Санкт-Петербургский государственный университет

Прикладная математика и информатика

Динамические системы, эволюционные уравнения, экстремальные задачи и математическая кибернетика

Веселова Диана Геннадьевна

Разработка системы управления проектными рисками с использованием мультиагентных технологий для виртуальных компаний

Магистерская диссертация

Научный руководитель:

канд. физ.-мат. наук, доц. Кияев В.И.

Рецензент:

канд. физ.-мат. наук, старший науч. сотр.

ИПМаш РАН Иванский Ю.В

Санкт-Петербург

2019

SAINT-PETERSBURG STATE UNIVERSITY

Applied Mathematics and Computer Science

Dynamical Systems, Evolution Equations, Extremal Problems and Mathematical Cybernetics

Veselova Diana

Development of a project risk management system using multi-agent technologies for virtual companies

Master's Thesis

Scientific supervisor:

PhD, Assist. Professor Kiyaev V.

Reviewer:

PhD, Researcher Ivanskiy Y

Saint Petersburg

2019

ОГЛАВЛЕНИЕ

Введение

В настоящий момент переход от информационной к цифровой экономике становится частью развития государственной и социальной политики. Данный процесс вовлекает в себя различные организационные структуры, которые выступают как в роли производителя, так и потребителя услуг информационного оборота. Именно концепция информационного общества позволяет нам выявить приоритет направленности социального развития. Информационное общество можно считать следующим этапом в развитии человеческой цивилизации именно благодаря развитию информационно-телекоммуникационных технологий. Информационное общество и его развитие — это плавно вытекающая стадия эволюции, происходящая при воздействии современнейших Интернет-технологий в различных областях. Концепции информационного общества представляют собой теоретический фундамент, благодаря которому можно рассматривать развитие цифровой экономики.

Помимо цифровизации информационного общества в наше время во всю идет технологизация социального пространства, то есть придание социальному действию динамического, целенаправленного окраса, а также обеспечение его эффективности и практичности. Общественные движения по обеспечению эффективной вовлеченности индивида в социальные связи должны поддерживать информационную технологичность всей социальной сферы. Технологизация задает направления движения по созданию новых служб и организаций, благодаря которым имеется возможность решать уже имеющиеся проблемы социума и отдельного человека.

Технологизация социального пространства — это одна из основных задач социального управления и ее главный способ познания социальной действительности. В основе технологизации лежит теория общественного управления. Общественные процессы обычно делятся на составляющие, решение проблем в которых соответствует необходимой специфике. Помимо

этого, синергетический подход к решению задач открывает новые методы в познании природной и социальной реальности. Синергетика изучает механизмы самоорганизации в открытых системах, она имеет тесную связь с кибернетикой и системным подходом, также она позволяет взглянуть на процессы перехода от хаоса к порядку совершенно по-другому.

С ростом актуальности технологизации социального пространства, цифровизации экономики, частоты использования информационных технологий, а также к повышению требований к качеству разрабатываемых продуктов, бизнес-процессы разработки не стоят на месте. Всего несколько десятилетий назад использовался только лишь водопадный подход жизненного цикла проекта, в котором каждая стадия начиналась только после окончательного завершения предыдущей. Данный подход неудобен и неустойчив к изменяющимся требованиям. Далее только в 2001 году в сферу информационных технологий входит понятие гибких методологий, что полностью меняет подход к управлению ИТ-проектами.

Как следствие изменений подходов к управлению программными продуктами, появляются организации совершенно нового формата. Примером таких организаций являются и виртуальные предприятия, которые взаимодействуют по средствам телекоммуникационных технологий и не принуждают иметь физического офиса.

Такие организации создаются с определенной целью для создания конкретного продукта. Поскольку требования к качеству разрабатываемого программного обеспечения растут, то соответственно увеличивается и необходимость в должном менеджменте программными продуктами, а также управлении рисками разработки. Во многих компаниях данному аспекту не уделяется должного внимания, что в свою очередь приносит как минимум увеличение сроков и бюджета, а как максимум — потерю актуальности разрабатываемого проекта.

Помимо этого, с ростом популяризации и активности использования ИТ во всех отраслях деятельности, многие профессии и сферы заменяются

автоматизированными рабочими местами. Одним из методов автоматизации процессов является использование мультиагентных технологий. В случае такого подхода сложные и масштабные задачи поручаются специализированным программным сущностям — агентам, которые посредством совместной систематической работы достигают необходимого результата намного быстрее и эффективнее, чем люди.

Таким образом, *целью* настоящей магистерской диссертации является разработка системы управления программными рисками в обозначенных выше виртуальных организациях с использованием мультиагентного подхода.

Объектом исследования является виртуальная организация, разрабатывающая сложный ИТ-продукт в распределенной среде разработки.

Предметом исследования является процесс управления рисками в программном проекте.

Для достижения представленной цели были поставлены следующие задачи:

1. изучить и описать особенности деятельности виртуальных организаций;
2. проанализировать и описать риски, связанные с проектной деятельностью в виртуальных организациях;
3. провести аналитическую работу по изучению мультиагентного подхода;
4. описать необходимые для работы аспекты мультиагентных технологий;
5. освоить агентную платформу JADE, для создания мультиагентной системы;
6. разработать состав агентов, а также описать принципы работы этих агентов;
7. разработать алгоритмы функционирования агентов и их взаимодействия между собой;
8. разработать схему базы данных мультиагентный системы;

9. разработать мультиагентную систему управления рисками в виртуальных организациях.

Глава 1. Изменение парадигмы проектного управления: виртуальные организации, занимающиеся разработкой программного обеспечения

1.1 Определение виртуальных компаний

Виртуальные предприятия — это разновидность компаний, организационная форма которой базируется на современных информационно-коммуникационных технологиях (ИКТ) [24]. На развитие этих форм организации и управления предприятием в значительной степени повлияли такие тенденции развития современных рынков, как их глобализация, рост значения качества товара, его цены и степень удовлетворения потребителей, повышение важности устойчивых отношений с потребителями (индивидуализация обслуживания заказчиков), а также растущее значение использования новых информационных и коммуникационных технологий. Как правило, речь идет о сети партнеров (предприятий, организаций, отдельных коллективов и людей), совместно разрабатывающих, производящих и сбывающих продукцию.

С маркетинговой точки зрения цель виртуального предприятия — получение прибыли благодаря максимальному удовлетворению потребителей в товарах путем объединения ресурсов и команды разработчиков различных партнеров в единую систему. Виртуальные предприятия, как правило, ориентируются не на удовлетворение индивидуальных запросов конкретных потребителей, а работают на требования соответствующей ниши рынка, в которой они себя позиционируют. [28]

В различных источниках приведена масса определений виртуальных организаций. Мы выделили ключевые признаки, благодаря которым можно

дать четкое определение таким сущностям как виртуальные компании. Таким образом, введем определение 1.1

Определение 1.1 Виртуальная организация — это разновидность компаний, обладающая следующими признаками:

1. отсутствие конкретного географического расположения;
2. отсутствие ограничений по временному поясу;
3. взаимодействие базируется на использовании информационно-коммуникационных технологий (ИКТ);
4. существование в форме партнерства;
5. партнерство имеет временный характер;
6. отсутствие иерархии в структуре (гибкая структура).

Для четкого понимания данного определения приведем пояснения по каждому из пунктов в Таблице 1.1.

Таблица 1.1 Пояснение к определению «Виртуальная организация»

Пункт определения	Пояснение
Отсутствие конкретного географического расположения	Главной особенностью виртуальных компаний является отсутствие физического офиса, а также распределенное расположение всех сотрудников предприятия.
Отсутствие ограничений по временному поясу	Исходя из распределенной структуры, все ресурсы располагаются в различных часовых поясах
Взаимодействие на основе ИКТ	Отсутствие лично-контактных коммуникаций между сотрудниками компании также является отличительной особенностью. Вся деятельность в предприятии осуществляется посредством информационных технологий, также как и коммуникация между работниками.

Существование в форме партнерства	В виртуальных организациях как правило отсутствует обширное количество ресурсов, таким образом, для реализации того или иного проекта заключаются партнерства между компаниями, имеющими специалистов в различных областях. Помимо этого партнеры юридически независимы друг от друга, что позволяет разделять риски между членами альянса, а также сохранять устойчивость.
Партнерство имеет временный характер	Существование партнерства ограничено временными рамками реализации проекта.
Наличие гибкой структуры	Деятельность данного рода компаний направлена на определенный проект, в рамках которого определена структура команды. Но само предприятие не имеет четкой иерархической структуры, это позволяет компенсировать денежные затраты на содержание ненужных ресурсов для нынешних проектов. Благодаря этому компании быстро подстраиваются под рыночные изменения, нанимая нужные ресурсы на время.

1.2 Модель деятельности в виртуальных компаниях

Под моделью деятельности будем понимать схему бизнес-процесса взаимодействия отдельных компонент, из которых состоит любая деятельность по проекту. Неотъемлемыми частями проекта являются:

1. Заказчик — лицо, формулирующие требования к результатам проекта и использующее полученные результаты проекта. К функциям, которые выполняет заказчик можно отнести определение необходимых результатов проекта, контроль выполнения проекта в контрольных точках, а также приемка промежуточных и окончательных результатов проекта.
2. Менеджер проекта — это руководитель, который несет ответственность за успешную реализацию проектов компании, то есть, когда по окончанию сроков проекта достигнуты все поставленные цели и задачи, при этом не нарушены сроки и не превышен бюджета, а заказчик доволен результатами. Помимо этого, менеджер проекта должен уметь грамотно планировать и контролировать выполнение работ. К функциям, которые выполняет менеджер можно отнести разработку плана-графика проекта, организация работы команды проекта, постановка четких задач и контроль их исполнения, решение конфликтов, формирование команды проекта, распределение ресурсов, общение с заказчиком, предоставление отчетности по проекту.
3. Команда проекта — группа людей, взаимодополняющих и взаимозаменяющих друг друга в ходе достижения поставленных целей проекта. Организационная структура, создаваемая на период осуществления всего проекта либо одной из фаз его жизненного цикла. Задачей руководства команды проекта является выработка политики и утверждение стратегии проекта для достижения его целей. В команду входят лица, представляющие интересы различных участников проекта.

Взаимодействие между этими тремя составляющими можно отразить в общем виде на схеме (рис. 1.1). Детальное взаимодействие зависит от методологии ведения проекта.

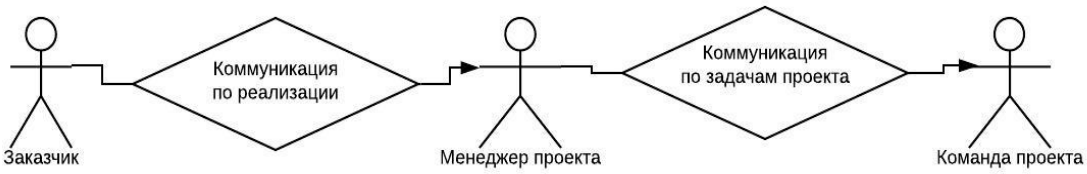

Рисунок 1.1. Схема взаимодействия по проекту

Данную схему можно детализировать, описав подробнее, каким образом происходит коммуникация между отдельными компонентами, а также как организован процесс внутри команды. Что касается виртуальных организаций, все коммуникации происходят путем использования современных информационных технологий и сети Интернет [28].

Организация деятельности в виртуальном предприятии имеет свои особенности. Например, организовать работу сотрудников, находящихся в разных часовых поясах очень сложно, что ведет нас к еще большим рискам по проекту. Именно для минимизации рисков деятельности таких компаний, в данной работе будет описана мультиагентная система, которая будет вести контроль над рисками ведения проектов.

1.3 Методологии управления проектами в виртуальных организациях

Для определения методологий управления проекта введем некоторые базовые понятия.

Определение 1.2. Управление проектами — это область деятельности, в ходе которой определяются и достигаются чёткие цели проекта при балансировании между объёмом работ, ресурсами (такими как деньги, сотрудники, материалы, энергия, пространство и другими), временем, качеством и рисками.

Определение 1.3. Жизненный цикл проекта (Software Life Cycle, SLC) — это набор стадий или этапов, которые необходимо пройти для получения заданного результата [41].

В стадии жизненного цикла разработки программного обеспечения обычно входят следующие стадии [26]:

1. исследование концепции;
2. исследование системы;
3. сбор требований;
4. разработка;
5. внедрение;
6. установка;
7. эксплуатация и поддержка;
8. сопровождение;
9. вывод из эксплуатации.

В зависимости от выбранной методологии, специфики и требований проекта некоторые стадии жизненного цикла продукта могут быть опущены. Каждый проект имеет тройственную ограниченность, под которой подразумевается баланс между такими составляющими как бюджет проекта, время и его содержание. Позже к данной сущности было добавлен четвертый критерий — требования к качеству (рис. 1.2).

Рисунок 1.2. Тройственная ограниченность

Рассмотрим особенности в подходах к управлению виртуальными командами. Организация удаленной работы распределенной команды уже давно не является проблемой, однако, имея команду, расположенную в разных

часовых поясах, сложно организовать по крайней мере коммуникацию внутри нее.

Помимо этого, одним из рисков работы внутри виртуальной команды являются культурные особенности и традиции участников команды, незнание которых может привести к недоразумениям и сбоям в работе. У коллег, которые не имеют личного контакта, не возникает того взаимопонимания, которое присуще командам, работающим в одном офисе. А это немаловажное условие для создания благоприятной атмосферы внутри коллектива и высокой эффективности деятельности.

Выделим основные правила построения эффективного рабочего процесса в виртуальной команде проекта [16]:

1. Предоставить Online-ресурс для коммуникации участников команды. Для продуктивной работы требуется инвестировать время и деньги в удобный ресурс, где участники команды смогут наладить взаимодействие и коммуникацию друг с другом. Например, основатель рекламной компании Ogilvy & Mather Дэвид Огилви обратил свое внимание на получение знаний внутри компании. Более десяти лет назад он инвестировал в развитие внутрикорпоративного интернет-сообщества, которое он назвал Truffles.
2. Выделить участников, которые уже знают друг друга.
3. Иметь минимум 15 % коммуникативных людей в команде.
4. Разбить проект на этапы, мало зависящие друг от друга. В данном случае, если после завершения одного из этапов проекта будут проблемы внутри команды, снижается риск отрицательного завершения проекта.
5. Создать проектный Web-ресурс, где участники команды смогут проявлять инициативу, высказывать идеи.
6. Инициировать эффективное общение команды, но не настаивать на регулярных сборах.

7. Поручать исполнителям задачи, которые являются интересными для них, а также стимулирующими. В данном случае менеджер покажет свою заинтересованность в развитии компетенций того или иного участника команды.
8. Показывать важность каждой задачи для проекта.
9. Привлекайте как можно больше добровольцев. Например, хорошо известные проекты Wikipedia и Linux показали, что виртуальные команды процветают, когда они состоят из добровольцев с ценными навыками, людей, которые своей готовностью присоединиться к команде уже доказали свою заинтересованность в проекте.

Все перечисленные выше пункты можно обобщить, сказав, что основным принципом продуктивной работы в виртуальной команде является наличие общей цели, заинтересованность в результате и прочно выстроенная коммуникация, основанная на доверии.

Помимо этих особенностей необходимо установить также нужный уровень доверия в команде. Эта обязанность лежит на менеджере проекта, который должен уметь управлять уровнем доверия в команде. Это одно из отличий между организацией работы в стандартных командах и виртуальных. В обычных командах, работающих в одном офисе, руководитель не всегда может уделять этому внимание, в то время как в виртуальной команде это важно. Руководитель виртуальной команды должен осознавать, какие факторы позволяют укрепить доверие в команде, а в идеале — уметь оказывать на них влияние.

Рисунок 1.3. Факторы управления доверием в команде

Основные аспекты, которые важны с точки зрения управления доверием отображены на рисунке 1.3. Обратим внимание, что можно выделить краткосрочное и долгосрочное доверие в команде. В данном случае, выбор стратегии построения доверия зависит от сроков существования проектной команды. Построение долгосрочного доверия строится на акцентировании внимания на профессиональной значимости каждого члена команды, а также на выделении личных качеств. К профессиональной значимости можно отнести компетентность сотрудника, его статус в команде и богатство его личных связей с другими людьми, которые можно использовать для выполнения проекта. В данном случае, руководителю необходимо уделять внимание этим факторам, принимая во внимание, что необходимо развивать профессиональные навыки каждого члена команды, поощрять новые идеи и инициативы, а также открытость каждого специалиста. Под безопасностью на данной схеме подразумевается уровень доверия между участниками команды, осознание каждого из них того, что можно передавать всю информацию друг другу.

