Федеральное государственное бюджетное учреждение науки
Институт систем энергетики им. Л.А. Мелентьева
Сибирского отделения Российской академии наук

На правах рукописи

Гальперов Василий Ильич

МЕТОДЫ, МОДЕЛИ И АЛГОРИТМЫ ПОСТРОЕНИЯ МНОГОАГЕНТНЫХ СИСТЕМ В ЭНЕРГЕТИКЕ (НА ПРИМЕРЕ ЗАДАЧИ ОЦЕНИВАНИЯ СОСТОЯНИЯ ЭЛЕКТРОЭНЕРГЕТИЧЕСКИХ СИСТЕМ)

Специальность 05.13.18 – Математическое моделирование,
численные методы и комплексы программ

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:

Доктор технических наук, профессор
Массель Людмила Васильевна

ИРКУТСК – 2017

Оглавление

Список сокращений, принятых в диссертации

ЕЭС – единая энергосистема

ИСЭМ СО РАН – Институт Систем Энергетики им. Л.А.Мелентьева Сибирского Отделения Российской Академии Наук

КУ – контрольное уравнение

МАС – мультиагентная система или многоагентная система

ОС – оценивание состояния

ПВК – программно-вычислительный комплекс

МВНК – метод взвешенных наименьших квадратов

ТИ – телеизмерения

ТС – телесигналы

УР – установившийся режим

ЦДУ – Центральное Диспетчерское Управление

ЭЭС – электроэнергетическая система

ACL – Agent Communication Language – язык взаимодействия агентов

FIPA – Foundation for Intelligent Physical Agents – стандарты для реализации интеллектуальных агентов

GPS – Global Positioning System – система глобального позиционирования

JADE – Java Agent Development Environment – среда реализации агентов на языке Java

KQML – Knowledge Query and Manipulation Language – язык межагентного взаимодействия

PMU – Phasor Measurement Unit – устройство измерения комплексных электрических величин

SCADA – Supervisory Control And Data Acquisition – система сбора и управления данными

TCP/IP – Transmission Control Protocol/Internet Protocol – протокол взаимодействия в локальных и глобальных сетях

WAMS – Wide-Area Measurement System–Широкомасштабная система сбора измерений

WACS – Wide-Area Control System – широкомасштабная система контроля

Введение

Актуальность работы. Агентные технологии являются одной из распространенных информационных технологий, востребованной во многих предметных областях. В частности, агентные технологии декларируются как одна из базовых технологий при реализации концепции интеллектуальных энергетических систем [1]. В то же время всегда существует разрыв между теоретическими разработками и их практическим применением в конкретных областях, что характерно и для энергетики. Диапазон предложений в данном сегменте со стороны ИТ-разработчиков весьма ограничен: решения зарубежных компаний довольно дороги и уязвимы с точки зрения кибербезопасности, качественных отечественных разработок недостаточно или они просто отсутствуют. Имеющиеся прикладные разработки в области энергетики, как правило, узкоспециализированы, нетиражируемы и требуют сопровождения (неотчуждаемы от разработчиков). Немногочисленные универсальные системы, в свою очередь, обладают излишней функциональностью и, как следствие, тяжеловесностью.

При реализации концепции интеллектуальных энергетических систем важная роль отводится подсистеме сбора и управления информацией о текущем режиме ЭЭС. В этой подсистеме одной из основных является задача оценивания состояний (ОС), которая состоит в расчете установившегося режима ЭЭС по данным телеизмерений и телесигналов. В современных условиях функционирования и управления ЭЭС возникает необходимость в выполнении ОС для энергосистем большой размерности -порядка нескольких тысяч узлов). При ОС схем ЭЭС такой размерности возникают проблемы, связанные с неоднородностью и большим объемом обрабатываемой информации, возрастанием нагрузки на доступные вычислительные ресурсы в центре управления ЭЭС. Распределенная обработка данных при декомпозиции задачи оценивания состояния является

эффективным методом решения этих проблем, повышения качества результатов и надежности вычислительной процедуры ОС ЭЭС.

Одними из основных направлений работ по развитию автоматизированной системы управления режимами являются разработка алгоритмов выявления предаварийных состояний на основе методов оценивания состояния и создание систем распределенного расчета режимов энергосистем. В ИСЭМ СО РАН был разработан, но не реализован программно алгоритм распределенной обработки телеинформации при оценивании состояния с использованием многоагентных технологий, который позволяет повысить эффективность расчетов ЭЭС большой размерности (И.Н. Колосок, А.С. Пальцев). Предложенный алгоритм основан на использовании методов декомпозиции схем ЭЭС. Представляется целесообразным продемонстрировать возможности многоагентного подхода при создании интеллектуальных энергетических систем на примере решения задач ОС ЭЭС.

Таким образом, актуальность диссертационной работы обусловлена, с одной стороны, необходимостью разработки теоретического подхода к построению типовых многоагентных систем (МАС) в энергетике, включая методы управления взаимодействием агентов, с другой стороны, потребностью в практической реализации таких систем, основанной на использовании гибкой архитектуры и базовых программных компонентов. Весьма актуальной является также задача разработки типовой системы ОС ЭЭС, позволяющей реализовать выбранный специалистом способ декомпозиции и создать конкретную конфигурацию МАС ОС для выбранного способа.

Использование многоагентных технологий в России и за рубежом рассматривается в работах В.И. Городецкого, В.Б. Тарасова, S. Russell, P. Norwig, P. Mars, M. Wooldridge, C. Hewitt, J. Inman, G. Weiss, J. Ferber и др. В ИСЭМ СО РАН вопросам применения методов искусственного интеллекта и

многоагентного подхода в задачах энергетики, а так же распределенному оцениванию состояния посвящены работы А.З. Гамма, А.М. Глазуновой, Ю.А. Гришина, Р.А. Заики, Ю.Б. Каштанова, И.Н. Колосок, В.Г. Курбацкого, Л.В. Массель, В.В. Новорусского, А.С. Пальцева, Д.А. Панасецкого, Н.В. Томина, Д.А. Фартышева, П.В. Этингова и др.

Цель работы: Разработка методических основ применения многоагентных технологий при реализации концепции интеллектуальных энергетических систем на примере задачи оценивания состояния ЭЭС.

Поставлены и решены следующие задачи:

1. Анализ предметной области, связанной с разработкой концепции интеллектуальных энергетических систем и решением задач оценивания состояния ЭЭС, существующих подходов к разработке многоагентных систем и возможностей их применения при создании интеллектуальных энергетических систем на примере задачи оценивания состояния ЭЭС.
2. Разработка методического подхода к построению типовых многоагентных систем на примере задачи оценивания состояния ЭЭС, включающего:
   - метод проектирования и реализации типовой МАС;
   - метод управления взаимодействием агентов на основе алгебраических сетей;
   - событийные модели сценариев взаимодействия агентов (агентных сценариев) и их графического представления с использованием элементов Joiner сетей;
   - алгоритмы взаимодействия и поведения агентов;
   - базовые компоненты типовой МАС с гибкой архитектурой.
3. Реализация, на основе предложенного методического подхода, конкретной МАС для одной из задач оценивания состояния ЭЭС и ее апробация
4. Разработка численного метода оценки надежности МАС на основе графовых моделей и его применение для выявления критичных компонентов конкретных конфигураций МАС.

5. Разработка усовершенствованной многоагентной технологии численного решения задачи оценивания состояния с использованием предложенного методического подхода и разработанного программного обеспечения.

Объектом исследования является электроэнергетическая система.

Предмет исследования: методы создания современных многоагентных систем, методы и средства управления взаимодействием агентов на основе алгебраических сетей и событийных моделей; информационная технология распределенного оценивания состояния ЭЭС, с учетом функциональной и структурной декомпозиции.

Методы исследования: Для решения поставленных задач использовались методы искусственного интеллекта, методы агентно-ориентированного проектирования и программирования и разработки многоагентных систем, методы и средства событийного моделирования, методы теории графов, методы теории вероятности и математической статистики, теория и методы оценивания состояния ЭЭС. Предлагаемые в диссертационной работе алгоритмы оценивания состояния ЭЭС базируются на разработанном в ИСЭМ СО РАН методе контрольных уравнений.

Составляют предмет научной новизны и выносятся на защиту следующие положения:

1. Разработан методический подход к построению типовых многоагентных систем на примере задачи оценивания состояния ЭЭС, включающий:
   - метод проектирования и реализации типовой МАС;
   - вычислительный метод управления взаимодействием агентов на основе алгебраических сетей;
   - событийные модели сценариев взаимодействия агентов (агентных сценариев) и их графического представления с использованием элементов Joiner сетей;
   - алгоритмы взаимодействия и поведения агентов;
   - базовые компоненты типовой МАС с гибкой архитектурой.

Пункт 1 новизны соответствует п. 3 паспорта специальности 05.13.18[1].

2. Построены событийные модели агентных сценариев с использованием аппарата Joiner-сетей, позволяющие пользователям самостоятельно вносить изменения в алгоритм работы системы без участия программистов. Пункт 2 новизны соответствует п. 8 паспорта специальности 05.13.18[2].
3. Предложен численный метод оценки надежности многоагентной системы с использованием теории графов. Пункт 3 новизны соответствует п. 4 паспорта специальности 05.13.18[3].
4. Разработана методика разработки конкретных МАС оценивания состояния ЭЭС на основе типовой многоагентной системы. Выполнена реализация многоагентной системы «EstateMAS», отличающаяся от известных применением авторского метода управления взаимодействием агентов, разработкой агентных сценариев и их моделированием с использованием Joiner-сетей. Пункт 4 новизны соответствует п. 4 паспорта специальности 05.13.18.
5. Предложена усовершенствованная многоагентная технология численного решения задачи оценивания состояния, отличие которой состоит в использовании разработанных методов и реализованного программного обеспечения для поддержки этапов технологии. Соответствует п. 3 паспорта специальности 05.13.18.

Практическая ценность работы:

Предлагаемый методический подход к построению типовых многоагентных систем ориентирован на его применение при реализации

[1] П. 3 «Разработка, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий»

[2] П. 8 «Разработка систем компьютерного и имитационного моделирования»

[3] П. 4 «Реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента»

концепции интеллектуальных энергетических систем и позволяет его тиражирование для различных энергетических задач. Его применение для задач оценивания состояния ЭЭС позволяет разрабатывать конкретные МАС, использующие различные методы декомпозиции схем ЭЭС, без привлечения программистов. Усовершенствованная технология оценивания состояния ЭЭС предусматривает децентрализацию расчетов установившегося режима ЭЭС, что позволит снизить нагрузку на главный диспетчерский центр и уменьшить количество данных, передаваемых по сети.

Результаты диссертационной работы применены в базовом проекте ИСЭМ СО РАН № 01201361373 «IV.35.1.3. Методы, технологии и инструментальные средства интеллектуализации поддержки принятия решений в интегрированных интеллектуальных энергетических системах», в проектах, поддержанных грантами РФФИ №13-07-140 (2013–2015), №16-07-00474 (2016–2018), №15-07-01284 (2015–2017), грантом Программы Президиума РАН №229 (2012–2014), №15-07-04074 Бел_мол_а (2015–2016), а так же грантом СО РАН и Института энергетики НАН Беларуси №18Б (2012-2014). В рамках двух последних проектов результаты диссертационной работы переданы и использованы в Институте энергетики НАН Беларуси.

Апробация работы: Результаты работы докладывались на Международной конференции «Computer science and information technologies (CSIT’2015), Рим, Италия, 2015; Байкальских Всероссийских с международным участием конференциях «Информационные и математические технологии в науке и управлении», г. Иркутск, 2013-2016 гг.; на международных семинарах «Contingency management, intelligent, agent-based computing and cyber security in energy sector», **Монголия – Россия, 2014-2017 гг.; на** XVI Всероссийской конференции молодых ученых «Математическое моделирование и информационные технологии»**,** г. Красноярск, 2015 г.; на конференциях молодых ученых ИСЭМ СО РАН,

2014-2016 гг., а также на заседаниях секции Ученого совета ИСЭМ СО РАН «Прикладная математика и информатика».

Личный вклад: Положения, составляющие новизну и выносимые на защиту, получены лично автором. Постановка задачи выполнена совместно с Л.В. Массель, И.Н. Колосок и А.С. Пальцевым.

Публикации: По теме диссертационной работы опубликованы 10 статей, в том числе 4 из них – в реферируемых журналах, рекомендованных ВАК. Получены 7 свидетельств о регистрации программ для ЭВМ.

В первой главе выполнен анализ предметных области, связанной с концепцией интеллектуальных энергетических систем. Важную часть в этой концепции занимает задача оценивания состояния. В главе рассматриваются основные положения и математическая постановка задачи ОС. Одним из способов расчета установившегося режима ЭЭС является распределенная обработка данных. Данные декомпозируются (разбиваются на небольшие группы), рассчитываются параллельно и затем вновь объединяются. Использование декомпозиции в задаче оценивания состояний обосновано в работах А.С. Пальцева и И.Н. Колосок. Для реализации данного подхода целесообразно использовать многоагентные системы, поскольку они являются одной из базовых технологий при реализации концепции интеллектуальных энергетических систем, а в данном случае позволяют организовать распределенные вычисления при реализации методов декомпозиции. Далее в главе выполнен аналитический обзор современного состояния в области агентных технологий, рассмотрены существующие подходы и платформы разработки агентных систем, выявлены их недостатки. Рассмотрена одна из разновидностей алгебраических сетей (Joiner-сети), как основа моделирования взаимодействия агентов. Делается вывод о необходимости разработки методического подхода к построению многоагентных систем в энергетике и реализации многоагентных систем на примере решения задачи оценивания состояния ЭЭС.

