Санкт-Петербургский государственный университет

Направление математика и механика

Прикладная математика и информатика

Величко Юлия Дмитриевна

Разработка компонентов системы интерфейсов для эффективного управления проектами в распределенной компании

Бакалаврская работа

Научный руководитель:

канд. физ.-мат. наук, доц. Кияев В.И.

Рецензент:

канд. физ.-мат. наук, науч. сотр. Иванский Ю.В.

Санкт-Петербург

2017

SAINT-PETERSBURG STATE UNIVERSITY

Main Field of Study Mathematics and Mechanics

Area of Specialisation Applied Mathematics and Computer Science

Iuliia Velichko

Development of interface systems components for efficient project management in a distributed enterprise

Bachelor's Thesis

Scientific supervisor:

PhD, Assist. Professor Kiyaev V.

Reviewer:

PhD, Researcher Ivanvskiy Y.

Saint-Petersburg

2017

Оглавление

Введение

В начале XXI века с развитием информационных технологий, в частности, интернета, стала реальна такая модель формирования бизнес-организаций, как *Виртуальная организация*. На тот момент она была потенциально новой и подавала большие надежды. Подразумевалось, что виртуальная организация позволит повысить гибкость, предложить товар всему миру, при этом географический (физический) центр оказывал бы малое влияние на ее эффективность. В связи с высокой скоростью обмена информацией члены команды могли бы работать над проектом, будучи на большом расстоянии друг от друга. Все это поспособствовало образованию такого тренда, как виртуальная организация. Спустя некоторое время выяснилось, что создание и сопровождение деятельности таких организаций требует особых технологий, моделей управления, информационных платформ, которые на тот момент еще только зарождались. Популярность виртуальных компаний стала спадать, потому что расходы на такую модель организации превышали прежние, а перед руководителями и менеджерами встал вопрос – как эффективно управлять такой компанией.

Существует множество определений виртуальной компании, но ее суть в отсутствии четкой структуризации и большой структурно-функциональной гибкости, поэтому обозначим признаки, характерные для большинства таких компаний и отличающие их от традиционных [1].

Незначительная физическая структура – в отличие от традиционных компаний количество офисов, складов не так велико и распределено географически.

Доверие коммуникационным технологиям – информационные технологии являются необходимым элементом системы управления и используются для динамической связи людей и активов. Они лежат в основе виртуальной организации, служат инструментом, который позволяет выполнять работу. Но это не означает, что люди будут взаимодействовать только посредством

интернета или других современных технологий. Они содействуют работе организаций, но не являются самой организацией.

Мобильность работы – физическое местоположение работника не важно, нет необходимости в сборе отдела и команды в одном помещении. Это значительно снижает расходы на аренду офисов, а члены команды могут сами выбирать себе удобное для работы место. Верным становится утверждение: офис там, где работают, а не наоборот.

Отсутствие физических границ и вовлечение – виртуальные организации сознательно не устанавливают четкие границы, это позволяет им вовлекать дистрибьютеров, поставщиков и клиентов в систему, где общий результат требует участие каждого из них – вне зависимости от того, где могут находиться участники общей деятельности.

Гибкость и ответная реакция – позволяет незамедлительно реагировать на изменения стратегии компании. Объединять различные элементы для достижения определенной цели и, если потребуется, производить быструю реструктуризацию. В связи с этим увеличивается скорость предоставления услуг.

Благодаря разнообразию информационных технологий сформировались несколько форм виртуальных организаций (рис. 1).

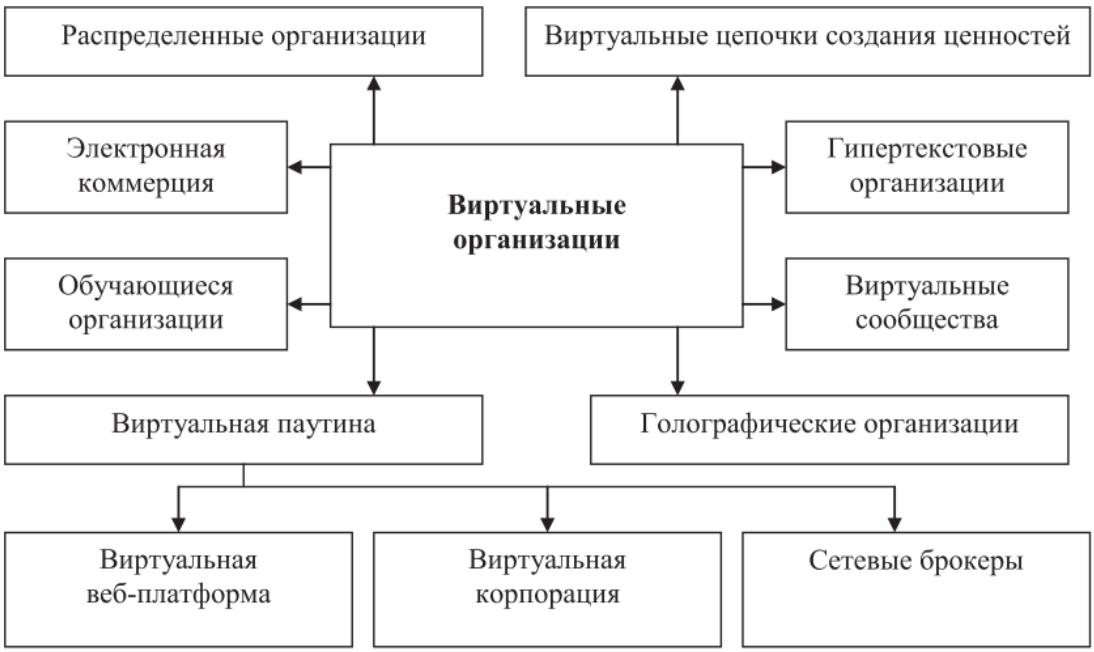


Рис. 1 Формы виртуальных организаций [2]

Рассмотрим *распределенные организации*. Они возникают в нескольких случаях, представленных ниже.

- Предприятие не нуждается в компактной физической структуре и результатом деятельности компании является реализованный проект или программа, которая может быть выполнена несколькими рабочими группами, не находящимися в одном офисе.
- Компании нуждаются в увеличении области действия, открываются региональные филиалы, бизнес расширяется и охватывает все большие территории.
- В работе компании используются удаленные структурные подразделения.
- Появляется необходимость в привлечении специалистов из других городов, регионов или стран.

Их отличительной особенностью является распределение капитала в различных географических точках. Одними из первых компаний, что использовали такую форму, были банки и международные торговые и трастовые компании. В настоящее время многие организации расположили свои офисы в разных уголках страны и обеспечивают связь между ними посредством современных коммуникационных технологий и общего информационного пространства.

Преимущества виртуального механизма координации проектной работы очевидны [2]:

- высокая скорость выполнения рыночного заказа;
- значительное снижение транзакционных издержек производственной деятельности;
- более эффективное взаимодействие с потребителем и более полное удовлетворение его потребностей;
- гибкая адаптация к изменениям окружающей деловой среды;
- снижение барьеров выхода на новые рынки и др.

Однако, оборотной стороной деятельности организаций такого типа является существенное увеличение организационных и функциональных рисков. Один из наиболее существенных – как эффективно управлять командами разработчиков и проектом в целом. Для эффективной работы распределённой организации в первую очередь необходимо единое рабочее и информационное пространство. Цель формирования такого организационного пространства – максимально автоматизировать систему бизнес-процессов, а также управление ими и упростить пользователю контроль всех этапов ее реализации.

Отметим, что, используя современные мультиагентные технологии, можно автоматизировать некоторые управленческие и технико-функциональные процессы, создав систему интеллектуальных агентов, которые будут отслеживать все этапы выполняемого процесса в критических точках, принимать стандартные решения или передавать полномочия человеку при возникновении нестандартных или угрожающих ситуаций. Важным аспектом здесь является разработать компоненты адаптированной системы интерфейсов взаимодействия «агент-агент» и «агент-человек», учитывая закономерности мышления и поведения. Под интерфейсом мы будем подразумевать точки взаимодействия активных элементов мультиагентной системы {«человек-агент», «агент-агент», «агент-человек»}, в которых происходят взаимодействия объектов и субъектов по определенным правилам на базе согласованных алгоритмов, протоколов и стандартов. Отметим также, что интерфейс может быть записан в виде соответствующего кода (программный интерфейс), реализован в виде набора программных и технических средств (программно-аппаратный интерфейс), а также в виде пользовательского интерфейса (человеко-машинный интерфейс). Указанный аспект обуславливает актуальность исследуемой проблемы.

Разработка компонентов системы интерфейсов является частью проектной работы, выполненной совместно с Веселовой Дианой, в соответствии с известными методологиями проведения совместных работ.

Нами были определены цели и задачи проекта, сформированы требования к разрабатываемой концептуально-функциональной модели, составлен план работ (построена диаграмма Гантта), в ходе работы проводились совместные встречи для обсуждения текущих результатов. Все проектные задачи были поделены на две группы в соответствии с темами ВКР, которые указаны в названии работ. Проект заключается в разработке прототипа (концептуально-функционального описания и архитектуры) Web-приложения, являющегося мониторинговой системой для эффективного управления проектами в распределенных компаниях.

Целью данной работы является разработка компонентов системы интерфейсов для эффективного управления проектами в распределенных компаниях. Модели, алгоритмы и блок-схемы, полученные в результате работы, впоследствии можно будет использовать для реализации систем подобного типа.

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать общие принципы управления программными проектами и выделить ключевые управленческие моменты, характерные для распределенных компаний;
- сравнить организацию деятельности команд разработчиков в традиционных и распределенных компаниях и выбрать методологии разработки программного обеспечения, опираясь на выявленные особенности;
- рассмотреть мультиагентный подход как организационное решение в распределенных компаниях;
- выявить ключевые процессы управления проектами в распределенных компаниях, наиболее подверженные рискам, и которые можно автоматизировать;
- разработать алгоритмы действий для агентов, автоматизирующие эти процессы;

- проанализировать модели, протоколы и стандарты взаимодействия в мультиагентных системах и разработать интерфейсы взаимодействия «агент-агент» и «агент-человек»;
- разработать и внедрить компоненты системы интерфейсов в Web-приложение.

1. Общие принципы и проблемы управления проектами

1.1. Управление проектами по разработке программного обеспечения

Рассмотрим проектно-ориентированную деятельность, то есть деятельность, результаты которой доставляются потребителю в виде реализованных проектов и программ, целью которых являются создание продуктов и представление разнообразных услуг [3]. Управленческая деятельность осуществляется на трех уровнях [4]:

- управление организацией и инфраструктурой;
- управление собственно проектами;
- разработка и управление программой измерений (разработка метрик показателей качества и эффективности производительности).

Важным аспектом организационного управления является политика управления персоналом и процедуры найма, обучения и мотивации персонала для развития их навыков и карьерного роста, причем не только на уровне проекта, но и для дальнейшего успеха организации.

Управление коммуникациями часто упускают, хотя оно во многих случаях является чрезвычайно важным аспектом производительности команд разработки, где необходимо точное понимание потребностей пользователей, требований к программному обеспечению и программным проектам. Получение такой информации очень важно для эффективного решения поставленных задач.

Помимо управленческой деятельности, специфичной для компаний по разработке программного обеспечения, ИТ-специалистам также необходимо осознавать ключевые моменты общепринятого менеджмента и управления проектами.

В компаниях по разработке программного обеспечения характерна «нематериальная деятельность», обусловленная виртуальным характером производимого продукта – это в большей степени работа менеджеров и аналитиков, нежели исполнителей, пишущих и тестирующих код. Менеджер

проекта является ответственным лицом за достижение целей проекта. От сотрудников требуется больше навыков и знаний, а также умение взаимодействовать с людьми на расстоянии посредством использования информационных технологий. Виртуальная работа требует большей персональной ответственности, поскольку каждый сотрудник отвечает за качество выполняемой работы – оплошность или нерадивость одного участника процесса может пагубно сказаться на результате работы всей команды. Менеджеры виртуальных организаций отвечают за выработку норм качества и обеспечение инструментами, которые позволят сотрудникам соблюдать требования международных стандартов. Менеджер каждого проекта должен управлять влиянием различных заинтересованных сторон проекта, он также имеет полные полномочия и руководит всеми членами команды, используя принципы и методы управления проектами с целью повышения эффективности выполнения проектов [1].

Общие принципы проектного подхода рационально использовать и в качестве управления распределенными компаниями, так как деятельность таких компаний [3]:

- ориентирована на достижение определенной цели;
- содержит взаимосвязанную реализацию взаимозависимых операций;
- нацелена на создание в своем роде уникального продукта;
- подразумевает временное ограничение на создание конечного продукта.

При этом, однако, как уже отмечалось выше, необходимо строго отслеживать специфику такой деятельности, связанной с повышенными рисками в ключевых точках проектной работы.

В нашей работе будем руководствоваться «Сводом знаний по управлению проектами» (A Guide to the Project Management Body of Knowledge – руководство PMBOK® [5]), который представляет собой совокупность профессиональных знаний по управлению проектами, признанных в качестве

стандарта. Стандарт – это официальный документ, в котором описываются установленные нормы, методы, процессы и практики [5].

Следуя проектному подходу распределенные компании должны создавать систему управления проектами, взяв за основу системную модель управления проектами (рис. 2).

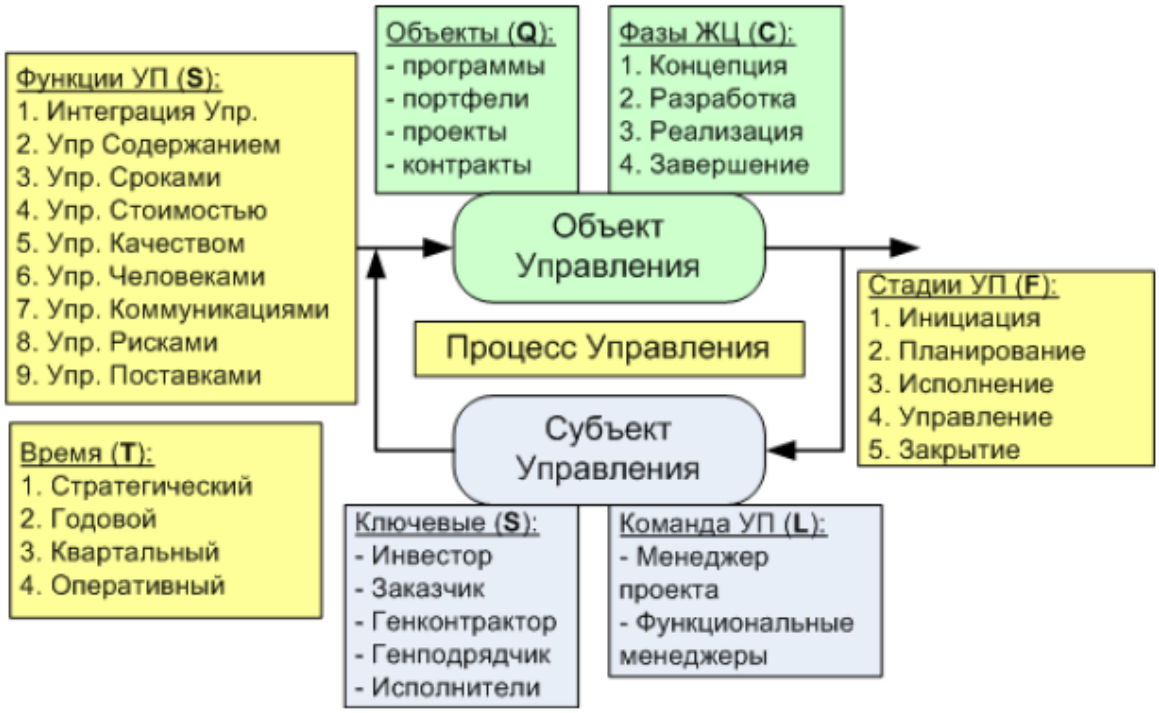


Рис. 2 Системная модель управления проектами согласно PMBOK [5]

В такой модели есть объект управлений с различными фазами жизненного цикла, субъект управления, содержащий основных участников и команду управления проектом, и сам процесс управления. В соответствии с PMBOK [5] все процессы управления проектами делятся на пять групп:

- ***инициализация***

  Получение разрешения и формальная авторизация начала проекта. Сбор информации необходимой для создания устава проекта, определение заинтересованных сторон. Определение и обсуждение требований, анализ осуществимости.

- ***планирование***

  Определение и планирование действий, необходимых для успеха проекта. Выход группы планирования проектов включает в себя все

аспекты содержания, сроков, стоимости, качества, коммуникаций, рисков и закупок.

- ***исполнение***

  Координирование людей и ресурсов для осуществления плана управления проектом, обеспечение качества.

- ***мониторинг и управление***

  Оценивание прогресса проекта, контроль хода и эффективности проекта, поиск областей, требующих внесения изменений в план, подготовка отчетности и др.

- ***закрытие***

  Осуществление действий, необходимых для завершений всех операций в рамках управления проектом, подтверждение правильности выполнения всех процессов в группах, документирование результатов, архивирование.

Применяя эти знания к разработке программного обеспечение правильнее было бы использовать словосочетание выполнение программного проекта, нежели исполнение. А также подразумевать под мониторингом и управлением обзор и оценку (определение удовлетворения требования, оценку продуктивности и результативности) [4].

В процесс управления включаются также функции, такие как управление проектом, его содержанием, сроками, стоимостью, качеством, человеческими ресурсами, коммуникациями, рисками, поставками и стейкхолдерами (заинтересованными сторонами). Задача менеджера – отслеживать влияние каждой функции на конкретный проект. Взаимоотношения между этими функциями таковы, что если одна из них изменится, то вероятнее всего, изменятся и некоторые другие. Изменение требований к проекту, сокращение сроков, уменьшение бюджета и другое, могут вызвать дополнительные риски. Важно уравновешивать все требования для достижения окончательного результата. Именно поэтому секции управления и измерения тесно взаимосвязаны и играют немаловажную роль в этой области знаний.

Управление, направленное на измерения, может сильно посодействовать в поиске решения проблем. Отсутствие количественных или качественных измерений влечет за собой неимению прогресса в достижения целей. Эффективное управление требует сбалансированной интеграции систематических и упорядоченных подходов и подходящего опыта.

Так как существует вероятность изменений первоначальных требований, план управления проектом имеет носит итерационную природу и выполняется последовательно в зависимости от стадии жизненного цикла. Основополагающая структура жизненного цикла проекта: предпроектная подготовка, начало, организация и подготовка, реализация, завершение проекта. Такая обобщенная структура может использоваться для обсуждения проекта с вышестоящим руководством, не имеющим детального представления о проекте [5].

Проект обычно делят на фазы – части проекта, требующие определенного контроля для эффективной реализации. В зависимости от структуры проекта фазы выполняются могут выполняться как последовательно, так и параллельно. Введение фаз дает возможность поделить проект на логические блоки, чтобы было надёжнее осуществлять управление и контроль. В зависимости от размера проекта и необходимой степени контроля количество фаз может меняться.

Разберем подробнее содержание жизненного цикла проекта. Первой стадией является исследование концепции проекта. На данном этапе формулируется проект в соответствии с запросом потребителя. Применяются методы проектного анализа, заключающиеся в экономическом, финансовом и организационным анализе рисков, проводятся переговоры с участниками проекта, разрабатывается примерный план реализации. Планирование проекта осуществляется на протяжении всего жизненного цикла проекта. Далее идет разработка проекта, на этом этапе формулируются задачи и связи между ними, распределяются роли, используются различные методологии, сетевые графики, диаграммы Гантта. Реализация начинается с утверждения детального

плана проекта. По мере продвижения проекта менеджер обязан отслеживать ход работы, собирая сведения о текущем ходе работ и сравнивая ее с планом. Когда достигнуты поставленные цели, проект завершается.

В зависимости от типа проекта выбираются различные подходы к управлению жизненным циклом проекта.

1.2. Сравнение организации деятельности команд разработчиков в традиционных и распределенных компаниях

В распределенных компаниях виртуальные команды можно классифицировать по типу коммуникаций и по цели создания.

