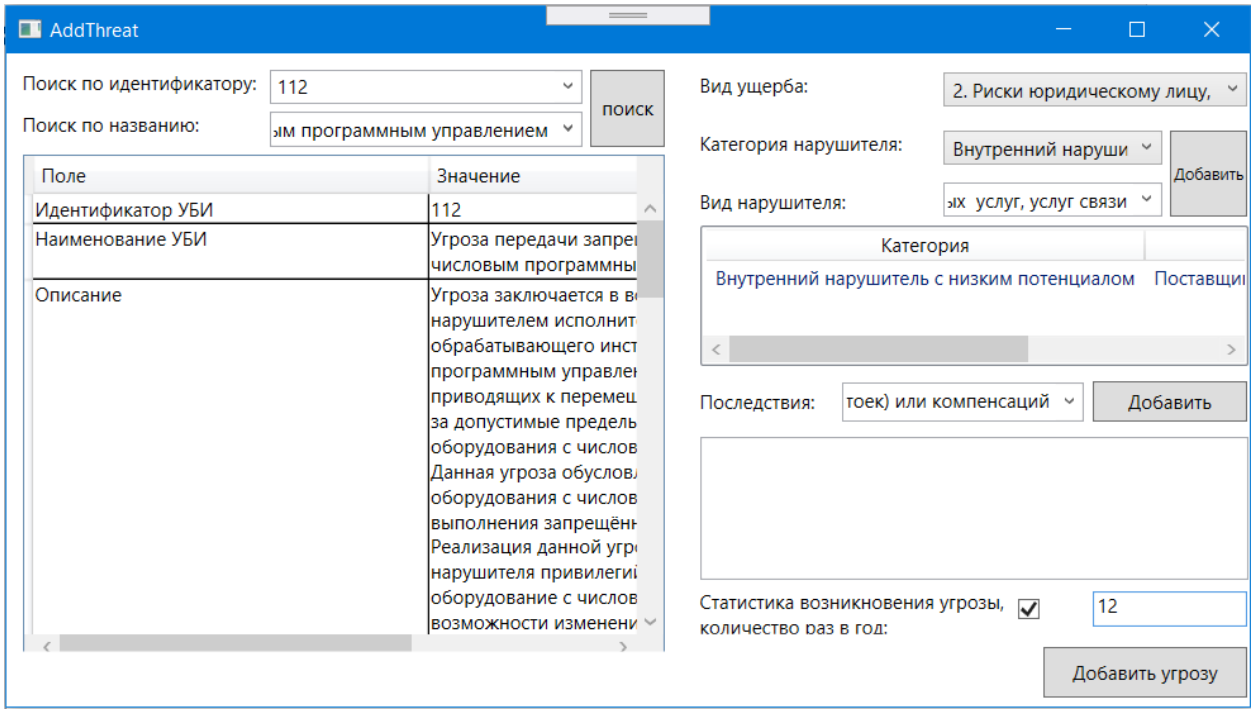


Рисунок 22 – Окно пользовательского интерфейса моделирования угрозы для некоторого актива

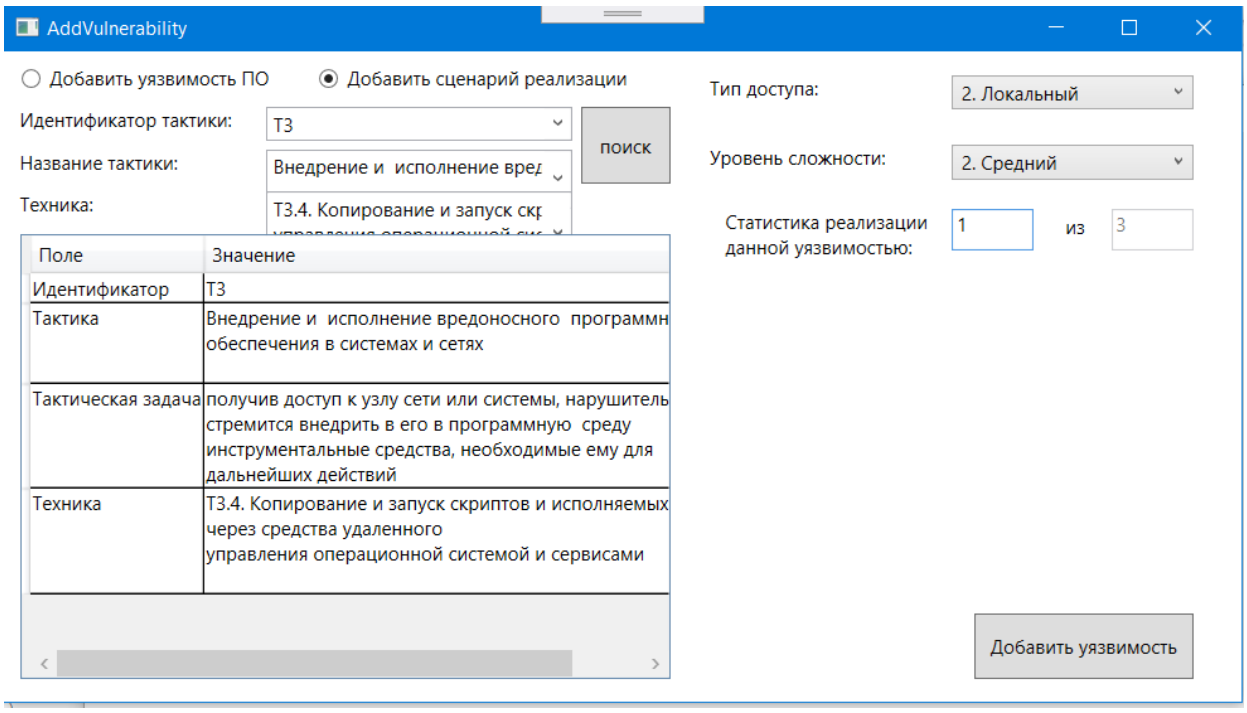


Рисунок 23 – Окно пользовательского интерфейса добавления уязвимости к смоделированной угрозе для некоторого актива

5.4. Модуль оценки рисков методикой ГРИФ 2005 и алгоритмом нечеткого логического вывода

Второй важный в рамках всей работы модуль включает в себя функции оценки рисков ИБ посредством методики ГРИФ 2005 и алгоритмом нечеткого логического вывода.

На первом шаге разработки модуля был реализован класс Risk, включающий в себя поля для обоих типов оценок, и содержащий два специальных конструктора: первый для создания экземпляра риска по связке «ресурс-угроза», т.е. риска по угрозе, и второй для экземпляра риска по ресурсу в целом. Поле GRIF_general_risk хранит оценку по методике ГРИФ 2005 в режиме без разделения по типам угроз, поле Fuzzy_Logic_Average_Risk – оценку риска посредством нечеткого вывода, рассчитанную как среднее арифметическое по всем уязвимостям и поле Fuzzy_Logic_Max_Risk – максимальную среди всех уязвимостей оценку для угрозы.

Начнем рассмотрение реализации с методики ГРИФ. Окно пользовательского интерфейса для методики представлено на рисунке 24.

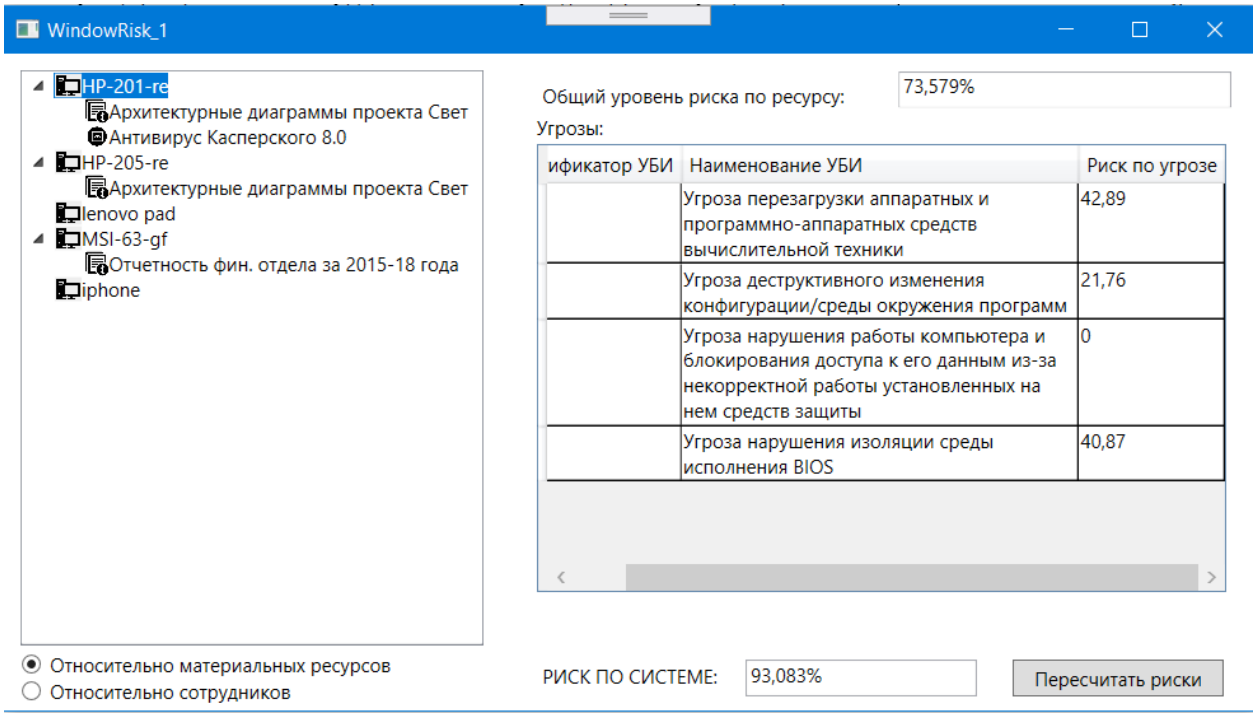


Рисунок 24 – Окно пользовательского интерфейса расчета рисков методикой ГРИФ 2005

При первом запуске окна риски не рассчитаны. Для расчета необходимо нажать кнопку «Пересчитать риски», используемая как для первичного расчета, так и для последующих обновлений оценок в случае пересмотра каких-либо параметров. В окне справа отображается дерево иерархии активов, и риски для любого актива можно просмотреть при нажатии на него. Слева от кнопки расположено поле, в котором отображается значение риска по системе в целом.

Для методики был реализован специальный класс GRIF_Risk (программный код в приложении 7), содержащий две основных функции: CalculateAll_General_AssetThreatRisks для расчета риска по каждой из угроз для каждого ресурса и CalculateAll_General_AssetRisks для расчета рисков по активу в целом (по всем угрозам для актива). Итоговый риск представляется и хранится в процентах. Функция CalculateAll_General_AssetThreatRisks может рассчитывать риски как по параметризованной вероятности, так и по статистической, в зависимости от подаваемого на вход булевого аргумента.