Во второй главе кратко характеризуются разработки в области энергетики, в которых был применен агентный подход. Далее рассматривается предлагаемый автором методический подход методический подход к построению типовых многоагентных систем на примере задачи оценивания состояния ЭЭС, включающий: метод проектирования и реализации типовой МАС; вычислительный метод управления взаимодействием агентов на основе алгебраических сетей; событийные модели сценариев взаимодействия агентов (агентных сценариев) и их графического представления с использованием элементов Joiner сетей;
алгоритмы взаимодействия и поведения агентов; базовые компоненты типовой МАС с гибкой архитектурой. Рассмотрен предложенный автором численный метод оценки надежности многоагентной системы с использованием теории графов, позволяющий выявить наиболее критичные элементы отдельных конфигураций МАС. Описаны разработанные автором алгоритм взаимодействия агентов через TCP/IP и состав XML-сообщений, используемых в системе и алгоритм поведения агентов при выполнении многоэтапных расчетов. Определены системно-концептуальные соглашения, принятые при разработке многоагентной системы, выделены и описаны базовые компоненты агентов, определены основные требования к многоагентной системе.

В третьей главе рассматривается реализация многоагентной системы «EstateMAS», выполненная на основе предложенного методического подхода, методов, моделей и алгоритмов. Подчеркивается гибкая архитектура МАС, обеспечиваемая возможностью задания агентных сценариев пользователями и использованием базовых компонентов. Описана усовершенствованная многоагентная технология численного решения задачи оценивания состояния, базирующаяся на использовании разработанных методов и реализованного программного обеспечения для поддержки этапов технологии.

Автор выражает глубокую благодарность своему научному руководителю, д.т.н. Массель Л.В., а также д.т.н. Колосок И.Н. и к.т.н. Пальцеву А.С., за оказанную помощь в постановке задачи и консультации в ходе выполнения работы. Автор благодарен своим коллегам, сотрудникам лаборатории Информационных технологий в энергетике ИСЭМ за поддержку, конструктивную критику и помощь в формулировке основных положений работы.

1 Анализ предметной области оценивания состояний ЭЭС

1.1 Концепция интеллектуальных энергетических систем

В последнее десятилетие за рубежом активно обсуждается и развивается концепция «интеллектуальных энергетических систем» (SmartGrid). В США и Европейском Союзе эта концепция рассматривается как технологическая концепция электроэнергетики будущего [2-5]. В России обсуждение проблемы активизировалось несколько позже [6, 7].

Необходимо отметить, что существует различное понимание существа концепции интеллектуальной электроэнергетической системы. Часто это понятие связывают с интеграцией в ЭЭС возобновляемых источников энергии на основе современных информационных, телекоммуникационных и Интернет-технологий [8, 9]. Другие трактовки данного направления делают акцент на распределительных электрических сетях, включающих возобновляемые источники энергии с формированием активных и адаптивных свойств распределительных сетей за счет развития распределенной системы адаптивной автоматики, широкого использования компьютерных технологий и современных систем управления [10-13]. Данная концепция рассматривается также применительно к основной (передающей) электрической сети с использованием систем широкомасштабного мониторинга режимов (Wide Area Monitoring System — WAMS) и управления ими (Wide Area Control System — WACS) на основе принципов адаптивного управления, устройств измерения комплексных величин PMU (Phasor Measurement Unit), FACTS (Flecsible Alternative Current Transmission System), интеллектуальных компьютерных методов [5, 14, 15]. Многие авторы придают большое значение обеспечению активного участия потребителей в управлении собственным электропотреблением путем применения «умных» счетчиков электроэнергии и регистраторов нагрузки, использования современных интеллектуальных

средств обработки и визуализации информации для потребителя, формирования цены на электроэнергию в реальном времени и т.п. [5, 9, 11]. В последнее время говорят о интегрированных интеллектуальных энергетических системах [13].

В США, Европейском Союзе, Канаде, Китае концепция интеллектуальных энергетических систем является, по сути, государственной политикой технологического развития электроэнергетики будущего. В настоящее время во всех развитых странах мира уделяется очень большое внимание системам электроэнергетики, использующим самое современное оборудование и технологии, средства измерения и управления, позволяющие на более высоком уровне обеспечить надежность и экономичность функционирования электроэнергетических систем. Речь идет о создании так называемой интеллектуальной ЭЭС, под которой понимается система, в которой все субъекты электроэнергетического рынка (генерация, сеть, потребители) принимают активное участие в процессах передачи и распределения электроэнергии.

Новая электроэнергетика России должна базироваться на ключевых ценностях, основанных на клиенто- и социальной направленности с высоким общественным имиджем, в т.ч. обеспечивать:

- достаточность (по мощности, объему и графику электропотребления) энергетических услуг надлежащего качества;
- допустимость (технологическую и социально-экологическую) совместной работы систем централизованного и децентрализованного энергоснабжения, поддерживая необходимый уровень резервирования и надежности энергоснабжения;
- доступность предоставления услуг (подключения) и передачи электроэнергии в соответствии с экономически обоснованным спросом.

Интеллектуальная ЭЭС представляет собой электроэнергетическую систему нового поколения с целью обеспечения эффективного

использования всех ресурсов (природных, социально-производственных и человеческих) для надежного, качественного и эффективного энергоснабжения потребителей за счет гибкого взаимодействия всех ее субъектов (всех видов генерации, электрических сетей и потребителей) на основе современных технологических средств и единой интеллектуальной иерархической системы управления [16].

В интеллектуальной ЭЭС важная роль отводится активно-адаптивной электрической сети, как технологической инфраструктуре электроэнергетики, наделяющей интеллектуальную ЭЭС принципиально новыми свойствами. Наиболее эффективными и гибкими системами управления являются системы, имеющие распределенную многопроцессорную архитектуру программных и аппаратных средств. Распределенная структура обеспечивает действенность системы, возможность ее расширения, модернизации ее программных и аппаратных средств без потери эксплуатационных свойств системы. Виртуализация вычислительных ресурсов и связанная с нею виртуализация баз данных позволяет осуществить распределенные сетевые вычисления, но одновременно создает определенные проблемы с обеспечением координации и планирования решения отдельных задач, вычислением разреженных матриц и обеспечением информационной безопасности сети передачи данных.

Одним из многообещающих направлений исследований по созданию самовосстанавливающихся энергетических систем является разработка многоагентных интеллектуальных систем управления для разных уровней управления на основе теории группового управления. МАС строится на основе программных интеллектуальных агентов, функционирующих на трех уровнях иерархического управления: «помощник»-«начальник»-«координатор».

Одним из важных элементов данной концепции является использование многоагентных технологий [16, 17]. Агенты как независимые, самостоятельные программы обладают всеми необходимыми характеристиками для проведения распределенных вычислений. Каждый агент будет отвечать за определенную часть алгоритма и контактировать с другими агентами для выполнения задания, формируя, таким образом, многоагентную систему. На данный момент существует очень небольшое количество платформ, позволяющих организовывать работу многоагентных систем. Именно поэтому разработка новых методических подходов к реализации многоагентных систем, является актуальной и востребованной задачей. Апробацию нового подхода предлагается выполнить на примере задачи оценивания состояния ЭЭС.

1.2 Агентные технологии и Joiner-сети

1.2.1 Понятие и определение агента

Практически для всех работ, где даются определения, что такое агент и каковы его основные свойства, общим является отсутствие единого мнения по этому поводу. Фактически, используя понятие “агент”, каждый автор определяет агента по-своему, с конкретным набором свойств. Понятие агента используется в разных областях, например, на производстве агентом может называться робот, в области телекоммуникаций – программа и т.п. Как следствие, в зависимости от среды обитания агенты обладают разными свойствами. Поэтому в процессе разработки и реализации систем в рамках данного направления появилось множество типов агентов, например: автономные агенты, мобильные агенты, интеллектуальные агенты, социальные агенты и т.д. Таким образом, вместо единого определения базового агента имеется множество определений производных типов.

В одном из наиболее авторитетных современных трудов по ИИ, изданном С. Расселом и П. Норвигом [18], под агентом понимается «любая

сущность, которая находится в некоторой среде, воспринимает ее посредством сенсоров, получая данные, которые отражают события, происходящие в среде, интерпретирует эти данные и действует на среду посредством эффекторов». Таким образом, здесь вычленяются четыре исходных фактора, образующих агента – среда, восприятие, интерпретация, действие.

Согласно П. Маэс [19], «автономные агенты – это компьютерные системы, функционирующие в сложной, динамической среде, способные ощущать и автономно действовать на эту среду и, таким образом, выполнять множество задач, для которых они предназначены». Здесь предложены два ограничения на среду агентов – «сложная и динамическая».

Нередко агенты понимаются как вычислительные единицы, поддерживающие локальные состояния и параллельные вычисления, а также способные в процессах коммуникации достигать состояния других агентов и автоматически выполнять действия в некоторых условиях среды [20].

Агент может быть рассмотрен как программный объект, отображающий восприятие среды и действия среды для достижения определенной цели. Восприятие среды агентом может включать восприятие физической среды, состоящее из компонентов системы (обычно представленного как база данных) и восприятие многоагентной среды, состоящей из других агентов. Полная последовательность восприятия физической среды в момент времени t обозначим как ENV_t, многоагентной среды – MAS_t. Полная последовательность восприятий до настоящего времени состоит из физических и многоагентных. Обозначив h как настоящий момент времени, полную последовательность восприятий P^* можно представить как:

$$P^* = ENV_h \cup MAS_h$$

Агент обычно обладает некоторыми знаниями об окружающем мире, закодированными как правила, эти закодированные правила обозначаются

как *K*. Эти знания включают общие знания, т.е. те, которые доступны всем агентам, и знания, уникальные для каждого агента в отдельности. В нашем случае общие знания могут включать алгоритмы задач, решаемых при управлении системами энергетики, и другие общедоступные правила и процедуры. Пока агент не активирован, он может обновлять свои внутренние состояния, основываясь на новых восприятиях и своих знаниях о мире; установочная функция, которая генерирует новые состояния *SG*, обозначается следующим образом:

$$SG:\left(P^{*},K\right)\rightarrow S$$

Поэтому агент неспособен различать некоторые состояния окружающей среды, другими словами, агент имеет одинаковые приоритеты для некоторых состояний среды. Это существенно в сложных средах, например, таких, как электроэнергетические системы, которые могут иметь множество состояний, но позволяют отдельному агенту иметь дело с ограниченным числом состояний.

Обычно агент обладает набором следующих свойств:

- адаптивность: агент обладает способностью обучаться;
- автономность: агент работает как самостоятельная программа, ставя себе цели и выполняя действия для достижения этих целей;
- коммуникативность: агент может общаться с другими агентами;
- коллаборативность: агент может взаимодействовать с другими агентами несколькими способами, например, играя роль поставщика/потребителя информации или одновременно обе эти роли, а также роль посредника;
- способность к рассуждениям: агент может обладать частичными знаниями или механизмами вывода, например, знаниями, как приводить данные из различных источников к одному виду; агент может специализироваться на конкретной предметной области;

•мобильность: способность к передаче кода агента от одного источника данных к другому.

С точки зрения разработчиков информационных систем: агент – это модуль (компонент) программного обеспечения, выполняющийся на определенной платформе, совершающий некоторые действия, такие, как отображение и ввод информации, и обменивающийся сообщениями с другими агентами или человеком [21]. Автор в дальнейшем изложении будет придерживаться именно этого определения.

1.2.2 Многоагентные системы

Многоагентные системы возникли на пересечении теории систем и искусственного интеллекта. С одной стороны, речь идет об открытых, активных, развивающихся системах, в которых главное внимание уделяется процессам взаимодействия агентов как причинам возникновения системы с новыми качествами. С другой стороны, достаточно часто МАС строятся как объединение отдельных интеллектуальных систем, основанных на знаниях.

Любая МАС состоит из следующих основных компонентов [22]:

1. Множество организационных единиц, в котором выделяются подмножество агентов, манипулирующих подмножеством объектов; множество задач.
2. Среда, т.е. некоторое пространство, в котором существуют агенты и объекты.
3. Множество отношений между агентами.
4. Множество действий агентов (например, операций над объектами).

Какие характеристики отличают многоагентные системы от одноагентных? Можно выделить следующие различия [23]:

Различия в структуре агента. Часто агенты сконструированы по-разному, например, программы, написанные разными людьми. Различия могут заключаться в оборудовании и программном обеспечении. Часто такие агенты называют гетерогенными в противоположность гомогенным,

которые сконструированы одним и тем же способом и имеют изначально одинаковые возможности. Однако различие не так очевидно; агенты, основанные на одинаковом оборудовании и программах, но ведущие себя по-разному, тоже могут называться гетерогенными.

Окружение. Окружение агентов может быть статическим (не зависящим от времени) или динамическим (нестационарным). Из-за простоты обработки и более строгого математического описания большинство методов искусственного интеллекта в одноагентном случае были разработаны для статического окружения. В многоагентных же системах хотя бы само присутствие нескольких агентов делает окружение динамичным.

Восприятие. Информация, которой обладает система агентов, является обычно распределенной: агенты могут следить за окружением из разных положений, получать информацию в различные моменты времени или интерпретировать эту информацию по-разному. Таким образом, состояние окружения является частично обозримым для каждого агента.

Многоагентная система также может быть определена как сеть асинхронных объектов, которые работают вместе для того, чтобы решать проблемы, которые не под силу решить одному из таких агентов. МАС состоит из ряда нецентрализованных сотрудничающих элементов, называемых агентами, каждый из которых действует автономно.

Специфический интерес представляют МАС, в которых отдельные агенты показывают существенные интеллектуальные возможности и существенную автономию. В последние несколько лет технологии МАС нашли применение во многих распределенных системах, таких, как распределенная обработка информации, а также распределенные научные вычисления.

Часть исследователей считают, что агенты обеспечивают более прогрессивный метод работы в сетевых приложениях [24]. Другие авторы

отмечают, что агенты привносят опасность с точки зрения обеспечения секретности информации и загруженности сети [25].

Агенты являются перспективным направлением, но в настоящее время нет единых стандартов их разработки и все еще остается нерешенным ряд проблем, в частности легальные способы перемещения агентов по сети, верификация агентов (в частности, защита от передаваемых по сети вирусов), соблюдение агентами прав частной собственности и сохранение конфиденциальности информации, которой они обладают, перенаселение сети агентами, а также совместимость кода агента и программно-аппаратных средств сетевой машины, где он исполняется. Проектирование каждого отдельного агента может выполняться с использованием какой-либо из существующих методик объектно-ориентированного проектирования [26–27].