По типу коммуникаций со соответствующими средствами взаимодействия [6]:

Синхронные-локальные Встречи (сбор команды в одном месте)	**Асинхронные-локальные** Электронные доски объявлений, сервисы распространения информации
Синхронные-удаленные аудио/видеоконференции, чат, звонки	**Асинхронные-удаленные** Интернет, письма, e-mail, голосовые сообщения

Йаел Сара Зофи из Колумбийского университета (США), специалист по виртуальному менеджменту, выделяет следующие типы команд по цели создания [7]:

- сетевые команды;
- параллельные команды;
- команды по разработке проектов/продуктов;
- производственные команды;
- сервисные команды;
- управленческие команды;
- офшорные команды;
- команды быстрого реагирования.

Мы будем рассматривать работу и специфические особенности команды проекта/продукта по разработке программного обеспечения. Выделим общие и различные черты в работе виртуальных и традиционных команд. Эти данные о командах и процессах разработки программного обеспечения послужат основой для выявления дополнительных рисков, связанных с деятельностью распределенных компаний.

В традиционных компаниях особое внимание уделяется *личным встречам и взаимодействию внутри команды*. Совместное расположение позволяет пользоваться такими инструментами, как дисплей с текущим статусом проекта, который могут видеть все разработчики, или белая доска, позволяющая размещать совместно используемые заметки. Но самое главное, что личный контакт помогает наладить доверие и взаимопонимание между членами команды. С развитием технологий электронные варианты этих инструментов стали доступными и для распределенных компаний.

Традиционные компании действуют в рамках *иерархической структуры*. Менеджер выступает в большей степени как ведущий, он обеспечивает безопасную среду для разработки и поощряет кооперацию между сотрудниками. Особое внимание уделяется совершенствованию навыков членов команды. Обучение может включать в себя наставничество от более опытных членов команды или тренеров, а также более формальный вариант – учебные пособия или классы. Для распределенных компаний все это тоже стало доступным с появлением онлайн-курсов, веб-хостингов для ИТ-проектов и их совместной разработки, интерактивных видеоконференций.

В *виртуальных компаниях*, для наиболее эффективной координации и слаженности команды менеджер проекта использует четкое структурирование обязанностей, проектных ролей и зон ответственности менеджеров и исполнителей. Он должен гарантировать, что методы связи и совместной работы согласованы, четко определены и контролируются. Более того, важно учесть возможное различие между уровнями квалификации участников. При

этом координирование членов команды не осуществляется совместно на производственных митингах, что свойственно традиционным компаниям.

Виртуальные команды практически полностью *зависят от информационных технологий*, используя их для общения и совместной работы, в то время как традиционные команды всегда могут отказаться от технологий и программных инструментов в пользу работы по принципу «лицом к лицу». Для виртуальных организаций важно уметь работать с такими технологиями, потому что эффективность команды может как стремительно возрасти, так и упасть. Менеджеры проектов виртуальных организаций должны обеспечивать своевременную техническую поддержку команде в случае неисправности технического оборудования или программного обеспечения. В традиционных компаниях менеджер проектов обычно не отвечает за проблемы технической поддержки, это в компетенции ИТ-отдела.

В целом, виртуальные и традиционные команды, преследующие аналогичные цели будут по-прежнему проходить через одинаковый жизненный цикл проекта для достижения поставленной задачи [8]. Но из-за отличительных особенностей распределенных компаний увеличивается количество рисков, которым они подвержены. Правильный выбор методологии разработки поможет вовремя обнаружить и контролировать возможные риски.

1.3. Методологии разработки программного обеспечения

Изучив большинство традиционных моделей разработки (каскадная, V-образная, инкрементная, итеративная и др. модели), можно сделать вывод, что в такой сложной структурной организации, как распределенная компания, необходимо комбинировать различные приемы разработки. В настоящее время большинство компаний, не использующих традиционные методы проектной разработки программных продуктов, применяют различные гибкие методологии, основанные на итерационной (спиральной) модели.

Спиральная модель

В 1988 году Барри Боэм впервые представил спиральную модель в журнале IEEE Computer (рис. 3). Спиральная модель учитывает недостатки традиционной каскадной модели и усиливает ее преимущества, в ней в большой степени учитываются анализ и управление рисками, связанные со спецификой программных проектов, процессы поддержки реализации и менеджмента. В своей работе [9] Б. Боэм описывает наиболее распространенные риски в разработке программного обеспечения:

- нехватка специалистов;
- «неправдоподобные» сроки и бюджет;
- реализация несоответствующего требованиям функционала;
- разработка неудобного пользовательского интерфейса;
- ненужная и/или излишняя оптимизация и универсальность;
- бесконечные изменения, которые могут привести к «порче» конечного релиза;
- разрыв между квалификацией специалистов и др.

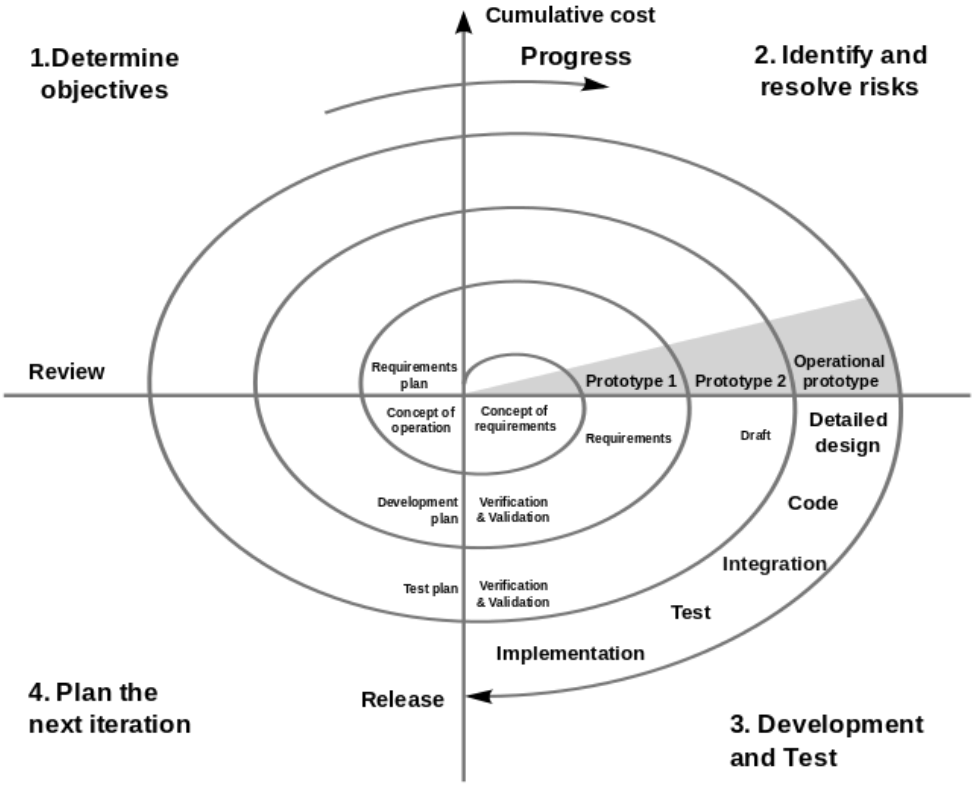


Рис. 3 Жизненный цикл программного продукта в рамках спиральной модели

Преимущества спиральной модели [10]:

- пользователи могут увидеть продукт на ранних этапах разработки;
- выявление непреодолимых рисков с наименьшими затратами;
- позволяет разбить большую по объему работу на небольшие части, в которых особое внимание уделяется сначала функциям с наибольшей степенью риска
- предусмотрено гибкое проектирование
- частое общение с заказчиком и пользователями
- нет необходимости заранее распределять финансовые ресурсы

Спиральная модель имеет следующие недостатки [10]:

- небольшие проекты с невысокой степенью риска могут оказаться дорогостоящими
- достаточно сложная для понимая структура
- необходимы квалифицированные специалисты
- спираль может продолжаться до бесконечности
- может появиться необходимость в дополнительной документации

Использование спиральной модели логично в больших дорогостоящих проектах с *высокой степенью риска*. Или, когда организация владеет необходимыми навыками для адаптации модели, при разработке новой серии продуктов или прототипировании, когда пользователи не могут в начале проекта сформулировать точно свои требования и/или требования могут динамически изменяться в ходе выполнения проекта. Применять спиральную модель можно также при разработке систем, требующих обработки больших объемов данных и вычислений, например, систем принятия решений или банковских систем.

Как примеры «гибких» методологий, рассмотрим «Экстремальное программирование» и методику «Scrum» [11].

Экстремальное программирование (Extreme programming, XP) появилось из проблем, вызванных длительными циклами разработки в

традиционных моделях [12]. Процесс XP может быть охарактеризован короткими циклами разработки, инкрементным планированием, непрерывной обратной связью, опорой на коммуникации и эволюционное проектирование. Со всеми вышеперечисленными качествами программисты XP гораздо быстрее реагируют на изменяющуюся среду.

Согласно Вильямс [13], члены команды, использующей методику XP тратят распределенное и согласованное количество времени на управление проектами, на программирование, на проектирование, на обратную связь и на укрепление духа команды (уверенность в успехе) – и так многократно, каждый день. В распределенных компаниях такой метод разработки можно использовать, несмотря на расстояние между членами команды, ведь существует множество сервисов для коммуникации и совместной разработки.

Scrum – это итеративный, инкрементный процесс разработки. Он концентрируется на том, как члены команды должны функционировать, чтобы обеспечить гибкость системы в постоянно меняющейся среде [11]. В конце каждой итерации (спринта) он создает потенциальный набор функциональных возможностей. Спринты – это процедуры адаптации к изменяющимся переменным среды (требованиям, времени, ресурсам, знаниям, технологиям и т.д.), которые должны приводить к потенциальной доработке программного обеспечения. Рабочими инструментами команды являются совещания по планированию спринта, журнал с пожеланиями проекта и ежедневные встречи. Спринт обычно фиксирован по времени и его длительность составляет 30 дней. Scrum не требует и не предоставляет какие-либо конкретные практики разработки программного обеспечения для использования. Вместо этого он требует определенных методов управления и инструментов на разных этапах Scrum, чтобы избежать хаоса из-за непредсказуемости и сложности [14].

Быстрая реакция на изменения окружающей среды и адаптация к ним – ключ к успеху в разработке программного обеспечения в распределенных компаниях. Использование спиральной модели позволит отслеживать и

контролировать многочисленные риски проекта, выполняемого в распределенной компании, а «гибкие» методологии, основанные на такой модели разработки, обеспечат необходимую гибкость деятельности распределенным компаниям, которая является одной из ее основных требуемых характеристик.

1.4. Управление рисками и качеством. Измерения в разработке программного обеспечения.

На этапе планирования программного проекта особое внимание необходимо уделить управлению качеством и рисками, а также измерениям в разработке программного обеспечения.

Управление качеством

В распределенных компаниях менеджерам проектов сложнее контролировать качество работы каждого члена команды, поэтому на базе семейств международных стандартов ISO, IEC, IEEE, CMM и CMMI создаются или используются уже существующие отраслевые и корпоративные стандарты качества [4]. В компаниях такого типа повышается персональная ответственность каждого сотрудника, он должен следовать стандартам качества, предоставленным менеджером проекта. Требования к качеству программного продукта должны быть определены как в качественном, так и в количественном отношении для программного проекта и соответствующих рабочих продуктов. Они разрабатываются в соответствии с потребностями и ожиданиями заинтересованных сторон. Все процедуры, связанные с обеспечением качества программного обеспечения и улучшения качества на протяжении всего процесса разработки должны быть указаны при планировании качества.

Управление рисками

Разработка программного обеспечения – деятельность, использующая различные технологические достижения и требующая высоких уровней знаний. Из-за этих и других факторов каждый проект разработки

программного обеспечения содержит элементы неопределенности, называющиеся рисками проекта [4]. Успех проекта по разработке программного обеспечения в значительной степени зависит от величины риска, которая соответствует каждой деятельности по проекту. Менеджеру проекта недостаточно просто знать о существовании риска. Для достижения успешного результата руководители проектов должны выявлять, оценивать, устанавливать приоритеты и управлять всеми основными рисками.

Целью большинства проектов по разработке программного обеспечения является создание уникального продукта, отличающегося функционалом или большей эффективностью. Стремление использовать передовые технологии не может протекать без рисков. Идентификация и ранжирование рисков является единственным прогностическим методом для определения вероятности того, что проект разработки программного обеспечения столкнется с незапланированными или недопустимыми событиями. К ним относятся прерывания, разрывы, задержки в графике, недооценка стоимости и перерасход ресурсов проекта. Процессы по управлению рисками проекта помогают увеличивать вероятность появления и осуществления благоприятных событий и снижать вероятности неблагоприятных для проекта событий.

Управление рисками включает в себя следующие задачи [4]:

- определение рисков и их тренды;
- классификация и расставление приоритетов по каждому из рисков;
- составление плана по смягчению каждого риска;
- мониторинг триггеров риска в течение хода проекта;
- разработка корректирующих воздействий при появлении какого-либо риска;
- информирование всех заинтересованных сторон о статусе риска на протяжении всего проекта.

Помочь менеджеру в классификации рисков может опыт из предыдущих проектов, необходимо точно описать все возможные угрозы успешности

проекта. Простой, но эффективной схемой классификации рисков является соотнесение (идентификация) рисков в соответствии с областями воздействия.

Для большинства проектов разработки программного обеспечения можно выделить пять основных областей воздействия рисков [15]:

1. новые, не апробированные в больших и сложных проектах технологии;
2. пользовательские и функциональные требования;
3. архитектура приложений и систем;
4. производительность;
5. организационная деятельность.

1. Новые, не апробированные технологии

Большинство программных проектов связано с использованием новых технологий. Постоянно меняющиеся инструменты, методы, протоколы, стандарты и системы разработки увеличивают вероятность того, что технологические риски будут возникать практически в любом существенном процессе разработки программного обеспечения. Обучение и знания имеют решающее значение, а неправильное использование новых технологий чаще всего ведет непосредственно к провалу проекта.

2. Пользовательские и функциональные требования

Требования к программному обеспечению охватывают все потребности пользователей в отношении особенностей, функций и качества обслуживания. Зачастую процесс определения требований является длительным, сложным и утомительным. Более того, требования меняются в процессе инициализации, прототипирования и интеграции. Изменения в элементных требованиях, скорее всего, будут распространяться по всему проекту, а изменения требований пользователей могут не соответствовать функциональным требованиям. Эти сбои часто приводят к критически важным ошибкам плохо спланированного проекта разработки программного обеспечения.

3. Архитектура приложений и систем

Неправильный выбор платформы, компонента или архитектуры может иметь катастрофические последствия. Как и в случае с технологическими

рисками, крайне важно, чтобы в группе разработчиков были эксперты, которые разбираются в архитектуре и могут сделать правильный выбор.

4. Производительность

Важно обеспечить, чтобы любой план управления рисками учитывал ожидания пользователей и партнеров в отношении производительности. Необходимо уделить внимание эталонам и пороговым испытаниям на протяжении всего проекта, чтобы гарантировать, что работа продуктов движется в правильном направлении.

5. Организационная деятельность

Организационные проблемы могут оказывать негативное влияние на результаты проекта. Руководство проектом должно планировать эффективное выполнение проекта и находить баланс между потребностями команды разработчиков и ожиданиями клиентов. Разумеется, грамотное кадровое обеспечение включает в себя выбор членов команды с набором навыков, которые хорошо сочетаются с проектом.

После категоризации всех рисков в соответствии с типом менеджер проекта разработки программного обеспечения должен разработать *план управления рисками*. План управления рисками описывает ответ, который будет приниматься для каждого риска, при его появлении [4].

Чтобы быть эффективным, мониторинг всех этапов разработки программного продукта – особенно в ключевых и критических точках – должен быть неотъемлемой частью большинства мероприятий проекта. По сути это означает частую проверку правильности выполнения хода проекта во время совещаний по проекту и отработку критических событий. Такой мониторинг включает в себя:

- своевременную (плановую) публикацию отчетов о состоянии проекта и включение вопросов управления рисками;
- проверку планов отработки рисков в соответствии с любыми крупными изменениями в расписании проекта;

- анализ и смену приоритетов рисков, отсеивая риски с наименьшей вероятностью;
- мозговой штурм при возможном появлении новых рисков после внесения изменений в расписание или в содержание проекта.

При возникновении риска, следует принять соответствующие меры по смягчению последствий, исходя из плана управления рисками. Снижение рисков включает в себя [15]:

- *Принятие*: Признание того, что риск влияет на проект и его принятие без каких-либо изменений в проекте.
- *Обхождение*: Урегулирование объема проекта, ограничений или графика чтобы минимизировать влияние риска.
- *Контроль*: Принятие мер для сведения к минимуму воздействия или снижения интенсивности риска.
- *Передача*: Внедрение организационных изменений в подотчетность, ответственность или полномочия для других заинтересованных сторон, которые будут принимать риск.
- *Постоянный мониторинг*: часто подходит для рисков с низким уровнем воздействия, контроль среды проекта с потенциально увеличивающимся воздействием риска.
- *Коммуникации*.

На протяжении всего проекта важно обеспечить обратную связь со всеми заинтересованными сторонами, менеджерами и разработчиками. Обмен информацией о рисках значительно повысит вероятность успеха проекта.

В связи с особенностями распределенных компаний появляются еще несколько видов рисков, которые могут стать критическими для проекта по разработке программного обеспечения [16]:

- отсутствие или недостаток общения
- проект, критически важный для организации
- интеграция компонентов проекта

Согласно А. Рид и Л. Найт [17], в распределенных командах (проектах) наибольшее влияние на успешность совместной работы оказывают коммуникационные риски, то есть отсутствие или недостаток общения. Существует несколько причин, по которым *коммуникационный риск* будет выше в распределенных организациях. Во-первых, как было отмечено выше, члены команды практически не общаются лицом к лицу, они должны полагаться на информационные технологии такие, как видеоконференции, электронные письма, средства совместной работы и обмен мгновенными сообщениями. Важным для менеджеров по управлению проектами является использование одного набора средств коммуникации, организованных единообразным образом, а затем обучение всего персонала инструментам стандартной организации и использования.

Фактор риска проекта, критически важный для организации, соответствует уровню важности проекта по сравнению с другими проектами в портфеле проектов организации.

Подчеркнем, что степень интеграции компонентов и участников проекта – это сложный показатель риска, указывающий на *количество и на сложность интерфейсов*, требуемых проектом. Исходя из всего сказанного ранее, можно сделать вывод, что распределенные компании подвержены большему количеству рисков, нежели традиционные организации. Поэтому распределенную организацию важно обеспечить специальными приемами риск-менеджмента, которые могли бы идентифицировать, ранжировать и управлять рисками, а также соответствующими интерфейсами «внешняя среда-проект», внутренними проектными интерфейсами между распределенными средствами и участниками проекта, программно-аппаратными интерфейсами распределенных программных приложений и соответствующими наборами пользовательских графических интерфейсов.

Измерения в разработке программного обеспечения

Важность измерений и их роли в совершенствовании методов управления и проектирования широко признается. Эффективное измерение стало одним

из краеугольных камней организационной зрелости. Измерения могут применяться к организациям, проектам, процессам и рабочим продуктам. Если управление проектом или компанией осуществляется по результатам измерений, а эти показатели недостаточно обоснованы, недостаточно понятны и не тесно связаны с атрибутами, которые они должны измерять, искажения и дисфункциональность измерений – закономерное явление, которое не принесет никакой пользы. Это в большой степени относится к измерениям, необходимым для идентификации рисков и эффективного управления ими. В создании и поддержке процесса измерений можно выделить следующие аспекты [4].