Что касается выбора методологии управления проектом, то основным фактором, на который стоит опираться при выборе — это особенности

самого проекта. Одним из самых эффективных и современных подходов является использование гибких проектных методологий Kanban и Scrum (ссылки в виде сносок). Схемы данных подходов проиллюстрированы на рисунках 1.4 и 1.5 соответственно [33].

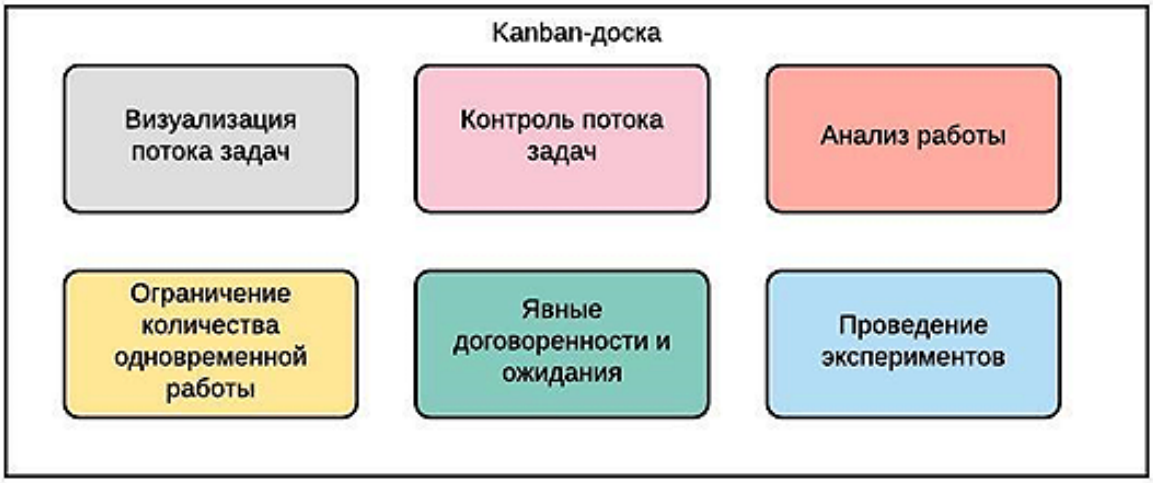

Рисунок 1.4. Схема Kanban

В основе подхода Kanban лежит использование специальной доски, на которой находятся задачи, выстроенные в порядке приоритета. Явным преимуществом является здесь то, что на доске можно визуально отделить задачи по категориям, соответствующим статусам исполнения той или иной задачи. Например, к ним можно отнести следующие статусы: в ожидании, в работе, тестирование, корректировки, завершена, архив и т.д. Преимуществом является то, что каждая команда определяет эти статусы под свою работу самостоятельно. Для того, чтобы вести такую доску, можно использовать уже готовые Web-ресурсы, например, Trello (из самых популярных систем управления проектами в режиме онлайн, позволяющая эффективно организовывать работу по японской методологии Kanban-досок). Для анализа работы проводятся митинги команды, на которых каждый член команды может высказать свои предложения по повышению эффективности выполнения задач.

Соответственно, гибкий подход к управлению проектами Scrum основан на «спринтах», длительностью от 1 до 4-х недель [33].

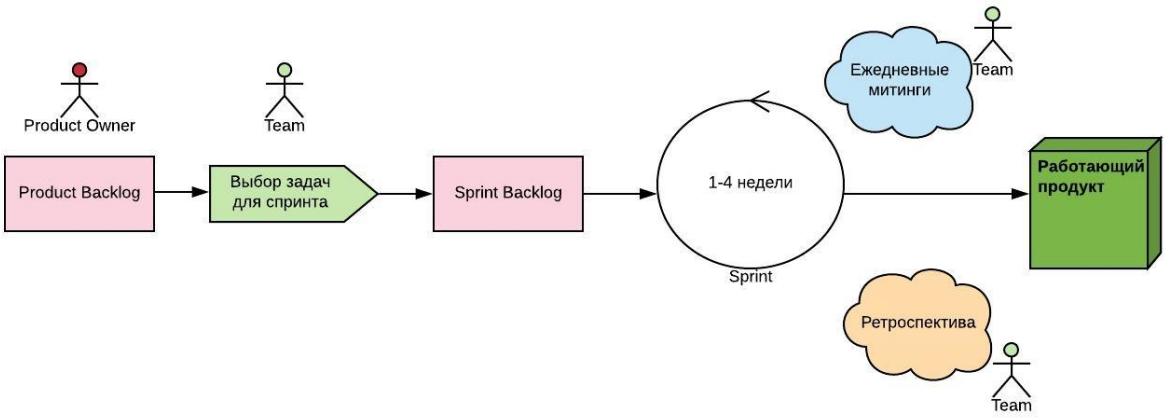

Рисунок 1.5. Схема Scrum

Определение 1.4. *Спринт* — это отрезок времени, за который Scrum-команда создаёт часть продукта, готовую к показу и ценную для клиента.

Введем дополнительные определения, которые фигурируют на рисунке 5.

Определение 1.5. *Product Backlog* — это стек всех задач проекта, которые рассортированы по приоритету выполнения. За данной сущностью следит член команды Product Owner. Sprint Backlog — стек задач, выбранных из Product Backlog, для их реализации во время нынешнего спринта.

По истечении каждого спринта на выходе команда и клиент должны получить версию готового работающего продукта. Задачей команды является проведение ретроспективы, для того, чтоб провести анализ проделанной работы и принять к сведению все допущенные ошибки.

Особенностью данного подхода является то, что каждый член команды всегда находится в курсе состояния каждой задачи на проекте. Таким образом, в виртуальной команде для ежедневных митингов необходимо выбирать удобное для всех членов команды время для проведения обязательных ежедневных митингов, используя, например, Skype. А планирование задач можно осуществлять в специализированных Web-сервисах, о которых говорилось выше.

В итоге, какую бы методологию не выбрал менеджер виртуальной команды, необходимо будет прибегнуть к использованию

Online-ресурсов для повышения качества работы и процесса исполнения задач. А это, как правило, может привнести дополнительные риски в проект.

1.4 Управление рисками в виртуальных предприятиях

Работая в сфере информационных технологий, мы сталкиваемся с различными рабочими ситуациями, сторонними системами, людьми и т. д. Каждый сторонний и внутренний фактор, связанный с командой и компанией, влияет на процесс исполнения того или иного проекта. Рассмотрим самый простой и примитивный жизненный цикл (далее ЖЦ) проекта. Институт управления проектами выделил 5 основных групп процессов, входящих в ЖЦ [23]:

1. инициализация;
2. планирование;
3. выполнение;
4. контроль и мониторинг;
5. завершение.

Каждая из этих групп является неотъемлемой частью любого проекта. Но в то же время, их можно конкретизировать и адаптировать под каждый продукт по-своему, в зависимости от целей и нужд.

Стадии инициализации и планирования являются одними из самых основных, так как именно на этих стадиях необходимо провести полный анализ требований к продукту и сделать планы по выполнению проекта (план управления содержанием, план управления расписанием, план управления стоимостью, план управления качеством, план управления рисками и т. д.). Соответственно, для составления данных планов необходимо провести детальный анализ рисков, которые могут возникнуть при работе над проектом. Общая схема анализа рисков приведена на рисунке 1.6 [16].

Рисунок 1.6. Схема работы с рисками

Для начала работы с рисками введем необходимые определения.

Определение 1.6. *Риск проекта* — это неопределенное событие или условие, которое в случае его появления имеет какое-либо влияние как минимум на одну из целей проекта, например: сроки, бюджет, содержание или качество (в зависимости от вида проекта) [16].

Классифицируем риски проекта следующим образом. Разделим их на две группы:

1. *Известные риски* — это риски, которые команда может идентифицировать с самого начала проекта, и затем подвергнуть анализу.
2. *Неизвестные риски* — это те риски, которые команда проекта не может предусмотреть или проанализировать. Для таких рисков необходимо выделить некоторый резерв на непредвиденные обстоятельства, включив туда и известные риски, для которых

разработка конкретных мер реагирования не представляется необходимой или возможной.

Разберем подробнее работу с рисками на примере проиллюстрированной выше схемы. На стадии планирования и идентификации рисков менеджеру необходимо собрать митинг со всеми участниками проекта, туда может быть приглашена не только команда, но также и представители со стороны заказчика. Всем причастным необходимо заполнить анкету для выявления рисков проекта. В своде знаний PMBoK приведена следующая классификация рисков (табл. 1.2) [41].

Таблица 1.2 Классификация рисков

Проект	Технический	Требования
		Технология
		Степень сложности и интерфейсы
		Эффективность и надежность
		Качество
	Внешний	Субподрядчики и поставщики
		Предписания контролирующих органов
		Рынок
		Заказчик
		Внешние условия окружающей среды
	Организационный	Организации, от которых зависит проект
		Ресурсы

		Финансирование
		Расстановка приоритетов
	Управления проектом	Оценки
		Планирование
		Контроллинг
		Коммуникация

В этой анкете лицам необходимо сформулировать влияние исполнения каждой из работ проекта на выполнение проекта в целом. А также нужно определить риски, которые могут оказать негативное влияние на выполнение проекта. После этого предстоит оценить вероятность их наступления, существенность последствий, а также возможные методы управления. Для этого необходимо использовать выше предложенную классификацию рисков и разработанные рабочей группой на основе практик риск-менеджмента шкалы влияния. Приведем пример таких шкал в таблицах 1.3 и 1.4. Данные таблицы мы будем использовать в дальнейшем.

Таблица 1.3 Шкала оценки вероятности наступления риска [26]

Коэффициент	Вероятность	Вероятность наступления, %
1	Риск не проявится	5
2	Риск, скорее всего, не проявится	10
3	Вероятность наступления и не наступления риска одинакова	50
4	Риск, скорее всего, проявится	75
5	Риск наверняка реализуется	95

Таблица 1.4 Шкала оценки существенности последствий от наступления риска [26]

Коэффициент	Последствия	Стоимость	Сроки	Содержание
1	Незначительные и минимальные	Незначительное увеличение	Незначительное увеличение	Едва заметное увеличение

				содержания
2	Допустимые	Увеличение менее чем на 10 %	Увеличение более чем на 5%	Затронуты второстепенные области содержания
3	Значительные	Увеличение на 10-20%	Увеличение на 5-10%	Затронуты основные области
4	Критические, проявление которых может привести к значительным потерям или приостановке работ	Увеличение на 20-40%	Увеличение на 10-20%	Уменьшение содержания неприемлемо для спонсоров
5	Катастрофические, реализация которых может привести к прекращению работ	Увеличение более чем на 40%	Увеличение более чем на 20%	Конечный продукт практически бесполезен

После заполнения данных анкет ответственный делает анализ рисков и заполняет первоначальную версию карты рисков. Для этого необходимо заполнить следующую таблицу (табл. 1.5):

Таблица 1.5 Первоначальная карта рисков

№	Наименование риска	Организация	Событие (содержание работы)	Вид риска	Описание риска	Причины	Последствия
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Указать наименование риска, конкретное событие и вид риска в соответствии с представленной выше классификацией рисков. Каждое событие (риск) должно однозначно соответствовать одной из групп риска					Определение риска с указанием причин, событий, описывающих вероятность проявления данного риска, а также обоснование возможны	Перечень причин, описывающий наступление данного события	Перечень возможных последствий от наступления рискового события

	х последствий		

На этапе анализа рисков данная карта дополняется еще 3-мя столбцами, описанными в таблице 1.6.

Таблица 1.6 Дополненная карта рисков (часть 1)

Общая оценка риска		
Вероятность риска	Последствия	Оценка
9	**10**	**11**
Оценивается в виде коэффициента из вышеприведенной таблицы 1.3	Оценивается в виде коэффициента последствий наступления рисковых событий исходя из таблицы 1.4	Равно произведению коэффициентов вероятности и последствия. Показывает общую оценку риска с предположением того, что данный риск не управляется и не контролируется

После построения данной таблицы можно построить уже более точную карту рисков, распределив риски в порядке увеличения или уменьшения потерь от его влияния. Также если рисков много, можно разделить данную карту на категории, относящиеся к рискам по срокам, бюджету и т.д.

После процедуры анализа рисков необходимо перейти к планированию реагирования на риски. Для этого необходимо заполнить следующую таблицу (табл. 1.7).

Таблица 1.7 Дополненная карта рисков (часть 2)

Владелец процесса управления риском	Воздействие на риск	Стоимость воздействия	Регламентирующие документы	Общий риск		
				Вероятность	Последствие	Оценка
12	**13**	**14**	**15**	**16**	**17**	**18**

В столбец 12 необходимо внести данные о лице, отвечающим за определенный риск, за его контроль и оценку. Под воздействием на риск подразумевается перечень мер, стратегий, которые необходимо предпринять для устранения сложившейся проблемы. В 14-м столбце определяется экспертная оценка стоимости воздействия риска, для ее идентификации

необходимо использовать шкалу: высокая, средняя, низкая, определенную для каждого проекта отдельно. К регламентирующим документам относится любой документ, который сможет даже косвенно помочь в снижении нужного риска. Общий риск (столбцы с номерами 16-18) оценивается по аналогии с таблицей 1.6. Единственным отличием между данными оценками и оценками из таблицы 1.6 является то, что здесь параметры оцениваются с учетом выше объявленных факторов, то есть с учетом того, что данный риск управляется и контролируется, т. е. после воздействия указанных выше методов управления.

Если наступает такая ситуация, что исключить все риски невозможно, то необходимо использовать стратегию принятия рисков, в этом случае требуется дополнительный контроль и мониторинг воздействия данного риска на ход проекта. В случае данной стратегии необходимо создать резерв на непредвиденные обстоятельства, включающий в себя время, деньги или ресурсы для управления принятыми (и неизвестными) рисками.

В конечном итоге, необходимо получить карту рисков, состоящую из 18 столбцов, которой необходимо придерживаться в ходе всего проекта. И на стадиях мониторинга и управления рисками данная карта может дополняться уже новыми, еще не выявленными рисками.

1.5 Риски управления проектами в виртуальных организациях

Приведем примеры возможных рисков в управлении проектами, касающихся именно виртуальных организации. Для формирования данного перечня рисков была проведена аналитическая работа по выявлению и изучению рисков менеджмента ИТ-проектов, их классификация, а также поиск мер по их устранению. Ознакомиться с данным перечнем можно в таблице 1.8 [23, 26].

Таблица 1.8 Риски проектов в виртуальных компаниях

Факторы и категории риска	Описание	Решение
Задачи и целевые факторы, организация управления		

Соответствие проекту	Здесь идет речь о целях проекта, четко ли они определены, правильно ли поставлены, идут ли работы по их выполнению.	Четкое формирование требований и их документирование, либо использование гибких методологий
Организационная стабильность	Здесь идет речь о менеджменте проекта, есть ли стабильность в управлении, есть ли вероятность реорганизации структуры менеджмента.	Документированная иерархическая структура проекта, либо быстрое обучение новых сотрудников
Стабильность команды разработчиков	Данный риск имеет высокую степень, если команда разработчиков не подобрана или подобрана неправильно, а также нет единого мнения по ее составу	Использование инструментов общего доступа, где хранится вся информация по проекту
Финансирование проекта	Прекращение финансирования проекта	Исполнитель не имеет механизмов управления данным риском. При прекращении финансирования проекта работы должны быть остановлены, дальнейшие действия участников проекта с обеих сторон регулируются договором и действующим законодательством
Факторы, связанные с заказчиком		
Вовлечение заказчика	Этот риск имеет высокую степень вероятности, если заказчик почти не принимает участия в разработке продукта, а также не имеет	Необходимо наладить коммуникацию с заказчиком

	представления о работе	
Соответствие контрактным условиям	Высокая степень риска наблюдается, когда контракт имеет обременительные задокументированные требования или нуждается в дополнительной работе по согласованию	Официальное согласование доработок, либо использование гибких методологий
Факторы, связанные с графиком выполнения работ		
График разработки	Здесь имеется в виду риск неправильно составленного графика работы, который не отображает работу команды в реальном времени	Корректировка плана по мере выполнения работ, использование гибких методологий
План-график	Некорректная детализация задач проекта и расстановка их приоритетов	Доскональное изучение архитектуры, согласование решений с командой архитекторов, технических руководителей и руководителей направления тестирования.
Оценка	Неправильная оценка приоритетов рисков и вероятности их актуализации	Построение матрицы рисков по результатам мозгового штурма проектной командой, актуализация приоритетов с утвержденной периодичностью, регулярный мониторинг актуализации и утилизации рисков
Факторы, связанные с персоналом		
Доступность персонала	Здесь идет речь о том, что риск возникает, когда идет передача	Налаженная коммуникация внутри команды

	большинства функций другим людям и команда тратит много времени на устранение различных проблем	
Часовые пояса	Исполнители(команда) находятся в разных городах и часовых поясах	Организовать дистанционное взаимодействие, используя e-mail, icq, skype и прочие современные средства коммуникации
Ключевые специалисты	Риск ухода ключевых специалистов из команды	Частичное дублирование обязанностей между различными участниками проекта, вовлечение в предметную область всех участников проекта, доступность для всех участников команды информации о распределении задач и ходе их выполнения
Область знаний	Недостаточность знаний предметной области	Обеспечить тесную коммуникацию сторон-участников проекта для своевременного прояснения возникающих вопросов и исключения неверной трактовки требований и ошибок при согласовании проектных решений; привлечь сторонних экспертов
Пользователи системы	Пользователи не имеют достаточных навыков работы с системой данного уровня сложности	Начать проводить обучение пользователей как можно раньше

1.6 Выводы к Главе 1

В данной главе были описаны основные аспекты деятельности в виртуальных организациях, были проанализированы модели бизнес-процессов для управления ИТ-проектами в компаниях данного типа. Помимо этого, были описаны основные методологии, которые актуальны в данное время в сфере управления ИТ-проектами. Также была проведена аналитическая работа по проверке работоспособности данных методологий в распределенных командах.