Перейдем к рассмотрению реализации алгоритма нечеткого вывода. Как уже описывалось в теоретической части, данный алгоритм требует задания функций-принадлежности для терм-множеств, поэтому было создано окно настройки параметров алгоритма, представленное на рисунке 25. В дополнение в данном окне была добавлена возможность выбора входной переменной: для угрозы можно выбрать уровень критичности либо статистическую вероятность угрозы, для уязвимости(сценария) – одну из двух вероятностей (параметрическую/статистическую) либо показатель $D+P$ из описанной в работе модификации методики оценки угроз безопасности информации ФСТЭК, для ценности по умолчанию всегда задается значимость актива. Более того, для каждой из переменных можно настроить количество уровней (термов в терм-множествах) и выбрать вид функции принадлежности из выпадающего списка при нажатии на соответствующую ячейку (треугольный или трапециевидный). Термы связываются с лингвистическими значениями, обозначающимися в зависимости от количества уровней, и нумеруются начиная с «1», начиная с наименее значимого. Например, для терм-множества с четырьмя термами это значения «Низкий» – 1, «Умеренный» – 2, «Средний» – 3 и «Высокий» – 4. В зависимости от выбранного количества терм-множеств генерируются продукционные правила с автоматическим заданием уровня риска по терм-множеству $\{\text{Низкий}, \text{Средний}, \text{Высокий}\}$. Для выбора уровня риска была использована формула:

$$R = \begin{cases} 1, \text{если } Th + Vu + Va \leq (n + m + k)/3; \\ 2, \text{если } 1/3(n + m + k) < Th + Vu + Va \leq 2/3(n + m + k); \\ 3, \text{если } 2/3(n + m + k) < Th + Vu + Va, \end{cases}$$

где Th, Vu, Va – номера термов переменных угрозы, уязвимости и ценности соответственно, $n,\ m,\ k$ – количество термов в терм-множествам для переменных угрозы, уязвимости и ценности соответственно, R – терм риска. Данная формула разбивает все возможные комбинации термов на три равных (в случае деления на 3) или приблизительно равных части с учетом уровня

каждой из переменных. При этом у пользователя остается возможность самостоятельно задать уровень риска в каждом из правил при нажатии на соответствующую ячейку и выборе уровня из выпадающего списка.

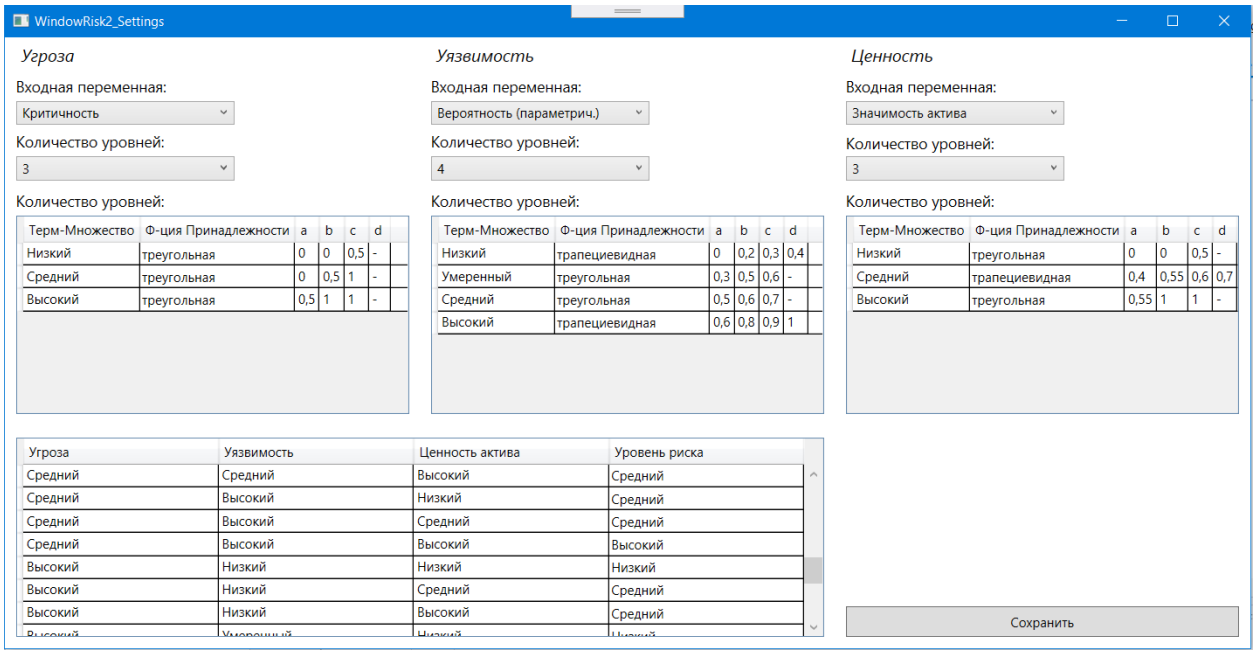


Рисунок 25 – Окно настройки параметров алгоритма нечеткого логического вывода оценки рисков

Для сохранения настроек и дальнейшего их применения были реализованы два вспомогательных класса Rule и AccessorFunction для представления в программе правил и функций принадлежности соответственно.

Нечеткий вывод производится в специальном пользовательском окне, которое можно увидеть на рисунке 26. Первичный расчет и дальнейшие пересчеты производятся по нажатию кнопки «Пересчитать риски». В окне справа находится дерево активов, и чтобы просмотреть риски для конкретного актива, нужно выбрать его нажатием мыши, тогда соответствующие ему угрозы с рассчитанными рисками отобразятся в таблице справа.

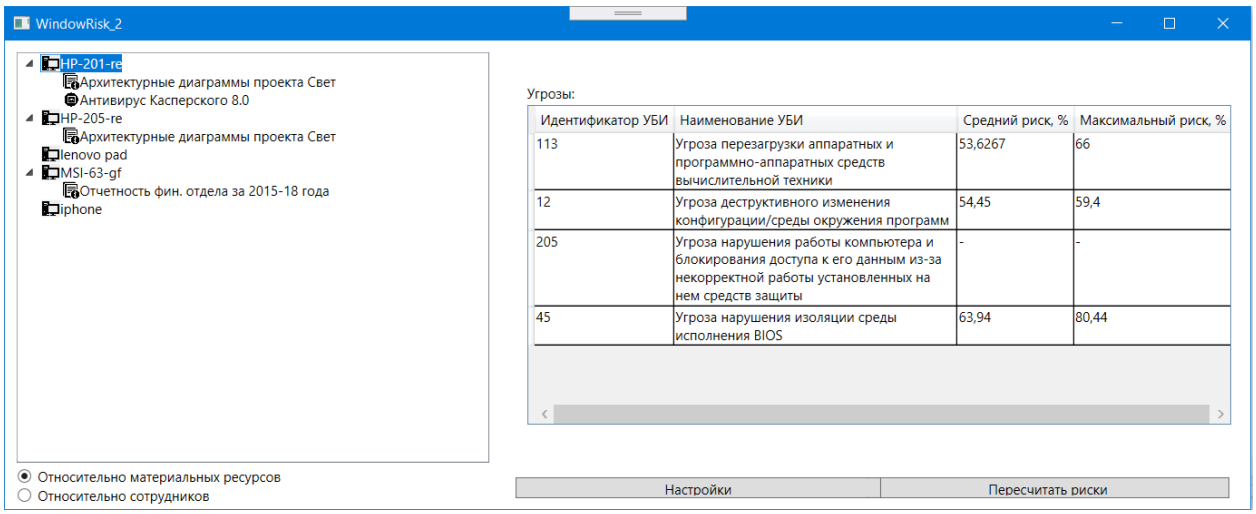


Рисунок 26 – Окно пользовательского интерфейса алгоритма нечеткого логического вывода оценки рисков

Для алгоритма нечеткого вывода был также реализован отдельный класс FuzzyLogicRisk, программный код которого представлен в приложении 8. Главной функцией класса является CalculateAll_Fuzzy_AssetThreatRisks, в которой и осуществляется вывод. Вспомогательными являются функции MinOperator и MaxOperator, соответствующие min- и max-операторам, обозначенным ранее в подразделе описания алгоритма, а также функция ExAccessorFunction, принимающая на вход число и экземпляр AccessorFunction, и возвращающая значение функции принадлежности для указанного числа.

Кратко обозначим особенности реализации вывода. Для вызова CalculateAll_Fuzzy_AssetThreatRisks необходимо создать экземпляр класса FuzzyLogicRisk, в котором будут записаны текущие настройки, извлеченные из базы данных приложения. В функции CalculateAll_Fuzzy_AssetThreatRisks в зависимости от указанных в настройках входных переменных, считываются соответствующие им значения, и если это требуется, то они нормализуются на вещественный отрезок [0;1]. К считанным значениям применяется функция ExAccessorFunction и рассчитываются все значения функций принадлежности. Далее определяется, какие из правил являются активными и для них выводится их уровень активности посредством применения min-оператора, а затем определяется уровень принадлежности каждому из трех термов риска

посредством max-оператора. И, наконец, методом центра тяжести рассчитывается «четкое» значение риска. После расчета риска по всем уязвимостям/сценариям для всех угроз для каждой в отдельности угрозы вычисляется среднее арифметическое значение риска и максимальное значение риска по заданным для нее уязвимостям/сценариям.

ЗАКЛЮЧЕНИЕ

Исследование методик и разработка соответствующей системы моделирования угроз и оценки рисков является актуальной задачей, которая значительно упрощает аудит менеджмент информационной безопасности и рисков, связанных с ней.

Теоретическая ценность проделанной работы заключается в новизне исследованных методик моделирования угроз и описании применения нечеткой логики к задаче оценки рисков информационной безопасности. В рамках данной работы рассмотрены две методики моделирования угроз 2020 и 2021 годов, разработанных государственным регулятором, предложены их объединение посредством внесения в методику оценки угроз безопасности информации ФСТЭК России параметра уровня опасности угрозы, а также способ расчета данного параметра таким образом, чтобы он соответствовал понятию риска информационной безопасности. Построение предложенных формул расчета обосновано, кроме того, проведена серия вычислительных экспериментов, подтверждающих их состоятельность. Также в работе предложен алгоритм нечеткого вывода оценки риска, и разобрано его применение на конкретном примере матричной методики [1].

В качестве итогового практического результата представлена автоматизированная информационная система, позволяющая вести базы данных угроз и уязвимостей, идентифицировать информационные активы и строить модель угроз для них, а также осуществлять расчет риска согласно предложенной модификации методики оценки угроз безопасности, посредством алгоритма нечеткого вывода и с помощью модификации алгоритма ГРИФ 2005. Данная система является полностью работоспособным программным продуктом, готовым к эксплуатации и далее будет размещена в свободном доступе в различных источниках.

Основные компетенции и навыки, которые были приобретены в ходе работы представлены в таблице 10.