- *Формирование требований к измерениям*. Каждое мероприятие по измерению должно основываться на целях организации и на наборе требований к измерениям, установленных организацией и проектом.
- *Обозначение содержания измерений*. Необходимо установить организационную единицу, к которой будет применяться каждое измерение. Это может быть функциональная область, один проект, сайт или всё предприятие. Дальнейшее сопровождение измерений, относящихся к соответствующим требованиям, должно проводиться в пределах выбранного содержания измерений. Также необходимо определить все заинтересованные стороны, задействованные в осуществлении измерений.
- *Участие менеджеров и команды* в осуществлении измерениях. Обязательство участвовать должно быть официально установлено, донесено до персонала и поддержано ресурсами.
- *Обеспечение измерений необходимыми ресурсами*. В данном случае очень важна поддержка организации в осуществлении измерений, так как для реализации процесса ведения измерений необходимы ресурсы. С выделением ресурсов, распределяется ответственность за различные задачи измерительного процесса. Необходимо провести обучение,

выделить адекватное финансирование и инструменты, а также обеспечить поддержку для проведения этого процесса.

- *Выделение организационной единицы*. Организационная единица обеспечивает контекст измерений, поэтому он должен быть явным и включать в себя ограничения, налагаемые организацией на процесс измерения. Характеристика может быть определена с точки зрения организационных процессов, областей применения, технологии, организационных интерфейсов и организационной структуры. Подробнее про организационные единицы можно посмотреть в соответствующем разделе стандарта ISO 15939-02.
- *Определение информационных потребностей*. Информационные потребности основаны на целях, ограничениях, рисках и проблемах организационной единицы. Они могут быть основаны на деловых, организационных, нормативных и / или продуктовых целях. Все, перечисленное выше, должно быть идентифицировано и расставлено по приоритетам, документировано и передано заинтересованным сторонам, которые проведут анализ аспектов измерений.
- *Выделение метрик*. Отбор кандидатных метрик должен производится в соответствии с приоритетами информационными потребностями, а также основываясь на других критериях, важных для проекта. Поскольку внутренние качественные характеристики часто не содержатся в обязательных для контракта условиях программного обеспечения, важно рассмотреть возможность измерения внутреннего качества программного обеспечения, чтобы обеспечить ранний индикатор потенциальных проблем, которые могут повлиять Внешних заинтересованных сторон.
- *Определение процедур сбора, анализа и отчетности*. Это могут быть такие процедуры, как сбор данных и графики, хранение, верификация, анализ, отчетность и управление конфигурацией данных.

- *Выбор критериев оценки информационных продуктов*. На них оказывают влияние определенные ранее технические и бизнес-цели организационной единицы. Полученные результаты измерений связанны с производимым продуктом, а также с процессами, используемыми для управления и измерения проекта.
- *Предоставление ресурсов для измерений*.
- *Внедрение и овладевание* вспомогательными технологиями.

После планирования измерений идет процесс выполнения измерений и их последующая оценка. Определяются сильные и слабые стороны процесса, рассматривается его дальнейшее улучшение. Специально отметим, что измерения крайне необходимы при автоматизации процесса мониторинга, так как они дают необходимую информацию для принятия эффективных решений в ключевых точках и критических ситуациях.

1.5. Выводы по главе 1

В соответствии с поставленными задачами в первой главе были проанализированы общие принципы управления программными проектами и выделены *ключевые управленческие моменты*, характерные для распределенных компаний, такие как управление требованиями, управление персоналом и коммуникациями, управление качеством, рисками и измерениями.

Было проведено сравнение организации деятельности команд разработчиков в традиционных и распределенных компания, где были выделены коммуникационные, структурные, технические особенности управления распределенными командами разработчиков.

Были выявлены *ключевые риски*, характерные для распределенных компаний, такие как отсутствие или недостаток общения, проект, критически важный для организации, интеграция компонентов проекта. Важно отметить, что помимо этих рисков распределенные компании в большей мере подвержены и большинству остальных рисков, характерных проектам

разработки программного обеспечения в традиционных компаниях. Основываясь на особенностях распределенных компаний, была выбрана наилучшая методология разработки программного обеспечения, а именно комбинирование гибких методов и спиральной модели, которая сочетает в себе усиленный мониторинг рисков и гибкость. Благодаря анализу управления измерениями мы отметили их важность в автоматизации процесса мониторинга проектной работы.

2. Мультиагентный подход к управлению проектами в распределенных компаниях

2.1. Мультиагентный подход как организационное решение управленческих задач в распределенных компаниях

В настоящее время выбор эффективных систем управления и организационных решений оказывает большое влияние на успешность компании, ее зрелость и возможности развития. Как уже говорилось ранее, существует множество подходов, мер, идей, на основе которых осуществляется управление. В этом плане рассмотрим перспективный агентно-ориентированный подход, который все чаще используется при разработке мониторинговых и управленческих систем [3]. В реализации данного подхода применяются мультиагентные технологии, которые представляют собой альтернативу решения проблем бизнес-процессов. Согласно [18] применение агентных технологий в информационных системах оправдано следующими ее характеристиками:

- область знаний включает в себя внутреннее распределение данных, возможности решения проблем и обязанности;
- необходимо сохранить автономность подчастей, не теряя организационную структуру;
- взаимодействия сложны, они включают в себя переговоры, обмен информацией и координацию;
- решение проблемы не может быть полностью описано из-за возмущений в окружающей среде в реальном времени (например, сбоев оборудования) и естественной динамики бизнес-процесса.

Подход к проектированию информационных систем требует моделей, которые представляли бы их статические, функциональные и динамические свойства. При разработке мультиагентной системы необходимо рассмотреть два уровня абстракции: микро (агентный) и макро (социальный). На микроуровне показывается архитектура агента и его внутреннее поведение.

Макроуровень представляет собой общество агентов, определяющее цель системы и ее требования.

Рассмотрим микроуровень, для этого сначала определим, что же является агентом. Агента можно рассматривать, как субъект (программу), который выполняет порученную ему задачу. В настоящее время активно ведутся исследования как в области искусственного интеллекта, так и прикладного и системного программирования, и для каждого конкретного случая понятия немного отличаются друг от друга.

Важными характеристиками интеллектуальных агентов является их способность совершать действия [19]:

- *автономно*, то есть действовать от лица владельца, не требуя вмешательства этого субъекта права собственности
- *рационально*, опираясь на желания и выбирая наиболее значимые цели – намерения
- *адаптивно*, реагируя в реальном времени на изменения внешней среды, в которую он помещен
- *взаимодействуя* с другими агентами или людьми, для достижения целей
- быстро *изучая* большие объемы данных

Мною была разработана модель интеллектуального агента, отражающая его основные действия и свойства (рис. 4).

Существует множество примеров, сложность которых может быть существенно понижена путем рассмотрения проблемы с точки зрения использования нескольких агентов. Например, светофоры, регулирующие движение транспортных средств в городе, или камеры наблюдения, отслеживающие движущиеся цели, как правило, осуществляются централизованным образом. Однако централизованное адаптивное поведение для всех светофоров или камер в городе – это дорогостоящие задачи из-за высоких вычислительных затрат, «проклятия размерности» и проблемы единственной точки отказа. Более того, многие сложные проблемы по сути

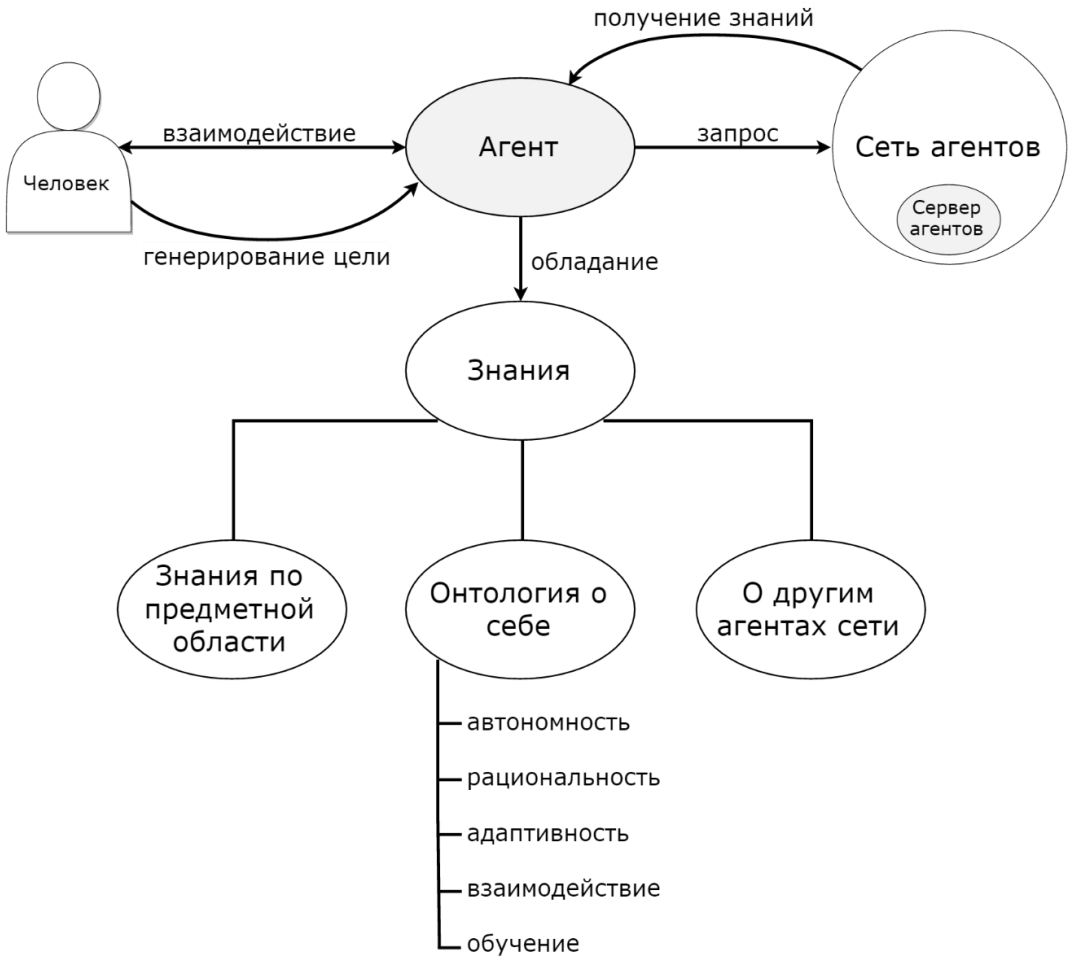


Рис. 4 Модель агента

децентрализованы. Централизованное управление просто недоступно и дорого стоит при решении таких проблем, как компьютерные устройства, обменивающиеся данными по беспроводному каналу, или передвижные роботы, исследующие незнакомые местности. В этих условиях отдельные агенты просто не смогут самостоятельно выполнить свои проектные задания [20].

Поэтому будем рассматривать децентрализованный класс мультиагентных систем, в котором принимается во внимание деятельность автономного агента в динамическом мультиагентном мире. Каждый агент обладает своими знаниями и ресурсами и может взаимодействовать с другими агентами, кооперироваться для достижения поставленной цели.

Мультиагентная система, по сравнению с адаптивной, сложнее, так как она способна быстрому обучению и обладает повышенной эффективностью за

счет переназначения задач и функций между агентами. В качестве миров агентов можно брать сложную систему, поскольку ее фундаментальные идеи совпадают с характеристиками интеллектуальных агентов. В результате взаимодействия агентов в сложной системе формируются структуры, часто граничащие между порядком и хаосом [19].

Распределенные компании зачастую создают сложную динамическую структуру, которой необходимо эффективно управлять. В качестве инструмента управления такими структурами целесообразно выбрать мультиагентные системы с интеллектуальными агентами. Для корректного функционирования мультиагентной системы важно, чтобы компании использовали ее непрерывно, позволяя системе накапливать информацию, обучаться и изменять свое поведение. Эффективность применения мультиагентных систем обуславливается ее основными свойствами: объектно-ориентированностью, адаптивностью, гибкостью, способностью быстро реагировать на изменения внешней среды и использованием интеллектуальных агентов.

2.2. Возможность автоматизации ключевых процессов проектного управления в распределенных компаниях на базе мультиагентного подхода

Как было сказано ранее, мы рассматриваем распределенные компании, которые по своей природе подвержены большему количеству рисков. Для того, чтобы снизить влияние рисков, необходимо определить и контролировать процессы, в которых они могут возникнуть. В традиционных компаниях обычно это задача ложится на плечи риск-менеджеров. В распределенных же компаниях стараются минимизировать число сотрудников для снижения финансовых затрат, поэтому в большинстве случаев рисками управляет менеджер проекта. Используя *мультиагентный подход*, можно значительно облегчить работу менеджера, автоматизируя некоторые мониторинговые и управленческие процессы и делегируя полномочия от агента-человека к агенту-программе. Существуют технико-формальные

процедуры, такие как сбор информации по мониторингу ситуации, анализ ситуации, сопоставление с утвержденным планом реализации по ключевым параметрам, поиск аналогов решения, и процедуры принятия решения, в которых необходимо участие экспертной системы или человека. Рассмотрим некоторые процессы управления проектами и выделим в них характерные управленческие моменты, которые могут контролироваться соответствующими взаимодействующими агентами в мультиагентной системе управления проектными инцидентами [5].

1. Обсуждение и определение требований

Требования являются основой всего, чему следуют в программном проекте. Поэтому важно, чтобы менеджер проекта понимал природу разработки требований и участвовал в процессе их разработки. Рассмотрим процесс сбора требований (рис. 5).

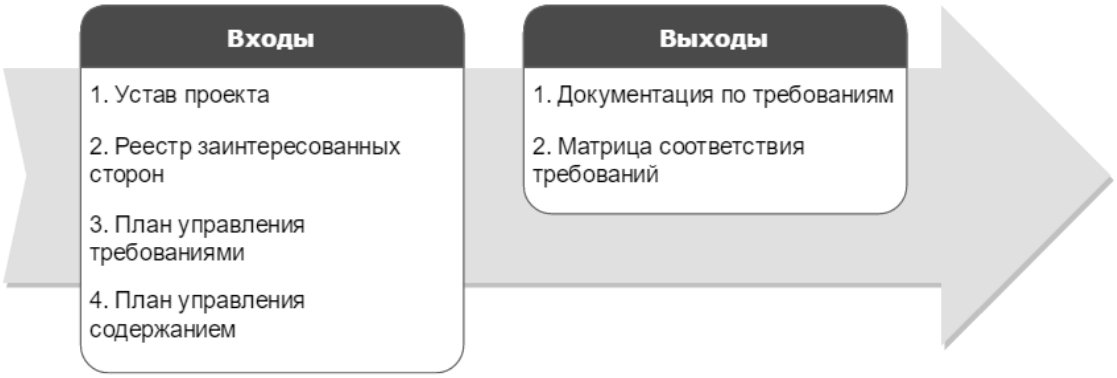


Рис. 5. Сбор требований: входы и выходы

Разработка требований охватывает следующие виды деятельности [21]:

- *сбор*: сбор и документирование потребностей пользователей, ожиданий клиентов и условий покупателя
- *анализ*: перевод потребностей пользователей, ожиданий клиентов и условий покупателя в технические требования к программному обеспечению
- *распределение*: прикрепление требований к аппаратным средствам, программным средствам и людям

- *спецификацию*: документирование технических требований в стандартных представлениях и форматах
- *верификацию*: определение того, что технические спецификации являются правильными, полными и совместимыми
- *согласование*: достижение консенсуса заинтересованными сторонами относительно требований
- *принятие*: подтверждение исходных требований всеми заинтересованными сторонами.

Некоторые управленческие моменты, связанные со сбором требований можно поручить агенту-программе. Например, спецификацию, верификацию и согласование требований.

Верификация требований определяет степень правильности, полноты и соответствия технических спецификаций в отношении требований к функционированию и ограничений процесса. Полнота технических спецификаций определяется демонстрацией того, что они охватывают все требования к функционированию, например, строится матрица соответствия. Таксономия проблем с требованиями может использоваться в качестве контрольного списка для проверки требований. Типичный такой список представлен в Приложении 1.

Мы рассматриваем распределенные компании, а значит чаще всего заинтересованные стороны и менеджер проекта не могут лично встретиться для обсуждения и принятия требований. Согласование требований в такой компании можно поручить программному агенту, который по введенному параметру оценивал бы степень принятия требований каждой стороной, принимал решение относительно сложных ситуаций и в итоге документировал требования с последующим построением матрицы соответствия требований.

2. План трудовых ресурсов

На данном этапе происходит распределение задач, основанных на требованиях к проекту, между членами команды разработчиков. Строится диаграмма Гантта и матрица ответственности проекта, по которым в

дальнейшем будет происходить мониторинг выполнения задач. На рисунке 6 можно видеть входы и выхода процесса составление расписания.

Для того, чтобы была возможна автоматизация этого процесса, менеджеру проекта необходимо создать базу данных членов команды разработчиков, с помощью которой агент смог бы принимать решение о назначении конкретного разработчика на задачу. Грамотно составленный список задач, основанный на требованиях, будет способствовать быстрому распределению членов команды по задачам.

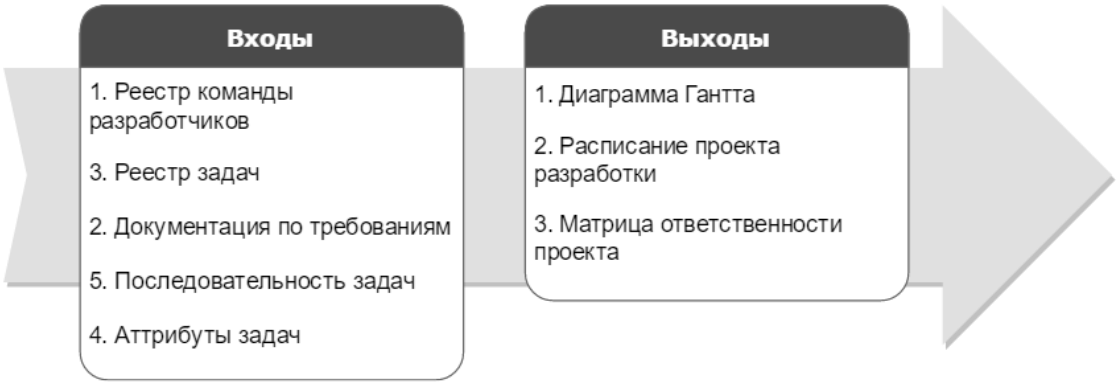


Рис. 6. Составление расписания: входы и выходы

3. Мониторинг и управление работами проекта

В эту группу процессов можно включить отслеживание, анализ, ведение отчетности о прогрессе проекта для достижения поставленных целей, определенных в плане проекта.

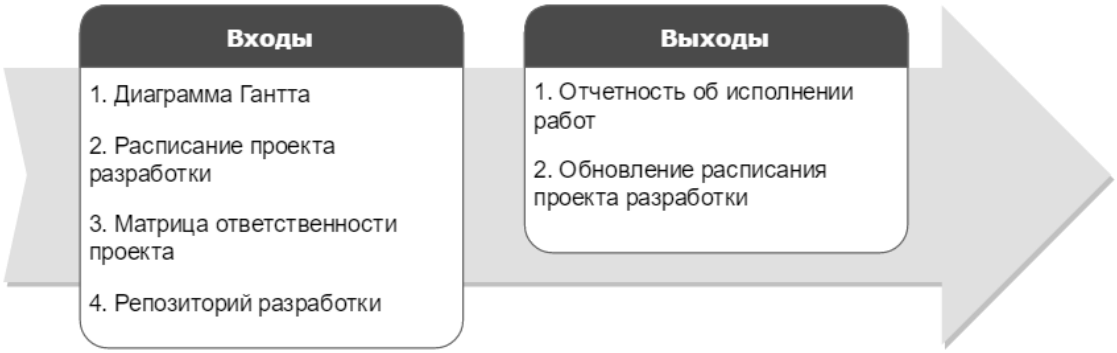


Рис. 7. Мониторинг выполнения задач: входы и выходы

Ключевым преимуществом этого процесса является то, что он позволяет заинтересованным сторонам понять текущее состояние проекта, предпринятые шаги, бюджет и график проекта.

В данном случае агенту можно поручить мониторинг выполнения задач и составление отчетности по проделанной работе. Работая с агентом, а не отдельно с каждым членом команды, менеджеру проекта будет значительно легче отслеживать прогресс по проекту. На рисунке 7 представлены входы и выходы процесса мониторинга выполнения задач.

4. Планирование коммуникаций.