Далее, были проанализированы подходы к риск-менеджменту, которые были описаны в рамках использования их в деятельности виртуальных проектов. Также были выделены конкретные группы и примеры рисков, которые могут быть использованы в распределенных проектах, было введено понятие карты рисков, которое является основным в данной работе и будет использоваться и далее.

Глава 2. Использование мультиагентных технологий для управления рисками в виртуальных организациях

2.1 Мультиагентные технологии и возможность их использования в управленческих целях

В эпоху широкого использования информационных технологий особую популярность набирает автоматизация процессов управления различными системами. Большинство крупных организаций стараются ввести автоматизированные рабочие места (программно-технический комплекс автоматизированных систем для автоматизации деятельности определенного вида) и сделать рабочие бизнес-процессы более гибкими и эффективными. Одним из вариантов автоматизации является использование мультиагентных технологий. Данные системы используются в различных видах деятельности, например, в компьютерных играх и графических приложениях, транспорте, логистике, робототехнике, сфере мобильных и сетевых технологий и т. д.

Определение 2.1. *Многоагентной (мультиагентной) системой* (МАС) называется совокупность в разной степени автономных объектов (агентов), взаимодействующих друг с другом и окружающей средой как в интересах решения собственных, так и общих задач [3].

Под агентом в данном случае могут подразумеваться различные программные системы и модели, роботы, а также люди и команды людей. Помимо этого, системы могут быть и смешанные, то есть состоящие из нескольких видов агентов. В нашей работе мы рассмотрим пример программных агентов, то есть определенную программу (программный модуль), имеющую особые характеристики, и выполняющую заданную функцию, приводящую к некоторому результату. Но так как достичь полной автоматизации процессов почти невозможно, то в качестве дополнительных агентов мы будем использовать членов проектной команды, имеющих соответствующие полномочия.

Мультиагентный подход применяется при решении масштабных и сложных задач. В случае составления сложных алгоритмов решений, больших объемов вычислений, работы с масштабным количеством информации использование ресурсов человека может быть недостаточно. В таких ситуациях действует человеческий фактор, при котором человек может не учесть какие-либо важные аспекты задачи, которые в дальнейшем повлияют на результат. В случае сложных и масштабных проектов потери бюджета и времени напрямую зависят от отклонения полученного результата от необходимого.

При использовании МАС решение же получается автоматически, что является результатом взаимодействия агентов в системе. Для реализации данных систем обычно используется 2 и более агентов, так как в теории один агент должен иметь частичное представление о всей решаемой проблеме.

Приведем важные характеристики агентов в мультиагентных системах [7]:

1. *Автономность*. В данном случае подразумевается некоторая независимость между агентами.
2. *Ограниченность.* То есть ни у одного из агентов нет полного представления о всей системе, но при этом каждый агент имеет свою определенную задачу.
3. *Децентрализованность.* В системе не предусмотрен агент, который управлял бы всей системой.
4. *Способность к коммуникации.* Агенты могут обмениваться знаниями, используя определенный язык и заданные правила. Примером такого языка является Knowledge Query and Manipulation Language (KQML) и Agent Communication Language (ACL).
5. *Обучаемость.* Каждый агент способен накапливать знания в процессе работы.

Для того, чтобы распределить задачи между агентами, можно использовать один из следующих подходов:

1. Использование системы распределенного решения.
2. Использование децентрализованного искусственного интеллекта, то есть совокупности средств, моделирующих человеческое мышление.

В случае первого подхода глобальная проблема разделяется на задачи, каждому агенту задается определенная роль и проектирование системы идет строго сверху вниз, используя иерархическую структуру. Все данные действия происходят под управлением некоторого единого «центра», под руководством которого происходит обратная связь между агентами.

При использовании децентрализованного искусственного интеллекта распределение задач носит зачастую случайный характер, то есть оно является результатом коммуникации агентов между собой. Данный подход явно не является преимущественным, так как случайное распределение носит неустойчивый характер.

Рассмотрим более подробно главный объект МАС – агента.

Определение 2.2. *Агент* – это некоторая открытая система, то есть система, которая взаимодействует с окружающей ее средой, обладающая своим определенным поведением и действующая по своим принципам [9].

Под окружающей средой мы будем понимать внешнюю среду и других агентов системы. Рассмотрим более подробно данное взаимодействие. На рисунке 2.1 представлена схема, иллюстрирующая обмен информацией между агентом и внешними сущностями. Как мы видим, двумя важными составляющими агента являются сенсоры, через которые агент принимает информацию от внешней сущности, и актуаторы, то есть те блоки, посредством которых агент воздействует на среду.

Примером таких блоков для человека-агента могут являться органы чувств, в то время как робот-агент может воспринимать и передавать информацию через специальные датчики, видеокамеры и прочие устройства. Что касается программных агентов, то для них в качестве сенсоров могут выступать различные комбинации нажатия клавиш, содержимое файлов, код программы, а в качестве актуаторов агент передает данные на экран, записывает их в файлы и т.д. Таким образом, агент может выполнять следующие функции [12]:

- воспринимать информацию;
- обрабатывать информацию;
- действовать на внешнюю среду;
- взаимодействовать с другими агентами.

Помимо этого, на рисунке 2.1 можно увидеть принцип построения правил и принципов агента. В случае простейшего агента, набор правил для его функционирования строится на основе условий вида:

ЕСЛИ <условие> **ТО** <действие 1> **ИНАЧЕ** <действие 2>

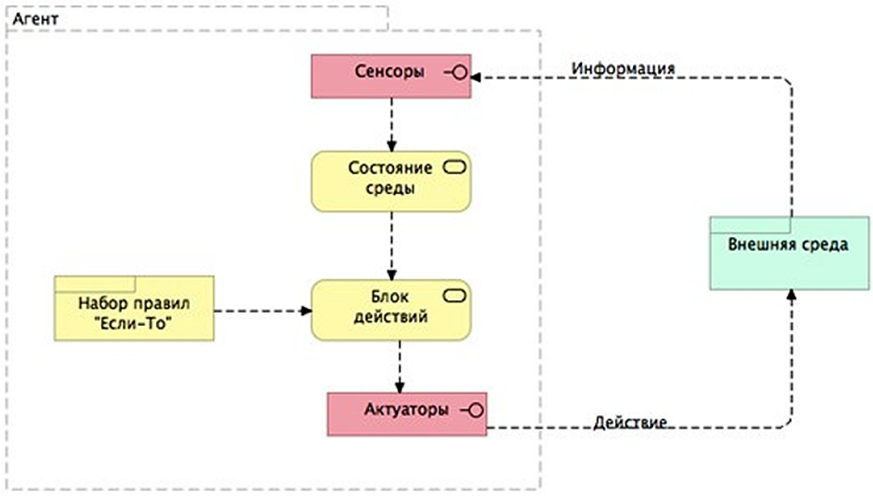

Рисунок 2.1. Схема взаимодействия агента и окружающей среды

Для наглядности представления об агентах приведем их классификацию (рисунок 2.2).

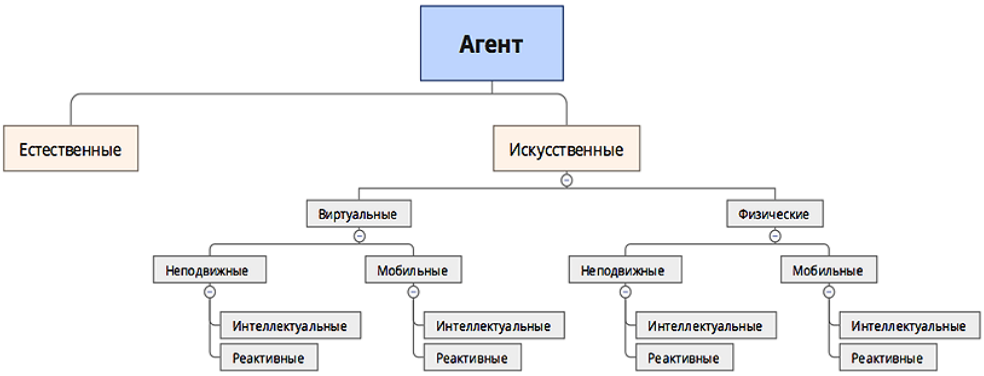

Рисунок 2.2. Классификация агентов

Определение 2.3. *Интеллектуальный агент* – это программно-аппаратная система, обладающая следующими свойствами:

- автономность;
- коммуникабельность;
- реактивность;
- обучаемость (своя база знаний, которая постоянно пополняется);
- убеждения (знания о себе и о среде);
- цели;
- намерения (список действий по отношению к другим агентам и среде);
- обязательства перед другими агентами.

Понятие агента в «слабом» смысле ограничивается наличием у агента наличием первых четырех свойств. Если агент обладает всеми вышеперечисленными качествами, то он называется интеллектуальным в «сильном» смысле [18].

Структура интеллектуального агента изображена на рисунке 2.3.

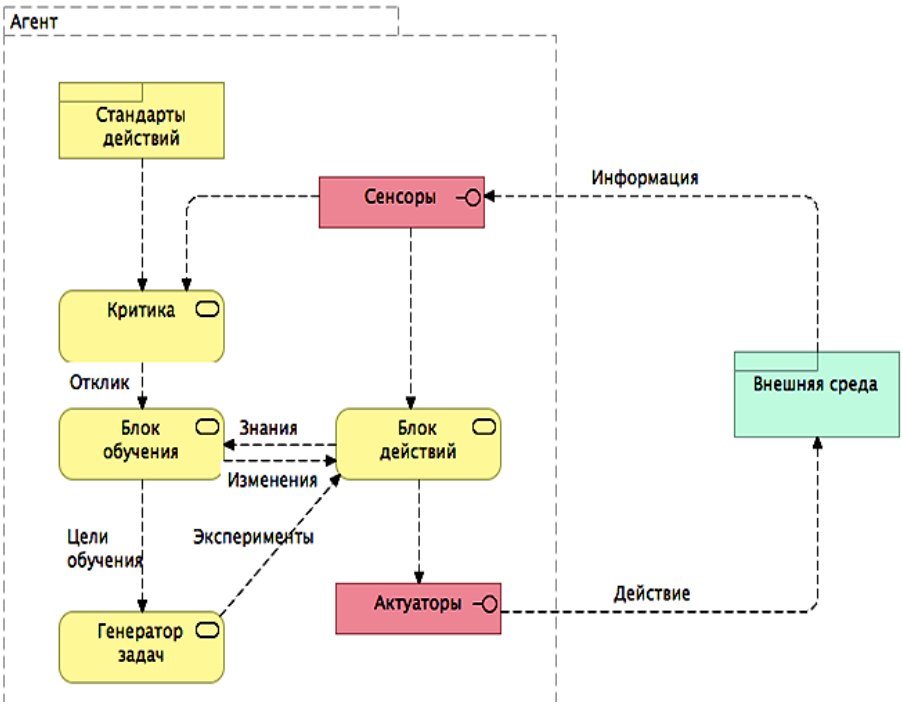

Рисунок 2.3. Структура интеллектуального агента

Интеллектуальные агенты можно разделить на 2 вида [20]:

- *интенциональные*, то есть наделенные мотивацией, моделирующие свои намерения и цели;
- *рефлекторные*, то есть не имеющие внутренней мотивации, как правило их поведение примитивно, они способны отвечать на вопросы и выполнять различные задания от других агентов, но при этом не создавая собственной цели.

Для моделирования агентов можно выделить 3 класса архитектур агентов, которым соответствуют модели их поведения:

- реактивные;
- делиберативные;
- смешанные.

Реактивные агенты построены на основе искусственного интеллекта, моделирующие взаимодействие среды и агента, имеющие перечень простых схем поведения, реагирующие на среду в форме «стимул – реакция». В свою очередь делиберативные агенты работают с

устоявшейся моделью окружающей среды (модель мира) и принимают решения при помощи конкретного планировщика (на основе формальных рассуждений, устоявшихся знаний и объектов).

2.2 Формирование мультиагентной системы

Для формирования агентов в систему необходимо определиться с видом организации структуры МАС. Выделим три вида систем [20]:

1. *Децентрализованный искусственный интеллект.* Все агенты автономны, окружающая среда динамическая.
2. *Распределенный искусственный интеллект*. В данном случае система создается для решения конкретной задачи и достижения конкретной цели, она имеет определенный центр управления.
3. *Искусственная жизнь.* В данном случае подразумевается децентрализованная система, которая может эволюционировать и состоит из реактивных агентов.

Главной чертой МАС является взаимодействие между агентами. Данное взаимодействие можно классифицировать следующим образом:

- кооперация, коллективная работа нескольких агентов для достижения общей цели;
- конкуренция, конфликт;
- поиск компромисса, готовность идти на уступки в использовании ресурсов;
- отказ от своих интересов в пользу других агентов;
- отказ от взаимодействия.

Рассмотрим более подробно *кооперацию агентов*. Она может быть организована при помощи удовлетворения каким-либо общим правилам поведения группы агентов. Обычно такое взаимодействие происходит, если установленные общие правила обязательно должны быть выполнены. Здесь каждый агент является конечным автоматом, но при этом их взаимодействие при выполнении обязательных правил создает систему с

высокоорганизованным поведением. Примером такого поведения является квазиразумное поведение роя пчел или колонии муравьев [6].

Вторым видом кооперации является взаимодействие на основе обмена информацией. В данном случае каждый агент имеет свое планирование, а их коммуникация ведется для принятия того или иного решения. В данном случае у каждого агента есть своя роль, а их коммуникация ведется по заданным протоколам.

Введем понятие конфликта между агентами. Обозначим символами r и m убеждения агента 1 и агента 2 соответственно. Данные убеждения могут быть либо ложными, либо истинными.

Определение 2.4. Под *конфликтом* будем понимать ситуацию, когда при взаимодействии агентов выполняется тавтология:

$$r \wedge m \Rightarrow \text{ложь},$$

где r и m могут принимать и значение *ложь*, и значение *истина* [9].

Для того, чтобы разрешить данный конфликт между агентами, необходимо либо отбросить одну из альтернатив, в силу какого-либо критерия, либо изменить r и m одновременно. Данное решение принимается либо исходя из правил поведения агентов, если таковое имеется, либо с использованием централизованного механизма, либо с использованием рандомного решения проблемы.

Для реализации взаимодействия между агентами будем использовать *протоколы передачи информации.* Под протоколом будем понимать определенный алгоритм, с использованием которого ведется коммуникация между агентами. Обычно к протоколам предъявляют следующие требования:

1. *эффективность*, в понимании того, что улучшение состояние одного агента не может быть достигнуто ухудшением состояния других агентов;
2. *простота*, то есть неусложнение архитектуры системы или агента;
3. *устойчивость* к изменениям;

4. *децентрализованность,* то есть проведение коммуникации без использования специального отдельного механизма принятия решений;
5. *равноправность,* то есть одинаковые правила участия в переговорах для всех агентов.

Обычно протокол строится из трех составляющих. Проиллюстрируем данные компоненты и требования к ним в таблице 2.1.