Таблица 10 – Приобретенные компетенции и навыки

Шифр компетенции	Расшифровка проверяемой компетенции	Расшифровка освоения компетенции
УК-1	Способен осуществить критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий	Произведен анализ новой методики оценки угроз безопасности информации ФСТЭК России и предложено решение для устранения недостатка методики в виде отсутствия численной оценки. Тестирование разрабатываемой системы и устранение ошибок в программном коде
УК-2	Способен управлять проектом на всех этапах его жизненного цикла	Последовательная реализация разработанной системы: исследование теоретического материала, составление и анализ требований, проектирование и разработка
УК-3	Способен организовывать и руководить работой команды, вырабатывая командную стратегию для достижения поставленной цели	Изучены способы руководства командой при проведении мероприятий по обеспечению информационной безопасности
УК-4	Способен применять современные коммуникативные технологии, в том числе на иностранном(ых) языке(ах), для академического и профессионального взаимодействия	Изучение технологии WPF, языка C# и возможностей СУБД SQLite в сети интернет, консультация по вопросам на профессиональных форумах, в том числе на английском языке
УК-5	Способен анализировать и учитывать разнообразие культур в процессе межкультурного взаимодействия	Изучение источников по теме диссертации авторов, являющихся представителями различных культур
УК-6	Способен определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки	Планирование этапов исследовательской работы и разработки системы. Приоритезация задач и распределение времени между ними
ОПК-1	Способен самостоятельно приобретать, развивать и применять математические, естественнонаучные, социально-экономические и профессиональные знания для решения нестандартных задач, в том числе в новой или незнакомой среде и в междисциплинарном контексте	Изучены различные методики моделирования угроз и оценки рисков. Предложен новый способ оценки угрозы посредством агрегированного уровня опасности по всем сценариям. Исследована возможность применения математического аппарата нечеткой логики в задаче оценки рисков ИБ

Продолжение таблицы 10 – Приобретенные компетенции и навыки

ОПК-2	Способен разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач	Разработана автоматизированная информационная система моделирования угроз и оценки рисков с использованием методологии программирования ООП и паттерна программирования Model-View-ViewModel
ОПК-3	Способен анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями	Изучение и анализ методик моделирования угроз, предложение использования в качестве оценки риска параметра уровня опасности угрозы, обоснование корректности предложенных методов вычисления параметра уровня угрозы, разбор алгоритма нечеткого вывода оценки риска на примере матричной методики
ОПК-4	Способен применять на практике новые научные принципы и методы исследований	Построено практическое обоснование состоятельности предложенных формул расчета уровня опасности угрозы, исследовано применение алгоритма нечеткого вывода оценки риска на примере матричной методики
ОПК-5	Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем	Разработана автоматизированная информационная система моделирования угроз и оценки рисков посредством использования следующего инструментария: технология Windows Presentation Foundation, язык программирования C#, СУБД SQLite и среда разработки Microsoft Visual Studio 2019
ОПК-6	Способен исследовать современные проблемы и методы прикладной информатики и развития информационного общества	Исследован важный для обеспечения информационной безопасности вопрос моделирования угроз и оценки рисков
ОПК-7	Способен использовать методы научных исследований и математического моделирования в области проектирования и управления информационными системами	Построение математических формул расчета уровня угрозы. Представление потока событий угроз в виде Пуассоновского потока для расчета вероятности возникновения. Исследование применения математического аппарата нечеткой логики к задаче оценки рисков
ОПК-8	Способен осуществлять эффективное управление разработкой программных средств и проектов	Для разработки системы выбран наиболее подходящий стек технологий, и разработка системы успешно завершена

Продолжение таблицы 10 – Приобретенные компетенции и навыки

ПК-1	Управление ресурсами информационных технологий	Использованы широкие возможности связки технологии WPF и языка программирования C# для реализации проекта согласно паттерну программирования MVVM и принципам ООП
ПК-2	Управление сервисами информационных технологий	Использование в качестве среды разработки MS Visual Studio 2019 и проектирование базы данных с помощью менеджера БД SQLite Studio
ПК-3	Выполнение работ и управление работами по созданию (модификации) и сопровождению информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы	Разработана информационная система, автоматизирующая процессы идентификации информационных активов, ведения баз данных угроз и уязвимостей, моделирования угроз и оценивания рисков
ПК-4	Управление проектами в области информационных технологий малого и среднего уровня сложности в условиях неопределенностей, порождаемых запросами на изменения, с применением формальных инструментов управления рисками и проблемами проекта	Управление проектом разработки автоматизированной информационной системы моделирования угроз и оценки рисков ИБ в условиях поступления новой информации в процессе написания диссертации. Использование паттернов программирования, позволяющих легко встраивать в проект новые особенности и модули, поддержка управления версиями проекта
ПК-5	Управление проектами в области информационных технологий любого масштаба в условиях высокой неопределенности, вызываемой запросами на изменения и рисками, и с учетом влияния организационного окружения проекта; разработка новых инструментов и методов управления проектами в области информационных технологий	Управление проектом разработки автоматизированной информационной системы моделирования угроз и оценки рисков ИБ в условиях поступления новой информации в процессе написания диссертации. Использование паттернов программирования, позволяющих легко встраивать в проект новые особенности и модули, поддержка управления версиями проекта
ПК-6	Непосредственное руководство процессами разработки программного обеспечения	Изучены руководства по управлению процессами разработки программного обеспечения на всем жизненном цикле проекта
ПК-7	Организация процессов разработки программного обеспечения	Организация процесса разработки ПО системы моделирования угроз и оценки рисков ИБ

Продолжение таблицы 10 – Приобретенные компетенции и навыки

ПК-8	Управление программно-техническими, технологическими и человеческими ресурсами	Управление программно-техническими, технологическими и человеческими ресурсами в процессе разработки автоматизированной системы моделирования угроз и оценки рисков ИБ
ПК-9	Концептуальное, функциональное и логическое проектирование систем среднего и крупного масштаба и сложности	Проектирование концепции, функциональности и бизнес-логики автоматизированной системы моделирования угроз и оценки рисков ИБ

СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ Р ИСО/МЭК 27005-2010. Информационная технология. Методы и средства обеспечения информационной безопасности. Менеджмент риска информационной безопасности. [Текст] / М.: Стандартинформ, 2011. URL: https://docs.cntd.ru/document/1200084141, последняя дата обращения: 21.05.2021

2. ГОСТ Р ИСО/МЭК 13335-1-2006. Концепция и модели менеджмента безопасности информационных и телекоммуникационных технологий. [Текст] / М.: Стандартинформ, 2007. URL: https://docs.cntd.ru/document/1200048398, последняя дата обращения: 21.05.2021

3. ГОСТ Р 53114-2008. Защита информации. Обеспечение информационной безопасности в организации. Основные термины и определения. [Текст] / М.: Стандартинформ, 2018. URL: https://docs.cntd.ru/document/1200075565, последняя дата обращения: 21.05.2021

4. Астахов А. Искусство управления информационными рисками [Текст] / Астахов А. – М.: ДМК Пресс, GlobalTrust, 2010. – ISBN 5-94074-574-1; 978-5-94074-574-7

5. ГОСТ Р 12.0.010-2009. Системы управления охраной труда. Определение опасностей и оценка рисков. [Текст] / М.: Стандартинформ, 2007. URL: https://docs.cntd.ru/document/1200048398, последняя дата обращения: 21.05.2021

6. Васильев Р.А. Управление информационной безопасностью, методическое пособие [Текст] / Васильев Р.А. – Н.Новгород, 2012, 169 с.

7. Методика моделирования угроз безопасности информации. [Текст] / ФСТЭК России, 2020. URL: https://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty/149-proekty/2070-metodicheskij-dokument, последняя дата обращения: 21.05.2021

8. Методика оценки угроз безопасности информации [Текст] / ФСТЭК России, 2021. URL: https://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty/114-spetsialnye-normativnye-dokumenty/2170-metodicheskij-dokument-utverzhden-fstek-rossii-5-fevralya-2021, последняя дата обращения: 21.05.2021

9. Кингман Дж. Пуассоновские процессы. [Текст] / Кингсман Дж. — М.: МЦНМО, 2007. — 136 с.

10. Нестеров С. Курс лекций «Анализ и управление рисками в информационных системах на базе операционных систем Microsoft». Лекция 4: Методики и программные продукты для оценки рисков. [Текст] / НОУ ИНТУИТ, 2009г. URL: https://intuit.ru/studies/courses/531/387/lecture/8996, последняя дата обращения: 26.12.2020

11. Куканова Наталья, аналитик ИБ, методика ГРИФ 2005. URL: http://citforum.ru/products/dsec/grif/

12. Сагитова В.В. Модели и алгоритмы анализа информационных рисков при проведении аудита безопасности информационной системы персональных данных [91-106] / Сагитова В.В. — Уфа: Уфимский государственный авиационный технический университет, 2019

13. Зак Ю.А. Принятие решений в условиях нечетких и размытых данных: Fuzzy-технологии. [Текст] / Зак Ю.А. – М. : Книжный дом «ЛИБРОКОМ», 2013. – 352 с.

14. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. —СПб.: БХВ Петербург, 2005. —736 с.

15. Сатыбалдина Д.Ж. Оценка рисков информационной безопасности на основе нечеткой логики. [Текст] / Сатыбалдина Д.Ж., Шарипбаев А.А. — Евразийский национальный университет им. Л.Н.Гумилева, 2017

16. Статья «Статистика настольных ОС. Февраль 2020: Небольшой рост Windows 10». [Текст]. URL: https://www.comss.ru/page.php?id=7123 Последняя дата обращения: 01.05.2021

ПРИЛОЖЕНИЯ

Приложение 1

1. Федеральный закон "Об информации, информационных технологиях и о защите информации" от 27 июля 2006 N 149-ФЗ.

2. Федеральный закон "О связи" от 07.07.2003 N 126-ФЗ.

3. Федеральный закон "О безопасности" от 28.12.2010 N 390-ФЗ

4. Федеральный закон "Об электронной подписи" от 06.04.2011 N 63-ФЗ.

5. Федеральный закон "О персональных данных" от 27.07.2006 N 152-ФЗ.

6. Федеральный закон "О ратификации Конвенции Совета Европы о защите физических лиц при автоматизированной обработке персональных данных" от 19.12.2005 N 160-ФЗ.

7. Федеральный закон "О внесении изменений в отдельные законодательные акты Российской Федерации в связи с принятием федерального закона "О ратификации Конвенции Совета Европы О защите физических лиц при автоматизированной обработке персональных данных" и федерального закона "О персональных данных" от 7.05.2013 N 99-ФЗ.

8. Федеральный закон "О внесении изменений в отдельные законодательные акты Российской Федерации в части уточнения порядка обработки персональных данных в информационно-телекоммуникационных сетях" от 21.07.2014 N 242-ФЗ.

9. Закон РФ "О государственной тайне" от 21.07.1993 N 5485-1.

10. Федеральный закон "О коммерческой тайне" от 29.07.2004 N 98-ФЗ.

11. Федеральный закон "О безопасности критической информационной инфраструктуры Российской Федерации" от 26.07.2017 N 187-ФЗ.

12. Федеральный закон "О внесении изменений в отдельные законодательные акты Российской Федерации в связи с принятием

Федерального закона "О безопасности критической информационной инфраструктуры Российской Федерации" от 26.07.2017 N 193-ФЗ.

13. Федеральный закон "О внесении изменений в Уголовный кодекс Российской Федерации и статью 151 Уголовно-процессуального кодекса Российской Федерации в связи с принятием Федерального закона "О безопасности критической информационной инфраструктуры Российской Федерации" от 26.07.2017 N 194-ФЗ.

Приложение 2

Таблицы из методики оценки угроз безопасности информации ФСТЭК России.