1.2.3 Существующие подходы к разработке многоагентных систем

В обзоре Гартнера, вышедшем в октябре 2015 года, МАС и технологии включены в список наиболее перспективных информационных технологий следующих лет [28]. Привлекательными для МАС технологий до настоящего времени являются задачи коллективной робототехники, однако можно указать и ряд других классов приложений, для которых многоагентная парадигма применима, например, для приложений управляющих сложными объектами сетевой структуры в таких сферах, как транспорт, производство, здравоохранение, энергетика и т.д. Одной из самых привлекательных сторон МАС - парадигмы является ее способность естественно и эффективно решать самую трудную задачу разработки сложных программ – взаимодействий множества компонентов программы.

Автор[29] считает, что разработчики МАС, в первую очередь, должны продемонстрировать способность МАС - технологии создавать сложные адаптивные распределенные системы промышленного уровня.

Авторы [30] пишут, что исследования и разработки в области МАС - технологий к 2005 г. еще не достигли уровня зрелости, необходимого для использования в индустриальных разработках. К этому времени было создано и тестировалось несколько глубоко проработанных методологий МАС: Gaia, Tropos, MaSE, ADELFE, MESSAGE, Prometheus и др. В работе [31] проанализированы 152 различных приложения. Из них наиболее заметные и удачные разработки были выполнены в областях управления производством, транспортной логистики, аэрокосмических приложений и в энергетике (Turkey energy forecast, California Energy Commission).

В период до 2010 г. было разработано более десяти методологий проектирования и разработки МАС. Большинство разработок в области МАС сопровождалось также созданием инструментальных программных средств их поддержки. К тому времени были разработаны такие средства, как agentTool, Zeus, agentBuilder, PASSI, MASDAK и др.

Агенты МАС могут взаимодействовать с целью кооперативного решения некоторой общей сложной задачи. В таком варианте взаимодействие агентов имеет целью координацию локальных решений для достижения требуемого качества решения задачи в целом. Эта координация может достигаться либо в полностью распределенном варианте, либо с той или иной степенью централизации, реализуемой агентом, специально выделенным для этих целей.

Другой характер взаимодействия агентов реализуется в случае, когда каждый агент имеет свои цели, однако он по каким-либо причинам не в состоянии решить задачу самостоятельно, а потому вынужден прибегать к помощи других агентов. Это взаимодействие агентов тоже имеет целью кооперацию, однако, в отличие от предыдущего, агенту в кооперации может быть отказано по причинам, принятым в соглашении о взаимных обязательствах агентов.

Еще один тип взаимодействия агентов имеет место тогда, когда агенты не кооперируются, а наоборот, конкурируют. В таком случае каждый агент имеет собственную цель и является «самозаинтересованным».

С самого начала развития теории агентов и МАС в качестве базовой формальной модели интеллектуального агента была выделена BDI-модель (Belief-Desire-Intention, Убеждения – Желание – Намерение) [32]. Суть ее состоит в том, что эта концепция оперирует двумя типами понятий: ментальными и поведенческими. К первым относятся события, убеждения, цели и взаимные убеждения. События отражают то, что происходит с агентами и объектами внешнего мира в реальном времени. Убеждения – знания агента о внешнем мире. Цели – состояния агента, к которым он стремится. Взаимные убеждения агентов складываются из убеждений группы агентов.

К поведенческим понятиям BDI-модели относятся индивидуальные обязательства и соглашения агента, его индивидуальные, а так же общие намерения, а также коллективные обязательства и намерения команды агентов. Индивидуальные обязательства – цели, которые необходимо достигнуть агенту в соответствии с теми задачами, которые агент взял на себя. Индивидуальные намерения – действия, которые агент собирается выполнять для достижения цели. Соглашениями называют условия, при выполнении которых агент может отказаться от индивидуальных обязанностей. Если МАС является командой агентов, то соглашения определяются однозначно и являются жесткими [33] .

К сожалению, в отношении агентных технологий пока нет четких методов разработки и алгоритмов функционирования. Основные достижения в этой части пока ориентируются на аспекты теоретической реализации и зачастую далеки от практики.

Для организации процесса распределения задачи в многоагентных системах создается либо система распределенного решения проблемы, либо

децентрализованная система искусственного интеллекта. В случае использования последней распределение заданий происходит в процессе взаимодействия агентов и носит больше спонтанный характер [34].

Децентрализованная система искусственного интеллекта используется в основном для агентного моделирования в том случае, когда более важен не финальный результат, а организация процесса, например, моделирование поведения толпы при эвакуации или же торги между продавцом и покупателем на рынке. Тогда каждый агент будет преследовать свои собственные цели, которые могут быть полностью противоположны интересам других агентов [35].

Одним из самых известных программных продуктов, реализующих данную концепцию, является AnyLogic. Это инструмент имитационного моделирования, который позволяет проектировать агентные, системно-динамические, дискретно-событийные и «многоподходные» модели [36]. Так же AnyLogic предоставляет широкий спектр отчетов и статистику по работе модели. Для создания моделей в основном используется графический редактор, где пользователь визуально описывает все составные элементы будущей модели. Для задания логики поведения отдельных элементов созданной модели применяется объектно-ориентированный язык программирования Java.

Так же децентрализованный подход используется, например, для программирования поведения реальных роботов. Одним из примеров такой реализации является платформа CogniTAO, разработанная на C++.

Однако же программные средства, придерживающиеся данной концепции, подходят исключительно для моделирования процессов и не могут помочь при решении каких-либо вычислительных задач.

Если нам необходимо решать комплексные вычислительные задачи, то в такой ситуации агенты должны действовать сообща, чтобы выполнить задачу максимально качественно и в наиболее короткие сроки. Также во

время работы системы в нее могут входить новые агенты, а другие перестают работать. Агенту необходимо получать информацию о других агентах, находящихся в системе, чтобы знать, с кем можно взаимодействовать. В таком случае процесс декомпозиции глобальной задачи и обратный процесс агрегирования найденных решений выполняются под управлением некоторого единого «центра». При этом многоагентная система проектируется строго сверху вниз, исходя из ролей, определенных для агентов, и результатов разбиения глобальной задачи на подзадачи. Несмотря на большое количество программных реализаций для этого типа задач, многие из них уже не развиваются и не поддерживаются разработчиками.

Кроме этого, одной из современных технологий для построения многоагентых систем является SOA (Service Oriented Architecture). Преимущества использования Web-сервисов для реализации агентов рассмотрены в [37-42]. Однако данный подход применялся лишь в единичных случаях.

Для разработки стандартов в области создания многоагентных систем была сформирована организация FIPA (Foundation for Intelligent Physical) [43]. Было достаточно много разработок, которые поддерживали стандарты предложенные FIPA, среди них такие как: Java Intelligent Agent Compontentware, The SPADE Multiagent and Organizations Platform, JACK Intelligent Agents, The Fipa-OS agent platform, AgentService, Zeus Agent Building Toolkit и другие. К большому недостатку FIPA-стандартов следует отнести то, что спецификации FIPA вообще не рассматривают проблемы параллельного программирования, хотя МАС – это концепция, изначально ориентированная на параллельные вычисления.

Одной из широко распространенных программных сред для разработки многоагентных систем является JADE**,** написанная на языке Java [44].

JADE использует концепцию распределенного решения задач. Основой данной системы является программная среда, без нее невозможно существование агентов. Внутри среды формируются контейнеры, в которые затем помещаются агенты. После запуска каждый агент должен передать данные о себе в один из контейнеров, чтобы зарегистрироваться в системе. Далее программная среда будет наблюдать за работой всей системы и при необходимости применять необходимые управляющие воздействия на отдельных агентов.

Между собой агенты обмениваются при помощи сообщений на языке ACL, который поддерживает стандарт FIPA [43, 45]. Сообщения могут носить разный характер: запрос, информирование, передача данных и т.д. Во время своей работы агент накапливает сообщения, которые приходят ему от других агентов, и по очереди занимается их обработкой.

JADE**,** на данный момент, является одним из наиболее активно развивающихся «Фреймворков» для разработки многоагентных систем. В JADE агенты активно взаимодействуют между собой и самоорганизуются для достижения поставленной цели. В нашем случае подобного тесного взаимодействия между агентами не требуется, они обмениваются лишь информацией, необходимой для расчетов. Использование такой комплексной платформы, как JADE, в данном случае не является целесообразным.

Таким образом, на данный момент существует очень небольшое количество платформ, позволяющих организовывать работу многоагентных систем. Большинство созданных ранее систем уже не поддерживаются и не развиваются, за исключение единичных представителей. Именно поэтому разработка новых подходов к реализации многоагентных систем является актуальной и востребованной задачей.

1.2.4 Joiner-сети как основа моделирования взаимодействия агентов

Аппарат Joiner-сетей является одной из разновидностей алгебраических сетей, предложенной проф. МФТИ Л.Н. Столяровым и

развиваемой его учениками [46, 47]. Joiner-сети (Joiner-Nets – JN) можно рассматривать как расширение сетей Петри, ориентированное на построение поведенческих моделей. В основе теории JN лежит описание логики взаимодействия асинхронных процессов в виде набора пусковых и флаговых функций, состоящих из булевых функций. Особенностью JN является то, что они предусматривают как графическое представление, так и описание в виде логических формул, обработку которых можно автоматизировать.

Рассмотрим формальное определение Joiner-сети (JN) [46].

Определение. Формулой JN называется направленный граф $JN = < \Phi, \Psi \subseteq (\Phi \times \Phi) >$, который состоит из двух видов вершин: позиций и переходов, которые связаны направленными однократными дугами. В этом смысле нотация для Joiner-сетей эквивалентна нотации сетей Петри.

Joiner-сети (**JN**) – сети Петри со специальной формой логической интерпретации – другими словами, в качестве пусковых и флаговых функций Joiner-сети могут выступать произвольные булевы функции. Теоретически JN являются обобщением практически всех подклассов сетей Петри [46, 48, 49]. Именно эта особенность делает Joiner-сети наиболее предпочтительным инструментом для событийного моделирования. Подробно аппарат Joiner-сетей рассматривается во второй главе.

На рисунке 1.1 показан элемент JN с входными позициями $P_1,...,P_i,...,P_n$ для событий, которые являются причинами «включения» процесса (или его состояния) S. Следствием является единственное событие P, которое говорит о том, что процесс S выполнен (завершен). Информация о событии «раздается» остальным элементам системы под другими именами: $P_{n+1},...,P_i,...,P_{n+m}$. Позиция P обычно специально не указывается, она размещается «внутри» процесса S, т.к. связана с вычислением соответствующего предиката процедурами, описанными внутри процесса.

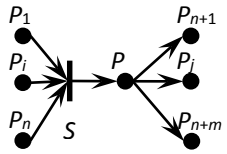


Рис. 1.1 Элемент Joiner-сети.

Основная цель теории Joiner-сетей – построение математической и информационной технологии для моделирования систем, состоящих из множества взаимодействующих между собой процессов. До появления представления в виде Joiner-сетей единственным формализованным механизмом описания и анализа сетей был граф, заданный на вершинах двух сортов (позициях и переходах), который позволял описывать взаимодействия между процессами. Средства анализа графов весьма ограничены, с одной стороны, матричной формой его представления, а с другой стороны, его графическим представлением. Матричная форма очень сложна для выявления причинно-следственных связей в асинхронной системе (фактически матрица их не отражает), а графическая форма столь нерегулярна, что не дает возможности выявить в этом «клубке» нужные для анализа структуры. Более того, каждый автор модели «рисует» сеть собственным способом, не поясняя при этом, придерживается ли он некоторого закона, либо располагает элементы сети случайным образом. При большом числе процессов в сети (например, при числе переходов и числе событий более ста) программная реализация, а тем более отладка асинхронной ее работы, представляет непреодолимые трудности. Использование Joiner-сетей позволяет снять большинство перечисленных ограничений.

В основе событийного моделирования лежит представление поведения некоторой системы как конечного набора системных событий, последовательная реализация которых отображает изменения состояния системы. Описание последовательности реализации этих событий и является предметом событийного моделирования в данной работе. Сущность

событийного метода моделирования заключается в отслеживании на модели последовательности событий в том же порядке, в каком они происходили бы в реальной системе. Вычисления выполняют только для тех моментов времени и тех частей (процедур) модели, к которым относятся совершаемые события. Другими словами, обращения на очередном такте моделируемого времени осуществляются только к моделям тех элементов, на входах которых в этом такте произошли изменения [49].

Событийная модель какой-либо системы представляет собой набор связанных между собой причинно-следственными связями событий, реализация которых отражает динамику поведения системы в ответ на возникновение инициирующего события. Задаваемые моделью деревья реализации событий описывают сценарии реакции системы на возникновение инициирующего события, стоящего в начале цепочки. Таким образом, событийная модель позволяет получить множество альтернативных сценариев развития заданной ситуации в системе. Такие модели относятся к классу поведенческих [50].

Основной целью событийного моделирования является выработка множества сценариев развития событий, отражающих варианты поведения системы. Множество вариантов формируются за счет получения из модели дерева реализации событий. Каждое такое дерево содержит в начале инициирующее событие – все остальные события дерева являются следствием его реализации. Если в процессе построения модели эксперт не описал ни одного события, имеющего альтернативные причинно-следственные связи с другими, то множество вариантов развития событий будет состоять лишь из одного дерева реализации событий.

Представляется, что применение Joiner-сетей для описания агентных сценариев позволит наглядным образом спроектировать алгоритм работы системы.

1.3 Оценивание состояния ЭЭС

1.3.1 Основные положения и важность задачи оценивания состояния

В современных условиях функционирования и управления ЭЭС требуется создание расчетной модели для схем большой размерности (порядка нескольких тысяч узлов) на базе методов оценивания состояния ЭЭС. Такие схемы не полностью наблюдаемы, возможно, искажение данных, плохая их синхронизация. Искажение результатов оценивания состояния вследствие появления ошибочных телеизмерений, несоответствия текущей расчетной схемы и используемых математических моделей реальному состоянию ЭЭС, потеря наблюдаемости вследствие отказа в системах сбора и передачи данных могут привести к искажению результатов оценивания состояния, замедлению сходимости вычислительного процесса, вплоть до расходимости, и, как следствие, к неправильным решениям, формируемым на базе полученной расчетной модели. Из этого следует, что нужно использовать методы декомпозиции.