Таблица 2.1. Компоненты протокола взаимодействия

Компонент	Требования
Пространство соглашений	Фиксировано и известно всем агентам
Правила взаимодействия	Модель «предложение – контрпредложение», представляющееся речевыми конструкциями
Пространство стратегий поведения	Должны образовывать равновесие «по Нэшу», то есть один агент не может изменить своё решение в одностороннем порядке, тем самым увеличив выигрыш, в том случае если другие агенты свое решение не меняют

Если подробнее описать процесс достижения консенсуса между агентами, то имеет смысл дать определение роевого интеллекта.

Определение 2.5. *Групповой* (*роевой) интеллект* – это система, которая реализует управление общим коллективным поведением децентрализованным самоорганизующимся множеством однородных объектов. Под однородными объектами мы понимаем агентов [10].

Приведем набор отличительных характеристик роевой системы:

- агенты независимы, то есть способны двигаться и взаимодействовать с внешней средой без централизованного управления;
- общая задача должна выполняться большим числом агентов;
- роевая система состоит из однородных групп агентов, а также упор делается на использование одинаковых объектов, нежели на совокупность агентов, которым назначена своя роль.

Определение 2.6. *Группой (роем*) будем называть множество однородных динамических агентов a_i, где $i = 1 \dots n$, которые при взаимодействии решают множество задач z_i, где $i = 1 \dots m$. Все агенты одинаковы, у каждого агента есть некая «зона видимости» радиусом R (внутри этой зоны агент обменивается сообщениями с другими агентами роевой системы и знает состояние этих агентов). Помимо этого, зададим расстояние *r,* на котором агент движется от своих ближних соседей.

Зададим карту потенциалов, которая определяет перспективу движения в заданном направлении, таким образом у каждого агента будет формироваться личная потенциальная картина мира. Движение для агента будем характеризовать направлением x_i, и значением потенциала выбранного пути q_i. Потенциал характеризует возможность достижения цели s, при сохранении движения в выбранном направлении x_t^i .

Таким образом, при заданной цели s_t^i для агента i в момент времени t можно положить потенциал $q_t^i = q_t^i(x_t^i) =< x_t^i, s_t^i >$. Также агент старается выбрать маршрут так, чтобы избежать столкновений. Обычно задается функция $\phi(x_t^i)$, которая отклоняет движение в случайном направлении в том случае, если агент i обнаруживает препятствие в направлении x_t^iна расстоянии ближе r .

Для достижения консенсуса между агентами используется протокол локального голосования. Здесь консенсусное управление агентами задается по формуле:

$$u_t^i = \alpha \sum_j b_t^{i,j} (y_t^{i,j} - y_t^{i,i})$$

В этой формуле используются все j из замкнутого множества зашумленных наблюдений о направлении движения своих соседей. Такое множество обозначим как A_t^i для агента a_i в момент времени t.

Здесь через α обозначим величину шага функции протокола управления, $\overline{A_t^i} \subset A_t^i$, величина $b^{i,j} > 0 \; \forall j \in \overline{A_t^i}, b_t^{i,j} = 0$ для остальных пар (i, j).

Введем функцию $g_t^i = q_t^i x_t^i$, тогда положим $y_t^{i,i} = g_t^i + \omega_t^{i,i}$ и $y_t^{i,j} = g_{t-h_t^{i,j}}^j + \omega_t^{i,j}$, $j \in A_t^i$, здесь $\omega_t^{i,i}$ и $\omega_t^{i,j}$ помехи, $0 \leq h_t^{i,j} \leq \overline{h}$ целочисленная задержка, а $\overline{h}$ максимальная задержка.

Обычно $h_t^{i,j} = 0$ и $\omega_t^{i,j} = 0$ для всех пар (i, j) для которых они не определены. И также необходимо добавить условие, что $j \in A_t^i \rightarrow t - h_t^{i,j} > 0$, так как начало работы системы происходит в момент времени $t = 0$.

Что касается динамики изменения направления движения, то оно будет описываться следующим уравнением:

$$x_{t+1}^i = x_t^i + f(u_t^i, x_t^i)$$

В этом уравнении u_t^i – это управление, описанное выше, а функция $f: \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}^3$ необходима для формирования окончательного значения управления, чтобы избежать столкновения объектов.[10]

Таким образом, чтобы агенты достигли консенсуса в группе агентов, необходимо следовать следующему алгоритму решения поставленной задачи:

1. Для начала задается множество задач.

2. Происходит инициация протокола голосования, начинается установление связей между соседями в группе, инициируется первичная коммуникация.
3. Все агенты группы получают информацию о нынешнем направлении движения для всех агентов, которые находятся в «зоне видимости», а также число, характеризующее потенциал движения этих объектов.
4. Используя выше представленные данные, вычисляются значения $y_t^{i,i}$ и $y_t^{i,j}$, и после этого, получаем значение управления u_t^i для каждого объекта роя.
5. Далее происходит переформирование движения x_{t+1}^i по разностной формуле, применяя функцию для избегания столкновений, а также пересчет потенциала движения, исходя из новой априорной информации каждого агента и дополнительной информации, полученной при исследовании соседей и окружающей среды [9].

2.3 Языки программирования агентов

Приведем основные требования, которым должен удовлетворять язык написания мультиагентной системы.

Во-первых, *мультиплатформенность* является одним из главных аспектов при выборе языка. Здесь подразумевается поддержка мобильности агента, то есть обеспечение переносимости кода на различные платформы, а также доступность и работоспособность на различных платформах. Во-вторых, язык должен поддерживать *сетевое взаимодействие*. Третьим, не менее важным требованием является *поддержка символьных вычислений* для коммуникации агентов. Помимо этого, основным, на наш взгляд, свойством является *объектная ориентированность языка*. А также язык должен *удовлетворять требованиям безопасности*, и языковой поддержке свойств агента для построения переговоров между агентами.

Можно выделить наиболее часто используемые языки программирования и самые популярные средства разработки МАС. Приведем необходимый перечень в таблице 2.2.

Таблица 2.2. Средства разработки МАС

Язык программирования	Наименование среды
Java	JADE
JVM (Java 2)	MadKIT
Java, C, C++	AgentBuilder
Java	Cougaar
NetLogo	NetLogo
Visual Basic	VisualBots
Java	MASON
Java, Python, Visual Basic, .NET, C++, J#, C#	REPAST

Самой популярной средой для разработки является среда JADE (Java Agent Development framEwork, так как она обладает рядом преимуществ перед остальными. В их число входит поддержка стандартов FIPA, широкая область применения, возможность расширения на всех уровнях, наличие большого количества плагинов, наличие лицензии LGPL (Lesser General Public License) [50]. Программная среда JADE включает:

- среду выполнения агентов — агенты регистрируются в ней и работают под управлением среды;
- библиотеку классов, которые используются для разработки многоагентных систем;
- набор графических утилит для администрирования и наблюдения за поведением и деятельностью активных агентов.

Среда JADE очень удобна для использования в нашем случае, так как она подключается к любому проекту на языке Java.

2.4 Мультиагентный подход к управлению рисками в виртуальных организациях

В разделе 1.4 данной работы мы подробно описали процесс управления рисками в виртуальных организациях. Нашей задачей в данном разделе является добавить мультиагентный подход к описанному ранее алгоритму. На рисунке 6 мы иллюстрировали схему работы с рисками, на данном этапе рассмотрим каждый из фрагментов схемы более подробно, при этом сформируем несколько этапов организации данного алгоритма.

Этап 1. Идентификация и анализ рисков

Как было описано ранее (пункт 1.4) на данном этапе команда формирует первоначальную карту рисков. Для ее формирования стандартная команда проводит митинги во главе с риск-менеджером, на котором первоначально формируется исходный перечень рисков, после чего он дополняется, исходя из заполненной анкеты от каждого члена команды. Первоначальный список рисков может быть сформирован, исходя из базы знаний команды (компании). Обычно в данный первичный перечень входят некие типовые риски, из которых можно исключить те пункты, которые не относятся к продукту. Формирование перечня типичных рисков в той или иной степени занимает некоторое количество времени, а также имеет большой объем используемой информации. В данном случае целесообразно поручить сбор типичных рисков отдельному агенту, который в течение работы над различными проектами будет формировать базу знаний, состоящую из тех угроз, которые были выявлены. Таким образом, мы минимизируем вероятность допущения ошибки в составлении данного перечня, а также минимизируем временные (а соответственно и денежные) затраты.

В момент анализа и оценки рисков не очень целесообразно привлекать агента, так как программа не может объективно оценить вред для продукта от воздействия того или иного фактора. Поэтому мы предлагаем ввести агента,

который будет формировать итоговый перечень рисков, исходя из анкет команды и итогов встреч.

Этап 2. Выбор стратегии

На этапе выбора стратегии управления тем или иным риском можно привлечь того же агента, который формирует конечный перечень рисков. Таким образом, данный агент получает от команды или риск-менеджера соответствующий перечень стратегий по борьбе с угрозами, и на данном этапе формирует дополненную версию карты рисков.

Этап 3. Распределение задач по менеджменту рисков

В этот момент происходит распределение ответственных за тот или иной риск. Выбор ответственного члена команды делается из перечня всех сотрудников проекта. К параметрам, по которым агент распределения рисков может выбрать исполнителя можно отнести навыки, область работы, загруженность того или иного сотрудника и т. д.

Этап 4. Мониторинг выполнения задач по управлению риском, коммуникация

На данном этапе необходимо производить контроль над итоговой картой рисков. Для этого обычно риск-менеджер совместно с владельцем риска производит актуализацию статусов работы с тем или иным риском. В этом случае мы можем подключить агента по мониторингу статусов рисков, или же группу агентов в зависимости от масштаба проекта. Данный агент будет общаться с владельцами рисков, а также узнавать у них актуальную информацию по состоянию той или иной угрозы. Также на данном этапе целесообразно добавить агента, который будет управлять коммуникациями между сотрудниками и агентами, так как речь идет о виртуальной компании, а организовать общение внутри такой команды является сложным процессом.

Если мы рассмотрим вариант с масштабным проектом и группой агентов мониторинга, то целесообразно распределить агентов мониторинга

по функциям управления проекта и стадиям жизненного цикла. К функциям управления проекта относятся [11]:

- Управление ресурсами, управление стоимостью (Ф1);
- Управление качеством разработки ПО (Ф2);
- Управление временем выполнения (Ф3);
- Управление персоналом (Ф4);
- Управление коммуникациями (Ф5);
- Управление интеграцией проекта (Ф6);
- Управление объемом работ (Ф7).

Опишем каждую функцию в таблице 2.3.

Таблица 2.3 Описание функций управления проектов

Функция управления проектом	Состав функции
Управление ресурсами, управление стоимостью	• *Планирование ресурсов* (Resource Planning) – какие ресурсы в каком количестве нужны для работ; • *Оценка стоимостей* (Cost Estimating) – ресурсов, необходимых для работ ; • *Разработка бюджета* (Cost Budgeting) – бюджетирование – распределение затрат по компонентам проекта; • *Контроль стоимости* (Cost Control) – управление изменениями бюджета;
Управление качеством разработки ПО	• *Планирование качества* (Quality Planning) – определение целей в области качества, стандартов качества и средств для их достижения; • *Обеспечение качества* процесса (Quality Assurance) – плановая, регулярная оценка исполнения – проверка производственных процессов; • *Контроль качества*

	результатов (Quality Control) – мониторинг результатов проекта, определение их соответствия стандартам, выявление и устранение причин несоответствия качества;
Управление временем выполнения	• *Определение состава работ* (Activity Definition); • *Определение взаимосвязей работ* (Activity Sequencing); • *Оценка длительностей работ* (Activity Duration Estimating); • *Составление расписания проекта* (Schedule Development);
Управление персоналом	• *Организационное планирование* (Organizational Planning) – идентификация, документирование, и назначение проектных ролей, обязанностей и структуры отчетности; • *Подбор кадров* (Staff Acquisition) – получение необходимых для проекта человеческих ресурсов, назначение персонала в команду проекта; • *Развитие команды проекта* (Team Development) – повышение производительности труда: индивидуальной и команды в целом (улучшение взаимодействия)
Управление коммуникациями	• *Планирование взаимодействия* (Communications Planning) – определение потребностей участников проекта в информации и планирование информационных потоков; • *Распределение информации* (Information Distribution) –

	регулярное и своевременное обеспечение участников проекта необходимой информацией; • *Оценка исполнения* (Performance Reporting) – сбор и распространение отчетности о текущем состоянии проекта, достигнутом прогрессе и ожидаемых результатах; • *Административное завершение* (Administrative Closure) – создание, распространение (уничтожение) информации, необходимые для формального завершения проекта/фазы
Управление интеграцией проекта	• *Создание плана проекта* (Project Plan Development); • *Исполнение плана проекта* (Project Plan Execution); • *Контроль изменений в проекте* (Integrated Change Control)
Управление объемом работ	• *Инициирование проекта* (Initiation) – формальное принятие решения о начале проекта (следующей фазы проекта); • *Планирование объема работ* (Scope Planning) – разработка документа, описывающего объем работ; • *Формализация объема работ* (Scope Definition) – декомпозиция всего объема работ (основных необходимых результатов) на мелкие, измеримые задачи; • *Верификация* (Scope Verification) – подтверждение объема работ – формальная проверка приемлемости результатов работы; • *Управление изменениями* объема работ (Scope Change

	Control) – контроль и утверждение изменений;

Введем матрицу агентов (Таблица 2.4) и следующие обозначения агентов:

$A_{i,j}$ – агент, у которого индекс i отвечает за номер стадии жизненного цикла программного продукта, а индекс j – номер функции управления программным продуктом.

Наличие агента на пересечении строки и столбца обозначено символом «+», его отсутствие соответственно символом «-». Отсутствие агента можно аргументировать тем, что при контроле данного пункта необходимо полное участие человека.

Таблица 2.4. Матрица агентов

№ стадии ЖЦ	Содержание этапа ЖЦ	Ф1	Ф2	Ф3	Ф4	Ф5	Ф6	Ф7
1	Исследование концепции	+	+	+	+	+	+	+
2	Исследование системы	+	+	+	+	+	+	+
3	Требования	+	+	+	+	+	+	+
4	Разработка проекта	+	+		+	+	+	+
5	Внедрение	+	+		+	+	+	+
6	Установка	+	+		+	+	+	+
7	Эксплуатация и поддержка	+	+		+	+	+	+
8	Сопровождение	+	+		+	+	+	+
9	Вывод из эксплуатации	+	+		+	+	+	+

Таким образом, исходя из данной матрицы мы видим, что можно задействовать 57 агентов. Для удобства работы с агентами, разделим их по группам функций управления программным продуктом. Итого, 7 групп агентов, отвечающих отдельной функции.

Внутри каждой группы агенты имеют жесткую связь, то есть взаимодействие 100%. К таким агентам относится множество агентов:

$$\{A_{i,j}, \text{где } i = 1 \dots 9, j \text{ фиксировано}\}.$$

Агенты, отвечающие отдельной стадии жизненного цикла имеют полуфункциональные связи – взаимодействие 50-70%:

$$\{A_{i,j}, \text{где } j = 1 \dots 7, i \text{ фиксировано}\}$$

Также можно определить слабофункциональные связи – взаимодействие 20-30%, но в данном случае это будет взаимодействие группы агентов мониторинга с другими агентами системы.

Именно в случае сложного проекта и использовании большего количества агентов, можно использовать групповой (роевой) интеллект.

Этап 5. Актуализация рисков проекта

Данное мероприятие актуализации карты рисков должно идти в течение всего проекта, для того, чтобы понимать нынешнее состояние продукта. Актуализацию перечня рисков можно поручить отдельному и самому основному агенту, который будет хранить в себе итоговую и полную карту рисков, и производить ее актуализацию.

Полный перечень агентов и описание их функциональных возможностей будет приведено в главе 3. На данном этапе нашей целью было показать, что функции, которые в стандартных компаниях должен выполнять риск-менеджер, могут быть поручены определенным агентам для того, чтобы избежать ненужных трудозатрат, денежных, временных затрат, а также для того, чтобы повысить эффективность работы с большим объемом информации, что обычному человеку не всегда удается сделать.

2.5 Выводы к Главе 2

В главе 2 были описаны основы теории мультиагентных систем. Была приведена теоретическая часть, описывающая классификации агентов, выделяющая основные определения данной сферы, представляющая различные подходы к формированию архитектуры мультиагентных систем. Также был описан подход к роевому управлению, который может быть применен к разрабатываемой системе. Были описаны особенности языков

программирования агентов. А также была проведена аналитическая работа по применению мультиагентного подхода к управлению рисками в виртуальных организациях. В данной главе были описаны стадии процесса риск-менеджмента и возможность добавления соответствующих агентов на определенные стадии.