Таблица 1 – Значимость информационных ресурсов и компонентов

Значимость информационных ресурсов и (или) компонентов	Описание
Низкая	В результате реализации угрозы будет нарушена безопасность информации, относящейся к одному пользователю, или система (сеть) сможет выполнять возложенные на нее функции (обеспечивать критический процесс) с недостаточной эффективностью или для выполнения функций (обеспечения критического процесса) потребуется привлечение дополнительных сил и средств
Средняя	В результате реализации угрозы безопасности информации будет нарушена безопасность информации, относящейся более чем к одному пользователю, или система (сеть) не сможет выполнять хотя бы одну из возложенных на нее функций или обеспечивать один критический процесс
Высокая	В результате реализации угрозы безопасности информации будет нарушена безопасность информации, относящейся ко всем пользователям, или система (сеть) не сможет выполнять возложенные на нее функции или обеспечивать критические процессы

Таблица 2 – Типы доступа при реализации сценария реализации угрозы безопасности информации

Тип доступа	Описание типов доступа
Физический	Сценарий реализации угрозы безопасности информации предусматривает физический доступ нарушителя к средствам обработки и хранения информации. Например, угроза вывода из строя или хищения машинного носителя информации
Локальный	Сценарий реализации угрозы безопасности информации предусматривает локальный доступ (наличие локальной учетной записи) нарушителя к системе или сети. Например, угроза несанкционированного изменения настроек программного обеспечения от имени системного администратора
Удаленный	Сценарий реализации угрозы безопасности информации предусматривает доступ нарушителя к системе или сети, реализуемым посредством сетевого удаленного взаимодействия. Угрозы безопасности информации могут быть удаленно реализуемый, когда реализация угроз происходит на уровне протокола одного или нескольких сетевых переходов (например, через несколько маршрутизаторов), и смежной, когда реализация угрозы происходит из общей физической или логической сети и ограничена только логически смежной топологией. Например, угроза отказа в обслуживании путем отправки специально созданного TCP-пакета

Таблица 3 – Уровни сложности при реализации сценария реализации угрозы безопасности информации

Уровень сложности	Описание уровней сложности
Умеренный	Сценарий реализации угроз безопасности информации может быть реализован нарушителем с базовым потенциалом
Повышенный	Сценарий реализации угроз безопасности информации может быть реализован нарушителем с базовым повышенным потенциалом
Средний	Сценарий реализации угроз безопасности информации может быть реализован нарушителем со средним потенциалом
Высокий	Сценарий реализации угроз безопасности информации может быть реализован нарушителем с высоким потенциалом

Таблица 4 – Соотнесение лингвистических и численных значений для параметров типа доступа, уровня сложности и значимости информационного ресурса

Показатель уровня опасности	**Значения показателей уровня опасности**	
Тип доступа (*d*)	Удаленный	3
	Локальный	2
	Физический	1
Уровень сложности (*p*)	Умеренный	4
	Повышенный	3
	Средний	2
	Высокий	1
Значимость информационных ресурсов, компонентов (*s*)	Низкая	1
	Средняя	2
	Высокая	3

Таблица 5 – Итоговые значения уровня опасности угрозы безопасности информации

Уровень опасности угрозы безопасности информации	**Диапазон значений (*w*)**
Низкий	$w \leq 4$
Средний	$5 \leq w \leq 7$
Высокий	$8 \leq w$

Приложение 3

Класс коннектора с базой данных SQLite.

```
        public class SQLLiteDB_Connector
        {
            SqliteConnection connection;

            public SqliteConnection Connection { get => 
connection; set => connection = value; }

            public SQLLiteDB_Connector(string DataSourse)
            {
                string DataSourseString = "Data Source=" + 
DataSourse;
                Connection = new 
SqliteConnection(DataSourseString);
            }


            //Функция чтения таблицы из базы данных
            public DataTable GetTable(string tableName)
            {
                DataTable table = new DataTable();

                //создание параметризированного запроса на 
извлечение данных таблицы с указанным именем
                string sqlExpression = "SELECT * FROM " + 
tableName;
                SqliteCommand select_command = new 
SqliteCommand(sqlExpression, Connection);
                using (Connection)
                {
                    Connection.Open();
                    using (SqliteDataReader reader = 
select_command.ExecuteReader())
                    {
                        table.Columns.Add(reader.GetName(0));
                        for (int i = 1; i < reader.FieldCount; 
i++)
                        {

table.Columns.Add(reader.GetName(i));
                        }

                        object[] values = new 
object[reader.FieldCount];

                        while (reader.Read()) // построчно 
считываем данные
                        {
                            DataRow row = table.NewRow();
                            reader.GetValues(values);
                            row.ItemArray = values.ToArray();
```

```
                            table.Rows.Add(row);
                        }
                    }
                    Connection.Close();
                }
                return table;
            }

            //Функция чтения конкретного столбца из базы данных,
упрощенная
            public List<string> GetColumn(string tableName,
string columnNameSearch, string value = "", string columnNameFind
= "", int limit = 100)
            {
                List<string> column_values = new List<string>();
                //создание параметризированного запроса на
извлечение данных таблицы с указанным именем
                string sqlExpression;
                if (value.Equals("")) sqlExpression = "SELECT
distinct [" + columnNameSearch + $"] FROM '{tableName}' LIMIT
'{limit.ToString()}'";
                else
                {
                    value = "%" + value + "%";
                    sqlExpression = "SELECT distinct [" +
columnNameFind + $"] FROM '{tableName}' WHERE [" +
columnNameSearch + $"] like @value LIMIT '{limit.ToString()}'";
                }
                SqliteCommand select_command = new
SqliteCommand(sqlExpression, Connection);
                SqliteParameter newParam = new
SqliteParameter("@value", value);
                select_command.Parameters.Add(newParam);
                using (Connection)
                {
                    Connection.Open();
                    using (SqliteDataReader reader =
select_command.ExecuteReader())
                    {

                        object[] row_values = new
object[reader.FieldCount];

                        while (reader.Read()) // построчно
считываем данные
                        {

//column_values.Add((string)reader.GetValue(0));
                            reader.GetValues(row_values);

column_values.Add(row_values[0].ToString());
                        }
                    }
```

```
                Connection.Close();
            }
            return column_values;
        }


        //Функция чтения строк по условию из базы данных
        public DataTable GetRows(string tableName, string[]
parameters, string[] values, List<string> columns = null)
        {
            if (columns == null) columns =
GetColumnNames(tableName);

            StringBuilder sqlExpressionBuilder = new
StringBuilder("SELECT * FROM " + tableName + " WHERE ");
            bool validation_flag = false;
            for (int i = 0; i < parameters.Length; i++)
            {
                if (!values[i].Equals(""))
                {
                    sqlExpressionBuilder.Append("[" +
columns[i] + "] " + "= " + parameters[i] + " and ");
                    validation_flag = true;
                }

            }
            //удаляем последний and
            if (validation_flag)
sqlExpressionBuilder.Remove(sqlExpressionBuilder.Length - 4, 3);
            else
sqlExpressionBuilder.Remove(sqlExpressionBuilder.Length - 6, 5);
            string sqlExpression =
sqlExpressionBuilder.ToString();

            DataTable table = new DataTable();
            using (Connection)
            {
                Connection.Open();

                SqliteCommand command = new
SqliteCommand(sqlExpression, Connection);
                // создаем параметры и добавляем их к команде
                for (int i = 0; i < parameters.Length; i++)
                {
                    // добавляем параметр к команде
                    SqliteParameter newParam = new
SqliteParameter(parameters[i], values[i]);
                    command.Parameters.Add(newParam);
                }
                using (SqliteDataReader reader =
command.ExecuteReader())
                {
                    table.Columns.Add(reader.GetName(0));
```

```
                        for (int i = 1; i < reader.FieldCount;
i++)
                        {

table.Columns.Add(reader.GetName(i));
                        }

                        object[]      row_values      =      new
object[reader.FieldCount];

                        while  (reader.Read())     //  построчно
считываем данные
                        {
                            DataRow row = table.NewRow();
                            reader.GetValues(row_values);
                            row.ItemArray                       =
row_values.ToArray();
                            table.Rows.Add(row);
                        }
                    }
                    Connection.Close();
                    return table;
                }
            }


            //функция получения названий столбцов
            public List<string> GetColumnNames(string tableName)
            {
                List<string> columns = new List<string>();
                string      sqlExpression      =      $"pragma
table_info('{tableName}');";
                using (Connection)
                {
                    Connection.Open();

                    SqliteCommand      command      =      new
SqliteCommand(sqlExpression, Connection);
                    using     (SqliteDataReader     reader     =
command.ExecuteReader())
                    {
                        object[]      row_values      =      new
object[reader.FieldCount];

                        while  (reader.Read())     //  построчно
считываем данные
                        {
                            reader.GetValues(row_values);

columns.Add(row_values[1].ToString());
                        }
                    }
                    Connection.Close();
```

```
                }
                return columns;
            }

    //функция вставки
            public bool InsertRowQuery(string tableName, string[]
parameters, string[] values)
            {
                if (parameters.Length != values.Length) return
false;
                //создание параметризированного запроса на
вставку данных в таблицу с указанным именем
                string sqlExpression = "INSERT INTO " + tableName
+ " VALUES(" + string.Join(",", parameters) + ")";
                using (Connection)
                {
                    Connection.Open();

                    SqliteCommand command = new
SqliteCommand(sqlExpression, Connection);
                    // создаем параметры и добавляем их к команде
                    for (int i = 0; i < parameters.Length; i++)
                    {
                        // добавляем параметр к команде
                        SqliteParameter newParam = new
SqliteParameter(parameters[i], values[i]);
                        command.Parameters.Add(newParam);
                    }
                    try
                    {
                        if (command.ExecuteNonQuery() == 1)
                        {
                            Connection.Close(); return true;
                        }
                        else { Connection.Close(); return false;
}
                    }
                    catch (SqliteException e)
                    {
                        Connection.Close();
                        return false;
                    }
                }
            }



            //функция поиска
            public DataTable FindRowQuery(string tableName,
string[] parameters, string[] values)
            {
                List<string> columns =
GetColumnNames(tableName);
```

```
                StringBuilder sqlExpressionBuilder = new
StringBuilder("SELECT * FROM " + tableName + " WHERE ");
                bool validation_flag = false;
                for (int i = 0; i < parameters.Length; i++)
                {
                    if (!values[i].Equals(""))
                    {
                        sqlExpressionBuilder.Append("["       +
columns[i] + "] " + "like " + parameters[i] + " and ");
                        validation_flag = true;
                    }

                }
                //удаляем последний and
                if                              (validation_flag)
sqlExpressionBuilder.Remove(sqlExpressionBuilder.Length - 4, 3);
                else
sqlExpressionBuilder.Remove(sqlExpressionBuilder.Length - 6, 5);
                string              sqlExpression              =
sqlExpressionBuilder.ToString();

                DataTable table = new DataTable();
                using (Connection)
                {
                    Connection.Open();

                    SqliteCommand      command      =      new
SqliteCommand(sqlExpression, Connection);
                    // создаем параметры и добавляем их к команде
                    for (int i = 0; i < parameters.Length; i++)
                    {
                        // добавляем параметр к команде
                        SqliteParameter    newParam    =    new
SqliteParameter(parameters[i], values[i]);
                        command.Parameters.Add(newParam);
                    }
                    using     (SqliteDataReader     reader     =
command.ExecuteReader())
                    {
                        table.Columns.Add(reader.GetName(0));
                        for (int i = 1; i < reader.FieldCount;
i++)
                        {

table.Columns.Add(reader.GetName(i));
                        }

                        object[]      row_values      =      new
object[reader.FieldCount];

                        while  (reader.Read())     //  построчно
считываем данные
```

```
                        {
                            DataRow row = table.NewRow();
                            reader.GetValues(row_values);
                            row.ItemArray                        =
row_values.ToArray();
                            table.Rows.Add(row);
                        }
                    }
                    Connection.Close();
                    return table;
                }
            }
     //функция поиска
             public string FindRowQuery(string tableName, string[]
parameters, string[] values, int g)
             {
                 List<string>              columns              =
GetColumnNames(tableName);

                 StringBuilder   sqlExpressionBuilder   =   new
StringBuilder("SELECT * FROM " + tableName + " WHERE ");
                 bool validation_flag = false;
                 for (int i = 0; i < parameters.Length; i++)
                 {
                     if (!values[i].Equals(""))
                     {
                         sqlExpressionBuilder.Append("["        +
columns[i] + "] " + "like " + parameters[i] + " and ");
                         validation_flag = true;
                     }

                 }
                 //удаляем последний and
                 if                                (validation_flag)
sqlExpressionBuilder.Remove(sqlExpressionBuilder.Length - 4, 3);
                 else
sqlExpressionBuilder.Remove(sqlExpressionBuilder.Length - 6, 5);
                 string              sqlExpression              =
sqlExpressionBuilder.ToString();

                 DataTable table = new DataTable();
                 using (Connection)
                 {
                     Connection.Open();

                     SqliteCommand      command      =      new
SqliteCommand(sqlExpression, Connection);
                     // создаем параметры и добавляем их к команде
                     for (int i = 0; i < parameters.Length; i++)
                     {
                         // добавляем параметр к команде
                         SqliteParameter    newParam    =    new
SqliteParameter(parameters[i], values[i]);
```

```
                        command.Parameters.Add(newParam);
                    }
                    using     (SqliteDataReader     reader     =
command.ExecuteReader())
                    {
                        table.Columns.Add(reader.GetName(0));
                        for (int i = 1; i < reader.FieldCount;
i++)
                        {

table.Columns.Add(reader.GetName(i));
                        }

                        object[]      row_values      =      new
object[reader.FieldCount];

                        while (reader.Read())
     // построчно считываем данные
                        {
                            DataRow row = table.NewRow();
                            reader.GetValues(row_values);
                            row.ItemArray                      =
row_values.ToArray();
                            table.Rows.Add(row);
                        }
                    }
                    Connection.Close();
                    return sqlExpression;
                }
            }


            //функция апдейта, выбираемые столбцы
            public bool UpdateQuery(string tableName, string[]
search_colums,      string[]      search_parameters,      string[]
search_values,
                string[]  upd_colums,  string[]  new_parameters,
string[] new_values)
            {

                StringBuilder   sqlExpressionBuilder   =   new
StringBuilder("UPDATE " + tableName + " SET ");
                bool validation_flag = false;
                for (int i = 0; i < new_parameters.Length; i++)
                {
                    sqlExpressionBuilder.Append("["            +
upd_colums[i] + "] " + " = " + new_parameters[i] + " , ");
                    validation_flag = true;
                }
                //удаляем последний and
                if                              (validation_flag)
sqlExpressionBuilder.Remove(sqlExpressionBuilder.Length - 2, 1);
```

```
                validation_flag             =             false;
sqlExpressionBuilder.Append(" WHERE ");
                for (int i = 0; i < search_parameters.Length;
i++)
                {
                    sqlExpressionBuilder.Append("["            +
search_colums[i] + "] " + " = " + search_parameters[i] + " and ");
                    validation_flag = true;
                }
                //удаляем последний and
                if                                (validation_flag)
sqlExpressionBuilder.Remove(sqlExpressionBuilder.Length - 4, 3);


                string              sqlExpression               =
sqlExpressionBuilder.ToString();

                //DataTable table = new DataTable();
                using (Connection)
                {
                    Connection.Open();

                    SqliteCommand      command      =      new
SqliteCommand(sqlExpression, Connection);
                    // создаем параметры и добавляем их к команде
                    for (int i = 0; i < search_parameters.Length;
i++)
                    {
                        // добавляем параметр к команде
                        SqliteParameter    newParam    =    new
SqliteParameter(search_parameters[i], search_values[i]);
                        command.Parameters.Add(newParam);
                    }
                    for (int i = 0; i < new_parameters.Length;
i++)
                    {
                        // добавляем параметр к команде
                        SqliteParameter    newParam    =    new
SqliteParameter(new_parameters[i], new_values[i]);
                        command.Parameters.Add(newParam);
                    }
                    for (int i = 0; i < command.Parameters.Count;
i++)
                        sqlExpression       +=       "{"       +
command.Parameters[i].Value.ToString() + "}";
                    try
                    {
                        if (command.ExecuteNonQuery() == 1)
                        {
                            Connection.Close(); return true;
                        }
                        else { Connection.Close(); return true; }
                    }
```

```
                    catch (SqliteException e)
                    {
                        Connection.Close();
                        return false;
                    }
                }
            }

            //функция удаления строк
            public bool DeleteQuery(string tableName, string[]
parameters, string[] values, List<string> columns = null)
            {
                if (columns == null)
                    columns = GetColumnNames(tableName);

                StringBuilder   sqlExpressionBuilder   =   new
StringBuilder("DELETE FROM " + tableName + " WHERE ");
                bool validation_flag = false;
                for (int i = 0; i < parameters.Length; i++)
                {
                    if (!values[i].Equals(""))
                    {
                        sqlExpressionBuilder.Append("["       +
columns[i] + "] " + " = " + parameters[i] + " and ");
                        validation_flag = true;
                    }

                }
                //удаляем последний and
                if                              (validation_flag)
sqlExpressionBuilder.Remove(sqlExpressionBuilder.Length - 4, 3);
                string              sqlExpression              =
sqlExpressionBuilder.ToString();
                using (Connection)
                {
                    Connection.Open();

                    SqliteCommand      command      =      new
SqliteCommand(sqlExpression, Connection);
                    // создаем параметры и добавляем их к команде
                    for (int i = 0; i < parameters.Length; i++)
                    {
                        // добавляем параметр к команде
                        SqliteParameter    newParam    =    new
SqliteParameter(parameters[i], values[i]);
                        command.Parameters.Add(newParam);
                    }
                    try
                    {
                        if (command.ExecuteNonQuery() == 1)
                        {
                            Connection.Close(); return true;
                        }
```

```
                        else { Connection.Close(); return true;
}
                    }
                    catch (SqliteException e)
                    {
                        Connection.Close();
                        return false;
                    }
                }
            }

            //удалить все строки
            public bool ClearTableQuery(string tableName)
            {
                using (Connection)
                {
                    Connection.Open();

                    SqliteCommand      command      =      new
SqliteCommand("DELETE FROM " + tableName, Connection);
                    // создаем параметры и добавляем их к команде
                    try
                    {
                        if (command.ExecuteNonQuery() == 1)
                        {
                            Connection.Close(); return true;
                        }
                        else { Connection.Close(); return true;
}
                    }
                    catch (SqliteException e)
                    {
                        Connection.Close();
                        return false;
                    }
                }
            }
        }
```