Планирование коммуникаций проекта осуществляется на основе требования заинтересованных лиц или команды разработчиков провести митинг. Так как мы рассматриваем распределенные компании, менеджеру проекта пришлось бы согласовывать время встречи с каждым ее участником, потому что время работы сотрудников может отличаться из-за часовых поясов. На рисунке 8 представлены входы и выходы процесса планирование коммуникаций.

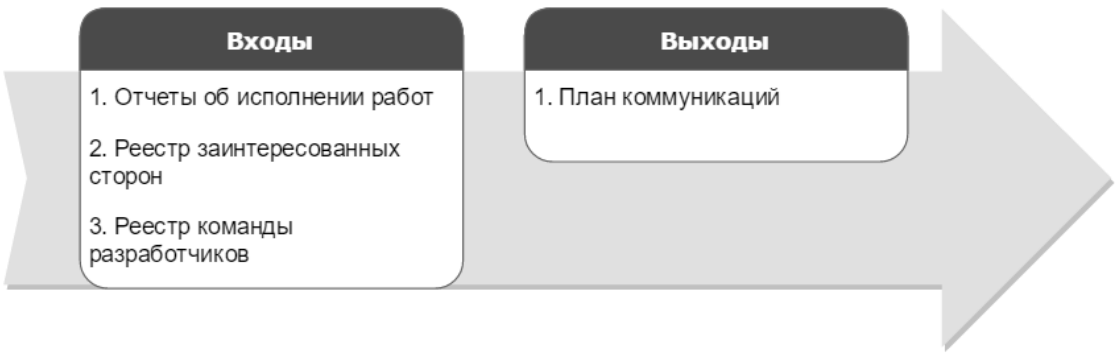


Рис. 8. Планирование коммуникаций: входы и выходы

Этот процесс можно автоматизировать, поручив агенту собирать информацию о предполагаемых участников митинга, корректировать время встречи и состав, а также заблаговременно оповещать участников встречи, освобождая менеджера проекта от этой технико-формальной деятельности.

5. Планирование управления рисками

Процесс управления рисками очень трудоемкий. Необходимо проанализировать все возможные риски, которые могут повлиять на проект и задокументировать их характеристики, чтобы найти способы контроля и предотвращения неблагоприятного влияния рисков. На рисунке 9 показаны входы и выходы процесса управления рисками.

Воспользуемся методологией FMEA (Failure Mode and Effects Analysis – анализ видов и последствий отказов) для проведения анализа и выявления критических моментов с целью управления качеством проекта разработки. FMEA проводится для оценки возможных проблем на ранних этапах

Рис. 9 Планирование управления рисками: входы и выходы

выполнения, где проще реализовать действия по преодоления этих проблем, но также может использоваться с уже существующими продуктами или процессами. FMEA может применяться для распознавания вероятных отказов, завершения их воздействия на процесс продукта и категоризации действий для уменьшения отказов.

Расчет числа риска потребители (RPN) в методе FMEA производится умножением трех компонент:

- Степени тяжести (S),
- Вероятности возникновения причины (O),
- Рейтинга обнаружения (D)

$$RPN = S * O * D$$

Эти компоненты описываются по 10-балльной шкале, следовательно, минимальное число риска потребителя – 1, а максимальное – 1000.

Процесс управления рисками можно автоматизировать: таблицу, состоящую из элементов отказов, их видов, возможных последствий, потенциальных причин, существующих методов контроля можно поручить

заполнять агенту, основываясь на базе знаний. Также в задачу агента будет входить подсчет числа риска потребителя, составления их рейтинга и документации.

2.3. Выводы по главе 2

Основываясь на задачах, поставленных во Введении, был рассмотрен мультиагентный подход, выступающий в качестве эффективного организационного решения для проектного мониторинга в распределенных компаниях.

Отмечено, что эффективность применения мультиагентных систем обуславливается ее основными свойствами: объектно-ориентированностью, адаптивностью, гибкостью, способностью быстро реагировать на изменения внешней среды и использованием интеллектуальных агентов.

Выявлены ключевые процессы управления проектами в распределенных компаниях, наиболее подверженные рискам и которые можно автоматизировать — такие как сбор требований, планирование трудовых ресурсов проекта, мониторинг работ проекта, планирование коммуникаций и планирование управление рисками. Были обоснована необходимость и описаны возможности автоматизации данных процессов.

3. Разработка прототипа управленческой подсистемы на базе интерфейсов и протоколов взаимодействия в агентной системе мониторинга проекта в распределенной организации

3.1. Методы моделирования корпоративных мультиагентных систем

В распределенных компаниях остро встает вопрос общения членов команды, менеджеров, других заинтересованных лиц и агентов. Например, необходимо с регулярной периодичностью проводить онлайн совещания, подстраиваясь под график работы и занятость сотрудников. Если компания сосредоточена в разных часовых поясах, то особое внимание нужно уделить согласованию времени митинга. Появляется необходимость в составлении динамического расписания в условиях ограниченных ресурсов. Расписание составляется исходя из уже существующих данных, предоставленных каждым участником встречи через своего агента, который знает все о «хозяине» и немного о других участниках. Затем между агентами группы ведутся переговоры до тех пор, пока не найдется согласованное решение, при этом агенты являются равноправными участниками переговоров со своей ролью, которая может динамически меняться.

Мы рассматриваем сотрудничество как воплощение *социальной способности агентов*. Агенты могут в определенной степени решать, когда, как и с кем взаимодействовать во время выполнения. Однако они должны соблюдать определенные протоколы сотрудничества для достижения поставленных целей.

Экспертами были исследованы общие проблемы, связанные с сотрудничеством. На основе теории речевого акта было предложено несколько ACL (Agent Communication Languages, языки коммуникаций агентов), в том числе KQML (Knowledge Query and Manipulation Language), FIPA ACL (Foundation for Intelligent Physical Agents) и другие. Зачастую для моделирования связи в мультиагентных системах используют графические обозначения и диаграммы. Например, AUML (Agent Unified Modeling

Language) описывает протоколы связи агентов в графической нотации, которая расширяет диаграммы последовательности UML. Существует совокупность *протоколов обмена информацией* между процессами, которые в нашем случае можно рассматривать в качестве информационных интерфейсов для взаимодействия конкретных агентов. В простейшем случае один агент действует как клиент и отправляет запрос другому агенту, действующему как сервер, а затем ожидает ответа. Приведем несколько вариаций моделирования сотрудничества в мультиагентных сетях:

1. Моделирование кооперативных мультиагентных систем, предложенное Л. Шанем и Х. Чжу.

Авторами этой статьи были разработаны SLABS (Specification Language for Agent-Based System, язык спецификации для агентной системы) и CAMLE (Caste-centric Agent-oriented Modelling Language and Environment, кастовый агентно-ориентированный язык моделирования и среда) [22]. В данной модели проблема рассматривается на трех уровнях:

- На *верхнем уровне* кастовая модель определяет архитектуру системы, группируя агентов в различные касты, которые можно грубо рассматривать как класс агента;
- На *среднем уровне* общение между агентами определяется моделью сотрудничества.
- На *низком уровне* модель поведения определяет внутреннее поведение различных агентов, чтобы их сотрудничество друг с другом осуществлялось посредством определенных действий в определенных сценариях.

Модель сотрудничества придерживается следующим принципам: инкапсуляция, свойства мультиагентов и совместное использование функций. Данная модель состоит из нескольких диаграмм кооперации. По горизонтали – диаграммы организованы как одна общая и некоторые диаграммы взаимодействия для конкретных сценариев. По вертикали – иерархия моделей

сотрудничества поддерживает моделирование кооперации с различной степенью детализации.

Агенты определяются как активные вычислительные объекты реального времени, которые инкапсулируют данные, операции и поведение и располагаются в их назначенных средах. Здесь данные представляют состояние агента. Операции - это действия, которые может предпринять агент. Поведение - это совокупность последовательностей изменений состояния и операций, выполняемых агентом в контексте его окружения. Под инкапсуляцией подразумевается, что состояние агента может быть изменено только им самим, и оно имеет свои собственные правила, которые управляют поведением агента в заданной среде.

$$\text{Агент} = < \text{Данные}, \text{Операции}, \text{Поведение} >_{\text{Окр. среда}}$$

В качестве расширения понятия класса в объектно-ориентированной архитектуре, берется понятие совокупности объектов, обозначаемое как *каста*, члены которой – набор агентов. Эти члены имеют набор структурных и поведенческих характеристик, определенных кастой. Агент может динамически изменять свое членство в касте во время своего существования, присоединяясь к касте или отступая от нее во время выполнения. Каста может наследоваться от ряда других каст.

$$\text{Каста} = \{\text{Агенты} \mid \text{структурные и поведенческие характеристики}\}$$

Механизм коммуникации состоит в том, что действия и состояния агента делятся на две части – видимую и невидимую. Агенты общаются друг с другом, предпринимая видимые действия и изменяя видимые переменные состояния, а также наблюдая за видимыми действиями и переменными состояниями другого агента.

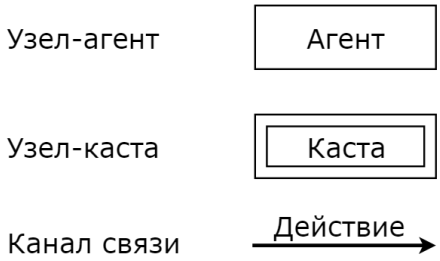


Рис. 10 Легенда диаграммы сотрудничества

На рисунке 11, узел-агент обозначает конкретных агентов, которые являются основными компонентами системы. Узел-каста обозначает любого агента в касте. Взаимодействие между агентами моделируется каналами связи, которые соединяют узлы агента /касты. Канал связи с подписанным действием от узла A до B, подразумевает, что агент A влияет на B: A принимает, а B наблюдает за действиями.

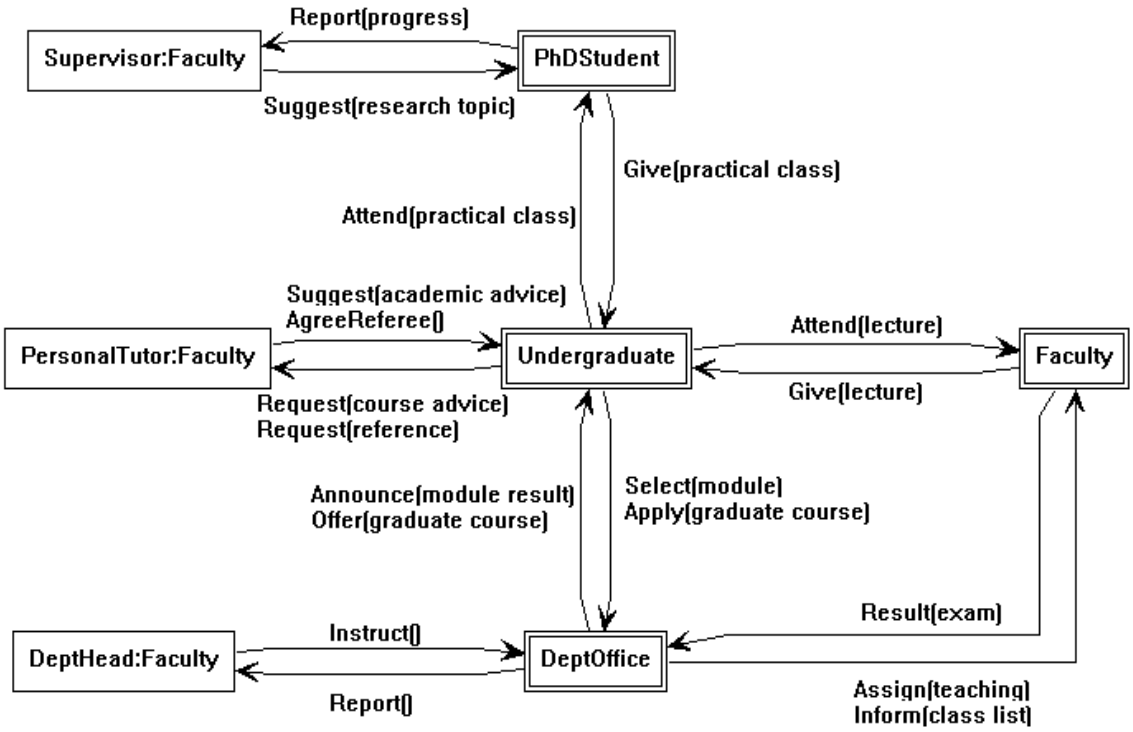


Рис. 11 Пример диаграммы сотрудничества [22]

2. Моделирование кооперативных мультиагентных систем Г. Гельфонда и Р. Уотсона.

В данной модели [23] в качестве языков программирования используются язык действий AL (Action Language) и CR-Prolog. Языки действий классифицируются здесь как декларативные языки для описания эффектов действий. Они имеют простой синтаксис и семантику, но все же достаточно мощны, чтобы представлять многие сложные логические рассуждения.

CR-Prolog – это расширение языка логического программирования A-Prolog, в котором представлены основанные на восстановлении правила согласованности.

Чтобы смоделировать способность агентов общаться, определение агента должно быть расширено. Это определение расширяется введением запросов, которые представляют собой язык для связи между агентами в данной системе. В зависимости от характера области моделирования эти запросы могут использоваться при формировании сообщений, которые агенты могут передавать между собой.

В данной модели агент определяется как $\text{Агент} = < F, A, R, D >$, где F – множество функций, A – множество элементарных действий, R – коллекция именованных множеств из F (запросы), D – описание действия на языке AL с сигнатурой $\Sigma = F \cup A$.

Мультиагентная система M определяется как конечное непустое множество агентов, которые:

$$\forall \alpha, \beta \in M, F_\alpha \cap F_\beta = \emptyset \wedge A_\alpha \cap A_\beta = \emptyset$$

Совместная мультиагентная система M определяется как $\langle C, S, M \rangle$, где

- C – это функция, которая задана агентом α и запросом $\mathrm{r} \in \mathrm{R}_\alpha$, возвращает набор агентов, известных как *клиенты* r.
- S – это функция, которая задана агентом α и запросом $\mathrm{r} \in \mathrm{R}_\alpha$, возвращает набор агентов, известных как *серверы* r.
- M – это множество упорядоченных пар вида $\langle r, \beta \rangle$, известных как *сообщения*, такие, что $\beta \in \mathrm{M}$, $\mathrm{r} \in \mathrm{R}_\alpha$ для некоторого $\alpha \neq \beta \in \mathrm{M}$ и $\beta \in \mathrm{S}(\alpha, \mathrm{r})$

В дополнение к вышесказанному C, S и M должны удовлетворять следующим свойствам:

- Если $\beta \in \mathrm{C}(\alpha, \mathrm{r})$, то $\forall \gamma \in \mathrm{M}, \beta \notin \mathrm{S}(\gamma, \mathrm{r})$.
- Если $\beta \in \mathrm{C}(\alpha, \mathrm{r})$, то $\alpha \in \mathrm{S}(\beta, \mathrm{r})$.
- Если $\beta \in \mathrm{S}(\alpha, \mathrm{r})$, то $\alpha \in \mathrm{C}(\beta, \mathrm{r})$.

Знание агента определяет пару диаграмм перехода, известных как локальные и глобальные диаграммы, которые используются агентом для обоснования его локальных и глобальных перспектив. Эти перспективы

зависят от того, как обрабатываются действия, связанные с передачей сообщений. Перспективы определяются модулем связи, который разделен на два подмодуля: локальный модуль, C_{local} и глобальный модуль, C_{global}. C_{local} определяется описанием действия на языке AL, в то время как C_{global} определяется логической программой в CR-Prolog.

При описании одноагентных систем поведение агента характеризуется тем, что называется агентом-циклом. По существу, агент выполняет следующие шаги:

- наблюдайте за состоянием мира;
- если эти наблюдения не совпадают с ожиданиями агента, определите причину несоответствия;
- выберите цель;
- сгенерируйте последовательность действий для достижения цели;
- выполните первый элемент последовательности.

Для мультиагентной системы структура цикла не меняется, изменяется лишь то, что агент выполняет шаги планирования и диагностики с использованием локальных и глобальных перспектив.

3. Моделирование связи и сотрудничества отказоустойчивых мультиагентных систем А.А. Гарсы, Х. Х. Серрано и др. (2007).

Авторы описали новую модель для интеллектуальных агентов [24], которые самостоятельно обрабатывают отказы, в которой рассматривают:

1) *рационального агента*: он всегда действует правильно.
2) *всеведущего агента*: он может быть определен как фактический результат действий, выполняемых агентами. Но авторы также отмечают, что в реальном мире всезнание вообще не существует.

Система на основе агентов может быть реализована без каких-либо структур программного обеспечения этого агента. Их можно назвать программными объектами, которые могут действовать от их имени. Особое внимание стоит уделить «встроенным знаниям. Если действия агента

полностью основаны на встроенных знаниях, так что ему не нужно обращать внимание на его восприятие, то мы говорим, что агенту не хватает автономии.

В данной модели используются:

- спецификации FIPA, которые представляют собой набор стандартов, предназначенных для содействия взаимодействию гетерогенных агентов и услуг, которые они могут представлять. Фонд интеллектуальных физических агентов (FIPA) является официальным комитетом стандартов IEEE.
- Язык коммуникации агентов ACL FIPA, в котором сообщение состоит из одного или нескольких элементов сообщения, необходимых для эффективной связи агентов. Такой набор элементов определяется в зависимости от ситуации, но подразумевается, что туда включаются отправитель, получатель и контент.
- Общение происходит на нескольких уровнях, содержимое сообщения – только часть коммуникации. С помощью KQML, программный агент передает содержимое сообщения, составленное на выбранном им языке.

Авторы представили различные коммуникационные протоколы агентов и диаграмму связи. Данный подход к системе мультиагентов помогает осуществлять отказоустойчивое управление, где каждый агент должен бороться с отказами, и обеспечивать такую систему необходимыми интерфейсами для эффективного взаимодействия агентов.

4. Моделирование совместной деятельности с использованием локальной связи для распределенных мультиагентных систем: А. Нагао, Т. Мики:

В данном методе [25] рассматривается распределенная мультиагентная система, в которой правила поведения агентов основаны на «boid» правилах («bird-oid object»), изображенных на рисунке 12.

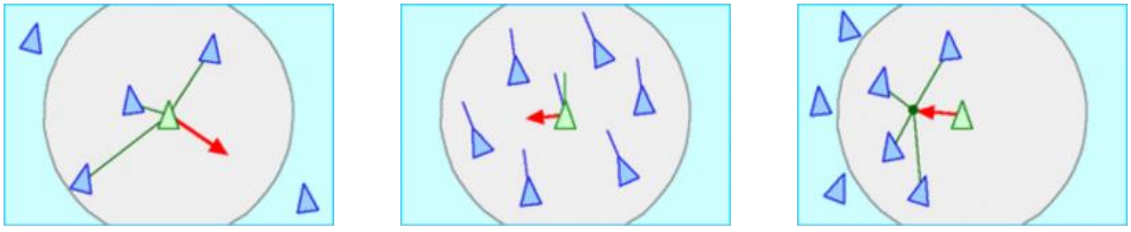


Рис. 12 «Boids» правила поведения

Данный метод предлагает простую архитектуру и низкие вычислительные затраты, в котором общая задача делится на подзадачи и достигается выполнением каждым агентом своей задачи автономно. В этом методе применяется подход, основанный на состоянии, в котором каждое состояние соответствует каждой подзадаче. Чтобы назначить подзадачи агентам, адаптируемым к прогрессу общей задачи, предложенный метод изменяет состояние с использованием локальной информации, полученной с помощью датчиков.