Оценивание состояния – одно из возможных средств повышения качества информации о текущем режиме электроэнергетической системы, используемой при управлении ЭЭС [12].

Результатом оценивания состояния является расчет установившегося режима (текущего состояния) ЭЭС на основе измерений. В качестве измерений при оценивании состояния ЭЭС до недавнего времени в основном использовались телеизмерения переменных режима и телесигналы о состоянии коммутационного оборудования, получаемые от системы SCADA [14]: модули узловых напряжений U_i, генерации активных и реактивных мощностей в узлах, перетоки мощностей по линиям P_{ij}, Q_{ij}, P_{ji}, Q_{ji}, реже – токи в линиях I_{ij} и узлах I_i .

$$\bar{y}=\{P_i, Q_i, P_{ij}, Q_{ij}, U_i, I_i, I_{ij}\} \quad (1.1)$$

Для получения узловых инъекций P_i Q_i , в дополнение к ТИ генераций использовались псевдоизмерения нагрузок в узлах.

Эффективность решения задачи оценивания состояния во многом определяется качеством исходной информации (телеизмерения, телесигналы). Далее в главе будет рассмотрено современное состояние системы сбора и обработки информации при управлении ЭЭС, описана математическая постановка задачи оценивания состояния, рассмотрены методы распределенного оценивания состояния, выполнен анализ проблем оценивания состояния в современных условиях и отмечены возможные пути их решения.

Информационная система, применяемая в диспетчерском управлении ЭЭС, предназначена для сопровождения процессов долгосрочного, краткосрочного и оперативного управления режимами энергосистемы и является основным технологическим средством, использующимся в упомянутых процессах. Получение исходных данных, их передача, преобразование и выполнение диспетчерских команд выполняются с помощью информационной системы. От качества компонентов системы и ее построения зависит успешность управления режимами энергосистемы.

С появлением новых программных и аппаратных средств в России был совершен переход к SCADA-системам. Концепция SCADA предопределена всем ходом развития систем управления и результатами научно-технического прогресса. Применение SCADA-технологий позволяет достичь высокого уровня автоматизации в решении задач разработки систем управления, сбора, обработки, передачи, хранения и отображения информации [14].

В настоящее время SCADA является основным и наиболее перспективным инструментом автоматизированного управления сложными динамическими системами (процессами), к которым относится система управления ЭЭС.

К недостаткам SCADA-систем применительно к задачам энергетики можно отнести недостаточный объем и невысокую точность ТИ. Кроме того,

в SCADA-системах невозможна абсолютная синхронизация данных из-за того, что производится последовательное сканирование измерений.

Существенно более высокий уровень наблюдаемости и управляемости в ЭЭС может быть достигнут с внедрением технологии WAMS (Wide-Area Measurement Systems). Развитие GPS дало возможность одномоментно регистрировать показания на всех объектах энергосистемы, оснащенных так называемыми GPS-синхронизированными устройствами. Современная система синхронизированных измерений параметров режима в энергосистемах – WAMS – и основанная на этих измерениях технология управления большими энергообъединениями – WAMS-технология – находит все более широкое применение в электроэнергетике. Стало ясно, что использование сигналов времени GPS, как входов в выборку времени в измерительной системе цифровых реле, становится мощным измерительным инструментом, способным обеспечить мгновенную картину состояния ЭЭС и фактически получить характеристики, которые сделают эти измерения эффективными, даже если не будет достигнута полная наблюдаемость этими новыми средствами.

Основным измерительным оборудованием систем WAMS, позволяющих контролировать состояние ЭЭС синхронно и с высокой точностью, являются приборы для измерения комплексных электрических величин – PMU [15]. Современные PMU вышли из SCDR (Symmetrical Component Distance Relay) в начале 70-х гг. XX в. Тогда у микрокомпьютеров, соединенных с реле, не было возможности обрабатывать переключения с помощью алгоритмов дистанционного управления. Новые принципы работы реле позволяют использовать симметричные компоненты напряжения и тока для получения уравнения с симметричными компонентами.

Измерения, поступающие от PMU, в сочетании с телеизмерениями, пришедшими от SCADA, более полно отражают режим рабочей схемы ЭЭС.

По сравнению со стандартным набором телеизмерений, получаемым от системы SCADA, PMU, установленное в узле, может обеспечить измерение фазы напряжения в этом узле и фаз токов в некоторых или во всех инцидентных этому узлу ветвях в зависимости от пропускной способности каналов связи.

1.3.2 Математическая постановка задачи оценивания состояния

Согласно [12], задача оценивания состояния для наблюдаемой ЭЭС может быть сформулирована как задача расчета оценок измеренных переменных $\hat{y}$ по данным измерений $\bar{y}$. Модель измерений, используемых при оценивании состояния, имеет вид:

$$\bar{y} = y_{ист} + \xi_y \,, \qquad (1.2)$$

$y_{ист}$ — вектор истинных значений измеренных величин;

$\bar{y}$ — значения измерений;

ξ_y — вектор ошибок измерений [51], имеющих нормальное распределение с нулевым математическим ожиданием и известной дисперсией

$$\xi_y \rightarrow N(0, \sigma_y^2)$$

Поскольку измерения $\bar{y}$ содержат погрешности, то ищутся оценки, которые наиболее близки к измеренным значениям $\bar{y}$ в смысле некоторого критерия и удовлетворяют уравнениям электрической цепи, связывающим измеренные **y** и неизмеренные *z* переменные режима.

$$w(y, z) = 0, \qquad (1.3)$$

Чаще всего в качестве критерия при оценивании состояния используется сумма взвешенных квадратов отклонений оценок от измерений:

$$J(y) = (\bar{y} - \hat{y})^T R_y^{-1} (\bar{y} - \hat{y}), \qquad (1.4)$$

где R_y^{-1} — диагональная матрица весовых коэффициентов, диагональные элементы которых обратны дисперсиям измерений: $r_{ii} = \sigma_i^2$.

При решении задачи оценивания состояния вводится понятие вектора состояния *x* размерностью *2n-1* (где *n* — число узлов расчетной схемы), включающего модули *U* и фазовые углы *δ* напряжений *x = (δ, U),* кроме фиксированной фазы базисного узла. Такой вектор состояния однозначно определяет все остальные переменные режима. В этом случае в качестве уравнений (1.3) используются явные зависимости измеренных и неизмеренных переменных от *x*:

$$y = y(x), \qquad (1.5)$$

$$z = z(x). \qquad (1.6)$$

Уравнения (1.5) используются для определения компонент вектора состояния *x* по измеренным переменным. В этом случае задача ОС сводится к минимализации критерия, т.е. поиску таких оценок вектора состояния $\hat{x}$, при которых вычисленные значения измеряемых переменных режима $y(\hat{x})$, были бы наиболее близки к измерениям также в смысле некоторого критерия, который при ОС по МВНК имеет вид:

$$J(x) = [\overline{y} - y(\hat{x})]^T - R_y^{-1}[\overline{y} - y(\hat{x})] \qquad (1.7)$$

Затем с использованием (1.6) ищутся оценки неизмеренных переменных $z(\hat{x})$. Этот метод подробно представлен в [12, 53].

Возможны и другие подходы к выбору критерия оценивания состояния [52, 54].

Вследствие нелинейной зависимости *y(x)* , задача решается итеративно.

$$\Delta x_k = \left[H_k^T R_y^{-1} H_k\right]^{-1} H_k^T R_y^{-1}[\overline{y} - y(x_k)]. \qquad (1.8)$$

Здесь H_k – матрица Якоби, вычисленная на *k* – ой итерации. Имеются и другие подходы к решению задачи оценивания состояния, обзор которых представлен в [53].

Существенно отличающаяся постановка была предложена в [55, 51]. Она состоит в использовании так называемых контрольных уравнений, которые могут быть получены при исключении неизмеренных переменных

из уравнений установившегося режима ЭЭС. Этот метод будет рассмотрен в 2.1 и использован в данной работе для реализации декомпозиционных алгоритмов оценивания состояния.

Процедура оценивания состояния включает в себя решение следующих основных задач [12]:

- формирование текущей расчетной схемы по данным телесигналов,
- анализ наблюдаемости,
- выявление грубых ошибок в телеизмерениях или обнаружение плохих данных (ОПД),
- фильтрация случайных погрешностей телеизмерений, т.е. получение их оценок,
- дорасчет неизмеренных переменных.

Основные проблемы, возникающие при решении задачи оценивания состояния, связаны с недостаточным объемом и низким качеством измерительной информации, поступающей от системы SCADA. Это приводит к ошибкам при формировании расчетной схемы, появлению ненаблюдаемых районов, критических измерений и критических групп, которые невозможно обнаружить [56], размазыванию грубых ошибок и, как следствие – искажению результатов оценивания состояния и низкой точности получаемых оценок.

При оценивании состояния ЭЭС большой размерности помимо проблем, вызванных недостаточным объемом и низким качеством измерительной информации, возникают проблемы, связанные с неоднородностью расчетных схем. Приведение системы (1.7) к нормальному виду еще больше усугубляет эту проблему и приводит к медленной сходимости вычислительного процесса (1.8) и существенному искажению результатов оценивания состояния из-за появления даже единичных плохих данных.

1.3.3 Алгоритм декомпозиции задачи оценивания состояния на основе метода контрольных уравнений

В ИСЭМ СО РАН для решения задачи ОС разработан метод контрольных уравнений. Суть метода состоит в использовании для оценивания состояния системы уравнений установившегося режима, описывающих состояние ЭЭС и включающих только измеренные переменные. Метод контрольных уравнений (КУ) позволяет значительно упростить процедуры ОС ЭЭС и анализа телеизмерений [52].

КУ – это уравнения электрических цепей, в которые входят только измеренные переменные режима y:

$$w_k(y) = 0. \quad (1.9)$$

Они могут быть получены из системы уравнений установившегося режима (УУР) после исключения неизмеренных переменных.

Впервые предложенные для достоверизации телеизмерений [57], контрольные уравнения затем стали применяться для решения практически всех задач, входящих в комплекс оценивания состояния в реальном времени [58]. Разработаны топологические и алгебраические методы получения контрольных уравнений [52]. Следует отметить, что все методы получения контрольных уравнений тесно связаны с методами анализа наблюдаемости и разделением измерений на базисные и избыточные [59], поскольку число контрольных уравнений определяется числом избыточных измерений в схеме.

Для проверки гипотез о наличии плохих данных, в том числе ошибок в топологии сети, разработаны алгоритмы логического анализа информации, входящей в контрольные уравнения. В результате применения указанных логических правил, все измерения делятся на 4 группы:

1) достоверные;

2) ошибочные – измеренные значения, которые могут быть заменены вычисленными;

3) сомнительные – измерения, вошедшие только в контрольные уравнения с большими невязками, но такие, что вычислить их оценки по данным достоверных измерений не удается (например, их число больше числа контрольных уравнений, в которые они входят);

4) непроверенные – это измерения, не вошедшие в контрольные уравнения, ошибки в них обнаружить нельзя.

Существенно улучшить свойства решения задачи ОС позволяет использование измерений, поступающих от устройств измерения комплексных электрических величин — PMU (Phasor Measurement Units). Измерения, поступающие от PMU, более полно отражают режим рабочей схемы ЭЭС. Использование данных PMU открывает новые возможности при декомпозиции задачи ОС [15].

В [61] был предложен алгоритм решения задачи оценивания состояния с учетом структурной и функциональной декомпозиции. Данный алгоритм состоит из следующих этапов:

1. Выполняется деление полной расчетной схемы ЭЭС на достаточно крупные подсистемы. В базисном узле полной схемы устанавливается устройство PMU.

А. В граничных (не транзитных) узлах подсистем устанавливаются устройства PMU для измерения комплексов узловых напряжений и токов в линиях. В подсистемах, не содержащих базисный узел полной схемы, в качестве базисного выбирается один из граничных узлов самого высокого класса напряжений с PMU. Измерения узловых инъекций в граничных узлах исключаются из вектора измерений.

Б. Если граничные узлы двух (и более) подсистем транзитные, или так задано изначально, то используется декомпозиция с граничными ветвями. В одном из граничных узлов устанавливается PMU, в другом узле ветви – расчетное PMU [15]. При расчете первой

подсистемы из вектора измерений исключаются измерения узловых инъекций в граничном узле второй подсистемы, при расчете второй подсистемы – в граничном узле первой.

2. На втором этапе декомпозиции расчетная схема каждой подсистемы в свою очередь делится на области, соответствующие уровням узловых напряжений. Границами областей являются узлы, смежные с узлами класса напряжения данной области. Так, для области класса напряжения 750-500 кВ граничными являются узлы напряжением 220 кВ, и наоборот.

3. Расчет начинается с области самого высокого уровня напряжения (750-500 кВ) для каждой подсистемы первого уровня декомпозиции. Как правило, эта часть схемы хорошо обеспечена телеизмерениями высокой точности и содержит базисный узел. Алгоритм оценивания состояния по подсистемам с граничными узлами состоит в следующем:

3.1. Для каждой области, включающей граничные узлы, решается задача обнаружения плохих данных методом контрольных уравнений.

3.2. В случае обнаружения всех плохих данных или их отсутствия решается задача оценивания состояния методом взвешенных наименьших квадратов.

3.3. В случае невозможности определения всех ошибочных телеизмерений, и, соответственно, невозможности обнаружения плохих данных, выполняется оценивание состояния с помощью робастного критерия (подавления плохих данных).

4. Последовательно рассчитываются остальные фрагменты схемы, ранжированные по уровням напряжений (220 кВ, 110 кВ и т.д.), всякий раз в качестве базисного узла выбирается узел, граничный с областью более высокого уровня напряжений. Оценки граничных переменных вектора состояния, полученные на верхнем уровне декомпозиции, фиксируются.

5. Рассчитываются инъекции в граничных узлах между областями разного класса напряжения.

6. После окончания расчета всех подсистем первого уровня декомпозиции аналогичная задача решается для граничных узлов с PMU, если узлы транзитные, то вычисляются перетоки в граничных связях.

7. Для формирования общего решения для всей схемы выполняется агрегирование результатов, полученных для отдельных подсистем, и результатов решения координационной задачи [61].