Приложение 4

Программный код класс угрозы IS_Threat

```
    //класс для угрозы безопасности информации
    class IS_Threat
    {
        int id;                       //идентификатор
        string name;                  //наименование БИ
        string description;           //описание угрозы БИ
        string source;                //источник
        string object_of_influence;  //объект воздействия

        //флаги угрозы нарушения конфиденциальности, целостности
и доступности
        int confidentiality;
        int integrity;
        int availability;

        public IS_Threat(int id, string name, string
description, string source,
            string object_of_influence, int confidentiality, int
integrity, int availability)
        {
            Id = id;
            Name = name;
            Description = description;
            Source = source;
            Object_of_influence = object_of_influence;
            Confidentiality = confidentiality;
            Integrity = integrity;
            Availability = availability;
        }

        public int Id { get => id; set => id = value; }
        public string Name { get => name; set => name = value; }
        public string Description { get => description; set =>
description = value; }
        public string Source { get => source; set => source =
value; }
        public string Object_of_influence { get =>
object_of_influence; set => object_of_influence = value; }
        public int Confidentiality { get => confidentiality; set
=> confidentiality = value; }
        public int Integrity { get => integrity; set =>
integrity = value; }
        public int Availability { get => availability; set =>
availability = value; }


        public static IS_Threat ParseFromDataRow(DataRow row)
        {
            // получаем все ячейки строки
            var cells = row.ItemArray;
```

```
            return new IS_Threat(int.Parse(cells[0].ToString()),
cells[1].ToString(), cells[2].ToString(), cells[3].ToString(),
cells[4].ToString(),
                int.Parse(cells[5].ToString()),
int.Parse(cells[6].ToString()), int.Parse(cells[7].ToString()));
        }
        public static ObservableCollection<IS_Threat>
ParseFromDataTable(DataTable dt)
        {
            ObservableCollection<IS_Threat> threats = new
ObservableCollection<IS_Threat>();
            foreach (DataRow row in dt.Rows)
            {
                // получаем все ячейки строки
                threats.Add(ParseFromDataRow(row));
            }
            return threats;
        }
        //возврат параметров для запроса на вставку в БД
        public List<string[]> ParamsForInsertSQL_query()
        {
            string[] parameters = { "@Id", "@Name",
"@Description", "@Source", "@Object_of_influence",
"@Confidentiality", "@Integrity", "@Availability",
            "@Date1", "@Date2"};
            string[] values = { Id.ToString(), Name,
Description, Source , Object_of_influence,
                Confidentiality.ToString(),
Integrity.ToString(), Availability.ToString(),
"20.03.2015","20.03.2015" };
            return new List<string[]>{parameters,values};
        }

        //возврат параметров для запроса на поиск в БД
        public static List<string[]>
ParamsForFindSQL_query(string[] vals)
        {
            string[] parameters = { "@Id", "@Name",
"@Description", "@Source", "@Object_of_influence",
"@Confidentiality", "@Integrity", "@Availability"};
            string[] values = { "%"+vals[0]+"%","%"+
vals[1]+"%","%"+ vals[2]+"%", "%"+vals[3]+"%" ,"%"+ vals[4]+"%",
                "%"+vals[5]+"%", "%"+vals[6]+"%",
"%"+vals[7]+"%"};
            return new List<string[]> { parameters, values };
        }


        //возврат параметров для запроса на изменение в БД
        public  List<string[]> ParamsForUpdateSQL_query()
        {
```

```
            string[] parameters = { "@Id_U", "@Name_U",
"@Description_U", "@Source_U", "@Object_of_influence_U",
"@Confidentiality_U", "@Integrity_U",
                "@Availability_U", "@Date1_U", "@Date2_U"};
            string[] values = { Id.ToString(), Name,
Description, Source , Object_of_influence,
                Confidentiality.ToString(),
Integrity.ToString(), Availability.ToString(),
"20.03.2015","20.03.2015" };
            return new List<string[]> { parameters, values };
        }
    }
```