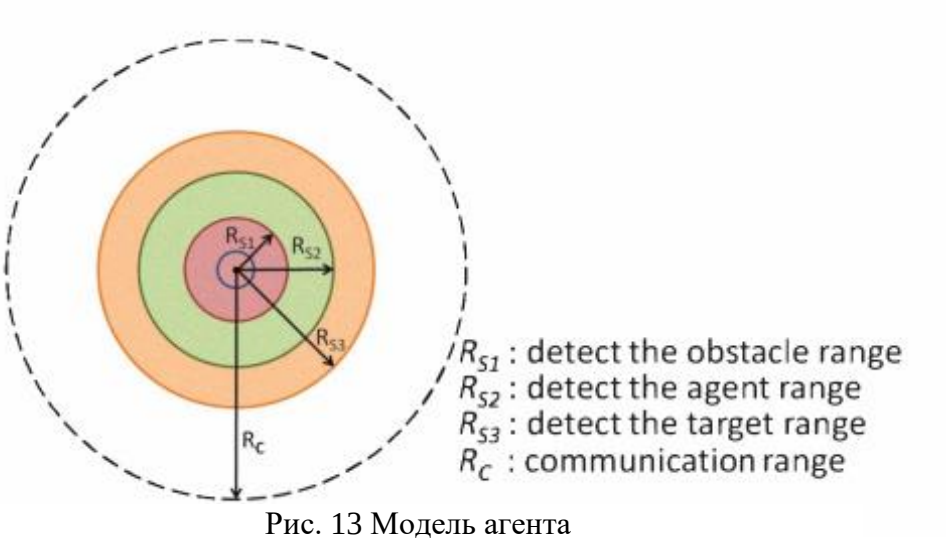


Рис. 13 Модель агента

Каждый агент воспринимает информацию об окружающей среде с помощью датчиков. Агент имеет три сенсора с разными диапазонами и локальное средство связи с ограниченным расстоянием, как показано на рисунке 13. Датчики с диапазонами обнаружения R_{S1}, R_{S2} и R_{S3} используются для обнаружения препятствий (датчик препятствий), состояния другого агента (датчик агента) и расстояния до цели (датчик дели) соответственно. Эти датчики имеют приоритеты, которые приводятся в следующем порядке:

датчик препятствий, датчик цели и датчик агента. Кроме того, локальная связь с ограниченным диапазоном используется для достижения прогресса соседнего агента в выполнении их локальной задачи. Каждый агент оценивает глобальную ситуацию, используя локальную связь. Каждый агент изменяет состояние в соответствии с ситуациями соседних агентов и прогрессом глобальной задачи.

В данной модели предложено пять состояний агента для решения сложной задачи:

1) *поиск*: поиск цели.
2) *найти*: приближение к цели.
3) *ждать*: ожидание около цели.
4) *окружение*: окружение цели.
5) *маяк*: трансляция позиции цели другим агентам.

5. Моделирование мультиагентных систем иерархическими цветными сетями Петри [26]:

Сети Петри хорошо применимы к системам с распределенными и параллельными событиями, поэтому их стали использовать в моделировании и анализе мультиагентных систем. С развитием мультиагентных технологий появились различные модификации сетей Петри, например, цветные сети Петри, которые сочетают в себе преимущества обычных сетей Петри и языки программирования высокого уровня. Использование цветных сетей Пети моделирует социальность, разговор и взаимодействие между агентами.

Цветные сети Петри, прикрепляющие цвет к каждому маркеру и набор цветов для каждого места, позволяют нам использовать меньше мест, чем необходимо в обычных сетях Петри, не теряя при этом никаких свойств. Основная идея иерархических цветных сетей Петри заключается в том, чтобы позволить разработчику модели построить большую модель, объединив ряд небольших цветных сетей Петри в большую сеть, которая аналогична ситуации. Всегда можно перевести иерархическую цветную сеть Петри в

неиерархическую, которая, в свою очередь, может быть преобразована в обычную сеть Петри, что означает, что теоретические моделирующие способности этих трех классов сетей Петри одинаковы.

В данной модели [26] структура агента основана на теории убеждений, желаний и намерений (BDI, Belief – Desire – Intention):

- профиль агента – оисание агента, имя, цели, намерения, убеждения
- модель процесса агента – описание агента, как работать, смоделированное с помощью цветных сетей Петри
- таблица сведений об агенте – реестр агентов, которые недавно с ним сотрудничали.

Модель процесса – это неиерархическая цветная сеть Петри, которая используется для моделирования механизма логики поведения агента, с помощью которой получают графики преобразования состояний и поведения агентов для достижения цели.

Мультиагентная система описывается на системном и на агентном уровнях. На системном уровне структура моделируется неиерархической цветной сетью, агенты представлены с данными об их переходах, благодаря чему можно считывать отношения между агентами и контролировать саму мультиагентную систему. На агентном уровне каждый агент моделируется, тем самым можно считать поведение каждого агента и его смену состояний для достижения целей. Так моделирует крупномасштабную систему на макро- и микроуровнях с иерархическим моделированием.

3.2. Моделирование агентов и их взаимодействия для эффективного управления проектами в распределенных компаниях

В соответствии с ключевыми процессами, выделенными в конце прошлой главы были разработаны алгоритмы действия для пяти агентов, предназначенных для мониторинга ситуаций и эффективного управления проектами в распределенных компаниях: агент сбора требований, агент планирования трудовых ресурсов, агент мониторинга работ проекта, агент

планирования коммуникаций и агент планирования анализа рисков. Рассмотрим подробнее агента мониторинга работ проекта.

В распределенных компаниях проект чаще всего хранится в некотором онлайн-репозитории. Благодаря агенту мониторинга у менеджера проекта не будет необходимости проверять каждого разработчика и требовать с него отчет. Агент сам соберет все необходимую информацию, составит отчет, напомнит разработчикам о необходимости загрузить прогресс в систему, определит, какие разработчики не укладываются в срок сдачи и оповестит об этом менеджера проекта. На рисунке 14 представлена разработанная нами блок-схема работы данного агента. Агент считывает базы данных «Задачи», «Команда», диаграмму Гантта и репозиторий, где хранится проект. Таблица «Задачи» содержит название задачи, ее описание, навыки, необходимые разработчику для ее осуществления, сложность задачи, сроки выполнения задачи, ее состояние («Активна», «Ожидает», «Завершена») и имя члена команды, которые ее реализует. В таблице «Команда» хранится информация о разработчиках, их навыках, занятости и времени работы. Следуя диаграмме Гантта и описанным выше базам данных, агент выбирает задачу с состоянием «Активна», определяет дату окончания выполнения задания в соответствии с расписанием, составленным агентом планирования трудовых ресурсов.

Если сегодня назначен «дедлайн» по каком-либо виду деятельности, то агент проверяет наличие отчетности по задаче в онлайн-репозитории, при ее отсутствии он принимает решение в соответствии с базой данных решений, в которой хранятся знания, накопленные из предыдущих проектов. При отсутствии решения агент передает свое полномочия по данной задаче менеджеру проекта.

На данном этапе происходит взаимодействия типа *«агент-человек»*. Менеджер проекта оценивает проблему, принимает решение и вносит его в базу данных, благодаря этого агент в будущем сможет сам справляться с проблемой такого рода.

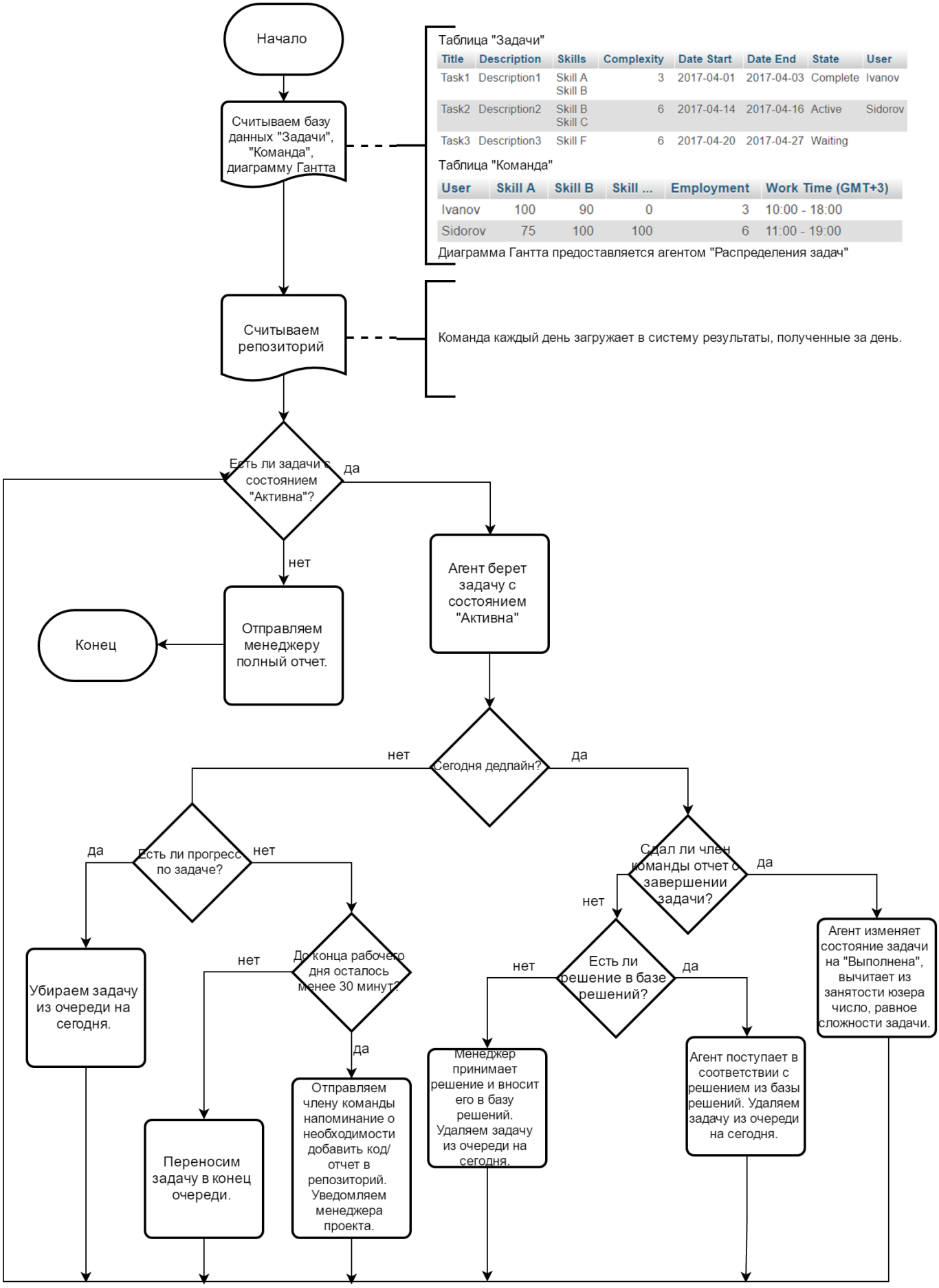


Рис. 14 Блок-схема действий «Агента мониторинга работ проекта»

Если «дедлайн» сегодня, и вся отчетность сдана, то агент мониторинга задач вычитает «сложность» задачи из «занятости» разработчика и оповещает агента планирования трудовых ресурсов о том, что освободился разработчик. Таким образом для данных агентов проявляется взаимодействие *«агент-агент»*. Если дата сдачи задачи еще не наступила, то агент снова «смотрит» на прогресс по выполнению задачи. При получении отчетности по задаче агент переходит к следующей активной задаче – если отчетности не поступало в репозиторий, то агент ждет, когда до конца рабочего дня разработчика останется полчаса и оповещает его о необходимости сдачи отчетности, переносит задачу в конец очереди и переходит к следующей. В конце рабочего дня последнего разработчика агент отправляет полную отчетность менеджеру проекта и оповещает его о все разработчиках, которые пропустили сдачу отчетности. В этом месте снова происходит взаимодействие *«агент-человек»*.

Модели остальных агентов представлены в следующих приложениях: Приложение 2 – «Агент сбора требований», Приложение 3 – «Агент планирования трудовых ресурсов», Приложение 4 – «Агент планирования коммуникаций» и Приложение 5 – «Агент планирования анализа рисков».

3.3. Моделирование взаимодействия между объектами и субъектами мультиагентной системы

Так как перед нами стоит задача разработать модель интерфейсного взаимодействия в мультиагентной системе, составим наглядную модель, представив ее в виде блок-схемы, которая будет показывать все объекты и субъекты системы и их каналы связи. На рисунке 15 представлены пять агентов системы (Агент сбора требований, Агент планирования трудовых ресурсов, Агент мониторинга работ проекта, Агент планирования коммуникаций и Агент планирования анализа рисков), менеджер проекта, команда разработчиков и заинтересованные стороны. Каждый агент системы имеет доступ к разделам базы знаний, которые хранят накопленный опыт принятия решений в различных ситуациях (пунктирная стрелка). При

возникновении критической ситуации, когда агент не может самостоятельно в рамках своего поведения и базы знаний решить текущую проблему, он отправляет менеджеру проекта запрос на добавление нового решения в базу знаний и передает ему полномочия, предоставляя полную информацию о ситуации (объемная стрелка).

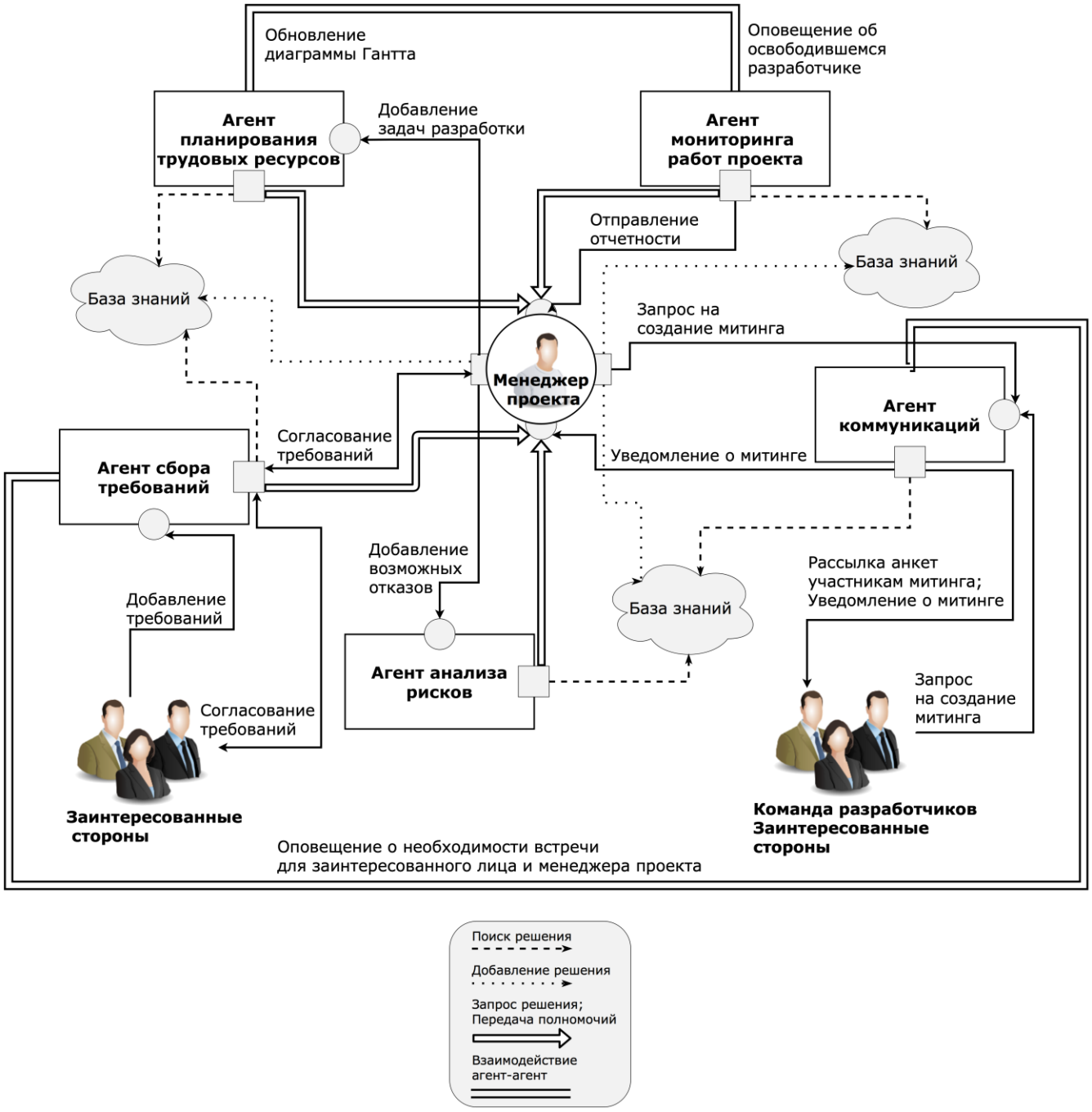


Рис. 15 Модель взаимодействия в мультиагентной системе для эффективного управления проектами в распределенной компании

В этом случае, менеджер проекта анализирует полученную информацию, принимает решение и вносит его в базу знаний (точечная пунктирная стрелка). Для простоты чтения схемы, база знаний была условна разбита на три части. Менеджер проекта, команда разработчиков и заинтересованные стороны могут взаимодействовать с агентами в рамках их поведения. Например, «Агенту планирования трудовых ресурсов» менеджер проекта может добавлять новые задачи, а «Агенту сбору требований» – новые требования.

Взаимодействие типа «агент-агент» было решено обозначить двумя сплошными линиями, поскольку общение между агентами может происходить в один момент времени. Например, «Агент планирования трудовых ресурсов» может обновить диаграмму Гантта в соответствии с новыми данными, а агент мониторинга работ проекта может отправить оповещение об освободившемся члене команды разработчиков для последующего распределения на задачи.

Для наглядности «входы» объектов взаимодействия были обозначены кружками, а «выходы» – квадратами. Стоит отметить, что во время сбора требований взаимодействие заинтересованных сторон и менеджера проекта происходит через агента сбора требований. Поскольку согласование – двусторонний процесс, для обозначения данного процесса мы используем двустороннюю стрелку.

3.4. Разработка компонентов интерфейса для взаимодействия «агент-человек»

В нашем случае агент – субъект (программа), и если рассматривать такой тип взаимодействия, как «агент-человек», то для человека «партнер» по диалогу – воображаемый. Для того, чтобы обеспечить пользователю более наглядную модель общения с агентом, нежели просто заполнение соответствующих полей баз данных, мы разработали компоненты системы интерфейсов в рамках нашего проекта по эффективному управлению проектами в распределенных компаниях.

Агент планирования коммуникаций

Рассмотрим действия «Агента планирования коммуникаций» (Приложение 4). Агент принимает запрос от пользователя на создание митинга, и для того, чтобы облегчить пользователю взаимодействие с агентом были разработаны соответствующие компоненты. В окне управления коммуникациями (рис. 16) пользователь может создавать, начинать и удалять митинги, а также добавлять участников в уже существующие встречи для дальнейшего согласования времени посредством агента. Для создания нового митинга была разработана специальная экранная форма (рис. 17). В ней нужно указать основную информацию о встрече (название, дату, время начала и описание встречи). Когда будут заполнены ключевые поля, агент начнет согласование, сначала отправит анкеты участникам митинга. У возможного участника встречи высветится оповещение с просьбой заполнить анкету, в которой будет дана вся информация о предстоящей встрече и поля для ответа («Да» или «Нет» и «Комментарии»). После этого агент будет действовать в соответствии с блок-схемой, представленной в Приложении 4.