Таким образом, была подготовлена вся теоретическая и аналитическая часть для разработки системы мониторинга проектных рисков с использованием мультиагентных технологий для виртуальных компаний.

Глава 3. Разработка системы управления рисками в виртуальных организациях с использованием мультиагентных технологий

В предыдущей главе были рассмотрены различные классификации мультиагентных систем. Исходя из представленных выше определений, система управления рисками будет разработана как *децентрализованная мультиагентная система*, состоящая из *делиберативных агентов* (Deliberative Agents). Система создана для решения определенной задачи, а именно для формирования и мониторинга карты рисков для различных проектов. Делиберативная архитектура удобна тем, что она содержит точную символическую модель мира агентов, которые принимают решения на основе логического вывода. Такая архитектура дает возможность использования строгих формальных методов и хорошо отработанных технологий искусственного интеллекта для представления знаний в определенной символьной форме и переноса их в разрабатываемую агентную систему.

В пункте 2.4 данной работы были описаны этапы процесса формирования и управления карты рисков. Таким образом, исходя из описанных стадий, выделим 6 агентов системы:

- агент анализа рисков – формирует первичную карту рисков и имеет базу знаний с предыдущих проектов;
- агент распределения рисков – находит наиболее оптимального владельца риска из участников команды проекта;
- агент мониторинга рисков (или мета-агент группы взаимосвязанных функциональных агентов) – проводит постоянный мониторинг карты рисков в течение всего проекта;
- агент актуализации карты рисков – производит регулярное обновление карты, исходя из актуальной информации по состоянию проекта;
- агент для уведомлений – производит рассылку уведомлений команде проекта;

- агент организации встреч – производит планирование митингов команды.

На рисунке 3.1 представлена схема взаимодействия данных агентов и потоки данных, передаваемых между объектами системы. В Приложении 4 данной работы представлен полный бизнес-процесс работы мультиагентной системы.

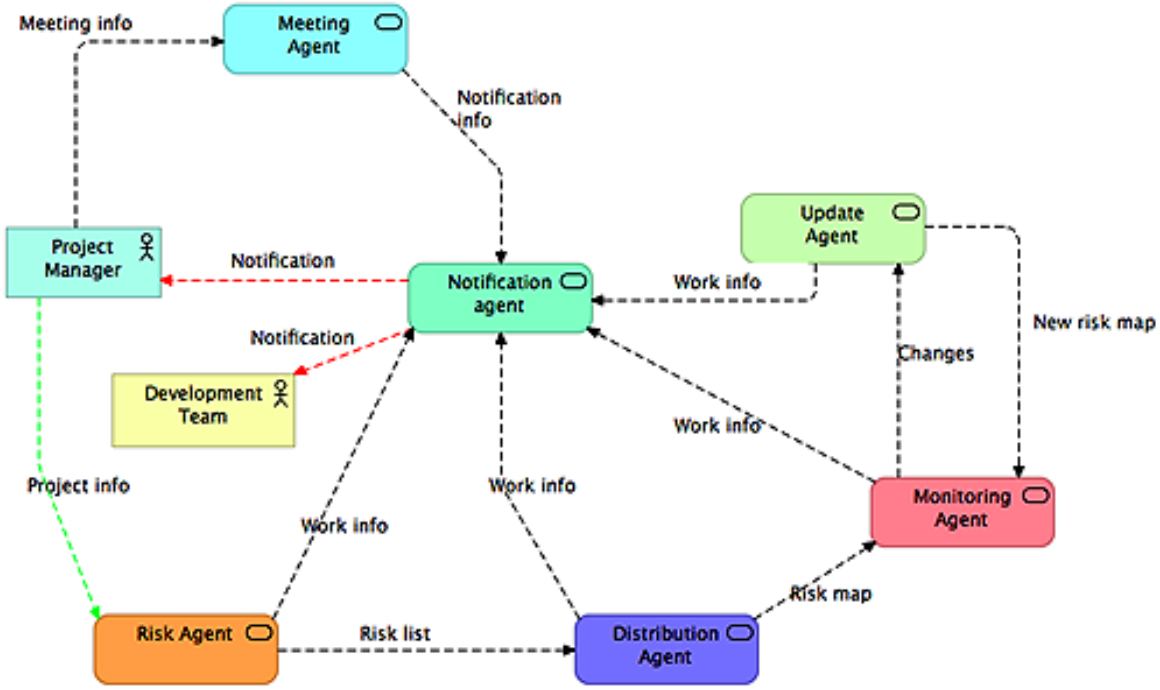

Рисунок 3.1. Схема взаимодействия агентов

3.1 Описание используемых технологий

В качестве языка программирования самих агентов был использован язык Java и агентная платформа (среда) JADE (Java Agent Development Framework). Экранные формы начала работы агентов в данной платформе представлены в Приложении 3. Помимо этого, для реализации системы были использованы следующие программные технологии:

- HTML (HyperText Markup Language) – для формирования интерфейса прототипа Web-приложения;
- Apache HTTP-сервер – для подключения внешних модулей для предоставления данных;

- CSS (Cascading Style Sheets) – для задания стилей интерфейса Web-приложения;
- JavaScript – для придания интерактивности страниц Web-приложения;
- PHP (Hypertext Preprocessor) – для реализации всего функционала приложения, не касающегося алгоритмов функционирования агентов;
- MySQL – свободная реляционная система управления базами данных.

Схема взаимодействия технологий изображена на рисунке 3.2.

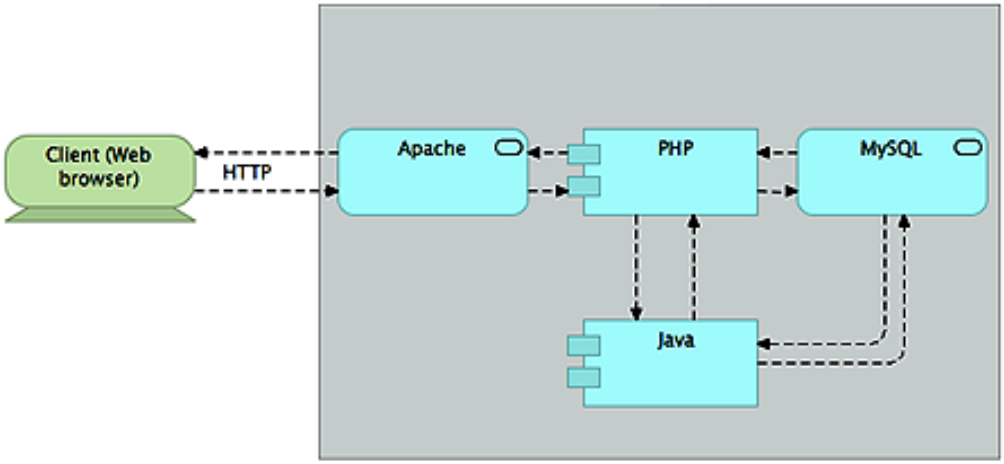

Рисунок 3.2. Взаимодействие технологий

3.2 Описание структуры базы данных

Схема базы данных, используемая в системе представлена в Приложении 1 к данной работе.

Концептуально-функциональный прототип Web-приложения имеет связь с базой данных, в которой хранится вся информация, необходимая для функционирования мультиагентной системы. Благодаря базе данных системы, можно получить всю информацию, которая соответствует некоему проекту. Опишем состав каждой таблицы используемой базы данных:

1. *Таблица User* (содержит информацию о пользователях системы):
    a. *id* – идентификационный номер пользователя;
    b. *name* – имя пользователя;

c. *login* – имя, под которым пользователь зарегистрирован в системе;
d. *password* – пароль пользователя для системы*;*
e. *role* – роль пользователя в проекте (admin, manager, implementer, customer),различные роли имеют различные приоритеты в системе;
f. *timezone* – часовой пояс, в котором живет работник*;*
g. *employment* – занятость сотрудника, которая вычисляется посредством сложения количества рисков, которые назначены пользователю, максимальное значение этого поля зависит от проекта.

2. *Таблица Project* (содержит информацию по конкретному проекту):
   a. *id* – идентификационный номер проекта;
   b. *name* – название проекта;
   c. *descr* – описание проекта, его цели и общие задачи;
   d. *project_type* – тип проекта (например, разработка ПО, техническая поддержка и т.д.);
   e. *status* – поле, отображающее статус проекта (*requirements* – идет сбор требований, *approving* – проект находится в процессе одобрения, *approved* – проект одобрен, *process* – проект находится в разработке, *completed* – проект завершен);
   f. *customer_id* – идентификационный номер заказчика проекта;
   g. *create_data* – дата создания проекта*;*
   h. *complete_data* – дата завершения проекта;
   i. *approving_stage* – стадия одобрения проекта;
3. *Таблица Project_member* (таблица, в которой отображены участники проекта):
   a. *project_id* – идентификационный номер проекта;
   b. *user_id* – идентификационный номер пользователя, принимающего участие в этом проекте.

4. *Таблица Skill* (в этой таблице приведен список навыков для пользователей):
    a. *id* – идентификационный номер навыка;
    b. *name* – название навыка.
5. *Таблица User_skill* (таблица, в которой хранятся уровни владения навыками пользователей):
    a. *user_id* – идентификационный номер пользователя;
    b. *skill_id* - идентификационный номер навыка*;*
    c. *level* – уровень владения навыком.
6. *Таблица Project_requirement (*содержит информацию по требованиям к проекту, заполняется при создании проекта менеджером и заказчиком, может корректироваться в течение проекта, если в проекте используется гибкая методология управления):
    a. *id* – идентификационный номер требования;
    b. *project_id* – идентификационный номер проекта*;*
    c. *name* – название требования;
    d. *possibility* – возможность выполнения данного требования (1 – возможно, 0 – не возможно, $\frac{1}{2}$ – возможно частично), заполняется менеджером*;*
    e. *comment* – комментарии по выполнению;
    f. *changed* – обозначение изменения в требовании;
7. *Таблица Meeting* (таблица для агента планирования встреч):
    a. *id* – идентификационный номер митинга;
    b. *name* – тема митинга*;*
    c. *descr* – описание митинга;
    d. *date* – дата планируемого митинга*;*
    e. *status* – статус митинга;
    f. *appointed* – поле, отвечающее за утверждение митинга;
8. *Таблица Meeting_participants* (содержит информацию по участникам митингов):

a. *meeting_id* – идентификационный номер митинга*;*
b. *user_id* – идентификационный номер участника митинга;
c. *possibility* – возможность пользователя принять участие в митинге (заполняется агентом планирования встреч из ответа на анкету)*;*
d. *comment* – комментарии ко времени;
e. *owner* – отметка для создателя митинга (1 – создатель, 0 – не создатель);

9. *Таблица Notification* (содержит сообщения, посланные агентом для уведомлений):
   a. *id* – идентификационный номер уведомления;
   b. *user_id* – кому направлено уведомление*;*
   c. *date* – дата отправки уведомления;
   d. *text* – текст уведомления;
10. *Таблица risk* (содержит в себе карту рисков проектов):
   a. *id* – идентификационный номер риска;
   b. *project_id* – идентификационный номер проекта, к которому относится риск;
   c. *name* – наименование риска;
   d. *company* – информация об организации, ответственной за риск;
   e. *content* – содержание риска;
   f. *type* – тип риска;
   g. *descr* – полное описание риска;
   h. *reasons* – причины возникновения риска;
   i. *effect* – последствия от наступления риска;
   j. *possibility_coeff* – коэффициент вероятности риска из таблицы 1.3;
   k. *effect_coeff* – коэффициент последствий наступления рисковых событий из таблицы 1.4;

l. *overall_coeff* – оценка риска, равна произведению коэффициентов вероятности и последствия. Показывает общую оценку риска с предположением того, что данный риск не управляется и не контролируется;
m. *responsible_user_id* – идентификационный номер владельца риска, ответственного за его мониторинг;
n. *impact* – описание воздействия на риск;
o. *impact_coeff* – информация по стоимости воздействия на данный риск;
p. *reglamentory_documents* – перечень регламентирующих документов;
q. *final_possibility_coeff* – вероятность наступления риска после применения мер по его мониторингу и контролю;
r. *final_effect_coeff* – коэффициент последствий риска после применения мер по его мониторингу и контролю;
s. *final_overall_coeff* – оценка риска, равна произведению *final_possibility_coeff* и *final_effect_coeff*.

3.3 Описание алгоритмов функционирования агентов системы

3.3.1 Агент организации встреч

Агент планирования митингов активируется любым участником команды, в том случае, когда какому-либо сотруднику проекта необходимо собрать встречу по проекту. Для этого организатору необходимо заполнить форму, проиллюстрированную на рисунке 3.3.

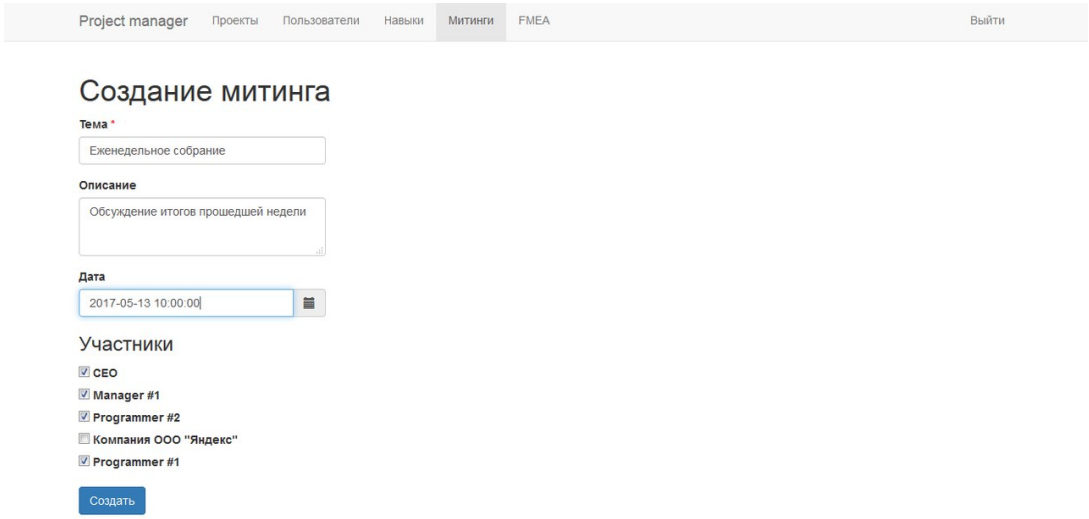

Рисунок 3.3. Форма создания митинга

После сохранения введенной информации, данные добавляются в таблицу *meeting*, агент организации встреч заполняет таблицу *meeting_participants* и всем участникам будущего митинга рассылается анкета, изображенная на рисунке 3.4:

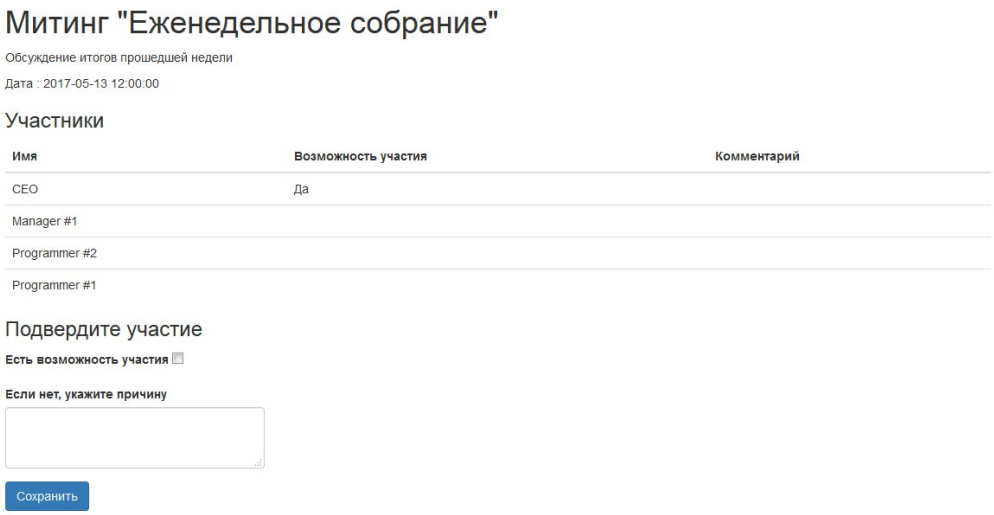

Рисунок 3.4. Анкета планирования митинга

Обозначим процедуру формирования таблицы *meeting_participants* - *GetTableParticipants.*

Ниже приведем алгоритм работы агента организации встреч. В случае работы данного агента, вмешательство в его работу со стороны членов команды предусмотрено только в том случае, если назначенные время и дата митинга не подходят участникам команды. В этом случае возникает

конфликт в работе агента, и ему необходимо обратиться к создателю риска для его разрешения.