Приложение 5

Класс Node, реализующий узел дерева иерархии

```
    public class TreeNode : INotifyPropertyChanged
    {
        MaterialResource materialResource=null;
        InformationResource informationResource = null;
        Software software = null;
        Employee employee = null;
        string name;
        string id;
        TreeNode parentNode;
        ObservableCollection <TreeNode> nodes= new
ObservableCollection<TreeNode>();

        string imageSource;

        public ObservableCollection<TreeNode> Nodes { get =>
nodes; set => nodes = value; }
        public MaterialResource MaterialResource { get =>
materialResource; set => materialResource = value; }
        public InformationResource InformationResource { get =>
informationResource; set => informationResource = value; }
        public Software Software { get => software; set =>
software = value; }
        public Employee Employee { get => employee; set =>
employee = value; }
        public string Name { get => name; set => name = value; }
        public string Id { get => id; set => id = value; }
        public TreeNode ParentNode { get => parentNode; set =>
parentNode = value; }
        public string ImageSource { get => imageSource; set =>
imageSource = value; }

        public TreeNode(MaterialResource materialResource,
ObservableCollection<TreeNode> childNodes= null, TreeNode
parentNode= null)
        {
            MaterialResource = materialResource;
            Name = materialResource.Name;
            Id = materialResource.Id;
            Nodes = childNodes;
            ParentNode = parentNode;
            ImageSource = @"\Images\material_asset.png";
        }
        public TreeNode(InformationResource informationResource,
TreeNode parentNode = null)
        {
            InformationResource = informationResource;
            Name = informationResource.Name;
            Id = informationResource.Id;
            ParentNode = parentNode;
            ImageSource = @"\Images\information_resource.png";
```

```
        }
        public TreeNode(Software software, TreeNode parentNode =
null)
        {
            Software = software;
            Name = software.Name;
            Id = software.Id;
            ParentNode = parentNode;
            ImageSource = @"\Images\software.png";
        }
        public TreeNode(Employee employee,
ObservableCollection<TreeNode> childNodes= null, TreeNode
parentNode = null)
        {
            Employee = employee;
            Name = employee.FIO;
            Id = employee.IdNum;
            Nodes = childNodes;
            ParentNode = parentNode;
            ImageSource = @"\Images\employee.png";
        }

        public void  AddChildNode(InformationResource
informationResource)
        {
            Nodes.Add(new TreeNode(informationResource, this));
        }
        public void AddChildNode(Software software)
        {
            Nodes.Add(new TreeNode(software, this));
        }
        public void AddChildNode(MaterialResource
materialResource)
        {
            Nodes.Add(new TreeNode(materialResource, null,
this));
        }

        public string NodeType()
        {
            if (Employee != null) return "Employee";
            if (MaterialResource != null) return
"MaterialResource";
            if (InformationResource  != null) return
"InformationResource";
            if (Software != null) return "Software";
            return "";
        }

        public bool
FindNodeInCollection(ObservableCollection<TreeNode> tree)
        {
            bool flag = false ;
```

```
            foreach (TreeNode treeNode in tree)
            {
                if (flag) break;
                if (treeNode.Id == this.Id &&
treeNode.Name==this.Name) flag = true;
                if (treeNode.Nodes.Count > 0 && !flag)
flag=this.FindNodeInCollection(treeNode.Nodes);
            }
            return flag;
        }

        public string[] Get_ID_Name()
        {
            if (Employee != null) return new
string[]{Employee.IdNum, Employee.FIO};
            else if (MaterialResource != null) return new
string[] { MaterialResource.Id, MaterialResource.Name };
            else if (InformationResource != null) return new
string[] { InformationResource.Id, InformationResource.Name };
            else return new string[] { Software.Id,
Software.Name };
        }

        public event PropertyChangedEventHandler
PropertyChanged;

        public void OnPropertyChanged([CallerMemberName]string
prop = "")
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new
PropertyChangedEventArgs(prop));
        }
         }
```

Приложение 6

Класс связи между активом, угрозой и уязвимостью

```
    public class Asset_Threat_VulnScenario : INotifyPropertyChanged
    {
        string asset_Id;
        string asset_Name;
        string threat_Id;
        string threat_Name;

        //номер тактики/уязвимости; имя; вендор ПО/техника -- сами значения ComboBox
        string id_tactic_numValue;
        string nameValue;
        string vendor_techniqueValue;
        //сценарий или техника
        string vuln_or_Scenario;

        //параметры необходимого доступа и сложности
        int d_parameter, p_parameter;

        //статистическая вероятность
        double statistic_Probability=0;
        //расчитанная параметрами вероятность
        double parameterized_Probability=0;
        int danger_lvl;
        int significance;

        public string Id_tactic_numValue { get => id_tactic_numValue; set { id_tactic_numValue = value; OnPropertyChanged("Id_tactic_numValue"); } }
        public string NameValue { get => nameValue; set { nameValue = value; OnPropertyChanged("NameValue"); } }
        public string Vendor_techniqueValue { get => vendor_techniqueValue; set { vendor_techniqueValue = value; OnPropertyChanged("Vendor_techniqueValue"); } }
        public string Vuln_or_Scenario { get => vuln_or_Scenario; set { vuln_or_Scenario = value; OnPropertyChanged("Vuln_or_Scenario"); } }
        public int D_parameter { get => d_parameter; set { d_parameter = value; OnPropertyChanged("D_parameter"); } }
        public int P_parameter { get => p_parameter; set { p_parameter = value; OnPropertyChanged("P_parameter"); } }
        public double Statistic_Probability { get => statistic_Probability; set { statistic_Probability = value;
                OnPropertyChanged("Statistic_Probability"); } }
        public double Parameterized_Probability { get => parameterized_Probability; set { parameterized_Probability = value;
                OnPropertyChanged("Parameterized_Probability"); }
}
```

```
        public string Asset_Id { get => asset_Id; set { asset_Id
= value; OnPropertyChanged("Asset_Id"); } }
        public string Asset_Name { get => asset_Name; set {
asset_Name = value; OnPropertyChanged("Asset_Name"); } }
        public string Threat_Id { get => threat_Id; set { threat_Id
= value; OnPropertyChanged("Threat_Id"); } }
        public string Threat_Name { get => threat_Name; set {
threat_Name = value; OnPropertyChanged("Threat_Name"); } }

        public int Danger_lvl { get => danger_lvl; set =>
danger_lvl = value; }
        public int Significance { get => significance; set =>
significance = value; }

        public Asset_Threat_VulnScenario(string asset_Id, string
asset_Name, string threat_Id, string threat_Name,
            string id_tactic_numValue, string nameValue, string
vendor_techniqueValue, string vuln_or_Scenario, int p_parameter,
int d_parameter,
            double          statistic_Probability,          double
parameterized_Probability, int danger_Lvl, int significance)
        {
            Asset_Id = asset_Id;
            Asset_Name = asset_Name;
            Threat_Id = threat_Id;
            Threat_Name = threat_Name;
            Id_tactic_numValue = id_tactic_numValue;
            NameValue = nameValue;
            Vendor_techniqueValue = vendor_techniqueValue;
            Vuln_or_Scenario = vuln_or_Scenario;
            D_parameter = d_parameter;
            P_parameter = p_parameter;
            Statistic_Probability = statistic_Probability;
            Parameterized_Probability                              =
parameterized_Probability;
            Danger_lvl = danger_Lvl;
            Significance = significance;
        }

        //Параметрический рассчет вероятности использования
уязвимости
        public static double[] ComputeParametrizedProbability(
Asset_Threat asset_Threat, int p = 1)
        {
            SQLLiteDB_Connector                    DBconnector=new
SQLLiteDB_Connector("FSTEC_Database.db");
            int Qfunction(int x)
            {
                if (x > 0) return x;
                else return 0;
            }
```

```
            ObservableCollection<Asset_Threat_VulnScenario>
VulnScenarios =
            Asset_Threat_VulnScenario.ParseFromDataTable(

DBconnector.GetRows("Asset_Threat_Vuln_Scenario", new string[] {
"@Asset_Id", "@Asset_Name", "@Threat_Id" }, new string[]
                    {                       asset_Threat.Asset_Id,
asset_Threat.Asset_Name,       asset_Threat.Threat_Id},       new
List<string> { "Asset_Id", "Asset_Name", "Threat_Id" }));
            double denominator = 0;
            int[] maxDifferences = new int[VulnScenarios.Count];
int i = 0;
            foreach (Asset_Threat_VulnScenario curVulnScenario in
VulnScenarios)
            {
                int maxDifference = 0;
                foreach                (Asset_Threat_VulnScenario
otherVulnScenario in VulnScenarios)
                {
                    if      ((curVulnScenario.P_parameter      +
curVulnScenario.D_parameter  -  otherVulnScenario.P_parameter  -
otherVulnScenario.D_parameter) >
                        maxDifference)
                        maxDifference                              =
curVulnScenario.P_parameter   +   curVulnScenario.D_parameter   -
otherVulnScenario.P_parameter - otherVulnScenario.D_parameter;
                }
                denominator   +=   curVulnScenario.P_parameter   +
curVulnScenario.D_parameter + p * Qfunction(maxDifference);
                maxDifferences[i] = maxDifference;
                i++;
            }
            i = 0;
            double[]          probabilities         =         new
double[VulnScenarios.Count];
            foreach (Asset_Threat_VulnScenario curVulnScenario in
VulnScenarios)
            {
                probabilities[i]= (curVulnScenario.P_parameter +
curVulnScenario.D_parameter + p * Qfunction(maxDifferences[i]))/
denominator;

DBconnector.UpdateQuery("Asset_Threat_Vuln_Scenario",
DBconnector.GetColumnNames("Asset_Threat_Vuln_Scenario").ToArray
(),

curVulnScenario.ParamsForInsertSQL_query()[0],

curVulnScenario.ParamsForInsertSQL_query()[1],
                    new string[] { "Parameterized_Probability" },
                    new string[] { "@U_Parameterized_Probability"
}, new string[] { probabilities[i].ToString() })
                    ;
```

```
                i++;
            }
            return probabilities;
        }

        public static void UpdateDangerLvl(string Asset_Id, string
Asset_Name, int new_Significance)
        {
            SQLLiteDB_Connector      DBconnector      =      new
SQLLiteDB_Connector("FSTEC_Database.db");
            int difference_in_SignificanceValues =
                new_Significance-
int.Parse(DBconnector.GetRow("Asset_Significance",    "Asset_Id",
Asset_Id)[2]);
            ObservableCollection<Asset_Threat_VulnScenario>
VulnScenarios =
           Asset_Threat_VulnScenario.ParseFromDataTable(

DBconnector.GetRows("Asset_Threat_Vuln_Scenario", new string[] {
"@Asset_Id", "@Asset_Name" }, new string[]
                  { Asset_Id, Asset_Name}, new List<string> {
"Asset_Id", "Asset_Name" }));
            foreach (Asset_Threat_VulnScenario curVulnScenario in
VulnScenarios)
            {

DBconnector.UpdateQuery("Asset_Threat_Vuln_Scenario",
DBconnector.GetColumnNames("Asset_Threat_Vuln_Scenario").ToArray
(),

curVulnScenario.ParamsForInsertSQL_query()[0],

curVulnScenario.ParamsForInsertSQL_query()[1],
                    new string[] { "Danger_lvl" , "Significance"
},
                    new      string[]      {      "@U_Danger_lvl",
"@U_Significance" },
                    new string[] {  (curVulnScenario.Danger_lvl +
difference_in_SignificanceValues).ToString(),
new_Significance.ToString() });
                    ;
            }
        }

        public          static          Asset_Threat_VulnScenario
ParseFromDataRow(DataRow row)
        {
            // получаем все ячейки строки
            var cells = row.ItemArray;
            return                                              new
Asset_Threat_VulnScenario(cells[0].ToString(),
cells[1].ToString(), cells[2].ToString(),
```

```
                cells[3].ToString(),        cells[4].ToString(),
cells[5].ToString(),
                cells[6].ToString(),        cells[7].ToString(),
int.Parse(cells[8].ToString()), int.Parse(cells[9].ToString()),
                double.Parse(cells[10].ToString()),
double.Parse(cells[11].ToString()),
int.Parse(cells[12].ToString()),
int.Parse(cells[13].ToString()));
        }

        public                                              static
ObservableCollection<Asset_Threat_VulnScenario>
ParseFromDataTable(DataTable dt)
        {
            ObservableCollection<Asset_Threat_VulnScenario>
asset_Threat_VulnScenarios                  =                   new
ObservableCollection<Asset_Threat_VulnScenario>();
            foreach (DataRow row in dt.Rows)
            {
                // получаем все ячейки строки

asset_Threat_VulnScenarios.Add(ParseFromDataRow(row));
            }
            return asset_Threat_VulnScenarios;
        }

        //возврат параметров для запроса на вставку в БД
        public List<string[]> ParamsForInsertSQL_query()
        {
            string[]  parameters  =  {"@Asset_Id",  "@Asset_Name",
"@Threat_Id","@Threat_Name","@Id_tactic_numValue", "@NameValue",
                "@Vendor_techniqueValue",  "@Vuln_or_Scenario"  ,
"@P_parameter"    ,    "@D_parameter",    "@Statistic_Probability",
"@Parameterized_Probability",
            "@Danger_lvl","@Significance"};
            string[] values = {Asset_Id, Asset_Name, Threat_Id,
Threat_Name, Id_tactic_numValue,
              NameValue,                  Vendor_techniqueValue,
Vuln_or_Scenario,P_parameter.ToString() ,D_parameter.ToString(),
            Statistic_Probability.ToString(),
Parameterized_Probability.ToString(),      Danger_lvl.ToString(),
Significance.ToString()};
            return new List<string[]> { parameters, values };
        }
        //возврат параметров для запроса на апдейт в БД
        public List<string[]> ParamsForUpdateSQL_query()
        {
            string[]     parameters     =          {"@U_Asset_Id",
"@U_Asset_Name",
"@U_Threat_Id","@U_Threat_Name","@U_Id_tactic_numValue",
"@U_NameValue",
```

```
                "@U_Vendor_techniqueValue", "@U_Vuln_or_Scenario"
,          "@U_P_parameter"          ,          "@U_D_parameter",
"@U_Statistic_Probability",

"@U_Parameterized_Probability","@U_Danger_lvl","@U_Significance"
};
            string[] values =  {Asset_Id, Asset_Name, Threat_Id,
Threat_Name, Id_tactic_numValue,
               NameValue,                  Vendor_techniqueValue,
Vuln_or_Scenario,P_parameter.ToString() ,D_parameter.ToString(),
            Statistic_Probability.ToString(),
Parameterized_Probability.ToString(),      Danger_lvl.ToString(),
Significance.ToString()};
            return new List<string[]> { parameters, values };
        }


        public event PropertyChangedEventHandler PropertyChanged;

        public  void  OnPropertyChanged([CallerMemberName]string
prop = "")
        {
            if (PropertyChanged != null)
                PropertyChanged(this,                          new
PropertyChangedEventArgs(prop));
        }

    }
```