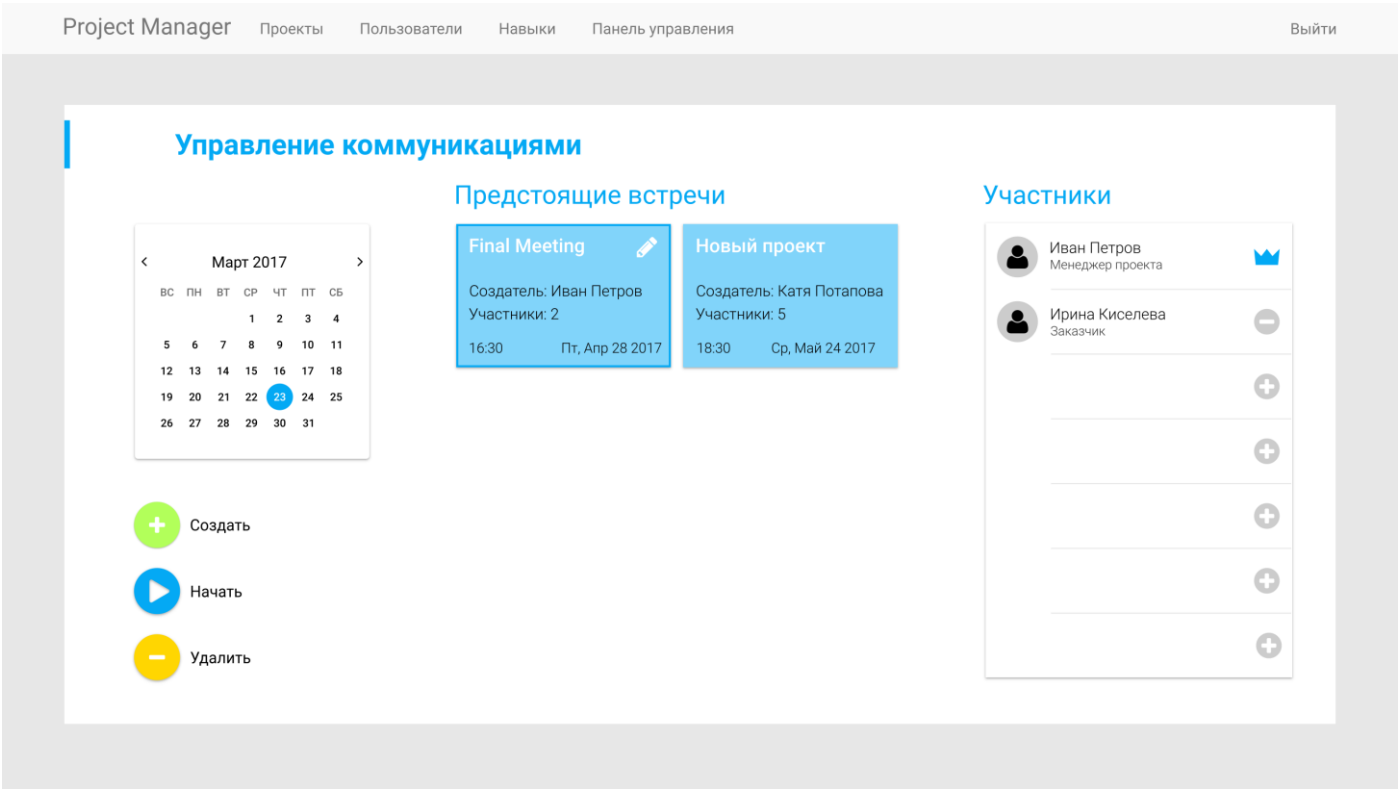


Рис. 16 Окно управления коммуникациями

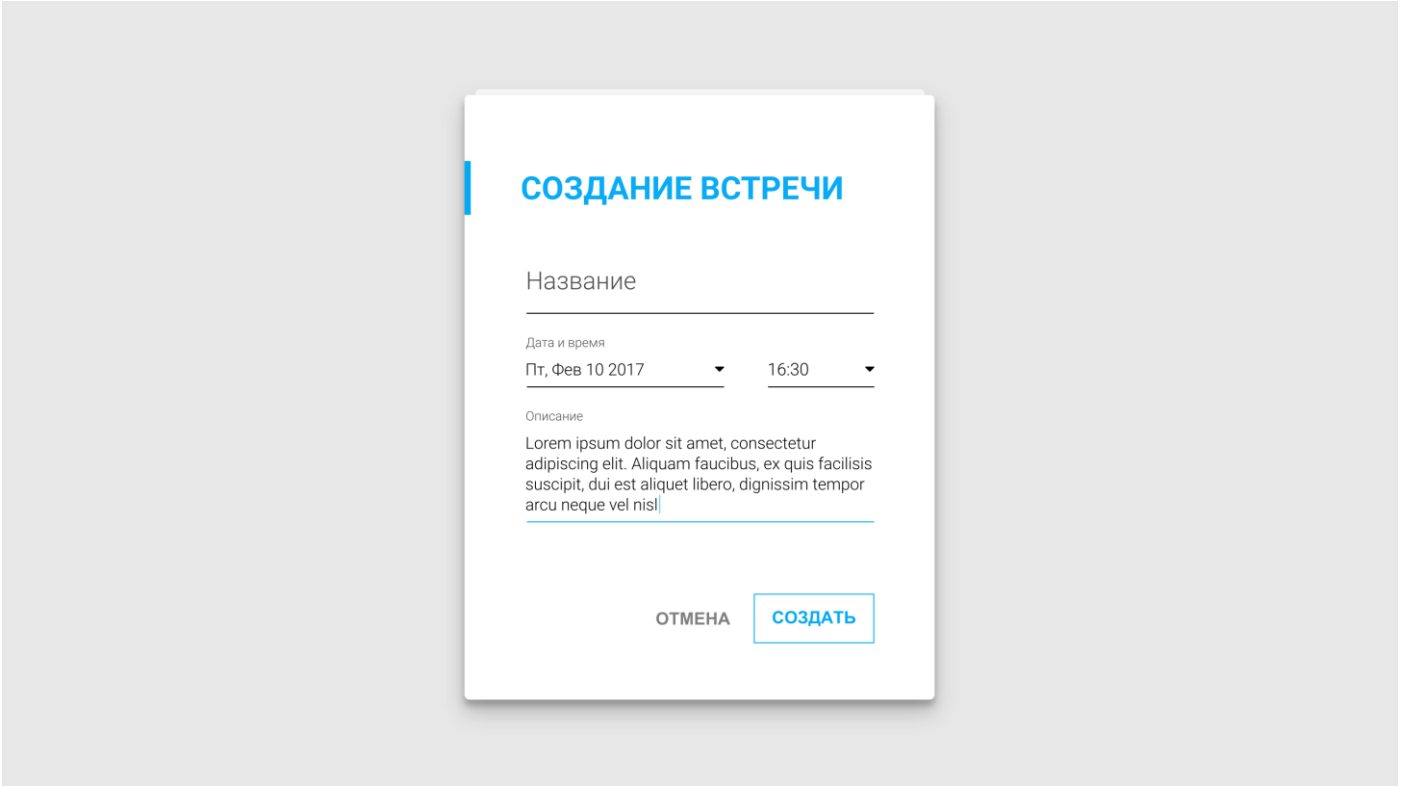


Рис. 17 Форма Создание встречи

Агент планирования трудовых ресурсов

Обратимся к агенту планирования трудовых ресурсов и определим ситуации, в которых происходит взаимодействие типа «агент-человек». Следуя блок-схеме действия Агента планирования трудовых ресурсов (Приложение 3), сначала менеджер проекта должен заполнить таблицы базы данных, такие как «Команда» и «Навыки» и закрепить необходимых исполнителей за *Проектом*. Данные действия являются подготовительными, но их также можно рассматривать с точки зрения взаимодействия между агентом и человеком, поскольку впоследствии агент будет обращаться к этим базам данных. Основное взаимодействие агента и менеджера проекта происходит, когда менеджер проекта добавляет новую задачу (рис. 18). После нажатия на кнопку *Создать*, агент, в соответствии с блок-схемой действий, приступает к назначению исполнителя на данную задачу. Если агент не смог справиться с задачей в рамках своего поведения и не нашел решения в текущей базе знаний, тогда агент сообщает менеджеру о неудаче и передает ему полномочия (рис. 19).

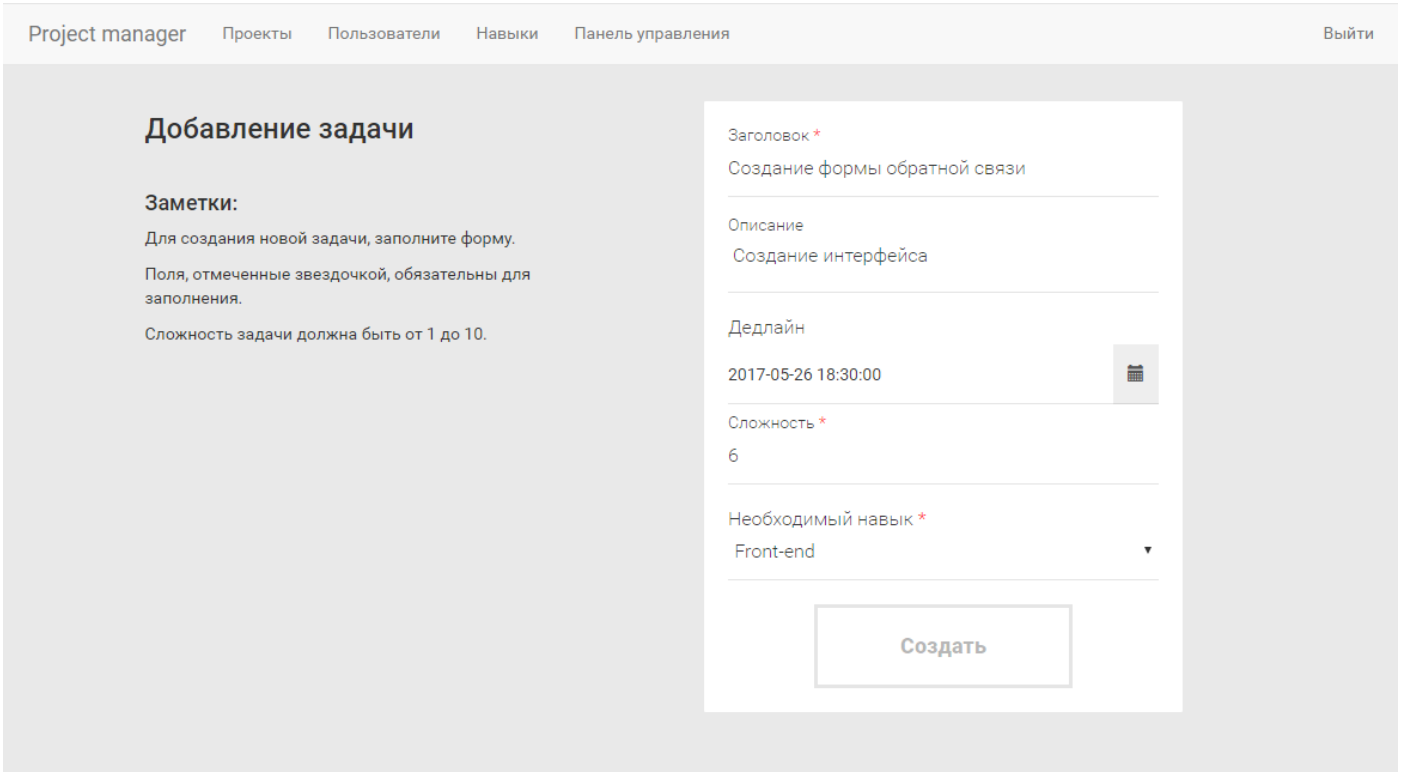


Рис. 18 Окно добавления задачи в проект

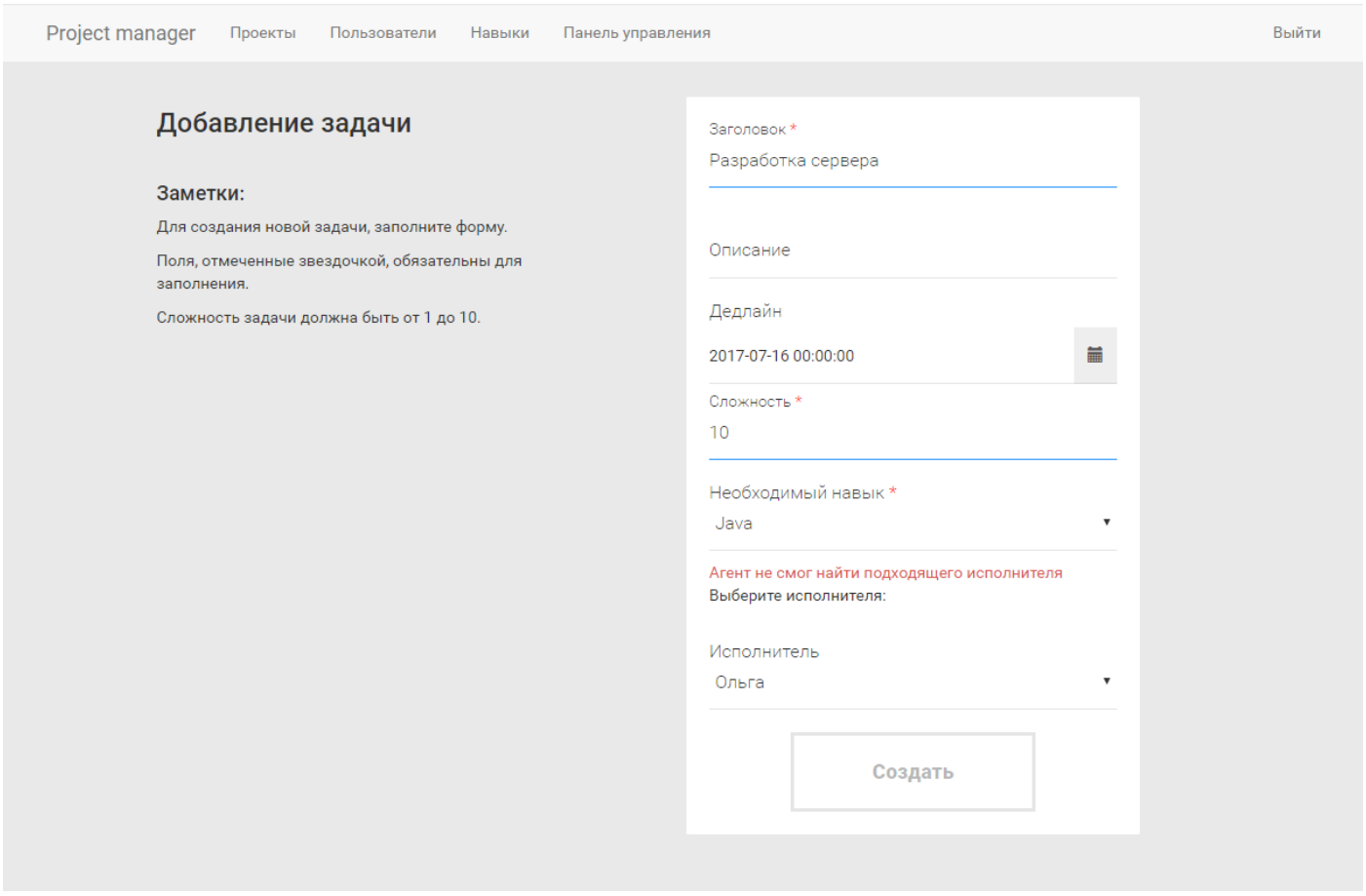


Рис. 19 Агент планирования трудовых ресурсов передает полномочия менеджеру проекта

Агент согласования требований

Теперь рассмотрим агента согласования требований, алгоритм действий которого представлен в виде блок-схемы в Приложении 2. Согласно данной блок-схеме агент принимает на вход таблицу «Требования», содержащую название и описание требований, которая заполняется заинтересованным лицом. Для данного взаимодействия «агент-человек» была создана соответствующая экранная форма (рис. 20). При получении требования агент начинает действовать согласно своему поведению, отправляя требование менеджеру проекта. В этом случае менеджер проекта получает форму согласования (рис. 21), в которой ему необходимо прочитать поля, заполненные заинтересованным лицом и заполнить поле *Возможность выполнения* и *Комментарий*. Поле *Возможность выполнения* может содержать числа 0, 0.5, 1, обозначающие «не можем выполнить», «частично», «полностью» соответственно. После заполнения этой формы, менеджер проекта отправляет ее агенту. Агент, руководствуясь алгоритму действий, добавляет требования с *Возможностью выполнения* «1» в базу данных, если значение поля отлично от единицы, то заинтересованной стороне отправляется форма изменения требований (рис.22). В случае, когда заказчик не готов поменять требование, агент уведомляет агента планирования коммуникаций о необходимой встрече, и на данном этапе происходит взаимодействие типа «агент-агент».