Одним из возможных подходов к реализации данного алгоритма на практике является использование многоагентной технологии [61]. Агенты как независимые, самостоятельные программы обладают всеми необходимыми характеристиками для проведения распределенных вычислений. Каждый агент будет отвечать за определенную часть алгоритма ОС и контактировать с другими агентами для выполнения задания, формируя, таким образом, многоагентную систему.

1.4 Выводы по главе 1

По результатам анализа, выполненного в главе, можно сделать следующие выводы:

1. Анализ предметной области, связанной с разработкой концепции интеллектуальных энергетических систем (ИЭС), позволил выявить проблемы, возникающие при попытках применения многоагентной технологии, как одной из базовых при реализации концепции ИЭС.
2. Выполнен аналитический обзор существующих на данный момент агентных решений, выделены их положительные и отрицательные стороны.
3. Обоснована целесообразность разработки методического подхода к построению типовых многоагентных систем в энергетике.
4. Предложена разработка методического подхода к реализации многоагентных систем на примере решения задачи оценивания состояния ЭЭС. Описаны существующие алгоритмы оценивания

состояния. Обоснована необходимость применения методов декомпозиции и их реализации на основе многоагентного подхода.

5. Обоснована целесообразность разработки собственного метода построения многоагентных систем для оценивания состояния ЭЭС.
6. Рассмотрен аппарат Joiner-сетей – одной из разновидностей алгебраических сетей, ориентированной на построение поведенческих моделей.
7. Проанализирована возможность использования Joiner-сетей для моделирования взаимодействия агентов.

2 Предлагаемый методический подход к построению типовых многоагентных систем на примере задачи оценивания состояния ЭЭС

2.1 Содержательное и формальное описание проблемы.

Использование многоагентных систем в энергетике является важной частью реализации концепции «Интеллектуальных энергетических систем». На данный момент в ИСЭМ СО РАН было выполнено несколько работ по применению многоагентного подхода в решении задач энергетики.

Построением интеллектуальных ПВК для решения энергетических задач занимались В.Г. Курбацкий и Н.В. Томин. Предложен ПВК на базе макрос-приложений, позволяющий из имеющихся математических «программно-готовых» моделей выбирать наиболее оптимальную для решения конкретной электроэнергетической задачи [62].

Д.А. Фартышев применил многоагентный подход для разработки программного комплекса «ИНТЭК-М» [41, 42], предназначенного для поддержки исследований проблемы энергетической безопасности. Базой для реализации подобных комплексов является сервис-ориентированная архитектура (SOA), и, в частности, Web-сервисы.

Д.А. Панасецкий использовал многоагентный подход для разработки системы противоаварийного управления ЭЭС. При реализации данной системы была использована многоагентная платформа Jade [63]. А.С. Пальцев предложил использовать многоагентный подход для распределенного решения задачи оценивания состояния ЭЭС [60].

Существуют частные случаи использования многоагентного подхода для решения энергетических задач, каждый из которых основан на различной технологической базе. Однако попытка каждый раз проектировать систему с самого начала требует слишком много времени, которое будет потрачено на решение проблем функционирования системы, а не на прикладные задачи,

которые являются первостепенными при разработке. Таким образом, необходим универсальный подход к разработке многоагентных систем в области энергетики. В этой главе автором предлагается универсальный метод разработки типовых многоагентных систем для решения энергетических задач, апробация которого будет выполнена на основе алгоритма А.С. Пальцева, поскольку практическое применение данного подхода для решения задачи оценивания состояния не было выполнено.

2.2 Математическое описание методического подхода к построению типовых многоагентных систем

Система должна создаваться с определенной целью G, которую перед созданием необходимо четко определить. Цель достигается за счет выполнения определенного ряда задач, как простых, так и комплексных, которые можно разделить на отдельные подзадачи. Требуется определить множество всех задач $\{T\}$. Система должна уметь решать данные задачи. Сформулированное множество задач определяет множество функций будущей системы $\{F\}$. Функции системы распределяются между агентами $\{A\}$ так, чтобы каждый агент решал свою задачу, либо часть какой-то большой задачи. Таким образом, необходимо построить цепочку отображений: $G \rightarrow \{T\} \rightarrow \{F\} \rightarrow \{A\}$.

Однако выполнять действия необходимо в определенном порядке, к тому же некоторые задачи можно решить несколькими разными методами, поэтому необходимо определить порядок вызова агентов $\{P_A\}$. Исходя из этого порядка и множества представленных в системе агентов, необходимо генерировать множество агентных сценариев $\{S_A\}$, для которых требуется построить событийные модели описания этих сценариев $\{Es\}$. Для ускорения разработки МАС целесообразно реализовать базовые программные компоненты для агентов $\{C_B\}$. Таким образом, модель многоагентной системы можно представить, как:

$$M_{MAS} = (A, P_A, S_A, Es, C_B);$$

где A – множество агентов, P_A – порядок вызова агентов, S_A – совокупность агентных сценариев, E_S – множество событийных моделей, C_B – множество базовых компонентов.

2.3 Методический подход к построению типовых многоагентных систем.

2.3.1 Предлагаемый методический подход к построению многоагентных систем в энергетике

Исходя из того, что на данный момент не существует единого подхода к разработке многоагентных систем, предложен авторский **методический подход к построению типовых многоагентных систем, на примере задачи оценивания состояния ЭЭС**. Основой подхода является метод проектирования и реализации типовой МАС, который реализован в виде методики, включающей следующие этапы [64]:

1 Описание будущей системы, исходя из специфики решаемой задачи.

1.1 Определение цели создания МАС.

1.2 Выделение множества задач *{T}*, которые необходимо решить. Поставленная цель может быть достижима в несколько этапов и включать в себя решение нескольких задач, каждая из которых может быть разбита на подзадачи.

1.3 Определение множества функций МАС *{F}*, исходя из задач.

1.4 Определение списка будущих агентов *{A}* на основании функций системы. Алгоритм «большой» задачи необходимо разбить на этапы, за выполнение каждого из которых будет отвечать определенный агент. Точно также решение «малых», последовательных задач можно выполнять в рамках одного агента. Для тех задач, которые подразумевают возможность проведения параллельных вычислений, необходимо создать специальный тип агента, в системе будет содержаться несколько экземпляров агента данного типа. Должен присутствовать главный модуль, который

занимается мониторингом работоспособности всей системы, а также следит за выполнением поставленных задач.

1.5 Разработка базовых компонентов МАС $\{C_B\}$, которые могут быть использованы при реализации конкретных МАС.

2 Разработка агентных сценариев.

2.1 Определение порядка вызова $\{P_A\}$ имеющихся в системе агентов, включая возможность ветвлений, циклов и распределенных вычислений. Данный порядок является агентным сценарием, которых в системе может быть несколько, в зависимости от поставленной цели и тех задач, которые необходимо решить.

2.2 Разработка сценариев вызова агентов $\{S_A\}$.

2.3 Описание сценариев в виде событийных моделей $\{Es\}$, с использованием элементов Joiner-сетей.

Методика построения системы, с указанием всех участвующих в создании лиц и управляющих воздействий, приведена на диаграмме (рис. 2.1).

Рис. 2.1. Диаграмма методики построения многоагентных систем.

На рисунке 2.2 изображена декомпозиция метода на 2 основных этапа.

Рис. 2.2 Диаграмма декомпозиции методики построения МАС.

На рисунках 2.3-2.4 представлены декомпозиции каждого из 2-х основных этапов на более мелкие задачи.

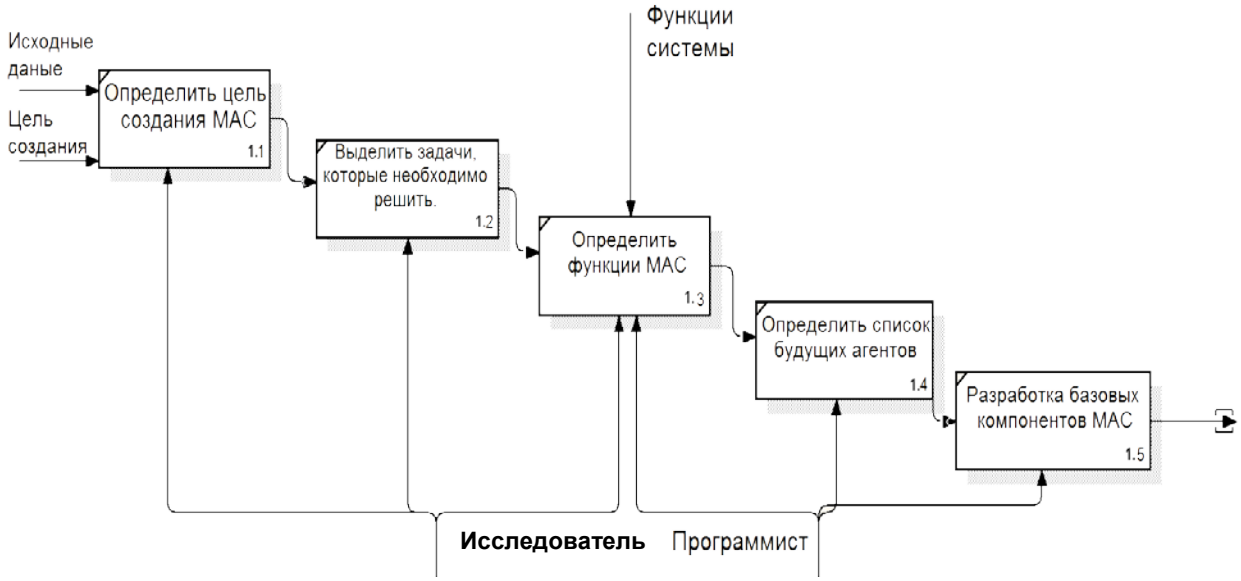


Рис. 2.3. Диаграмма декомпозиции этапа описания будущей системы.

Рис. 2.4. Диаграмма декомпозиции этапа написания агентных сценариев.

После того, как цель и задачи будущей системы определены, а список агентов и сценарии составлены, необходимо приступать к проектированию и реализации МАС [65] (вторая часть методики):

3 Разработка архитектуры МАС.

3.1 На основании созданного списка агентов и сценариев построить общую схему будущей системы с их участием.

3.2 Сформулировать функциональные, системные и технические требования к системе.

3.3 Определить список технологий, которые будут использованы при реализации системы.

4 Проектирование МАС.

4.1 Построить диаграмму прецедентов, IDEF0-модель и диаграммы классов для каждого агента.

4.2 На основании событийной модели разработать алгоритм взаимодействия между агентами. Составить список возможных сообщений, которыми агенты обмениваются в системе.

4.3 Выполнить проектирование пользовательского интерфейса.

5 Реализация МАС.

5.1 Разработать каждый агент в отдельности и провести его тестирование, чтобы убедиться, что заложенные в него алгоритмы решения задач работают корректно.

5.2 Разработать главный модуль, проверить правильность сетевого взаимодействия его и других агентов в системе.

5.3 Провести тестирование функции мониторинга работы системы.

5.4 На тестовом примере проверить корректность, как отдельных элементов, так и комплексную работу системы.

Этапы проектирования и реализации МАС, с указанием всех участвующих в создании лиц и управляющих воздействий, приведены на диаграмме (рис. 2.5). На рис. 2.6 показана декомпозиция на 3 основных этапа.

Рис. 2.5. Диаграмма методики построения многоагентных систем.

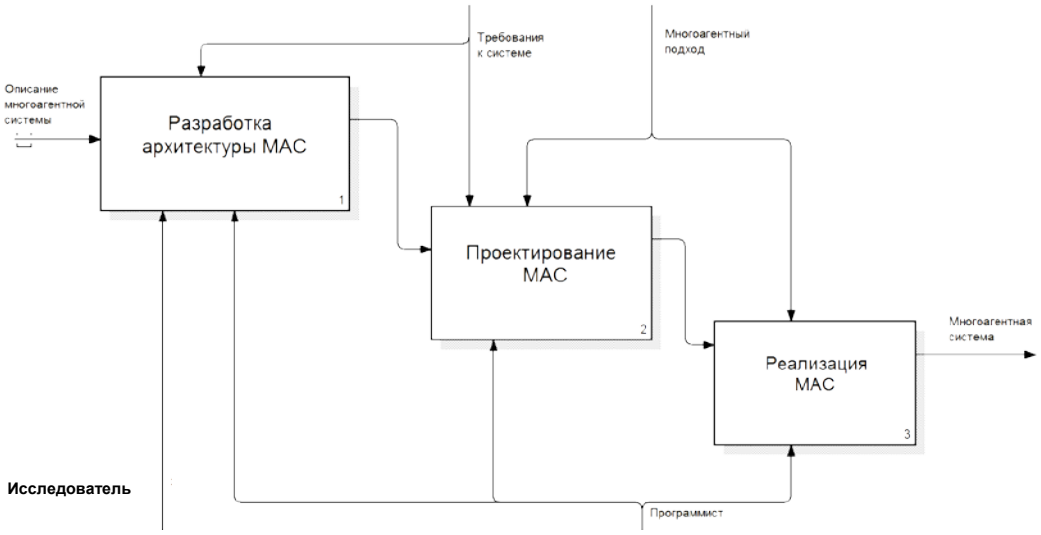


Рис. 2.6. Диаграмма декомпозиции методики построения многоагентных систем.

На рисунках 2.7-2.9 представлены декомпозиции каждого из 3-х основных этапов на более мелкие задачи.