Приложение 7

Реализация методики ГРИФ 2005

```
    class GRIF_Risk
    {
        SQLLiteDB_Connector dBconnector;
        ObservableCollection<Asset_Threat> asset_Threats = new
ObservableCollection<Asset_Threat>();
        ObservableCollection<Asset_Threat_VulnScenario>
asset_Threat__VulnScenarios = new
ObservableCollection<Asset_Threat_VulnScenario>();

        public GRIF_Risk()
        {
        }

        //расчет рисков по ресурсу
        public static void CalculateAll_General_AssetRisks()
        {
            SQLLiteDB_Connector DBconnector = new
SQLLiteDB_Connector("FSTEC_Database.db");
            ObservableCollection<Risk> AssetThreatRisks =
Risk.ParseFromDataTable(DBconnector.GetTable("AssetThreatRisks")
,true);
            Dictionary<string, double> Asset_Risk = new
Dictionary<string, double>();
            Dictionary<string, string> Assets = new
Dictionary<string, string>();
            foreach (Risk asset_risk in AssetThreatRisks)
            {   //если не встречался, добавляем в словарь
                if
(!Asset_Risk.ContainsKey(asset_risk.Asset_Id))
                {
                    Asset_Risk.Add(asset_risk.Asset_Id, 1 -
double.Parse(asset_risk.GRIF_general_risk) / 100);
                    Assets.Add(asset_risk.Asset_Id,
asset_risk.Asset_Name);
                }
                else
                    Asset_Risk[asset_risk.Asset_Id] *= 1 -
double.Parse(asset_risk.GRIF_general_risk) / 100;
            }
            foreach(string asset_id in Asset_Risk.Keys)
            {
                double asset_risk = Math.Round( 1 -
Asset_Risk[asset_id], 5);
                Risk risk = new Risk(asset_id,
Assets[asset_id]);
                if (risk.ThereIsRisk(DBconnector, false))
                {
                    risk.GRIF_general_risk =
(asset_risk*100).ToString();
```

```
                    risk.UpdateGeneralGRIFRisk(DBconnector,
false);
                }
                else
                {
                    risk.GRIF_general_risk =
(asset_risk*100).ToString();
                    risk.InsertRisk(DBconnector,false);
                }
            }
        }

        //расчет рисков по связке Ресурс-Угроза
        public static void
CalculateAll_General_AssetThreatRisks(bool
ParameterizedProbability=true)
        {
            SQLLiteDB_Connector DBconnector = new
SQLLiteDB_Connector("FSTEC_Database.db");
            ObservableCollection<Asset_Threat> Asset_Threats
=Asset_Threat.ParseFromDataTable(DBconnector.GetTable("Asset_Thr
eat"));
            ObservableCollection < Asset_Threat_VulnScenario >
Asset_Threat__VulnScenarios
                                                =
Asset_Threat_VulnScenario.ParseFromDataTable(DBconnector.GetTabl
e("Asset_Threat_Vuln_Scenario"));
            foreach (Asset_Threat at in Asset_Threats)
            {
                double general_risk;
                double temp_param=1;
                foreach (Asset_Threat_VulnScenario atvs in
Asset_Threat__VulnScenarios)
                {
                    if (at.Asset_Id==atvs.Asset_Id &&
at.Threat_Id==atvs.Threat_Id)
                        temp_param *=( 1 -
double.Parse(at.Crit_lvl)/100 *

(ParameterizedProbability?atvs.Parameterized_Probability:atvs.St
atistic_Probability));
                }
                general_risk = 1-Math.Round(temp_param, 4);
                Risk risk = new Risk(at.Asset_Id, at.Asset_Name,
at.Threat_Id, at.Threat_Name);
                if (risk.ThereIsRisk(DBconnector,true))
                {
                   risk.GRIF_general_risk =
(general_risk*100).ToString();
                   risk.UpdateGeneralGRIFRisk(DBconnector,
true);
                }
                else
```

```
                {
                    risk.GRIF_general_risk = (general_risk *
100).ToString();
                    risk.InsertRisk(DBconnector, true);
                }
            }
        }

        //возвращает -1, если общий риск не считался ни разу, 0,
если считался, но не хватает значений (например, добавляли
активы), 1- если посчитан
        public static int
General_AssetThreatRisks_WereCalculated()
        {
            SQLLiteDB_Connector DBconnector = new
SQLLiteDB_Connector("FSTEC_Database.db");
            List<string>
Risks=DBconnector.GetColumn("AssetThreatRisks",
"GRIF_general_risk", "", "GRIF_general_risk");
            if (Risks.Count == 0) return -1;
            foreach (string risk in Risks)
                if (risk.Equals("")) return 0;
            return 1;
        }

        //возвращает -1, если общий риск не считался ни разу, 0,
если считался, но не хватает значений (например, добавляли
активы), 1- если посчитан
        public static int General_AssetRisks_WereCalculated()
        {
            SQLLiteDB_Connector DBconnector = new
SQLLiteDB_Connector("FSTEC_Database.db");
            List<string> Risks =
DBconnector.GetColumn("AssetRisks", "GRIF_general_risk", "",
"GRIF_general_risk");
            if (Risks.Count == 0) return -1;
            foreach (string risk in Risks)
                if (risk.Equals("")) return 0;
            return 1;
        }

        public SQLLiteDB_Connector DBconnector { get =>
dBconnector; set => dBconnector = value; }
        public ObservableCollection<Asset_Threat> Asset_Threats
{ get => asset_Threats; set => asset_Threats = value; }
        public ObservableCollection<Asset_Threat_VulnScenario>
Asset_Threat__VulnScenarios { get =>
asset_Threat__VulnScenarios; set => asset_Threat__VulnScenarios
= value; }
         }
```