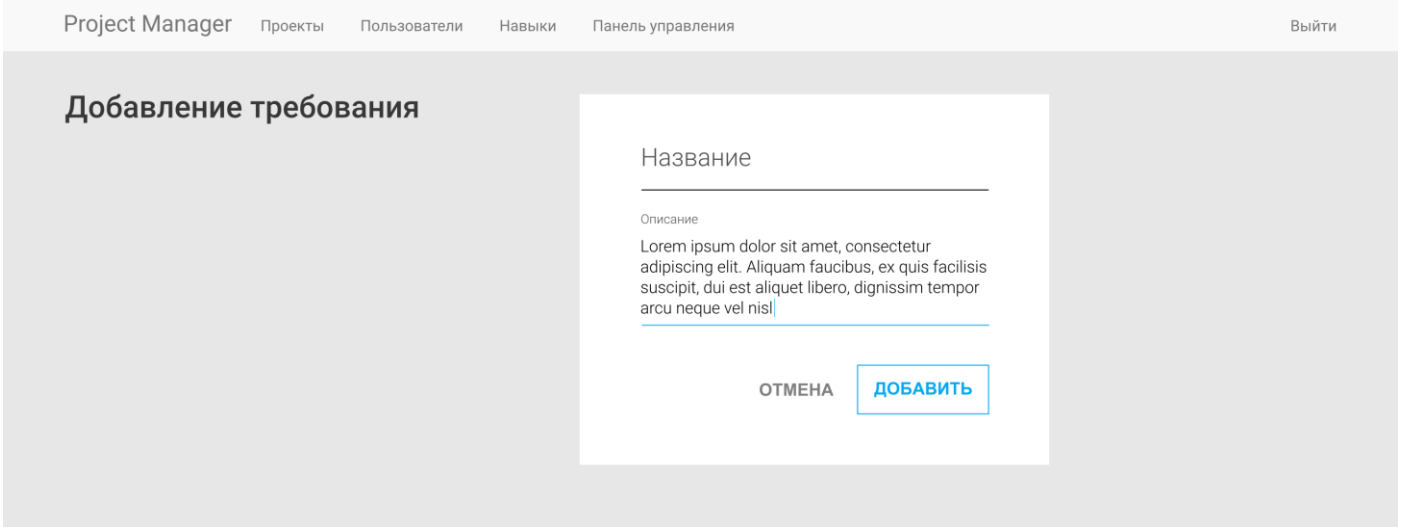


Рис. 20 Окно с формой *Добавление требования*

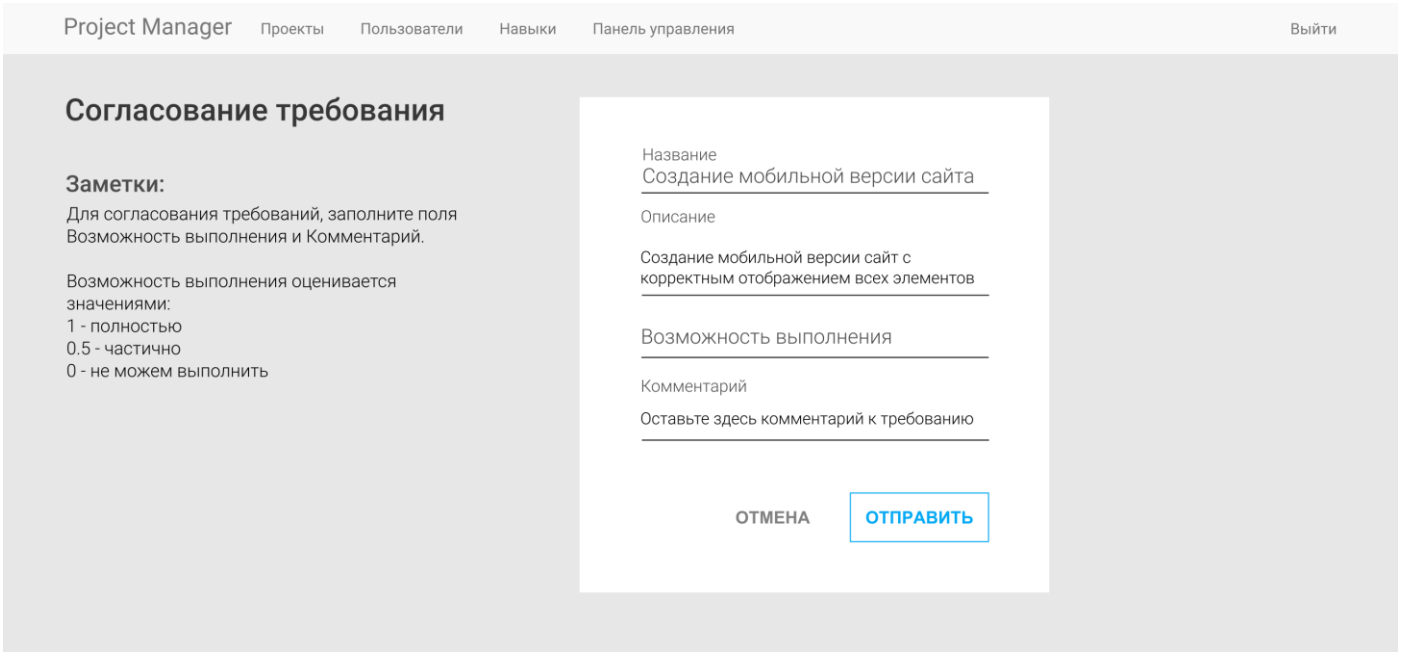


Рис. 22 Окно с формой *Согласование требования* для менеджера проекта

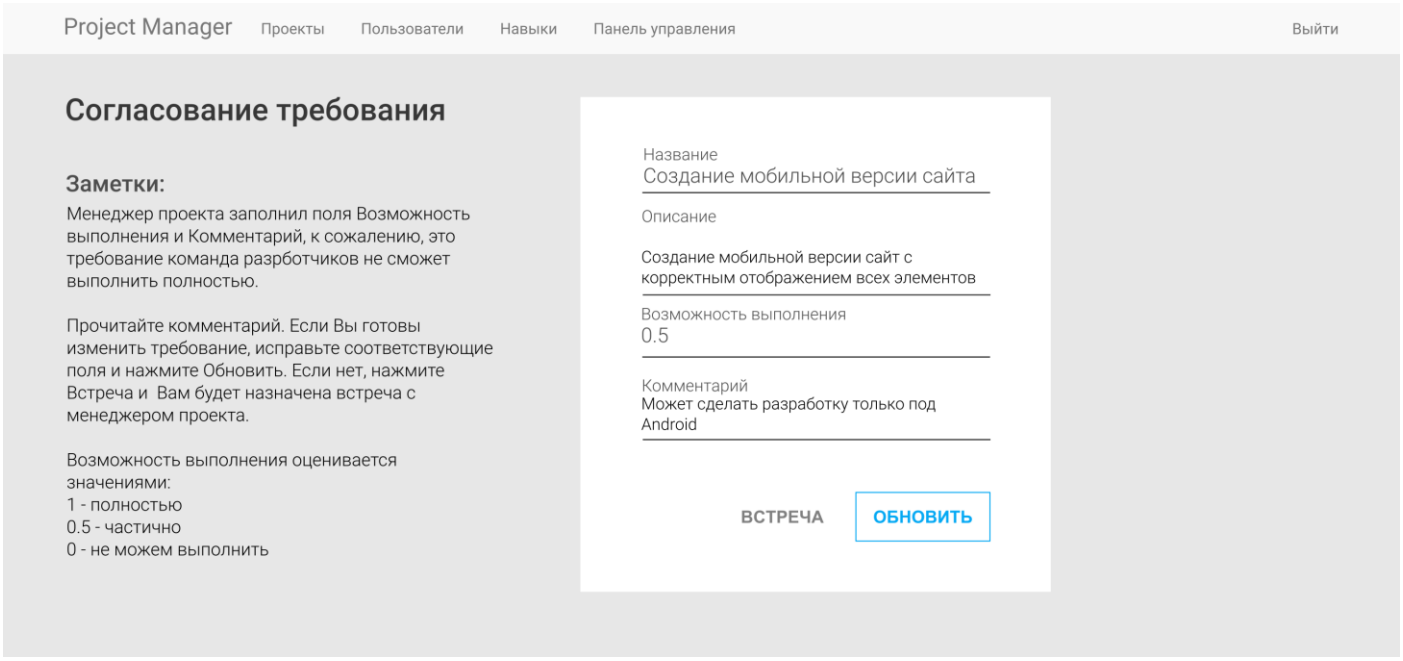


Рис. 21 Окно с формой *Согласование требования* для заказчика

Интерфейсы взаимодействия типа «агент-человек» остальных агентов были разработаны по такому же принципу: пользователь заполняет таблицы соответствующих баз данных не напрямую, а посредством специализированных экранных форм. Такой способ взаимодействия с агентом позволяет менеджеру визуально представить «невидимого» агента, а заметки к форме помогут новому пользователю быстро освоить Web-приложение.

Панель управления деятельностью агентов

Разработка Панели управления очень важна, поскольку одной из целей разработки Web-приложения для эффективного управления проектами в распределенных компаниях, был именно мониторинг. Построение данного приложения на базе мультиагентной системы позволило автоматизировать некоторые ключевые процессы управления проектами но, чтобы отслеживать деятельность системы и взаимодействовать с ней необходим соответствующий интерфейс, отображающий все пять агентов с их текущим состоянием (рис. 23).

Под каждым из агентов размещен блок, который окрашивается в синий, зеленый, оранжевый или красные цвета в зависимости от типа сообщения. Если агенту необходимо вмешательство менеджера, блок окрашивается в красный цвет, агент отправляет сообщение с информацией о проблеме и передает по ней полномочия менеджеру проекта. Если агент сталкивается с проблемой, но может решить ее в рамках своего поведения, то блок становится оранжевого цвета и носит информативный характер. По желанию, менеджер проекта может нажать на блок и посмотреть подробную информацию, а также

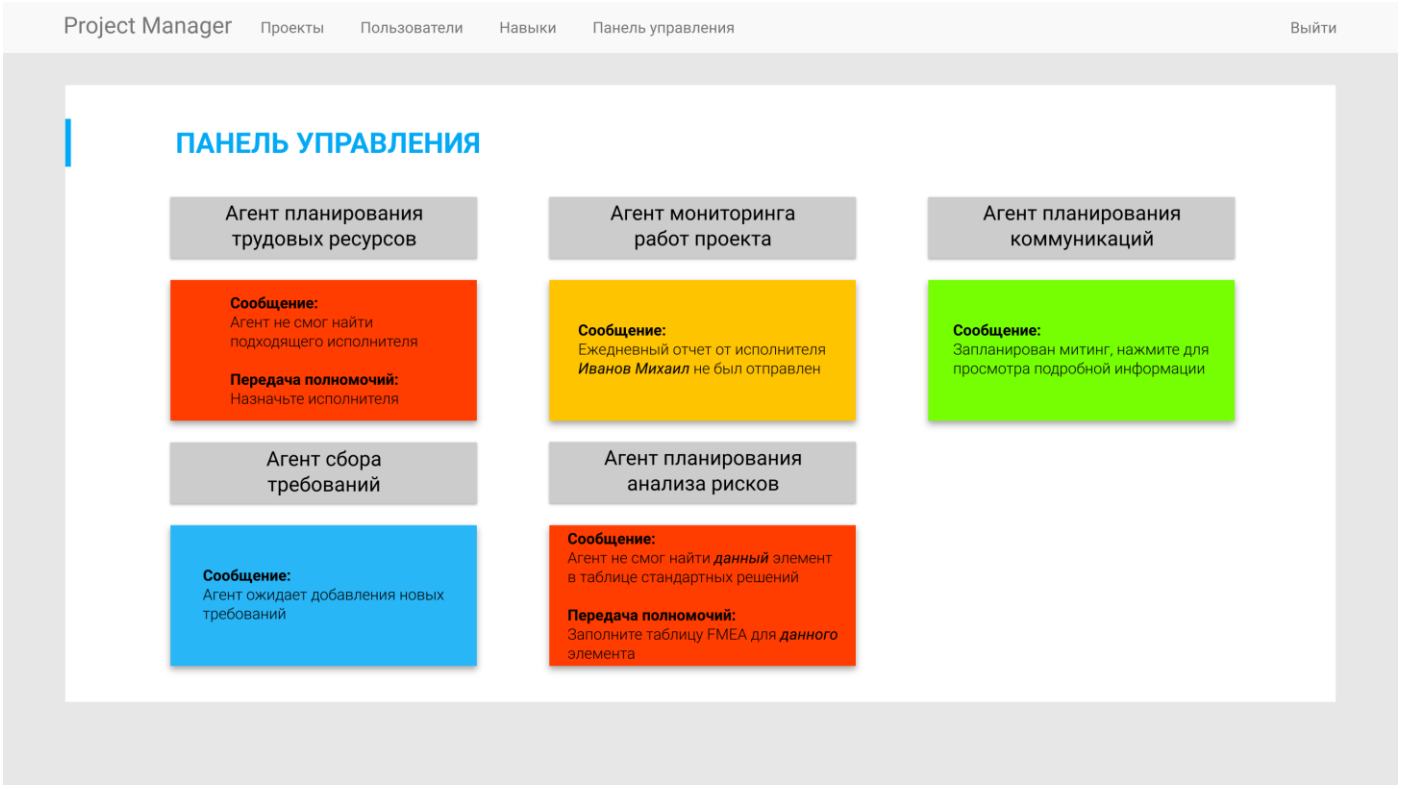


Рис. 23 Панель управления агентами

взять полномочия в свои руки. Цвет блока – зеленый, если агент выполняет свои задачи без возникновения критических ситуаций. В таком случае в блоке размещается информация о текущей деятельности агента. Блок, окрашенный в синий цвет говорит о том, что агент выполнил все поставленные перед ним задачи и ожидает новых.

Используя приложение, менеджер не всегда будет находиться в разделе Панель управления, поэтому не сможет постоянно следить за изменением состояния агентов. В этом случае агент уведомит менеджера проекта посредством всплывающего оповещения, содержащего сообщение. Для того, чтобы быстро понять тип сообщения, уведомления также были окрашены в разные цвета. Помимо оповещения менеджера проект, агенты также могут отправлять уведомления членам команды и всем заинтересованным лицам, если этого требует модель поведения агента. Примеры уведомлений по типу сообщений показаны на рисунке 24.

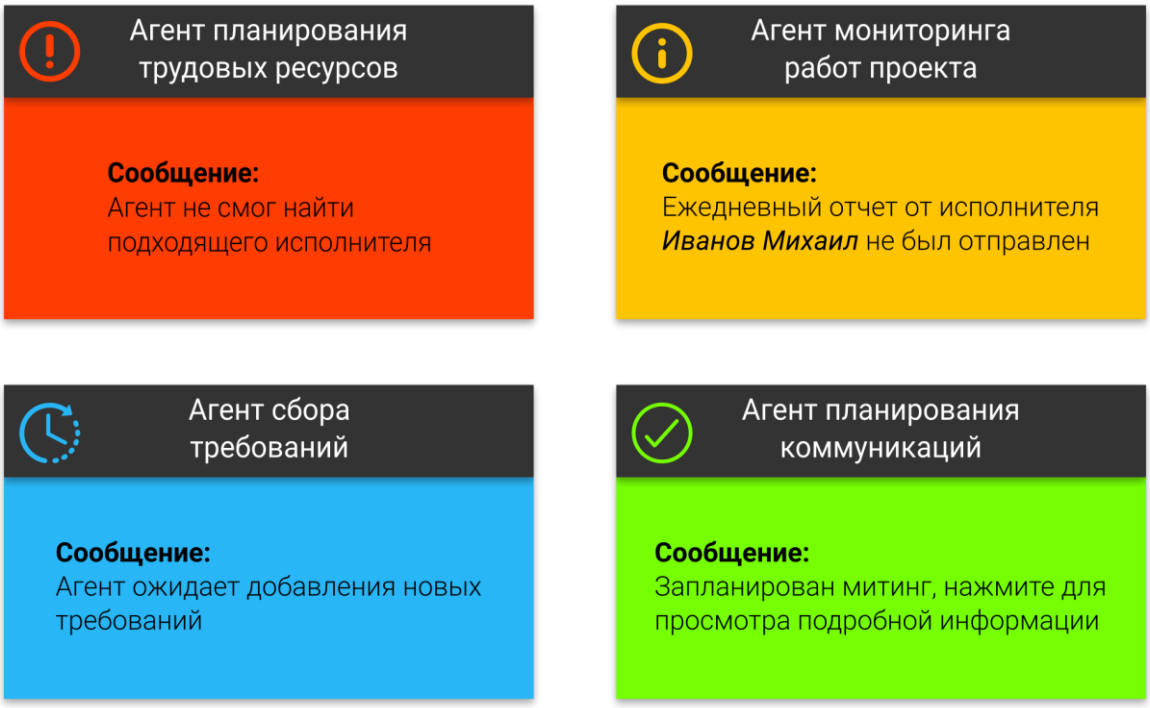


Рис. 24 Примеры уведомлений агентов

В Приложении 6 приведен пример кода реализации интерфейса прототипа Web-приложения.

Заключение

В настоящее время распределенная компания – относительно новая форма организации. Как было отмечено ранее, таким компаниям необходимое единое информационное рабочее пространство, которое позволило бы менеджерам эффективно управлять проектами. Стоит отметить, что менеджеры проектов распределенных организаций проделывают колоссальную работу по мониторингу и анализу.

Мы с Веселовой Дианой в совместном проекте по созданию концептуально-функциональной модели мониторинговой системы, позволяющей разгрузить менеджеров проектов и повысить эффективность контроля реализации программного проекта в распределенной компании, выбрали мультиагентный подход. Благодаря мультиагентным технологиям, нам удалось создать систему из нескольких агентов, которые автоматизируют некоторые ключевые управленческие процессы. Использование агентных систем наилучшим образом отвечает вопросам управления и организации эффективного мониторинга проекта в компаниях такого типа.

В нашем случае показано, что особую роль играют интерфейсы взаимодействия между активными объектами и субъектами системы. Разработка компонентов системы интерфейсов была основной целью данной работы. Мы рассматривали взаимодействие менеджеров, команды разработчиков и заинтересованных сторон с интеллектуальными агентами. Речь шла не только о привычных пользовательских интерфейсах, но и интерфейсах, в которых взаимодействие происходит по определенным правилам на базе согласованных алгоритмов, протоколов и стандартов (например, программный и программно-аппаратный интерфейс).

Для достижения поставленной цели были проанализированы общие принципы управления программными проектами и выделены ключевые управленческие моменты, характерные для распределенных компаний. Было проведено сравнение организации деятельности команд разработчиков в традиционных и распределенных компаниях. На основе полученных знаний

мы выбрали методологию разработки программного обеспечения, наиболее подходящую таким видам организаций.

Были изучены теоретические аспекты мультиагентных технологий в применении к распределенным компаниям, помимо этого были выделены ключевые процессы управления, наиболее подверженные рискам в распределенных организациях. Нами были разработаны алгоритмы действия интеллектуальных агентов, автоматизирующие некоторые такие процессы. Проанализировав модели, протоколы и стандарты взаимодействия в мультиагентных системах, мы создали модель взаимодействия агентов, разработали компоненты системы интерфейсов, отвечающие за взаимодействие типа «агент-агент» и «агент-человек». Завершающим шагом было внедрение созданных компонентов системы интерфейсов в прототип мониторинговой системы, разработанной совместно с Веселовой Дианой.

Подводя итог всей работы, хочется отметить, что разработанный нами прототип мониторинговой системы можно использовать для создания реального Web-приложения по эффективному управлению проектами в распределенных компаниях.

Список литературы

1. Уорнер М., Витцель М. Виртуальные организации. Новые формы ведения бизнеса в XXI веке/ Пер. с англ. Ю. Леонов – М.: Добрая книга, 2005 – 296с.
2. Коблова Ю. А. Виртуальные организации как новейшая форма сетевых структур// Вестник Саратовского государственного социально-экономического университета – 2013. – №3 – С.18-21.
3. Трофимов В.В., Горбунов И.Г. Методологические основы управления проектами виртуальных предприятий – СПб.: Изд-во СЗТУ, 2007. – 174 с.
4. Bourque P., Fairley R.E., eds. Guide to the Software Engineering Body of Knowledge, Version 3.0/ IEEE Computer Society, 2014 – 346 p.
5. Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Newtown Square, Pa: Project Management Institute, 2013 – 589 p.
6. Хусаинова А.Т. Понятие, сильные и слабые стороны виртуальных команд и их отличие от традиционных команд проекта //Вестник КазНУ. Серия экономическая – 2015. – Vol. 2, №108 – С.165-168
7. Zofi Y. A manager's guide to virtual teams – New York, NY: American Management Association, 2011 – 272 p.
8. Trautsch B. R. Managing virtual project teams – San Francisco, California – 2003 – 50 p.
9. Boehm B. W. A Spiral Model of Software Development and Enhancement // IEEE Computer, IEEE – 1988. – Vol. 21, №5 – pp.61-72
10. Фатрелл Р. Т., Шафер Д. Ф., Шафер Л. И. Управление программными проектами: достижение оптимального качества при минимуме затрат/ Пер. с англ. М.: Издательский дом «Вильямс», 2004 – 1136 с.
11. 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.
12. Beck K., Embracing change with Extreme Programming// IEEE Computer, IEEE – 1989. – Vol. 32, №10 – pp.70-77