Procedure *meeting_agent*

GetMeetingTable;

Всем разработчикам, находящимся в столбце *user_id* агент рассылает анкету;

Агент считывает анкеты участников;

*GetTableParticipants***;**

If все участники ответили «Да» **then**

Агент передает информацию *notification_agent*

End if

Else

агент дает команду *notification_agent* отправить создателю митинга сообщение со списком ответов («Внесите изменения в список участников»);

агент считывает ответ создателя;

агент вносит изменения в базы *meeting* и *meeting_participants*;

агент дает команду *notification_agent* об отправке времени и даты митинга;

End else

End procedure.

Таким образом, на *выходе* агент выдает конечный список участников, время и дату митинга.

3.3.2 Агент анализа рисков

Данный агент отвечает за создание первичной карты рисков (часть таблицы *risk*). На первом этапе работы агент получает команду от менеджера о запуске нового проекта, а также информацию о том, к какой категории относится данный проект. К категориям можно отнести разработку мобильного приложения, разработку Web-приложения, сопровождение системы, техническая поддержка и т.д. В каждой компании может быть

введена собственная классификация. Таким образом, данный агент имеет собственную базу знаний (таблица *risk*), основанную на предыдущем опыте компании при работе с различными видами проектов. Исходя из нее, агент может предложить первичный перечень возможных рисков, выбрав нужный *project_type* из таблицы *project*.

Таким образом, *на вход* данный агент получает параметр *project_type*, а по завершении работы агент формирует перечень рисков проектов и первую версию карты рисков, то есть без назначения риска на конкретного исполнителя(Таблица 1.3, 1.4, 1.5). Процедуру вывода данной таблицы обозначим *GetFirstRiskMap*.

Процедуру формирования списка рисков из базы знаний обозначим *GetRiskBase(project_type)*.

В процессе работы данного агента конфликтных ситуаций не происходит.

Таблица с картой рисков представлена на рисунке 3.5 и 3.6.

Карта рисков

Создать

							Общая оценка риска		
Наименование	Организация	Событие	Вид риска	Описание	Причины	Последствия	Вероятность	Последствия	Оценка
Соответствие проекту	Менеджмент	Отклонение от изначальных требований	Планирование	Несоответствие требованиям проекта, прописанным в Техническом задании	Отсутствие налаженной коммуникации между командой и менеджером, а также менеджером и заказчиком	Изменение сроков и бюджета, неактуальность итогового функционала	0.50	0.50	0.25
Организационная стабильность	Исполнитель	Стабильность менеджмента, вероятность реструктуризации управления	Контроллинг	Непредвиденная реорганизация структуры менеджмента	Нечеткая организационная структура, личностные взаимоотношения	Потеря времени, бюджета	0.10	1.00	0.10
График разработки	Команда	Несоблюдение сроков	Планирование	Выход за оценку задач, вылет по срокам	Неправильная оценка	большие трудозатраты и временные затраты, потеря	0.50	0.50	0.25

Рисунок 3.5. Карта рисков проекта (часть 1)

				Общий риск			
Владелец процесса управления риском	Воздействие на риск	Стоимость воздействия	Регламентирующие документы	Вероятность	Последствие	Оценка	
Менеджер Антон	наладить коммуникацию	Высокий	ТЗ, Устав проекта	0.10	0.10	0.01	
Генеральный директор	Документированная иерархическая структура, быстро обучение новых сотрудников	Средний	Устав проекта, устав компании	0.10	0.80	0.08	
Разработчик Анатолий	Корректировка плана работ, митинги, ретроспективы	Средний	План-график	0.20	0.40	0.08	

Рисунок 3.6. Карта рисков проекта (часть 2)

Экранные формы создания описания риска приведены на рисунках 3.7 и 3.8.

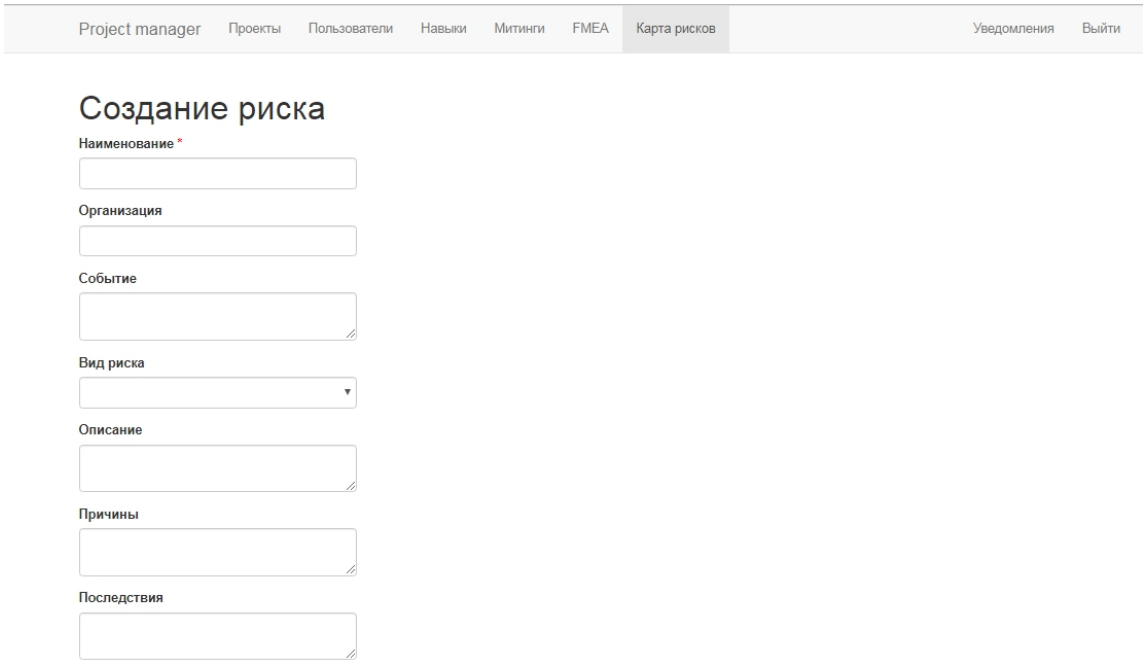

Рисунок 3.7. Создание описания риска (часть 1)

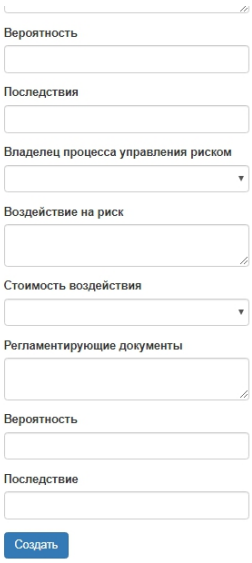

Рисунок 3.8. Создание описания риска (часть 2)

Procedure *risk_analysis*

GetRiskBase(project_type);

GetFirstRiskMap;

WHILE команда не утвердила перечень рисков **DO**

открыть карту рисков для заполнения командой;

сформировать перечень рисков;

отдать перечень рисков на утверждение команде

END DO

GetFirstRiskMap;

Направить запрос менеджеру на обновление базы рисков;

If менеджер выбрал риски для переноса в базу **then**

дополнить базу рисков

End if

End procedure.

3.3.3 Агент мониторинга рисков

Агент мониторинга рисков отвечает за управление картой рисков в ходе всего проекта (таблицей *risk*). Данный мета-агент может состоять из группы агентов, о чем было упомянуто ранее. В данном случае опишем алгоритм работы в общем виде. В случае группы агентов необходимо добавить параметр, определяющий стадию жизненного цикла проекта.

Данный мета-агент получает на вход актуальную карту рисков. После окончания работы агент должен передать перечень изменений в данной карте. Агент получает информацию посредством коммуникации с командой. Для удобства добавим в карту категорию рисков – параметр *type* из таблицы *risk*. Данный параметр необходим для удобства управления картой рисков и корректной организации информации об угрозах. Владельца риска обозначим как *risk_owner* (данные из поля *responsible_user_id* в таблице *risk* для данного *project_id*). Процедуру формирования перечня изменений обозначим как *GetChanges*.

В случае, если в таблицу *project_requirement* были внесены изменения, то агент также незамедлительно начинает мониторинг карты рисков.

В ходе функционирования данного агента конфликтные ситуации не происходят.

Procedure *monitoring_agent*

WHILE все *type* не просмотрены **DO**

Выбрать ранее не просмотренный *type*;

WHILE все риски из *type* не просмотрены **DO**

Выбрать ранее не просмотренный риск;

Направить агенту *notification_agent* запрос на уведомление *risk_owner* («Необходимо актуализировать информацию о риске № *risk_id*»);

Получить информацию от *risk_owner*;

If состояние риска изменилось **then**

Добавить информацию в перечень для актуализации

End if

END DO

END DO

Обратиться к менеджеру через *notification_agent* («Появились ли новые риски в проекте? Если да, то внесите их в карту»)

Ожидание ответа менеджера;

GetChanges;

Направить перечень изменений агенту *update_agent*

End procedure.

В случае использования мета-агента и группы агентов мониторинга, которые обозначены в таблице 2.4, могут возникнуть следующие конфликтные ситуации:

1. конфликт внутри группы агентов с полуфункциональными связями:

$$\{A_{i,j}, \text{где } j = 1 \dots 7, i \text{ фиксировано}\}$$

2. конфликт внутри агентов с жесткими связями:

$$\{A_{i,j}, \text{где } i = 1 \dots 9, j \text{ фиксировано}\}.$$

Общую схему разрешения конфликтов в группе приведена на рисунке 3.9.

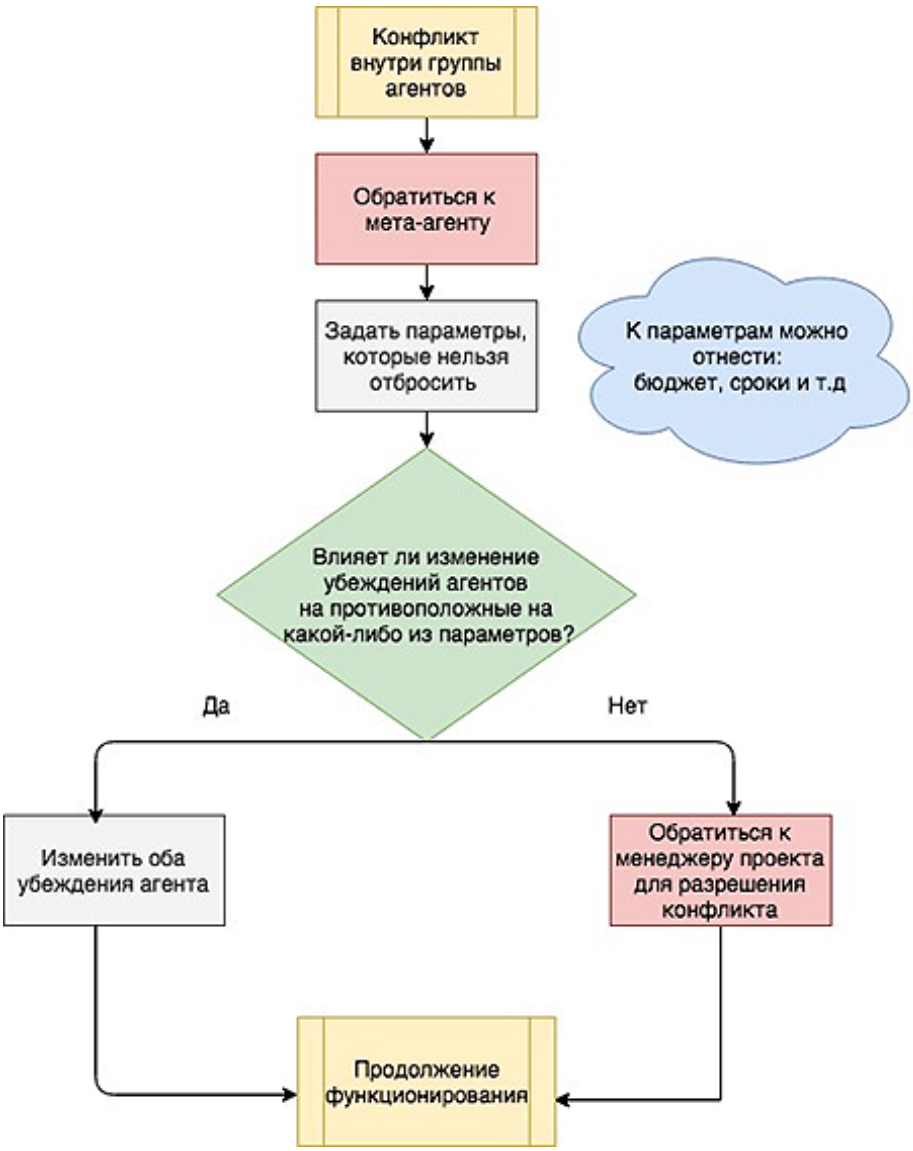

Рисунок 3.9. Разрешения конфликта в группе агентов

Конфликты в первой ситуации могут произойти из-за формирования неточных требований, использования неактуальной информации и т.д. Но они происходят в рамках одной из стадий жизненного цикла проекта.

Конфликты второго случая могут произойти в той ситуации, когда при работе над проектом на стадии *j* изменилась информация по стадиям от *1* до *j-1*. Это может произойти как и в случае водопадной модели жизненного цикла, так и в случае использования гибких методологий. В случае гибких методологий такие конфликты происходят постоянно и необходимо под них адаптироваться.

3.3.4 Агент актуализации карты рисков

Необходимость ввода данного агента возникла из-за того, что масштабы проекта могут быть различными и полностью загрузить мета-агента мониторинга рисков не целесообразно. Тем более, если будет необходимость ввода группы агентов, то необходимо, чтобы вся информация была собрана в одном месте и по всем стадиям жизненного цикла проекта.

Принцип работы агента строится следующим образом: на вход агент получает перечень изменений от мета-агента мониторинга рисков. Процедуру получения данной информации обозначим *GetListChange*s. На выходе агент отдает актуализированную карту рисков. Процедуру ее формирования обозначим *GetActualRiskMap*.

В процессе работы данного агента конфликтных ситуаций не происходит.

Procedure *update_agent*

*GetListChange*s;

WHILE все изменения не обработаны **DO**

Поиск старой версии риска в таблице *risk*;
Изменение информации о риске

END DO

GetActualRiskMap

End procedure.

3.3.5 Агент для уведомлений

Агент такого типа собирает информация о работе остальных агентов, и после завершения их функционирования, формирует соответствующие уведомления. Процедуру отправки уведомления обозначим *SendNotification*.

В ходе работы данного агента конфликтных ситуаций не происходит.

Procedure *notification_agent*

If один из агентов закончил работу **then**

SendNotification всем *user_id* связанным с *project_id*

End if

If получена команда от любого агента об отправке уведомления **then**

SendNotification всем указанным *user_id*

End if

End procedure.

Перечень возможных уведомлений представлен в таблице 3.1 также агенты в качестве параметра обращения к данному агенту могут направить произвольный текст, который описан в алгоритме агентов.

Таблица 3.1. Перечень уведомлений

Событие	Текст уведомления
Агент анализа рисков закончил работу	Агент анализа рисков закончил работу, вы можете ознакомиться с перечнем рисков
Агент распределения рисков закончил работу	Агент распределения рисков закончил работу, вы можете ознакомиться с картой рисков
Агент распределения рисков не может выбрать исполнителя	Просьба менеджеру добавить владельца риска вручную
Агент мониторинга рисков закончил работу	Агент мониторинга рисков закончил работу
Агент актуализации карты рисков закончил работу	Карта рисков может быть изменена, ознакомьтесь с ней
Агент организации встреч закончил	Митинг состоится *Дата* в *Время*

работу	
Другие события	Текст может быть назначен передающим агентом

3.3.6 Агент распределения рисков

Благодаря функционированию данного агента происходит итоговое формирование полной карты рисков проекта. Основным этапом работы является поиск владельца риска из перечня пользователей, относящихся к команде данного проекта. Процедуру запроса первичной карты рисков от агента анализа рисков обозначим *GetFirstRiskMap.* Процедуру, благодаря которой агент получает информацию о составе команды проекта обозначим *GetTeam.*

На входе агент получает первичную карту рисков от агента анализа рисков, помимо этого для каждого риска агент получает значение *skill_level* – уровень необходимых навыков, который будет использоваться для выбора исполнителя риска. Таким образом, по окончании работы алгоритма данного агента мы получаем итоговую карту рисков всего проекта. Процедуру формирования и вывода данной карты обозначим *GetRiskMap*.