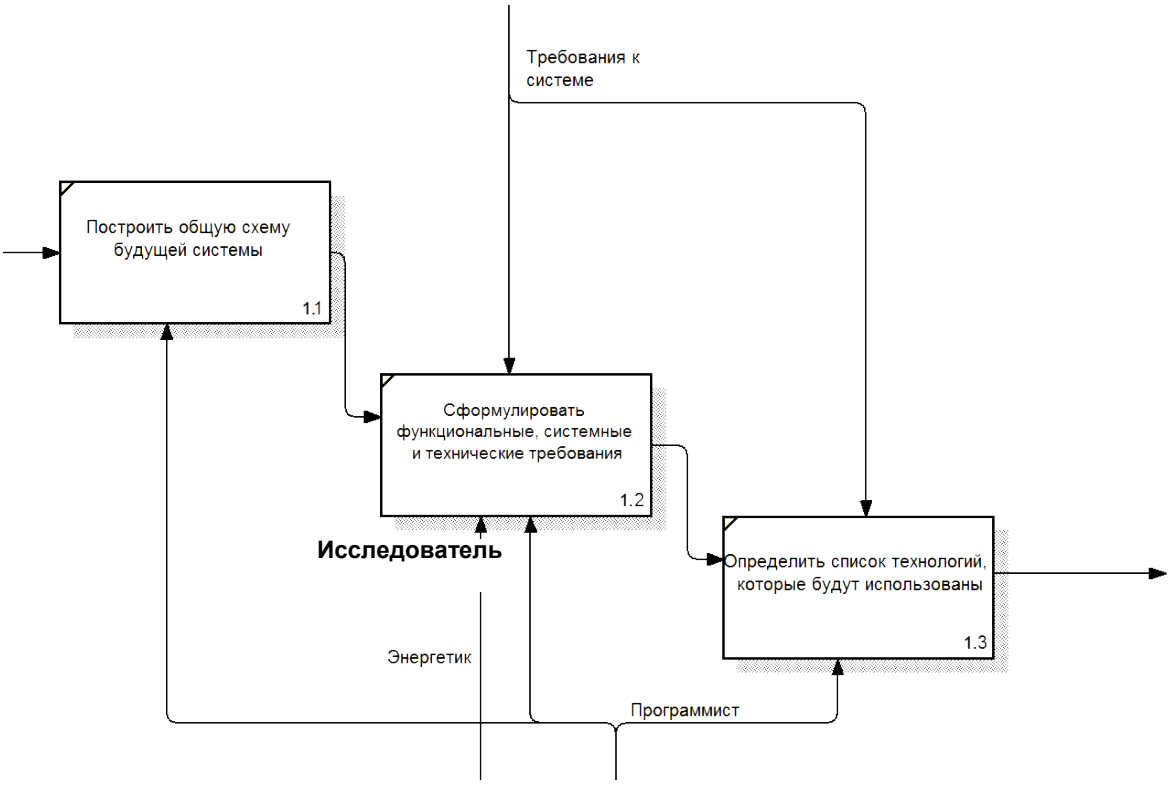


Рис. 2.7. Диаграмма декомпозиции этапа разработки архитектуры МАС.

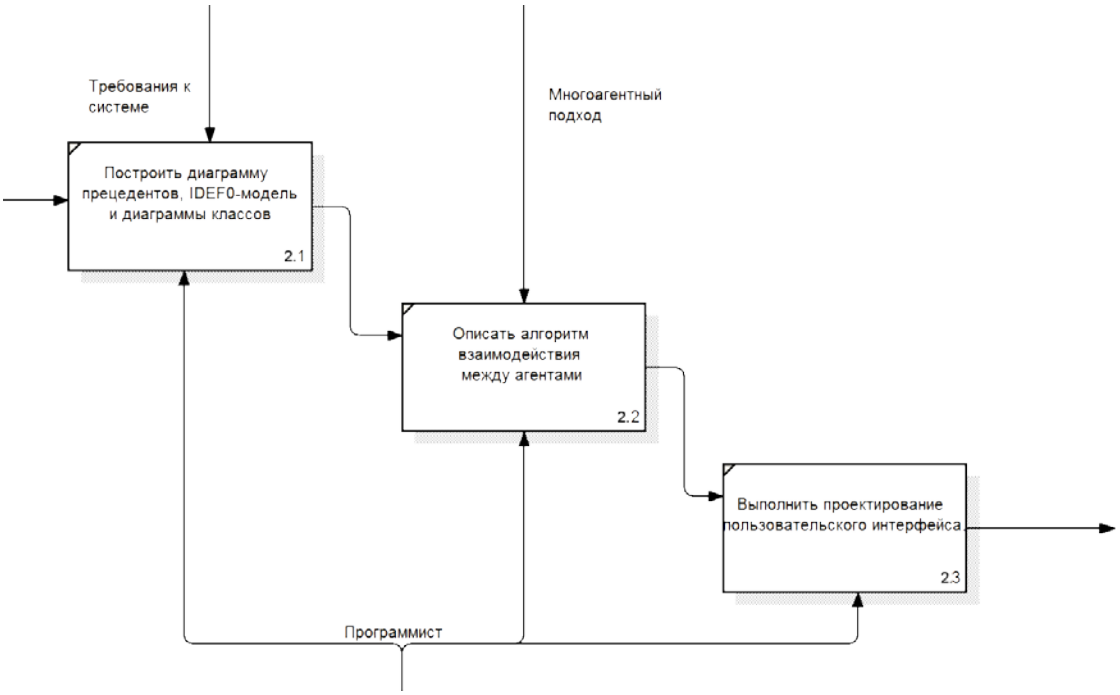


Рис. 2.8. Диаграмма декомпозиции этапа проектирования МАС.

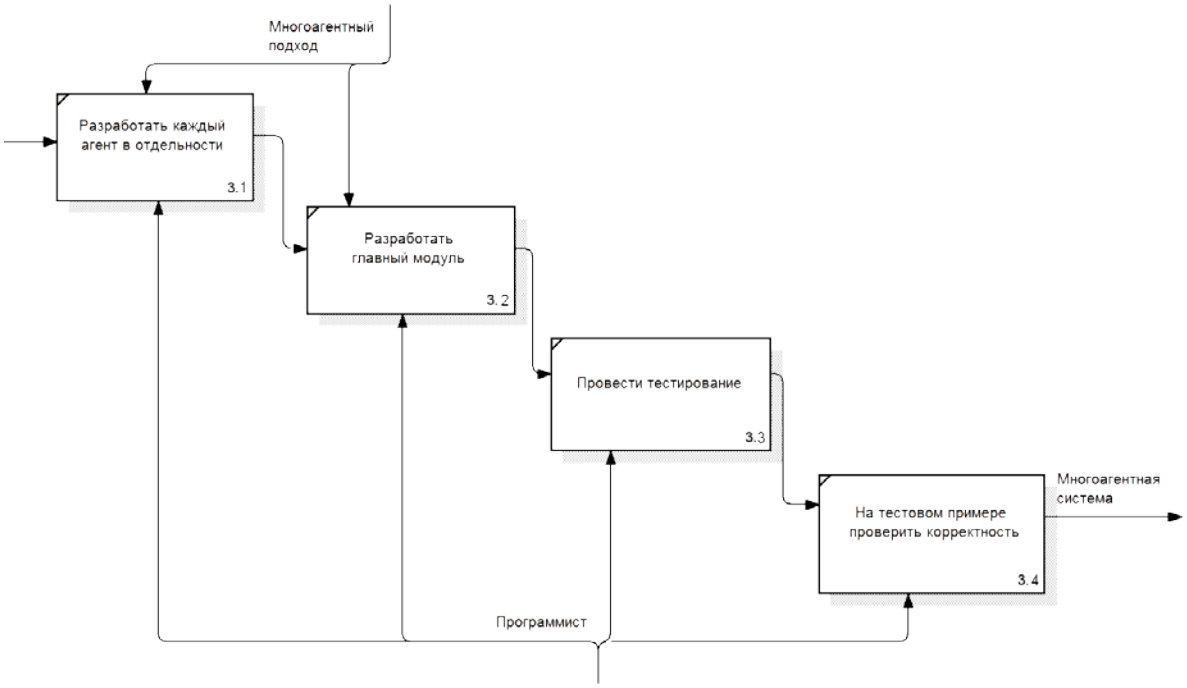


Рис. 2.9. Диаграмма декомпозиции этапа реализации.

2.3.2 Сценарии взаимодействия агентов (агентные сценарии)

Агентные сценарии предлагаются для редактирования алгоритма решения задачи без привлечения программистов. Сценарии формируются исследователями-энергетиками. Каждый агент в системе выполняет

определенный этап того или иного алгоритма. Однако возможны случаи, когда необходимо внести изменения в процесс решения задач.

Для ускорения процесса оценивания состояний применяется метод декомпозиции схемы ЭЭС на отдельные части, которые рассчитываются отдельно. Однако с течением времени появляются новые методы декомпозиции схемы. Включение этих методов в уже работающую систему потребует изменения как агентов, отвечающих за саму декомпозицию, так и главного модуля. Для того, чтобы не приходилось вносить изменения в главный модуль, и чтобы пользователь мог сам редактировать существующий алгоритм, предлагается использовать агентные сценарии. После включения агент должен передать главному модулю информацию не только о своем местонахождении в сети, но и о тех задачах, которые он умеет решать. Таким образом, составляется список различных типов агентов. Когда пользователю нужно провести расчеты, он может собственноручно выстраивать порядок, в котором будут вызваны те или иные агенты. При необходимости, он сможет отредактировать существующий алгоритм, добавить новые этапы расчетов или убрать существующие, чтобы получить промежуточные результаты [66].

Пример агентного сценария для решения задачи оценивания состояния ЭЭС приведен на рис. 2.10. На нем изображены два сценария оценивания состояния ЭЭС с разными методами декомпозиции исходной схемы.

В некоторых ситуациях агенту для решения задачи необходимо получить от пользователя дополнительные параметры. В таком случае агенты могут передавать не только информацию о местонахождении и решаемых задачах, но и о параметрах, которые требуются от пользователя. На основании полученных от агента данных главный модуль сгенерирует форму для ввода значений и передаст ее пользователю. Форма должна состоять из графических элементов интерфейса, таких, как: списки, поля

ввода текста, поля ввода чисел и т.д. После ввода необходимой информации пользователь отправляет ее главному модулю.

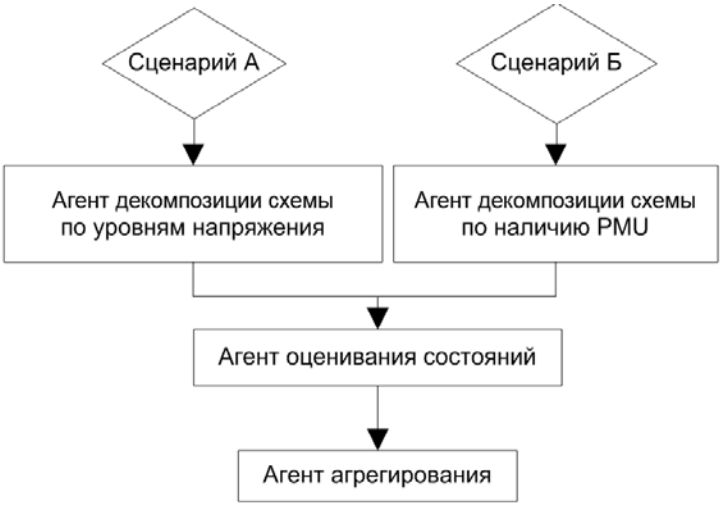


Рис. 2.10. Агентные сценарии для задачи оценивания состояний ЭЭС.

Главный модуль проверяет корректность введенной пользователем информации на основании тех ограничений, которые ему передал агент. После проверки данные возвращаются агенту, и на их основании он выполняет требуемые вычисления.

Применение предложенного подхода позволяет:

- Добавлять новые и изменять существующие алгоритмы в системе без необходимости привлекать программиста.
- Расширять функции системы за счет введения новых типов агентов.
- Предоставить пользователю автоматически сгенерированный графический интерфейс для ввода необходимых параметров.

Основное преимущество применения многоагентного подхода – это возможность организовать распределенную обработку данных. При создании нового или редактировании существующего сценария у пользователя должна быть возможность указать те места, в которых данные можно разделить между несколькими агентами одного типа. Также в сценарии можно будет задать максимальное и минимальное количество агентов, между которыми будет распределена задача. Данная ситуация иллюстрируется на рис. 2.11.

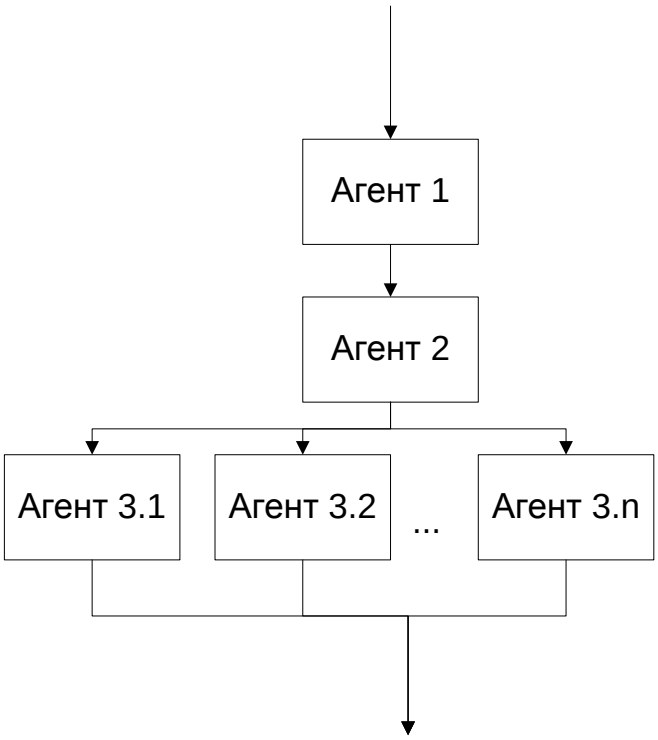


Рис. 2.11. Распределение задачи между агентами одного типа.

Для создания более «гибких» алгоритмов предлагается добавлять в сценарии условные операторы и циклы (рис. 2.12). В таком случае полученные на текущем этапе результаты будут проходить логическую проверку, по итогам которой будет выполняться то или иное действие.