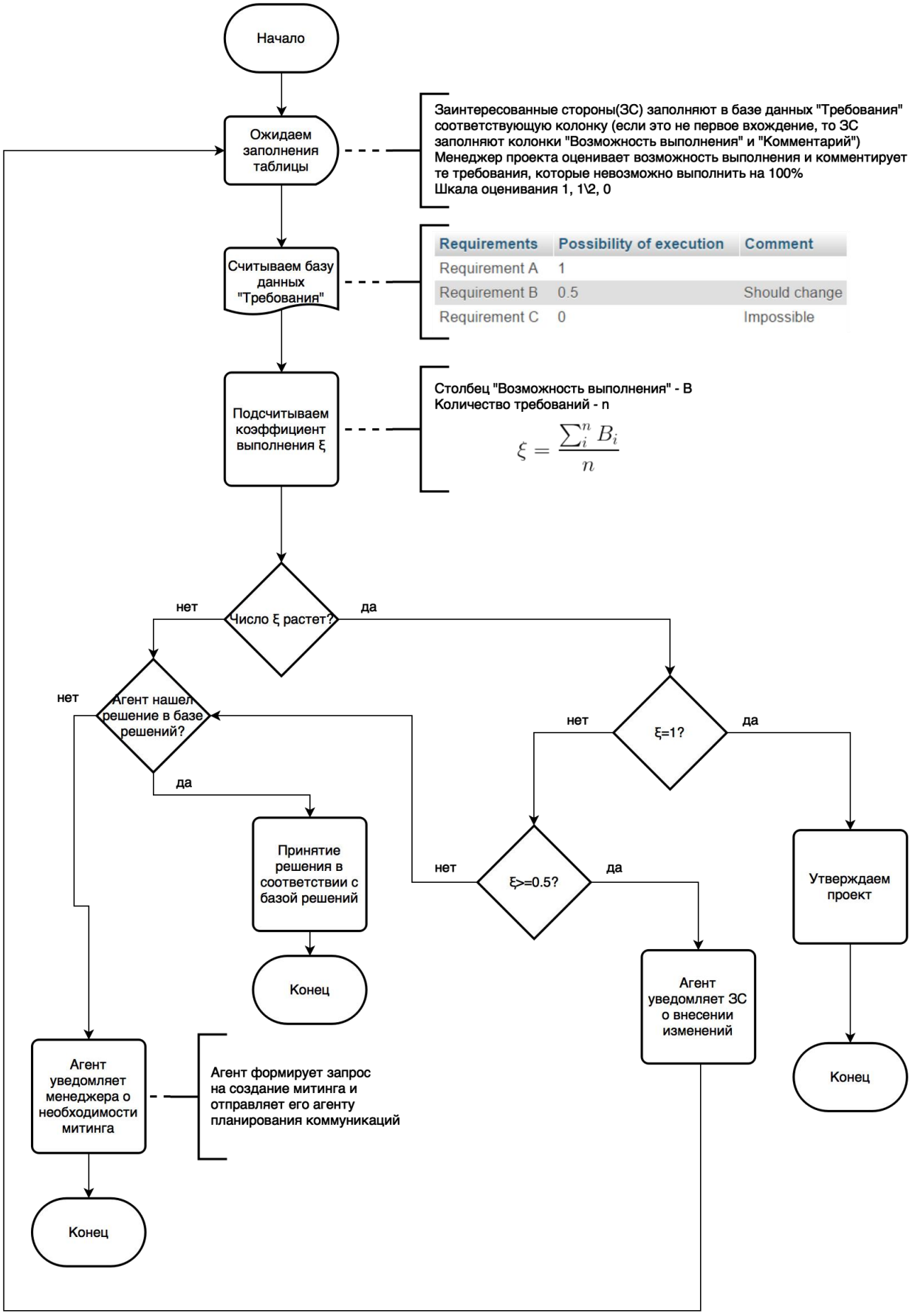
13. Williams L. A. The XP Programmer: The Few-Minutes Programmer// IEEE Software – 2003. – Vol. 20, №3 – pp.16-20
14. Schwaber K., Beedle M. Agile Software Development with Scrum – Upper Saddle River, NJ: Prentice-Hall, 2001 – 158 p.
15. Požgaj Ž., Sertić H. Strategies for Successful Software Development Project Preparation// Proceedings of the 26th International Conference on Information Technology Interfaces ITI 2004 – Cavtat: IEEE – 2004. – pp. 679-684
16. Basharat I., Nafees T. Risks factors identification and assessment in virtual projects of software industry: A survey study// Science and Information Conference (SAI), 2013 – pp. 176-181
17. Reed A.H., Knight L.V. Major virtual project risk factors// Journal of Information Technology Management – 2011. – Vol. 22, №4 – pp.1-12
18. Jennings N.R. et ali. Using intelligent agents to manage business processes, Proceedings of Practical Applications of Intelligent Agents and Multi-Agent Technology – PAAM'96, London, UK, 1996.
19. Граничин О.Н., Кияев В.И. Информационные технологии и системы в современном менеджменте – СПб: Изд-во СПбГУ-ВВМ, 2014. – 897 с.
20. Mihaylov M. Decentralized Coordination in Multi-Agent Systems// Ph.D. thesis, Vrije Universiteit Brussel, Belgium. – 2013. – 208 p.
21. Fairley R.E. Managing and Leading Software Projects/ Wiley-IEEE Computer Society Press, 2009 – 512 p.
22. Shan L., Zhu H. Modelling Cooperative Multi-agent Systems // Lecture Notes in Computer Science. – Berlin, Heidelberg: Springer, 2004. – Vol. 3033 – pp. 994-1001
23. Gelfond G., Watson R. Modeling Cooperative Multi agent systems// Proc. of ASP, 2007
24. Garza A., Serrano J. Fault Tolerant Multi agent systems: its communication and cooperation// Engineering Letters, 2007

25. Nagao A., Miki T. Cooperative behavior generation method using local communication for distributed multi agent system// Systems Man and Cybernetics (SMC), 2010
26. Ma B. Modeling Multi-Agent Systems with Hierarchical Colored Petri Nets// In: Li D., Wang B. (eds) – Artificial Intelligence Applications and Innovations. AIAI 2005. – Boston, MA: Springer, 2005. – Vol. 187. – pp. 167-171

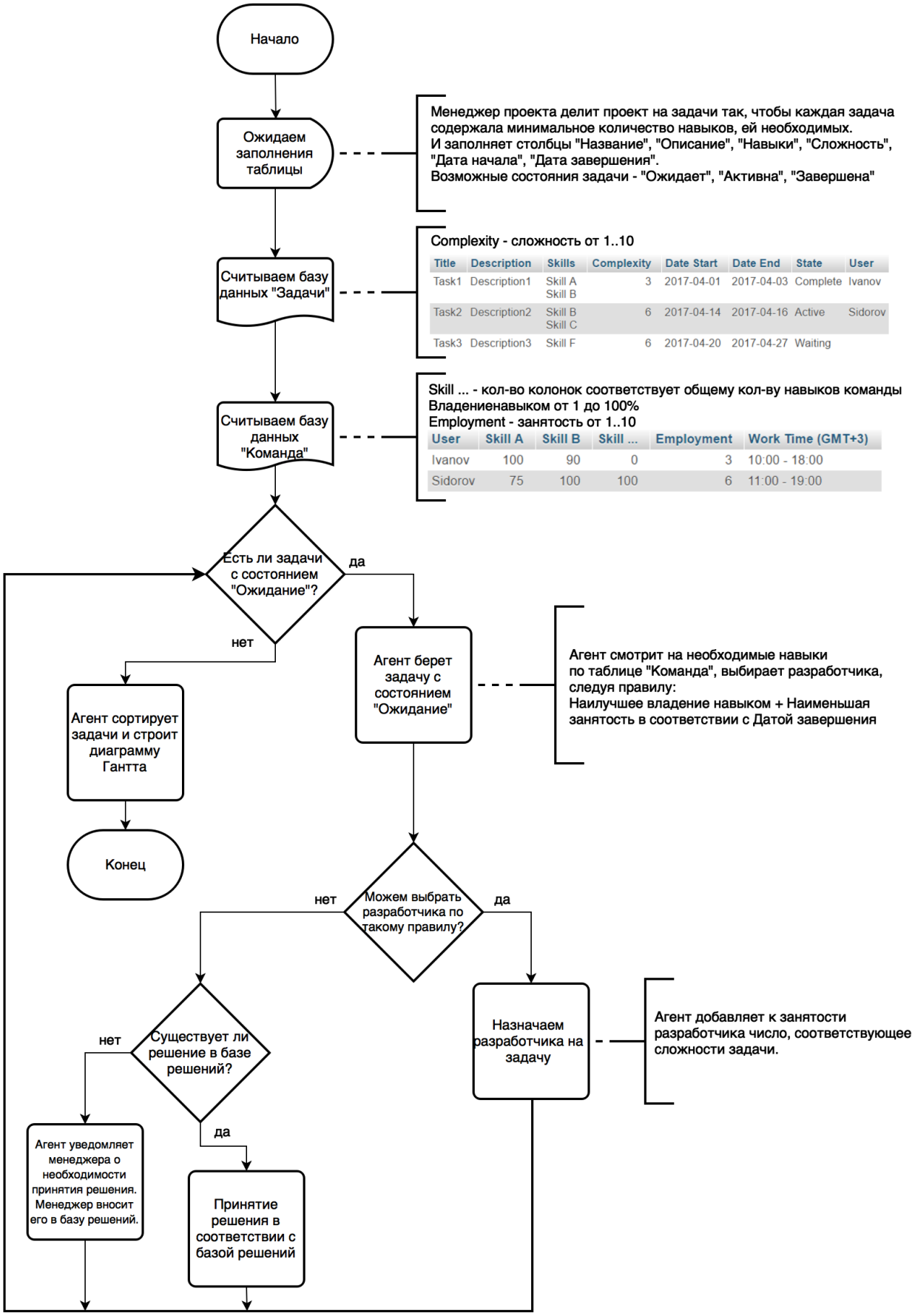
Приложение 1. Таксономия проблем с требованиями

Категории проблем	Примеры
Некорректные требования	Неправильная формулировка требования к функционированию Некорректный перевод требования к функционированию в техническую спецификацию Некорректное значение или переменная в требовании Неверные внешние константы Завышение или занижение вычислительных ресурсов, присвоенных требованию Неправильное описание исходного состояния системы
Неполные требования	Не указано требование Неполная декомпозиция требования Неполное описание требования Ошибка при попытке полностью описать вход и выход системы Ошибка при попытке указать начальное состояние Неполное описание исходного состояния системы
Несовместимые требования	Попарно несовместимые требования Требования к параллельным процессам, если их брать попарно, несовместимы
Неоднозначность требований	Смысл требования неясен Требования, изложены в манере, которую трудно понять
Недостижимые требования	Требование невозможно достичь, учитывая: состояние технологий; современную науку об алгоритмах; текущие навыки наших разработчиков; программного обеспечения; другие системные факторы, такие как скорость; процессора или доступная память;
Труднодостижимые требования	Требование трудно реализуется с помощью доступных ресурсов, расписания и технологий Требование трудно достичь, учитывая текущие навыки наших разработчиков программного обеспечения
Переопределенные требования	Требования превышают требования к функционированию, вызывая дополнительные затраты
Требования с излишними ограничениями	Ограничения на производительность и надежность, чрезмерные для эксплуатационных потребностей и вызывают дополнительные затраты
Непрослеживаемое требование	Требование не может быть/не было прослежено до предыдущего или последующих этапов
Неподтвержденные требования	Полнота, правильность и непротиворечивость требования не могут быть проверены каким-либо разумным методом верификации
Неуместные требования	Требование относится к другому разделу документа
Избыточные требования	Требование уже было специфицировано
Несоответствующая информация	Расписание, бюджет, учебные планы и другие позиции, которые должны находиться в других документах

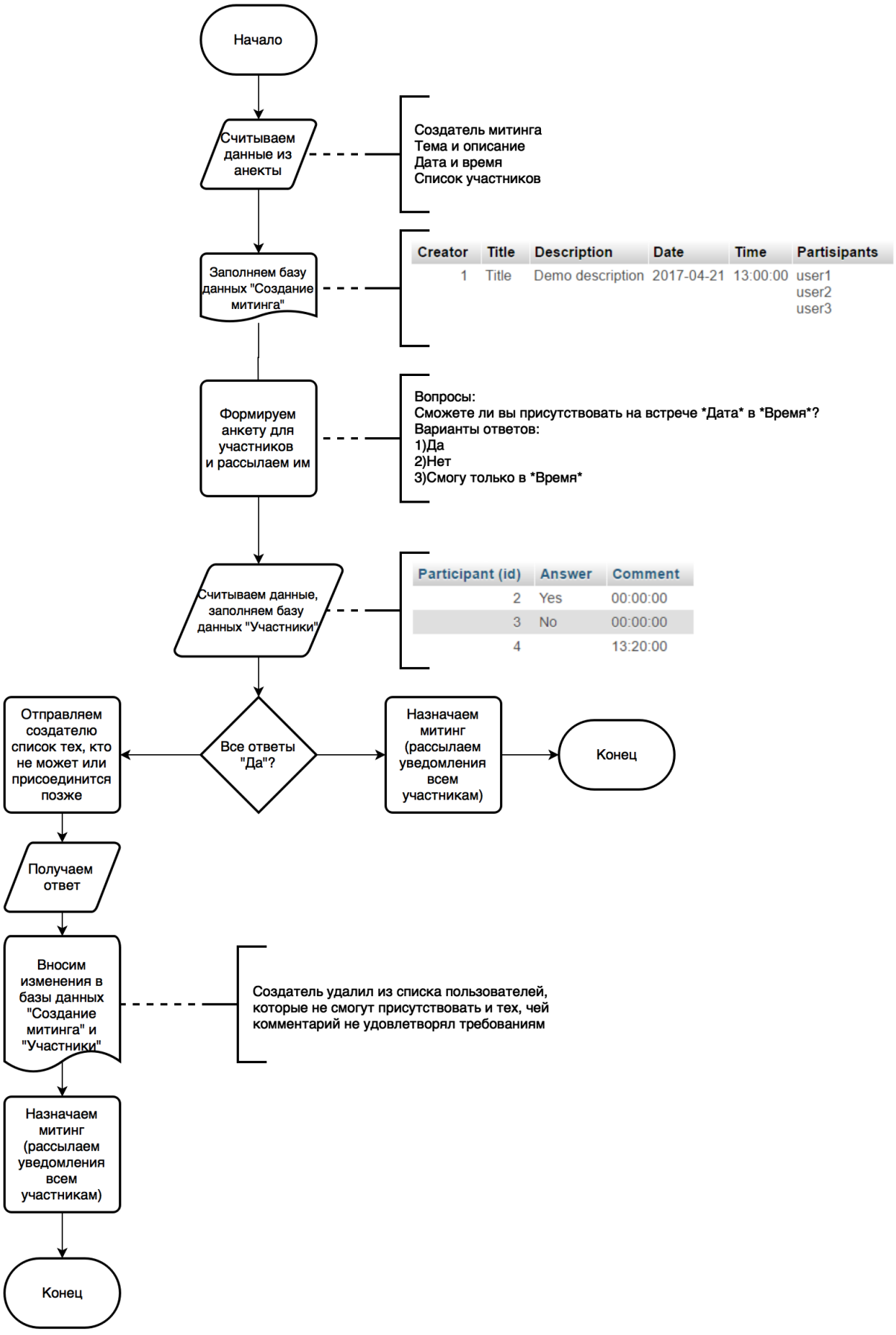
Приложение 2. Блок-схема действий «Агента сбора требований»



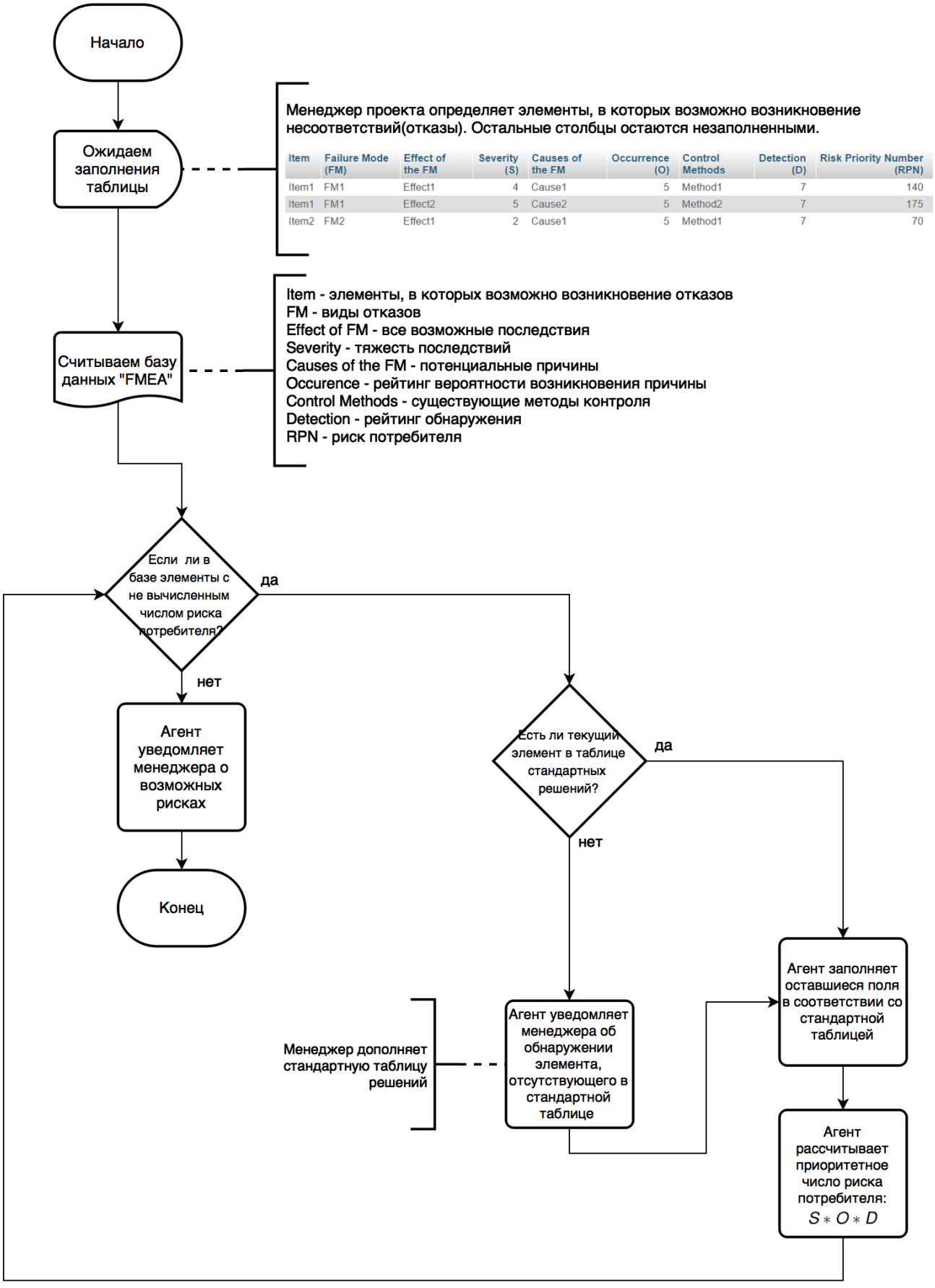
Приложение 3. Блок-схема действий «Агента планирования трудовых ресурсов



Приложение 4. Блок-схема действий «Агента планирования коммуникаций»



Приложение 5. Блок-схема действий «Агента планирования анализа рисков



Приложение 6. Пример кода реализации интерфейса прототипа Web-приложения

```
<div class="row">
   <div class="col-md-offset-1 col-md-4">
      <h3> Добавление задачи</h3>
      <h4>Заметки:</h4>
      <p>Для создания новой задачи, заполните форму.</p>
      <p>Поля, отмеченные звездочкой, обязательны для заполнения.</p>
      <p>Сложность задачи должна быть от 1 до 10.</p>
   </div>
   <div class="col-md-6">
      <div class="containe">
         <div class="form">
            <?php $form=$this->beginWidget('CActiveForm', array(
               'id'=>'task-form',
               'enableAjaxValidation'=>false,
            )); ?>
            <div class="input">
               <?php echo $form->labelEx($model,'header',array('class' => 'label')); ?>
               <?php echo $form->textField($model,'header'); ?>
               <span class="spin"></span>
            </div>
            <div class="err">
               <?php echo $form->error($model,'header'); ?>
            </div>
            <div class="input">
               <?php echo $form->textArea($model,'descr'); ?>
               <?php echo $form->labelEx($model,'descr',array('class' => 'label')); ?>
               <span class="spin"></span>
            </div>
            <div class="err">
               <?php echo $form->error($model,'descr'); ?>
            </div>
            <div class="form-group">
               <?php echo $form->labelEx($model,'deadline_date'); ?>
               <div class="input-group date datetimepicker">
                  <?php echo $form->textField($model,'deadline_date'); ?>
                  <span class="input-group-addon">
                     <span class="glyphicon glyphicon-calendar"></span>
                  </span>
                  <span class="spin"></span>
               </div>
            </div>
            <div class="err">
               <?php echo $form->error($model,'deadline_date'); ?>
            </div>
            <?php if ($model->isNewRecord) { ?>
               <div class="input">
                  <?php echo $form->labelEx($model,'complexity',array('class' => 'label')); ?>
                  <?php echo $form->textField($model,'complexity'); ?>
                  <span class="spin"></span>
               </div>
               <div class="err">
                  <?php echo $form->error($model,'complexity'); ?>
               </div>
               <div class="input">
                  <div class="sel">
                     <?php $skills = Skill::model()->findAll(); ?>
                     <?php echo $form->labelEx($model,'skill_id',array('class' => 'label')); ?>
                     <?php echo $form->dropDownList($model,'skill_id', CHtml::listData($skills, 'id', 'name')); ?>
                  </div>
               </div>
               <div class="err">
                  <?php echo $form->error($model,'skill_id'); ?>
               </div>
               <?php if ($model->hasErrors('implementer_id')) { ?>
               <div class="err">
                  <?php echo $form->error($model,'implementer_id'); ?>
               </div>
               <p>Выберите исполнителя:</p>
               <div class="input">
                  <div class="sel">
                     <?php $implementers = User::model()->findAllByAttributes(array('role' => 'implementer')); ?>
                     <?php echo $form->labelEx($model,'implementer_id',array('class' => 'label')); ?>
                     <?php echo $form->dropDownList($model,'implementer_id', CHtml::listData($implementers, 'id', 'name')); ?>

                  </div>
               </div>
            <?php } } ?>
            <div class="button">
               <?php echo CHtml::submitButton($model->isNewRecord ? 'Создать' : 'Сохранить'); ?>
            </div>
            <?php $this->endWidget(); ?>
         </div><!-- form -->
      </div>
   </div>
</div>
```