Данный агент имеет свою базу знаний для принятия решений по выбору исполнителя. Данная база знаний в начале работы агента пуста. После каждого обращения к менеджеру, агент запоминает принцип выбора исполнителя менеджером, и в последующих подобных ситуациях может не обращаться к менеджеру проекта.

Помимо этого, добавим процедуру вывода списка исполнителей из таблицы *user*, отсортированного по степени занятости – *GetUserList*.

Также при выборе исполнителя в случае спорной ситуации, когда в списке возможных исполнителей два и более человека, будем использовать параметр *employment* из таблицы *user*.

Алгоритм данного агента, реализованный в агентной платформе JADE приведен в Приложении 2.

В процессе функционирования агента распределения рисков могут произойти следующие конфликты:

1. В списке исполнителей несколько пользователей. Этот конфликт агент разрешает сам, выбирая из них наименее занятого члена команды.
2. В списке исполнителей нет пользователей. Данный конфликт агент разрешает посредством обращения к менеджеру проекта. После этого, он вносит действие менеджера в базу знаний.

Procedure *risk_distribution*

GetFirstRiskMap;

GetTeam;

WHILE все риски не распределены **DO**

GetUserList;

WHILE не просмотрены все исполнители **DO**

If *skill_level* исполнителя >= *skill_level* риска **then**

Добавить пользователя в список возможных исполнителей

End if

If у риска 2 и более возможных исполнителей **then**

выбрать исполнителя с наименьшим значением в поле *employment*

End if

END DO

If список возможных исполнителей пуст **then**

поиск решения в базе знаний агента

End if

If решение в базе знаний найдено **then**

добавить исполнителя в список

End if

Else

направить запрос *notification_agent* на уведомление менеджера выбрать исполнителя вручную

End else

END DO

GetRiskMap

End procedure.

3.4 Выводы к Главе 3

В данной главе был описан мультиагентный подход к созданию функционально-концептуального прототипа Web-приложения для управления рисками в виртуальных организациях. Помимо этого, были описаны технологии, которые использовались при создании данного Web-приложения, а также описана структура используемой базы данных системы.

Для каждого агента был приведен алгоритм его функционирования, а также было описано то, как эти агенты функционируют и взаимодействуют между собой.

При реализации и функционировании системы невозможно исключить все конфликтные ситуации, поэтому одной из основной задач данного раздела было проанализировать возможные конфликты агентов при их взаимодействии между собой и при работе каждого агента отдельно. Была представлена схема решения конфликтов.

Также в данной главе были представлены интерфейсы концептуально-функционального прототипа Web-приложения управления рисками для виртуальных компаний.

Заключение

Проблема управления является на данный момент одной из самых актуальных. Очень часто в компаниях любого типа риск-менеджменту не уделяется должного внимания. Данное упущение ведет за собой большой перечень последствий, таких как увеличение бюджета, затрачиваемого на проект, увеличение сроков, и в худшем случае потеря актуальности разрабатываемого продукта.

Данная тема работы была выбрана исходя из актуальности и роста популярности виртуальных организаций, в которых риск-менеджменту необходимо уделять еще более пристальное внимание, исходя из особенностей данных компаний. Управление рисками в данных организациях более сложное и требует больших временных и ресурсных затрат. В ходе работы была проведена аналитическая работа для определения отличий классических и виртуальных организаций, исходя из чего, можно сделать вывод о сложности мониторинга рисков.

Помимо этого, набирает рост и интерес к мультиагентным технологиям, которые могут стать решением многих сложных проблем. Для решения объемных задач, человеческих ресурсов может быть недостаточно, человек физически не сможет учесть абсолютно всех факторов для решения проблем, именно для этого необходимо вводить агентов, которые могут обучаться и работать с большим объемом данных.

Виртуальные организации – это новый вид организаций, на данный момент только набирающий популярность. Также очень актуальным является создание и освоение мультиагентных систем. Был выполнен обзор теоретических работ, чтобы приблизиться к пониманию мультиагентного подхода.

В ходе данной работы были достигнуты следующие результаты:

1. Проведена аналитическая работа по изучению работы и организации процессов в виртуальных компаниях.

2. Изучены особенности и методологии управления рисками в IT-компаниях.
3. Проведена аналитическая работа по адаптации карты рисков под деятельность в виртуальных организациях.
4. Изучены и описаны особенности работы с мультиагентными технологиями.
5. Изучена платформа JADE для программной реализации агентов.
6. Применен и описан мультиагентный подход к системе мониторинга рисков для виртуальных компаний, выделен перечень необходимых агентов.
7. Разработаны алгоритмы действий 6 агентов и схема взаимодействия между агентами.
8. Разработаны схема базы данных с учетом мониторинга рисков и прототип Web-сервиса системы с учетом карты рисков.
9. Реализован концептуально-функциональный прототип системы управления рисками для виртуальных компаний.
10. Реализованы алгоритмы 6 агентов с использованием агентной платформы JADE.

Данная сфера деятельности является передовой в текущий период времени, и на некоторые вопросы ответов не найдено, что может являться предметом дальнейших исследований.

Список литературы

1. Амелин К.С., Граничин О.Н., Кияев В.И., Корявко А.В. Введение в разработку приложений для мобильных платформ. – Санкт-Петербург: BBM, 2011. – 535 с.
2. Амелина, Н.О., Задача достижения дифференцированного консенсуса при стоимостных ограничениях/ Н.О. Амелина, Ю.В. Иванский // Вестник СПбГУ. Сер. 1: Математика. Механика. Астрономия, 2015. Т.2(60). Вып. 4. С. 3–14.
3. Амелина Н.О., Амелин К.С., Граничин О.Н., Кияев В.И. Развитие нефтегазовых комплексов и сетей: мониторинг и мультиагентное управление // В сб. материалов V научно-практической конференции «Суперкомпьютерные технологии в нефтегазовой отрасли. Математические методы, программное и аппаратное обеспечение», Москва, февраль 2015, С. 17- 21.
4. Глазкова, И.Н. Риск-менеджмент как механизм повышения конкурентоспособности предпринимательских структур в условиях нестабильной среды. Автореферат. - СПб: Изд-во СПбГЭУ, 2015. - 19 с.
5. Граничин, О.Н., Информационные технологии и системы в современном менеджменте/ О.Н. Граничин, В.И. Кияев – СПб: Издательство BBM. – 2014. – 897 с.
6. Грузенкин, Д. В. Области применения мультиагентных систем. Обзор / Д.В. Грузенкин, Е.А. Карпова, Е.Д. Кулаков //Новая наука: От идеи к результату. – 2016. – №. 12-3. – С. 61-64.
7. Гайдук, А.Р. Распределенные системы планирования действий коллективов роботов/ А.Р. Гайдук, И.А. Каляев, С.Г. Капустян, М.: Янус-К. 2002. — 292 с
8. Годлевский, В. Е. Применение метода анализа видов, причин и последствий потенциальных несоответствий (FMEA) на различных этапах жизненного цикла автомобильной продукции / В.Е. Годлевский,

А.Я. Дмитриев, Г.Л. Юнак. /Под ред. В.Я. Кокотова. — Самара: Перспектива, 2002. — 160 с.

9. Граничин, О.Н. Мониторинг и мультиагентное управление/ О.Н. Граничин, В.И. Кияев // В сб. материалов Тринадцатой Всероссийской конференции «Преподавание информационных технологий в Российской Федерации», АПКИТ, 14-15 мая 2015 г., Пермь, с. 90-91
10. Ерофеева В.А., Управление роем динамических объектов на базе мультиагентного подхода/ В.А. Ерофеева, Ю.В. Иванский Ю, В.И. Кияев // Компьютерные инструменты в образовании – 2015, № 6 – с. 36-44.
11. Кияев, В.И. О терминологии и требованиях международного стандарта качества разработки программного обеспечения. //В сб. «Системное программирование» (под ред. проф. А.Н.Терехова), С.-Петерб. ун-та, 2004. — с. 311-334.
12. Кияев, В.И. Интеллектуальный CRM на базе мультиагентного подхода/ В.И. Кияев, Р.В. Герасимов // В сб. «Стохастическая оптимизация в информатике», 2012.
13. Кияев, В.И. Стандартизация, метрология и качество разработки программного обеспечения и информационных технологий. — СПб: Изд-во СПбГЭУ, 2016. — 475 с.
14. Кияев, В.И. Мониторинговые системы безопасности на базе мультигентного подхода / В.И. Кияев, А.С. Шкарбан // В сб. Докладов IX Санкт-Петербургской региональной конференции «Информационная безопасность регионов России" (ИБРР-2015), СПб, 28-30 октября 2015 г. – с. 93.
15. Муравьев, Е.В. Управление рисками распределенной разработки программного обеспечения // Экономика России в современных условиях: пути инновационного развития и повышения конкурентоспособности Сборник научных трудов по итогам всероссийской научно-практической конференции молодых ученых

Санкт-Петербургского государственного экономического университета. Под ред. Е.А. Горбашко. Санкт-Петербург, 2017. С. 446-450

16. Николаенко, В.С. Внедрение риск-менеджмента в ИТ-проекты // Государственное управление. Электронный вестник. 2016. №54 С.63-88.
17. Трофимов, В.В. Методологические основы управления проектами виртуальных предприятий / В.В. Трофимов, И.Г. Горбунов – СПб.: Изд-во СЗТУ, 2007. – 174 с.
18. Трофимов, В. В., Ильина О. П., Кияев В. И, и др. Информационные технологии в 2 т. Том 2: учебник для академического бакалавриата / отв. ред. В. В. Трофимов. — Москва: Издательство Юрайт, 2017. — 390 с. — (Серия: Бакалавр. Академический курс).
19. Разработка приложений для мобильных интеллектуальных систем на платформе Intel Atom /К.С. Амелин, Н.О. Амелина, О.Н. Граничин, В.И. Кияев, СПб., BBM, 2012 – 220с.
20. Радченко, И.А. Интеллектуальные мультиагентные системы: учебное пособие / И.А. Радченко, Балт. гос. техн. ун-т. — СПб. – 2006. — 88 с.
21. Скобелев, П.О. Открытые мультиагентные системы для оперативной обработки информации в процессах принятия решений // Автометрия. – 2002. № 6. — с. 45-61
22. Тарасов, В.Б. Агенты, многоагентные системы, виртуальные сообщества: стратегическое направление в информатике и искусственном интеллекте// Новости искусственного интеллекта. – 1998. – № 2. — с. 5-63.
23. Товб, А.С. Управление проектами. Стандарты, методы, опыт/ А.С. Товб, Г.Л. Ципес – Олимп – Бизнес, 2005. – 240 с.
24. Уорнер, М. Виртуальные организации. Новые формы ведения бизнеса в XXI веке/М. Уорнер, М. Витц, М. – Добрая книга, 2005. – 296с.

25. Уокер Ройс, Управление проектами по созданию программного обеспечения, изд. Лори-2007г, 448с.
26. Роберт Т. Фатрелл, Дональд Ф. Шафер, Линда И. Шафер, Управление программными проектами. Достижение оптимального качества при минимуме затрат, изд. Вильямс-2004г., 1136с.
27. Чекинов, Г.П. Применение технологии многоагентных систем интеллектуальной поддержки принятия решения / Г.П. Чекинов, С.Г. Чекинов // Системотехника, 2003. – №1
28. Хусаинова, А.Т. Понятие, сильные и слабые стороны распределенных команд и их отличие от традиционных команд проекта // Вестник КазНУ, Сер. Экономическая, 2015. № 2.- С.165-169
29. Мир управления проектами. /Под ред. Х. Решке, Х. Шелле. – М.: Аланс, 1993. – 304 с.
30. Системное программирование. Вып. 2 : Сб. статей/ А.Н. Терехов, Д.Ю. Булычев, Д.В. Кознов, А.Н. Иванов /Под ред. А.Н. Терехов, Д.Ю. Булычев. – СПб.: Изд–во С.Петерб. ун-та, 2006. – 298с.
31. Системное программирование: Сб. статей/ Под ред. А.Н. Терехов, Д.Ю. Булычев. – СПб., 2004. – 412с.
32. Amelina, N. Simultaneous perturbation stochastic approximation in decentralized load balancing problem / Amelina N., Erofeeva V., Granichin O., Malkovskii N. // In: Proc. of 1st IFAC Conference on Modelling, Identification and Control of Nonlinear Systems, June 24–26, 2015, Saint Petersburg, Russia. P. 946-951. (IFAC Proceedings Volumes (IFAC-PapersOnline) Volume 48, Issue 11).
33. Awad, M.A. A comparison between agile and traditional software development methodologies// School of Computer Science and software Engineering, The University of Western Australia. – 2005.
34. Bourque P., Fairley R.E., eds. Guide to the Software Engineering Body of Knowledge, Version 3.0/ IEEE Computer Society, 2014 – 346 p.
35. Granichin, O. Simultaneous Perturbation Stochastic Approximation for

Tracking under Unknown but Bounded Disturbances / Granichin O., Amelina N. // IEEE Transactions on Automatic Control, vol. 60, issue 6, June 2015, pp. 1653–1658.

36. Ivanskiy, Y., Amelina N., Granichin O., Granichina O., Jiang Y. Optimal step-size of a local voting protocol for differentiated consensuses achievement in a stochastic network with cost constraints // In: Proc. of the 2015 IEEE Conference on Control Applications, September 21-23, 2015, Sydney, Australia, pp. 1367–1372.
37. He, R. et al. E-leadership strategy in virtual organizations and virtual teams. – 2008 // Technical report, Helsinki Universitty of Technology, Faculty of Electronics, Communications, and Automation, Department of Communications and Networking.
38. Leyton-Brown, K. Multiagent Systems: Algorithmic, Game-Theoretic and Logical Foundations/ Leyton-Brown K., Shoham Y., London: Cambridge University Press. – 2009. — pp. 513
39. Mohtashami, M. et al. Risk Management for Collaborative Software Development //EDPAC: The EDP Audit, Control, and Security Newsletter. – 2007. – Т. 35. – №. 3. – С. 10-24.
40. Prikladnicki, R. A reference model for global software development: findings from a case study / Prikladnicki R., Audy J. L. N., Evaristo R. //Global Software Engineering, 2006. ICGSE'06. International Conference on. – IEEE, 2006. – С. 18-28.
41. Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK® Guide); 6 edition– Newtown Square, Pa: Project Management Institute, 2017 – 756 p.
42. McDermott, Robin E. The Basics of FMEA/ McDermott, Robin E., Mikulak, Raymond J., Beauregard Michael R. — Productivity Press. – 1996. — pp. 80

43. William Ibbs, C. The benefits of Project Management: financial and organizational rewards to corporations/ C. William Ibbs, Young-Hoon Kwak – Project Management Institute Education Foundation, 1997
44. Williams, R. C., Pandelios G. J., Behrens S. G. Software Risk Evaluation (SRE) Method Description: Version 2.0. – Carnegie Mellon University, Software Engineering Institute, 1999.
45. Vanita Yadav, A. Flexible Management Approach for Globally Distributed Software Projects // Global Journal of Flexible Systems Management, March 2016, Volume 17, Issue 1, pp 29–40
46. Vasileva, O. Generation of Efficient Cargo Operation Schedule at Seaport with the Use of Multiagent Technologies and Genetic Algorithms / Vasileva O., Kiyaev V. // Proceedings of the Third International Scientific Conference «Intelligent Information Technologies for Industry» (IITI'18). — «Advances in Intelligent Systems and Computing». — Volume 874-1, 2019.—p.401-410
47. Lin, Y. et al. Multi-agent system for intelligent scrum project management //Integrated Computer-Aided Engineering. – 2015. – Т. 22. – №. 3. – С. 281-296.
48. ArchiMate® 3.0 Specification. [Электронный ресурс]. URL. - https://publications.opengroup.org/c162 (дата обращения: 05.04.2018).
49. Домашняя страница проекта FIPA // The Foundation for Intelligent Physical Agents. [Электронный ресурс]. URL: http://http://fipa.org/ (дата обращения: 10.04.2019).
50. Использование JADE для разработки компьютерных систем поддержки дистанционного обучения агентного типа [Электронный ресурс]. URL: https://scinse.donntu.edu.ua/ius/kirgaev/library (дата обращения: 15.02.2019).
51. Project of multi-agent technology in difficult systems // Open University of the Netherlands http://www.ouh.nl.