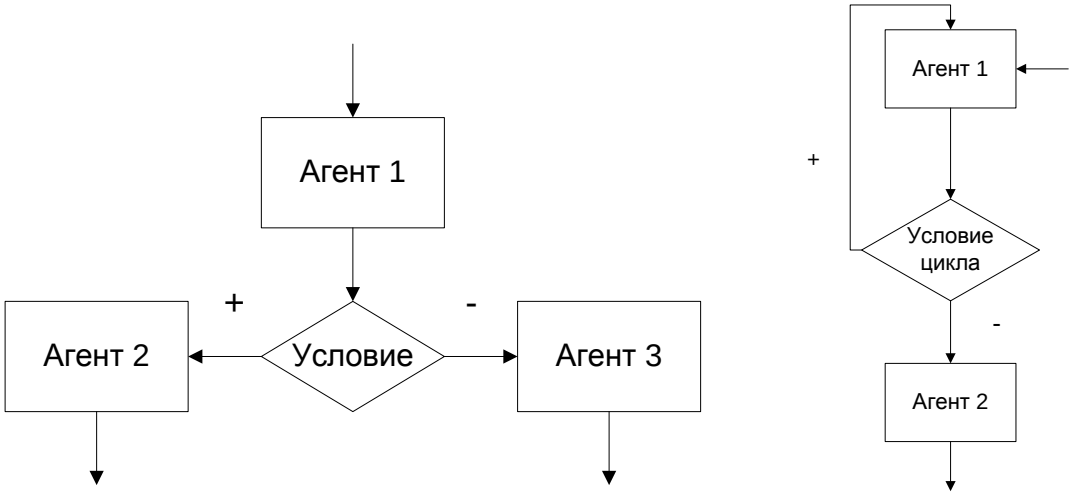


Рис. 2.12. Условные операторы и циклы в агентных сценариях.

2.3.3 Вычислительный метод управления взаимодействием агентов на основе алгебраических сетей (Joiner-net)

Предлагается вычислительный метод управления взаимодействием агентов на основе алгебраических сетей для решения задач оценивания состояний ЭЭС. Управление взаимодействием агентов осуществляется с помощью агентных сценариев.

Система состоит из следующих элементов:

1. Три типа агентов: агент – пользователь (U), агент – менеджер (M), агент – вычислитель (C).
2. Список агентов вычислителей: $L\ (c_1, c_2, \ldots, c_n)$.
3. Список задач G ($g_1, g_2, \ldots, g_n$).
4. Взаимно-однозначное соответствие: $g_i \leftrightarrow c_i$.
5. Множество пусковых ($\boldsymbol{\Psi}$) и флаговых ($\boldsymbol{\Phi}$) функций.

Для каждого сценария строится событийная модель на основе Joiner-сети. Модель можно представить в виде набора пусковых и флаговых функций. Предлагается использовать пусковые и флаговые функции для управления МАС и выполнения заданных пользователем сценариев.

Метод управления взаимодействием агентов на основе алгебраических сетей (рис. 2.13) включает следующие этапы:

1. Все флаговые функции принимают значение, равное 0.
2. Агент-пользователь (U) определяет сценарий (порядок вызова агентов и условия их запуска), по которому будет проходить расчет, и передает агенту-менеджеру (M) файл с исходными данными (F_D), генерируя событие, задавая флаговую функцию менеджера = 1, $\boldsymbol{U: F_D \rightarrow \Phi_M}$. Из события формируется пусковая функция ($\boldsymbol{\Psi_M}$) для агента-менеджера $\boldsymbol{U: \Phi_M \rightarrow \Psi_M}$.
3. Пусковая функция инициирует работу агента-менеджера $\boldsymbol{M: \Psi_M \rightarrow M^{\Psi}}$. Агент-менеджер формирует из исходных данных XML-файл $\boldsymbol{M: F_D \rightarrow F_{XML}}$.

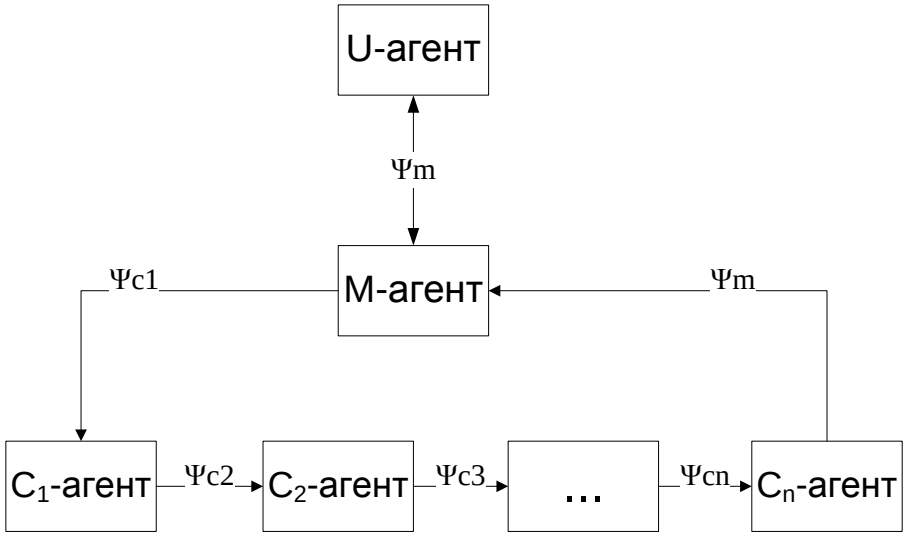


Рис. 2.13 Схема вычислительного метода управления взаимодействием агентов на основе Joiner-сетей.

4. В зависимости от выбранного сценария, агент-менеджер определяет первого агента-вычислителя (c_1) в цепочке вызовов, передает исходные данные, генерируя событие, и формирует пусковую функцию данного агента $\boldsymbol{M: M^{\Psi} \rightarrow M^{\Phi} \rightarrow \Phi_M \rightarrow \Psi_{C1}}$.
5. Пусковая функция инициирует работу агента-вычислителя, который получил данные $\boldsymbol{c_1: \Psi_{c1} \rightarrow c_1^{\Psi}}$. Агент выполняет свою часть алгоритма, после чего определяет агента-вычислителя (или агентов) для следующего этапа алгоритма $\boldsymbol{c_1: F_{XML} \rightarrow F'_{XML}}$. В зависимости от задачи на этом этапе возможен выбор из нескольких типов агентов, в том случае, если в алгоритме возможны несколько вариантов развития событий $\boldsymbol{c_1: L(g) \rightarrow c_i}$. Когда агент-вычислитель для следующего этапа определен, для него формируется пусковая функция, а на вход подаются события, которые запускают процесс работы агентов $\boldsymbol{c_1: c_1^{\Psi} \rightarrow c_1^{\Phi} \rightarrow \Phi_{c1} \rightarrow \Psi_{ci}}$.
6. 4-й этап повторяется до тех пор, пока все задачи из списка G не будут решены и процесс не подойдет к возврату данных агентом-вычислителем агенту-менеджеру $\boldsymbol{c_n: c_n^{\Psi} \rightarrow c_n^{\Phi} \rightarrow \Phi_{cn} \rightarrow \Psi'_M}$.

7. Повторное формирование пусковой функции Ψ'_M у агента-менеджера сигнализирует о том, что все этапы сценария были выполнены и последний агент-вычислитель из цепочки вызовов вернул данные после проведения всех расчетов $\boldsymbol{M:\Psi'_M \rightarrow M^{\Psi}}$, $\boldsymbol{M:\ F'_{XML} \rightarrow F'_D}$. Данные возвращаются к агенту-пользователю.

Это общий метод взаимодействия агентов, не зависящий от конкретной конфигурации и работающий для каждого из возможных сценариев.

2.3.4 Событийные модели агентных сценариев на основе Joiner-сетей

Основные Joiner-элементы описания агентных сценариев приведены в табл. 1.

Таблица 1. Joiner-элементы

	Процесс работы агента.
	Событие, которое генерирует агент после окончания своей работы.
	Пользователь, взаимодействующий с программой.
	Направление развития событий.

Использование подобных диаграмм позволит экспертам определить последовательность вызова агентов, а также наглядно описать весь алгоритм решения задачи. Комбинацией из описанных выше элементов можно создать несколько базовых шаблонов для различных случаев [67, 68]. Когда один агент начинает выполнять свою работу сразу после другого агента, то процесс идет последовательно, графическое представление показано на рис. 2.14.

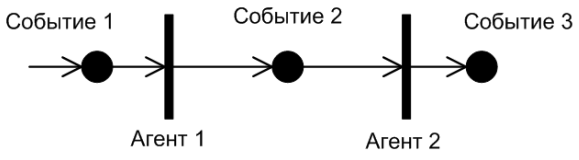


Рис. 2.14. Последовательное развитие событий.

Если после окончания работы агента существуют два варианта дальнейших действий, то получается, что он может генерировать два различных события. Каждое сгенерированное событие ведет к собственному следующему агенту. Графически подобный случай изображен на рис. 2.15.

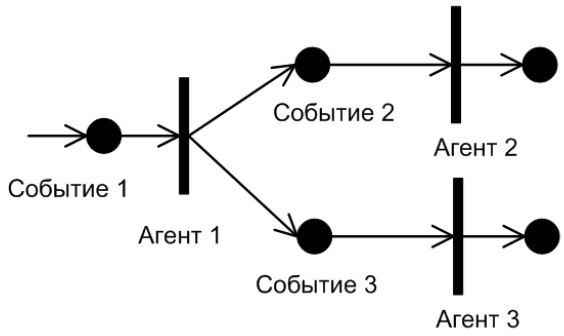


Рис. 2.15. Выбор одной ветки событий из двух.

Использование агентов подразумевает такие этапы в алгоритме, в которых расчеты должно проходить параллельно, в противном случае целесообразность использования агентного подхода крайне мала. Агент, который закончил выполнять свою часть алгоритма, генерирует одновременно несколько событий, равное количеству агентов, которые будут продолжать вычисления. Это могут быть как агенты одного типа (выполняющие одну и ту же задачу, но с разными массивами данных), так и агенты разных типов (каждый выполняет уникальную задачу). После этого вычисления могут проходить параллельно, последний вариант изображен на рис. 2.16.

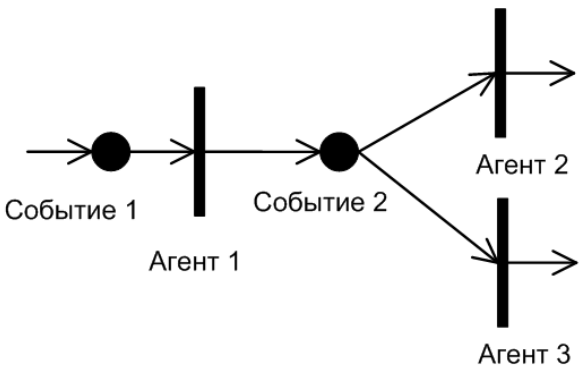


Рис. 2.16. Параллельное развитие двух событий.

Выполнение алгоритма начинается с того, что пользователь передает системе исходные данные и запускает процесс. Окончание работы алгоритма также связано с взаимодействием с пользователем. Система должна предоставить результаты своих вычислений, либо отчет о возникших в ходе работы проблемах и ошибках. Пользователь, в данной нотации, идентичен по свойствам агенту, поэтому какое-то выходное событие агента может быть входным для пользователя и точно так же пользователь может генерировать события, которые являлись бы входными для агентов. Графически взаимодействия с пользователем показаны на рис. 2.17.

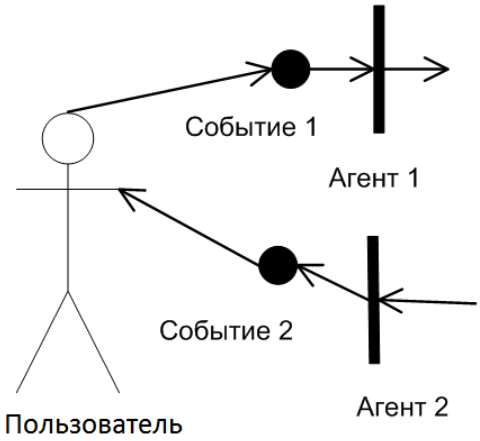


Рис. 2.17. Взаимодействие с пользователем.

Список описанных выше шаблонов не является окончательным, эксперт может создавать собственные, не нарушая при этом правила построения Joiner-сетей. Используя данные шаблоны, построим Joiner-сеть агентного сценария работы системы оценивания состояний ЭЭС. Графически представить взаимодействие агентов в нотации Joiner-сетей можно следующим образом (рис. 2.18).

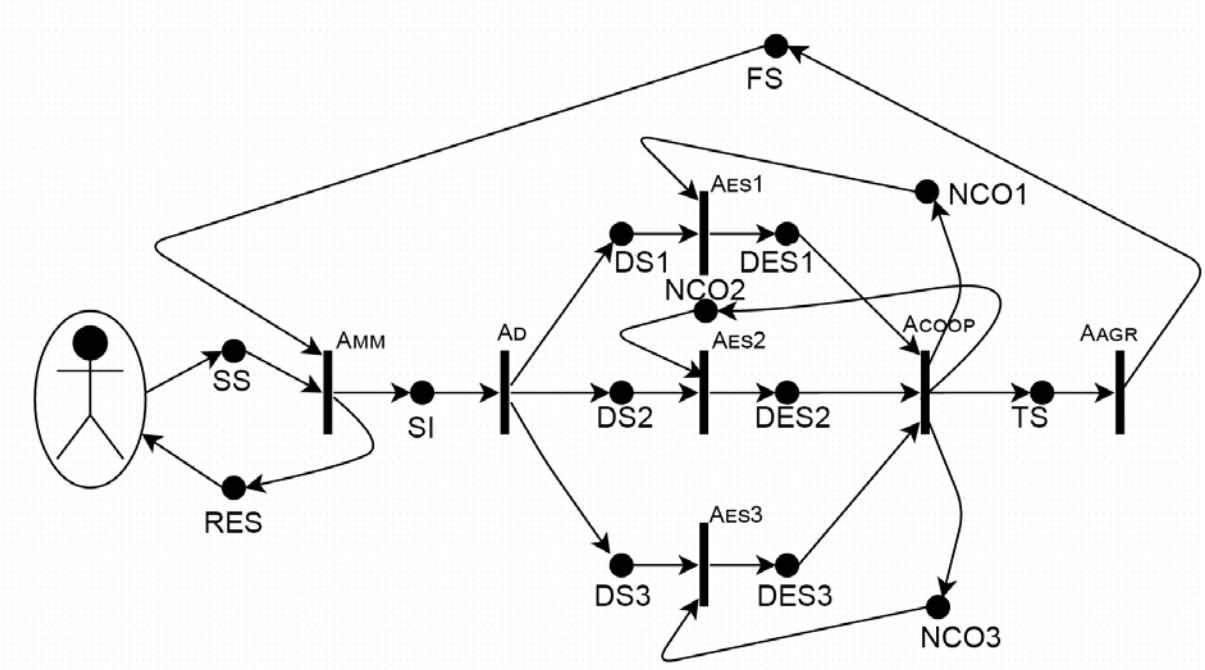


Рис. 2.18. Пример событийной модели агентного сценария.