Приложение 8

Реализация алгоритма нечеткого вывода оценки риска.

```
    class FuzzyLogicRisk
    {
        SQLLiteDB_Connector dBconnector;
        ObservableCollection<AccessorFunction>
threatAccessorFunctions                    =                    new
ObservableCollection<AccessorFunction>();
        ObservableCollection<AccessorFunction>
vulnAccessorFunctions                     =                    new
ObservableCollection<AccessorFunction>();
        ObservableCollection<AccessorFunction>
valueAccessorFunctions                     =                    new
ObservableCollection<AccessorFunction>();
        ObservableCollection<Rule>       rules       =       new
ObservableCollection<Rule>();

        string threatVariable = "";
        string threatLvlNumber = "";
        string vulnVariable = "";
        string vulnLvlNumber = "";
        string valueVariable = "";
        string valueLvlNumber = "";

        public FuzzyLogicRisk()
        {
      DBconnector = new SQLLiteDB_Connector("FSTEC_Database.db");
      ThreatLvlNumber  =  DBconnector.GetRow("FuzzyLogicSettings",
"Type", "Threat", "LvlNumber")[0];
      VulnLvlNumber  =  DBconnector.GetRow("FuzzyLogicSettings",
"Type", "Vuln", "LvlNumber")[0];
      ValueLvlNumber  =  DBconnector.GetRow("FuzzyLogicSettings",
"Type", "Value", "LvlNumber")[0];
      ThreatAccessorFunctions                =                new
ObservableCollection<AccessorFunction>();
      VulnAccessorFunctions                  =                new
ObservableCollection<AccessorFunction>();
      ValueAccessorFunctions                 =                new
ObservableCollection<AccessorFunction>();
         foreach         (AccessorFunction         af         in
(AccessorFunction.ParseFromDataTable(DBconnector.GetTable("Fuzzy
LogicAccessorFunctions"))))
           {
               switch (af.Variable_type)
               {
                   case                "Threat":                {
ThreatAccessorFunctions.Add(af); break; }
                   case "Vuln": { VulnAccessorFunctions.Add(af);
break; }
                   case                 "Value":                {
ValueAccessorFunctions.Add(af); break; }
               }
```

```
            }

        ThreatVariable = DBconnector.GetRow("FuzzyLogicSettings",
"Type", "Threat", "Variable")[0];
         VulnVariable = DBconnector.GetRow("FuzzyLogicSettings",
"Type", "Vuln", "Variable")[0];
         ValueVariable = DBconnector.GetRow("FuzzyLogicSettings",
"Type", "Value", "Variable")[0];
         Rules   =   Rule.GenerateRules(ThreatAccessorFunctions,
VulnAccessorFunctions, ValueAccessorFunctions);
        }


        //расчет рисков по связке Ресурс-Угроза
        public void CalculateAll_Fuzzy_AssetThreatRisks()
        {

DBconnector.ClearTableQuery("AssetThreatVulnFuzzyLogicRisk");
            //все угрозы
            ObservableCollection<Asset_Threat>  Asset_Threats  =
Asset_Threat.ParseFromDataTable(DBconnector.GetTable("Asset_Thre
at"));
            //все уязвимости
            ObservableCollection<Asset_Threat_VulnScenario>
Asset_Threat__VulnScenarios
                                                =
Asset_Threat_VulnScenario.ParseFromDataTable(DBconnector.GetTabl
e("Asset_Threat_Vuln_Scenario"));

            foreach (Asset_Threat at in Asset_Threats)
            {
                double averagerisk = 0; int countvuls = 0;
                double maxrisk = 0;
                double threat_value = 0;
                switch (ThreatVariable)
                {
                    case "Критичность":
                        {             threat_value             =
double.Parse(at.Crit_lvl) / 100; break; }   //с нормализацией на
[0;1]
                    case "Вероятность угрозы":
                        {
                            if  (!double.TryParse(at.Probability,
out threat_value))
                                threat_value = 0; break;
                        }
                }
                double asset_value = 0;
                switch(ValueVariable)
                {
                    case "Значимость актива":
```

```
                        {               asset_value                =
double.Parse(at.Significance)/ 3; break; }   //с нормализацией на
[0;1]
                }
                Dictionary<string, double> TermsetValue_Threat =
new Dictionary<string, double>();
                Dictionary<string,                            double>
TermsetValue_AssetValue = new Dictionary<string, double>();
                //фаззификация переменных угрозы и ценности
                foreach       (AccessorFunction       af       in
ThreatAccessorFunctions)
                    TermsetValue_Threat.Add(af.Term_set,
ExAccessorFunction(threat_value, af));
                foreach       (AccessorFunction       af       in
ValueAccessorFunctions)
                    TermsetValue_AssetValue.Add(af.Term_set,
ExAccessorFunction(asset_value, af));

                foreach   (Asset_Threat_VulnScenario   atvs   in
Asset_Threat__VulnScenarios)
                {

                    double vuln_value=0;
                    switch (VulnVariable)
                    {
                        case "Вероятность (статистич.)":
                            {             vuln_value              =
atvs.Statistic_Probability; break; }
                        case "Вероятность (параметрич.)":
                            {             vuln_value              =
atvs.Parameterized_Probability; break; }
                        case "Уровень доступа+Сложность":
                            {  vuln_value  =  (atvs.P_parameter  +
atvs.D_parameter) / 8; break; }   //с нормализацией [0;1]
                    }
                    //если угроза принадлежит уязвимости
                    if   (at.Asset_Id   ==   atvs.Asset_Id   &&
at.Threat_Id == atvs.Threat_Id)
                    {   //фаззификация уязвимости
                        Dictionary<string,                double>
TermsetValue_Vuln = new Dictionary<string, double>();
                        foreach    (AccessorFunction    af    in
ValueAccessorFunctions)
                            TermsetValue_Vuln.Add(af.Term_set,
ExAccessorFunction(vuln_value, af));
                        Dictionary<Rule, double> rules_activity =
new Dictionary<Rule, double>();
                        //цикл выбора активных правил и расчета
для них уровня активности
                        foreach(string     threat_termset     in
TermsetValue_Threat.Keys)
                        {
```

```
                            foreach(string    vuln_termset    in
TermsetValue_Vuln.Keys)
                            {
                                foreach (string value_termset in
TermsetValue_AssetValue.Keys)
                                    if
(TermsetValue_Threat[threat_termset]        !=        0        &&
TermsetValue_Vuln[vuln_termset] != 0
                                        &&
TermsetValue_AssetValue[value_termset] != 0)
                                        rules_activity.Add(

Rule.ParseFromDataTable(

DBconnector.GetRows("FuzzyLogicRules",new       string[]       {
"@ThreatTermSet", "@VulnTermSet", "@ValueTermSet" },
                                            new    string[]    {
threat_termset, vuln_termset, value_termset }))[0],
                                            MinOperator(new
double[]                                                    {
TermsetValue_Threat[threat_termset],TermsetValue_Vuln[vuln_terms
et],

TermsetValue_AssetValue[value_termset] })
                                            );
                                }
                            }
                        List<double> LowRisk = new List<double>();
                        List<double> MediumRisk = new List<double>();
                        List<double> HighRisk = new List<double>();
                           foreach (Rule rule in rules_activity.Keys)
                            {
                                switch (rule.RiskTermSet)
                                {
                                    case          "Низкий":          {
LowRisk.Add(rules_activity[rule]); break; }
                                    case         "Средний":         {
MediumRisk.Add(rules_activity[rule]); break; }
                                    case         "Высокий":         {
HighRisk.Add(rules_activity[rule]); break; }
                                }
                            }
                            double            LowRiskAFV            =
MaxOperator(LowRisk.ToArray());
                            double           MediumRiskAFV          =
MaxOperator(MediumRisk.ToArray());
                            double            HighRiskAFV           =
MaxOperator(HighRisk.ToArray());

                            double RiskValue = (LowRiskAFV * 0.33 +
MediumRiskAFV  *  0.66  +  HighRiskAFV  *  0.99)  /  (LowRiskAFV  +
MediumRiskAFV + HighRiskAFV);
```

```
DBconnector.InsertRowQuery("AssetThreatVulnFuzzyLogicRisk",
                           new    string[]    {    "@Asset_Id",
"@Asset_Name",           "@Threat_Id",           "@Threat_Name",
"@Id_tactic_numValue",
                                    "@NameValue",
"@Vendor_techniqueValue", "@FuzzyLogicRisk"},
                           new                          string[]
{at.Asset_Id,at.Asset_Name,     at.Threat_Id,     at.Threat_Name,
atvs.Id_tactic_numValue,
                                    atvs.NameValue,
atvs.Vendor_techniqueValue, Math.Round(RiskValue,4).ToString()});
                        if (maxrisk < Math.Round(RiskValue, 4))
maxrisk = Math.Round(RiskValue, 4);
                        averagerisk += Math.Round(RiskValue, 4);
                        countvuls += 1;
                    }
                }
                Risk risk = new Risk(at.Asset_Id, at.Asset_Name,
at.Threat_Id, at.Threat_Name);
                if (risk.ThereIsRisk(DBconnector, true))
                {
                    risk.Fuzzy_Logic_Average_Risk              =
Math.Round((averagerisk/ countvuls*100),4).ToString();
                    risk.Fuzzy_Logic_Max_Risk                  =
Math.Round((maxrisk * 100), 4).ToString();
                    if
(risk.Fuzzy_Logic_Average_Risk.Equals("не число"))
                    {   risk.Fuzzy_Logic_Average_Risk   =   "-";
risk.Fuzzy_Logic_Max_Risk = "-"; }
                    risk.UpdateFuzzyRisks(DBconnector, true);
                }
                else
                {
                    risk.Fuzzy_Logic_Average_Risk              =
Math.Round((averagerisk / countvuls * 100), 4).ToString();
                    risk.Fuzzy_Logic_Max_Risk                  =
Math.Round((maxrisk), 4 * 100).ToString();
                    if  (risk.Fuzzy_Logic_Average_Risk.Equals("не
число"))
                    {   risk.Fuzzy_Logic_Average_Risk   =   "-";
risk.Fuzzy_Logic_Max_Risk = "-"; }
                    risk.InsertRisk(DBconnector, true);
                }
            }
        }

        public static double MinOperator(double[] values)
        {
            double min = 1;
            for (int i = 0; i < values.Length; i++)
                if (min > values[i]) min = values[i];
            return min;
```

```
        }
        public static double MaxOperator(double[] values)
        {
            double max = 0;
            for (int i = 0; i < values.Length; i++)
                if (max < values[i]) max = values[i];
            return max;
        }

        public static double ExAccessorFunction(double x,
AccessorFunction af)
        {
            if (af.AccessorFunctionType.Equals("треугольная"))
            {
                double a = double.Parse(af.Param_a), b =
double.Parse(af.Param_b), c = double.Parse(af.Param_c);
                if (x < a) return 0;
                if (a <= x && x <= b) return (x - a) / (b - a);
                if (b <= x && x <= c) return (c - x) / (c - b);
            }
            if (af.AccessorFunctionType.Equals("трапециевидная"))
            {
                double a = double.Parse(af.Param_a), b =
double.Parse(af.Param_b), c = double.Parse(af.Param_c), d =
double.Parse(af.Param_d);
                if (x < a) return 0;
                if (a <= x && x <= b) return (x - a) / (b - a);
                if (b <= x && x <= c) return 1;
                if (c <= x && x <= d) return (d - x) / (d - c);
            }
            return 0;
        }
        public static double TriangularFunction(double x,
AccessorFunction af)
        {
            double a= double.Parse(af.Param_a), b=
double.Parse(af.Param_b), c= double.Parse(af.Param_c);
            if (x < a) return 0;
            if ( a <= x && x <= b) return (x-a)/(b-a);
            if (b <= x && x <= c) return (c-x)/(c-b);
            return 0;
        }
        public static double TrapezoidFunction(double x,
AccessorFunction af)
        {
            double a = double.Parse(af.Param_a), b =
double.Parse(af.Param_b), c = double.Parse(af.Param_c), d =
double.Parse(af.Param_d);
            if (x < a) return 0;
            if (a <= x && x <= b) return (x - a) / (b - a);
            if (b <= x && x <= c) return 1;
            if (c <= x && x <= d) return (d - x) / (d - c);
            return 0;
```

```
        }

        public             ObservableCollection<AccessorFunction>
ThreatAccessorFunctions { get => threatAccessorFunctions; set =>
threatAccessorFunctions = value; }
        public             ObservableCollection<AccessorFunction>
VulnAccessorFunctions { get => vulnAccessorFunctions; set =>
vulnAccessorFunctions = value; }
        public             ObservableCollection<AccessorFunction>
ValueAccessorFunctions { get => valueAccessorFunctions; set =>
valueAccessorFunctions = value; }
        public string ThreatVariable { get => threatVariable; set
=> threatVariable = value; }
        public string ThreatLvlNumber { get => threatLvlNumber;
set => threatLvlNumber = value; }
        public string VulnVariable { get => vulnVariable; set =>
vulnVariable = value; }
        public string VulnLvlNumber { get => vulnLvlNumber; set =>
vulnLvlNumber = value; }
        public string ValueVariable { get => valueVariable; set =>
valueVariable = value; }
        public string ValueLvlNumber { get => valueLvlNumber; set
=> valueLvlNumber = value; }
        public   SQLLiteDB_Connector   DBconnector   {   get   =>
dBconnector; set => dBconnector = value; }
        public ObservableCollection<Rule> Rules { get => rules;
set => rules = value; }
    }
```