Приложение 1. База данных мультиагентной системы управления проектными рисками в виртуальных компаниях

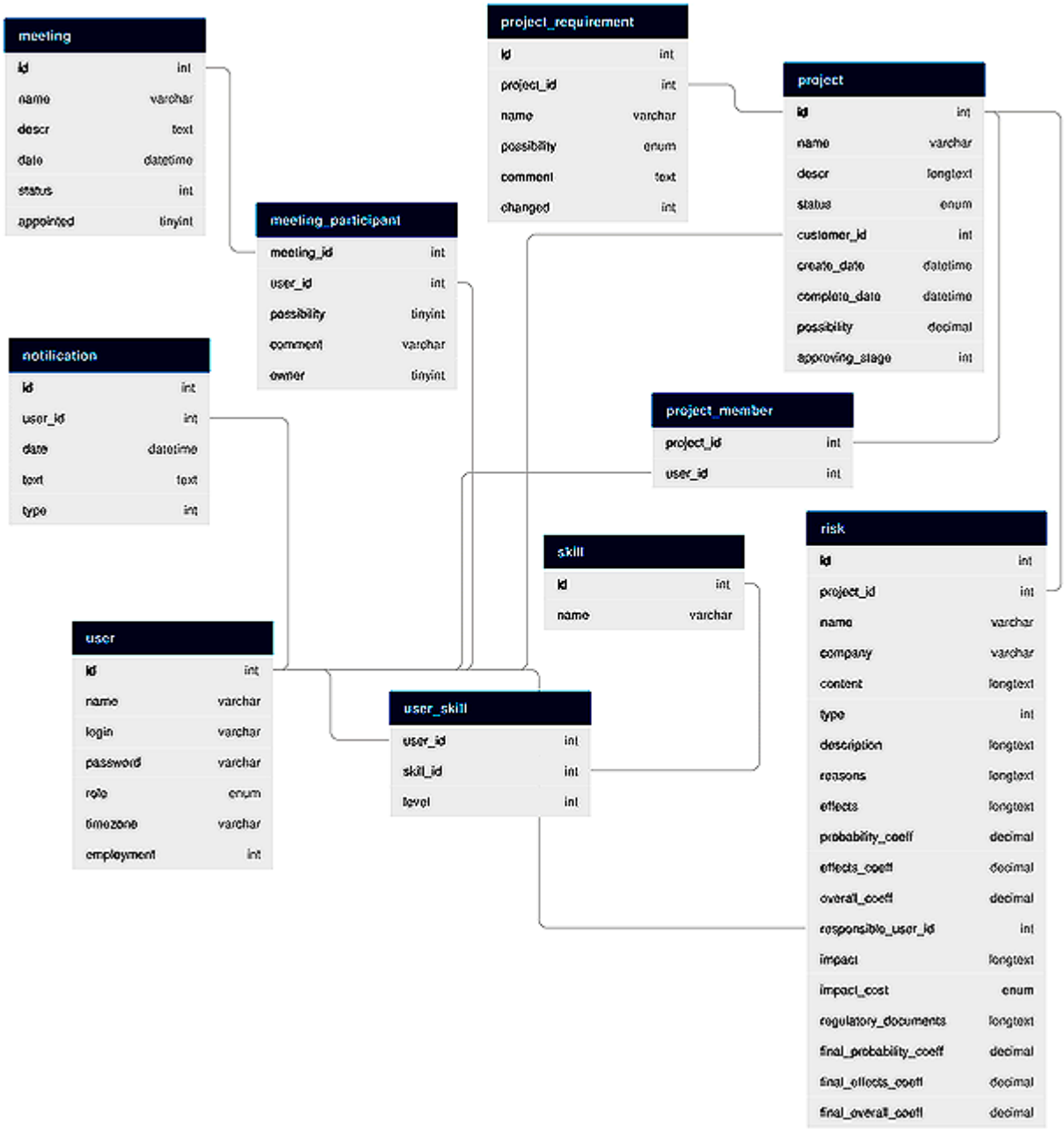

Рисунок П 1.1 База данных системы

Приложение 2. Фрагменты кода концептуально-функционального прототипа Web-приложения

П 2.1. Агент распределения рисков

```
1.  package com.general.agents;
2.
3.  import com.general.models.Risk;
4.  import com.general.models.User;
5.  import com.general.notifications.RiskResponsibleUserMissingNotification;
6.  import com.general.repositories.RiskRepository;
7.  import com.general.repositories.UserRepository;
8.  import jade.core.AID;
9.  import jade.core.Agent;
10. import jade.core.behaviours.SimpleBehaviour;
11. import jade.lang.acl.ACLMessage;
12.
13. import java.util.ArrayList;
14.
15. public class RiskDistibutionAgent extends Agent {
16.     @Override
17.     protected void setup()
18.     {
19.         addBehaviour(new Behavior(this));
20.     }
21.
22.     class Behavior extends SimpleBehaviour {
23.         private boolean finished = false;
24.         private RiskDistibutionAgent agent;
25.
26.         public Behavior(RiskDistibutionAgent a) {
27.             super(a);
28.             agent = a;
29.         }
30.
31.         public void action() {
32.             Object[] args = agent.getArguments();
33.             String notificationAgentId = args[0].toString();
34.             ArrayList<Risk> riskMap = getFirstRiskMap();
35.             riskMap.forEach(risk -> {
36.                 ArrayList<User> userList = getUserList(risk.getProjectId());
37.                 userList.forEach(user -> {
38.                     if (user.getSkillLevel() >= risk.getSkillLevel()) {
39.                         User current = risk.getResponsibleUser();
40.                         if (current == null || current.getEmployment() > user.getEmploy
    ment()) {
41.                             risk.setResponsibleUser(user);
42.                         }
43.                     }
44.                 });
45.                 if (risk.getResponsibleUser() == null) {
46.                     RiskResponsibleUserMissingNotification notification = new RiskRespo
    nsibleUserMissingNotification();
47.                     notification.setRisk(risk);
48.
49.                     ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
50.                     msg.setContent(notification.toString());
51.                     msg.addReceiver(new AID(notificationAgentId, AID.ISLOCALNAME));
52.                     send(msg);
53.                 }
54.             });
```

```
            getRiskMap(riskMap);
            finished = true;
        }

        public boolean done() {
            return finished;
        }

        private ArrayList<Risk> getFirstRiskMap() {
            RiskRepository db = RiskRepository.getInstance();
            return db.getFirstRiskMap();
        }

        private ArrayList<User> getUserList(int projectId) {
            UserRepository db = UserRepository.getInstance();
            return db.getUserListByProject(projectId);
        }

        private void getRiskMap(ArrayList<Risk> riskMap) {
            RiskRepository db = RiskRepository.getInstance();;
            db.updateRiskMap(riskMap);
        }
    }
}
```

П 2.2 Класс Main для запуска агента распределения рисков

```
package com.general;

import jade.core.Profile;
import jade.core.ProfileImpl;
import jade.core.Runtime;
import jade.wrapper.AgentController;
import jade.wrapper.ContainerController;

public class Main {

    public static void main(String[] args) {
        Runtime rt = Runtime.instance();
        Profile p = new ProfileImpl();
        p.setParameter(Profile.MAIN_HOST, "localhost");
        p.setParameter(Profile.MAIN_PORT, "10098");
        p.setParameter(Profile.GUI, "true");
        ContainerController cc =  rt.createMainContainer(p);
        try {
            AgentController notificationAgent = cc.createNewAgent("notification", "com.general.agents.NotificationAgent",null);
            notificationAgent.start();
        } catch (Exception e) {}
        try {
            Object[] agentArgs = new Object[1];
            agentArgs[0] = "notification";
            AgentController riskDistibutionAgent = cc.createNewAgent("risk_distibution", "com.general.agents.RiskDistibutionAgent",agentArgs);
            riskDistibutionAgent.start();
        } catch (Exception e) {}
    }
}
```

Приложение 3. Примеры экранных форм для инициализации работы агентов, разработанных с использованием платформы JADE

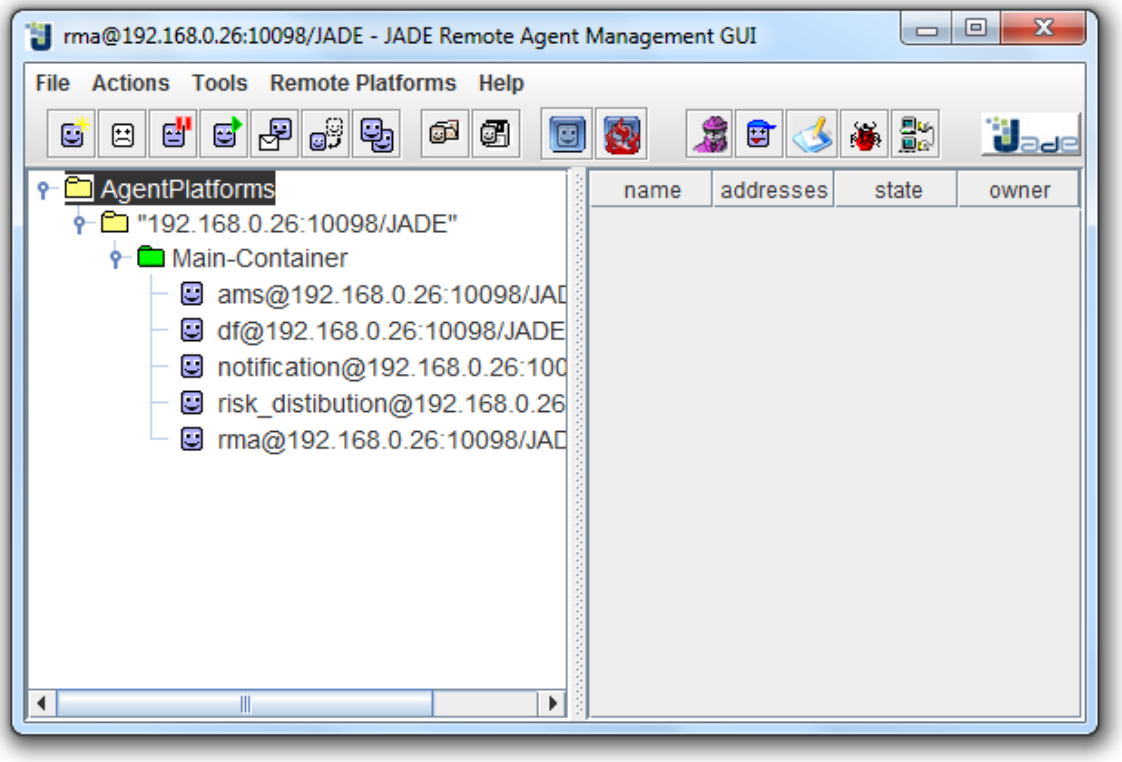

Рисунок П 3.1 Список активных агентов

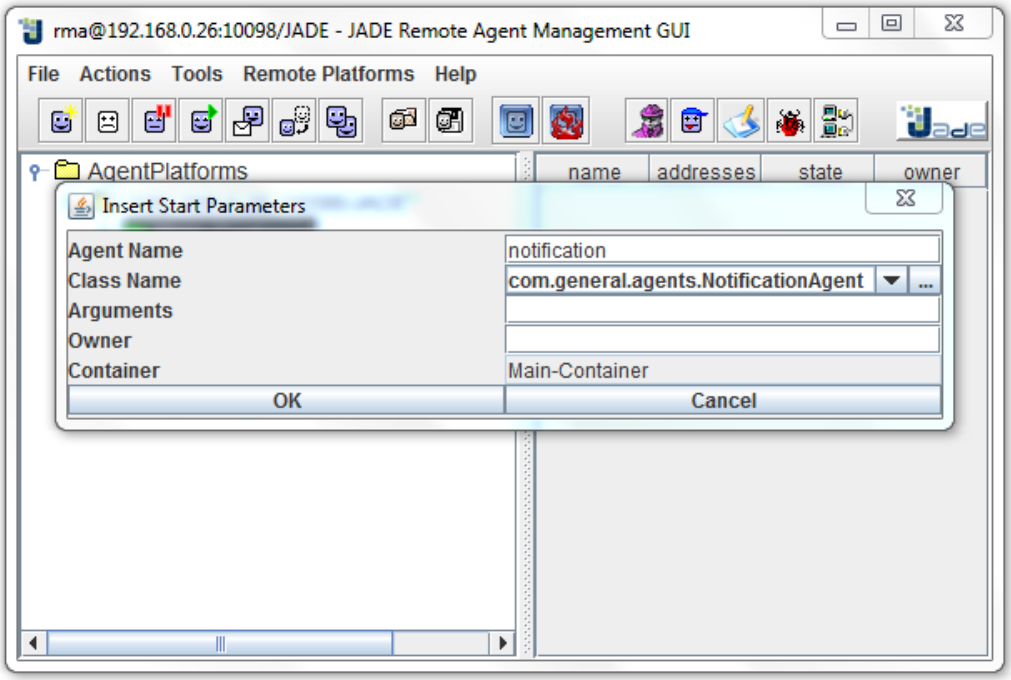

Рисунок П 3.2. Запуск агента вручную

Приложение 4. Бизнес-процесс работы мультиагентной системы управления проектными рисками для виртуальных компаний

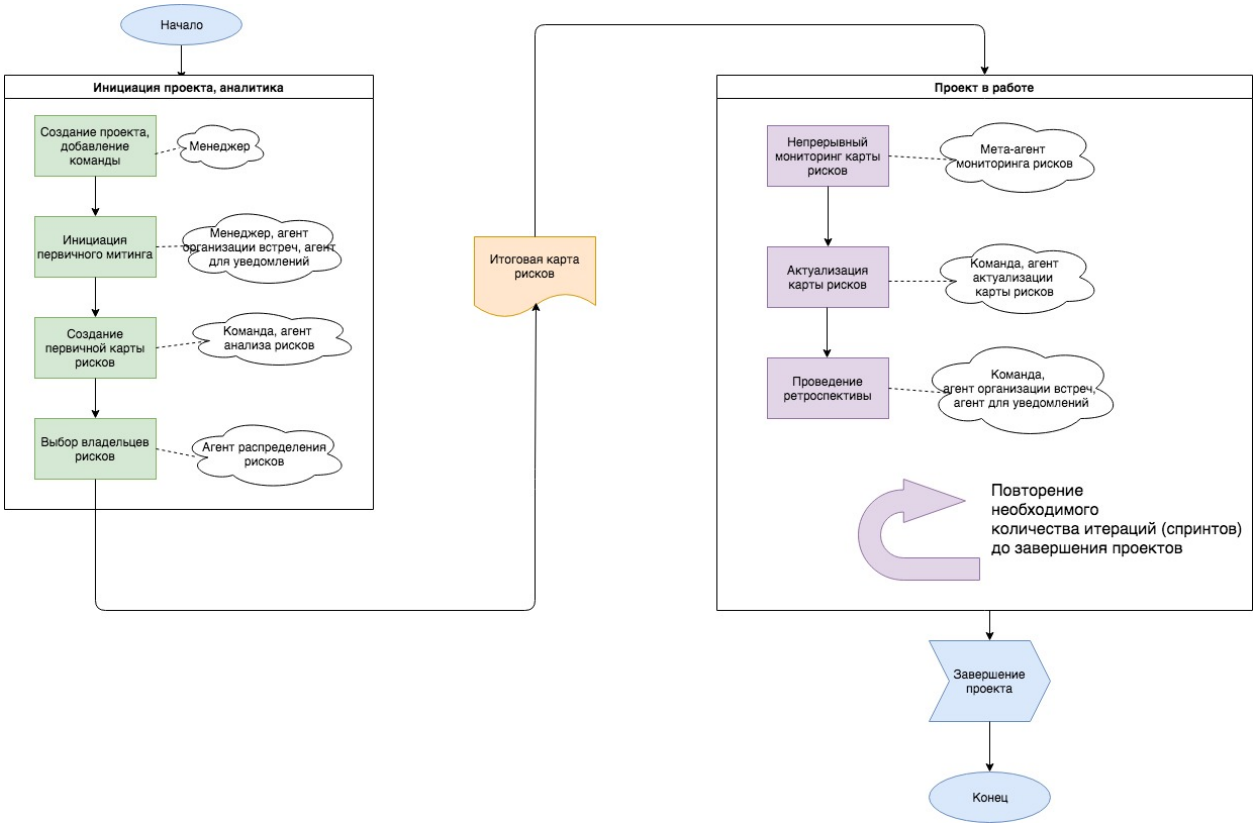

Рисунок П 4.1. Бизнес-процесс работы МАС