Процессы, показанные на рис. 2.18:

A_{MM} – процесс работы главного модуля.

A_D – процесс работы агента декомпозиции схемы.

A_{ES1}, A_{ES2}, A_{ES3} – процесс работы агентов оценивания состояния.

A_{COOP} – процесс работы агента кооперации.

A_{AGR} – процесс работы агента агрегирования.

События:

SS – пользователь отправил исходную схему.

SI – передача исходных данных.

DS1, DS2, DS3 – схема разделена на 3 подсистемы.

DES1, DES2, DES3 – оценивание состояния подсистем завершено.

NCO1, NCO2, NCO3 – согласование между схемами не достигнуто.

TS – согласование достигнуто.

FS – исходная схема собрана из разделенных подсистем.

RES – результаты отправлены пользователю.

Изображенную на рис. 2.18 схему можно представить в виде списка пусковых и флаговых функций. Эти функции описывают события, которые инициируют наступление процесса или свидетельствуют о его завершении.

Пусковые функции:
$\Psi mm = (SS \vee FS) \cdot \overline{SI}$
$\Psi d = SI \cdot \overline{DS1 \cdot DS2 \cdot DS3}$
$\Psi es_i = (DS_i \vee NCO_i) \cdot \overline{DES_i}$
$\Psi coop = (DES1 \cdot DES2 \cdot DES3) \cdot \overline{NCO1 \cdot NCO2 \cdot NCO3 \cdot TS}$
$\Psi agr = TS \cdot \overline{FS}$

Флаговые функции:
Фmm: SI := 1, SS:=0
Фd: SI := 0, DS1:=1, DS2:=1, DS3:=1
Фes$_i$: DS$_i$:= 0, DES$_i$:=1
Фcoop: DES1:=0, DES2:=0, DES3:=0, NCO1:=1, NCO2:=1, NCO3:=1, TS:=1
Фagr: TS := 0, FS:=1

Здесь пусковая функция $\Psi d = SI \cdot \overline{DS1 \cdot DS2 \cdot DS3}$ соответствует условию запуска агента декомпозиции исходных данных о текущем режиме ЭЭС ─ при реализации события «Передача исходных данных», а также отсутствия события, сигнализирующего, что данный процесс уже запущен. При выполнении этого условия агент запускается и после выполнения своей работы генерирует соответствующие события: «Схема разделена на 3 подсистемы», что описывается флаговой функцией Фd: SI := 0, DS1 := 1, DS2 := 1, DS3 := 1. Остальные пусковые и флаговые функции интерпретируются аналогичным образом.

Для иллюстрации метода применим его на примере задачи оценивания состояния ЭЭС и модели, изображенной на рисунке 2.19:

1. Ввод исходных данных пользователем. Инициация события передачи исходных данных (**SS**).
2. **Формирование пусковой функции для главного модуля$A_{ММ}$. $\Psi mm = (SSVFS) \cdot \overline{SI}$**. Дальнейшие действия происходят в зависимости от текущего состояния системы (флаговых функций). При срабатывании события на получение исходных данных (***SS***), либо при получении рассчитанной схемы ЭЭС (***FS***).

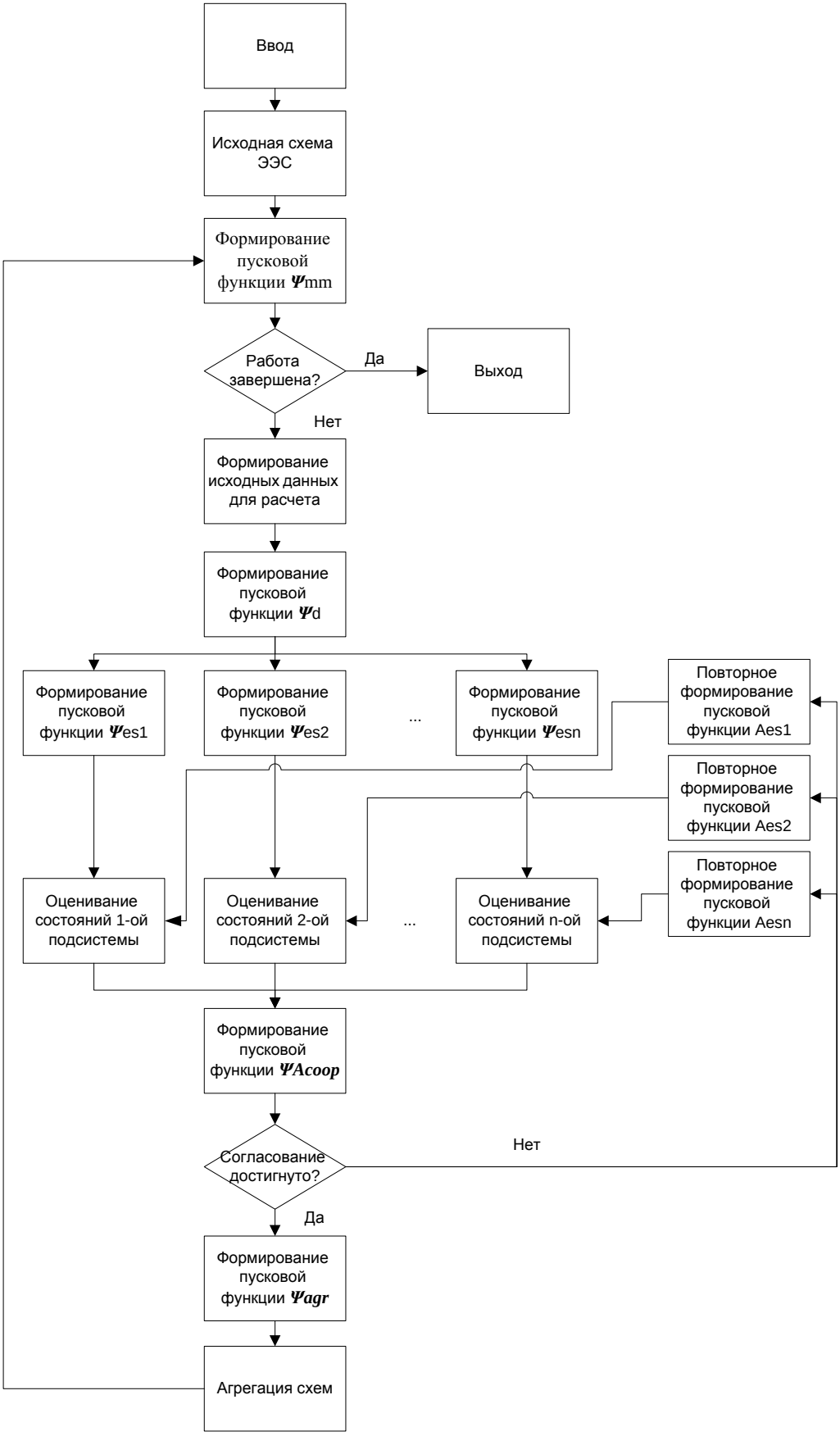


Рис. 2.19. Схема реализации численного метода взаимодействия агентов.

В случае срабатывания события ***FS***, рассчитанные данные возвращаются пользователю, инициировавшему расчет. Если срабатывает событие ***SS***, то главный модуль формирует из исходных данных готовый к дальнейшему расчету XMLфайл и инициирует событие по отправке этого файла (***SI***), агенту, стоящему первым в текущем агентном сценарии (A_D).

3. **Формирование пусковой функции агента декомпозиции A_D.** $\Psi d = SI \cdot \overline{DS1 \cdot DS2 \cdot \ldots DSn}$. Агент декомпозиции начинает работу в тот момент когда инициируется событие по передаче исходной схемы ЭЭС. Получив эти данные,агент производит разбивку схемы на несколько подсистем и инициирует события по пересылке полученных подсистемы дальнейшим агентам (***DS1, DS2,…DSn***). Генерируется n событий, где ***n*** это модуль разности количества активных агентов следующего этапа (***aes***) и количества полученных подсистем после декомпозиции (***ds***): $n=|aes-ds|$.
4. **Формирование пусковых функций агентов оценивания состоянияA_{ESi}.** Количество задействованных в этом этапе агентов зависит от количества агентов данного типа в системе и количества подсистем, на которые была разбита исходная схема. Для каждого из участвующих агентов генерируется пусковая функция следующего вида: $\Psi esi = (DSi \ V \ NCOi) \cdot \overline{DESi}$, где ***i*** – номер агента оценивания состояний. Данный агент может быть вызван при срабатывании двух событий, при получении подсистемы после декомпозиции (***DSi***)и при получении скорректированной подсистемы после процесса координации (***NCOi***). После проведения процесса оценивания состояния i-ый агент генерирует событие по передаче оцененной подсистемы следующему агенту (***DESi***).
5. **Формирование пусковой функции агента координацииA_{coop}.** $\Psi coop = (DES1 \cdot DES2 \cdot \ldots DESn) \cdot \overline{NCO1 \cdot NCO2 \cdot \ldots NCOn \cdot TS}$. Агент координации начинает свою работу после получения всех оцененных подсистем ***(DES1, DES2, …, DESn).*** После сравнения значений в граничных узлах могут возникнуть два возможных исхода. В первом случае значения в узлах

сходятся (с учетом погрешности), тогда формируется события по передаче подсистем следующему агенту в сценарии (***TS***). Во втором случае значения разнятся, тогда в них вносятся корректировки и подсистемы возвращаются на этап 4, для повторного проведения оценивания состояния ***(NCO1, NCO2,…, NCOn).***

6. **Формирование пусковой функции агента агрегированияA_{agr}.** ***Ψagr = TS · $\overline{FS}$.*** После получения подсистем прошедших процесс координации (***TS***) начинает работы агент агрегирования. Он производит слияние подсистем вновь в единую схему и возвращает их обратно главному модулю(***FS***), что означает переход на 2 этап алгоритма.

В случае полного согласования значений в граничных узлах на первой итерации, процесс работы агентов можно представить следующим образом:

=> SS:=1 =>Ψmm = (SS V FS) ·$\overline{SI}$=> SI:=1=>

=>Ψd = SI·$\overline{DS1 \cdot DS2 \cdot \ldots DSn}$ => DS1:=1, DS2:= 1, …,DSn:=1 =>

=>Ψesi = (DSi V NCOi) ·$\overline{DESi}$ =>DESi :=1 =>

=>Ψcoop = (DES1 · DES2 ·… DESn) · $\overline{NCO1 \cdot NCO2 \cdot \ldots NCOn}$·· $\overline{TS}$ => TS:=1 =>

=>Ψagr = TS ·$\overline{FS}$ => FS:=1 =>

=>Ψmm = (SS V FS) ·$\overline{SI}$=> FS:=1 =>

Вопрос о корректности вышеописанной модели зависит от описания действий пользователя, так как простейший анализ этой сети на причинно-следственную и логическую связанность выявляет зависимость этих свойств модели от событий, генерируемых пользователем.

2.3.5 Алгоритм взаимодействия агентов

Алгоритм решения задачи может включать в себя несколько этапов. В таком случае возможна ситуация, когда для выполнения различных этапов алгоритма используются специальные библиотеки, написанные на разных программных платформах. Таким образом, существует необходимость

разработки алгоритма взаимодействия агентов, работающих на разных программных платформах.

Конкретный агент не изолирован от других агентов. Предлагается осуществлять обмен информацией между отдельными агентами системы посредством передачи XML-файлов по сети. Для передачи данных по локальной сети распространен стек протоколов TCP/IP. Большинство программных платформ могут использовать TCP/IP для передачи данных по сети. Таким образом, при помощи TCP/IP мы можем обмениваться информацией между программами, написанными на разных программных платформах. То же самое касается и агентов. В том случае, если какую-то задачу проще реализовать и рассчитывать на конкретной платформе, можно использовать агента, работающего именно под ней – он точно так же сможет взаимодействовать с другими агентами. Алгоритм межплатформенного взаимодействия агентов показан на рис. 2.20.

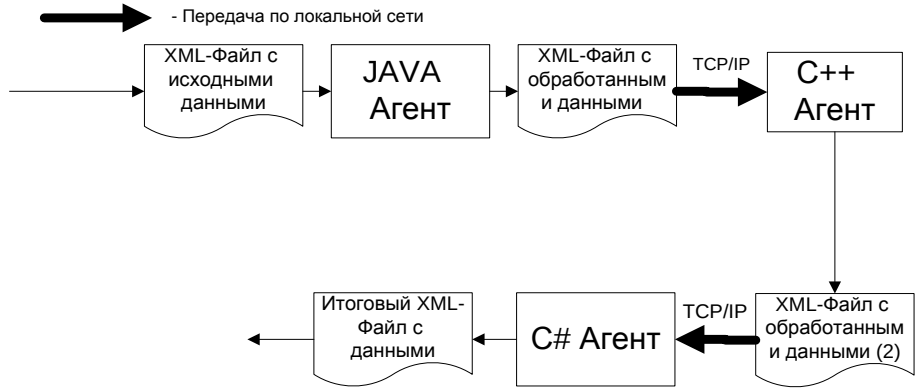


Рис. 2.20. Алгоритм межплатформенного взаимодействия агентов по локальной сети.

XML-файлы, которые используются в многоагентной системе, состоят из двух частей, их структура показана на рис. 2.21. В первой части находится информация о местонахождении агента в локальной сети, чтобы агент-получатель знал, по какому адресу ему необходимо отправлять ответ. В этой части указываются сетевые координаты агента (IP-адрес и порт), тип сообщения (запрос, результат и подтверждение), тип агента (зависит от тех

задач, которые он может выполнять) и его состояние (свободен или занят). Также этот список может быть расширен при необходимости.

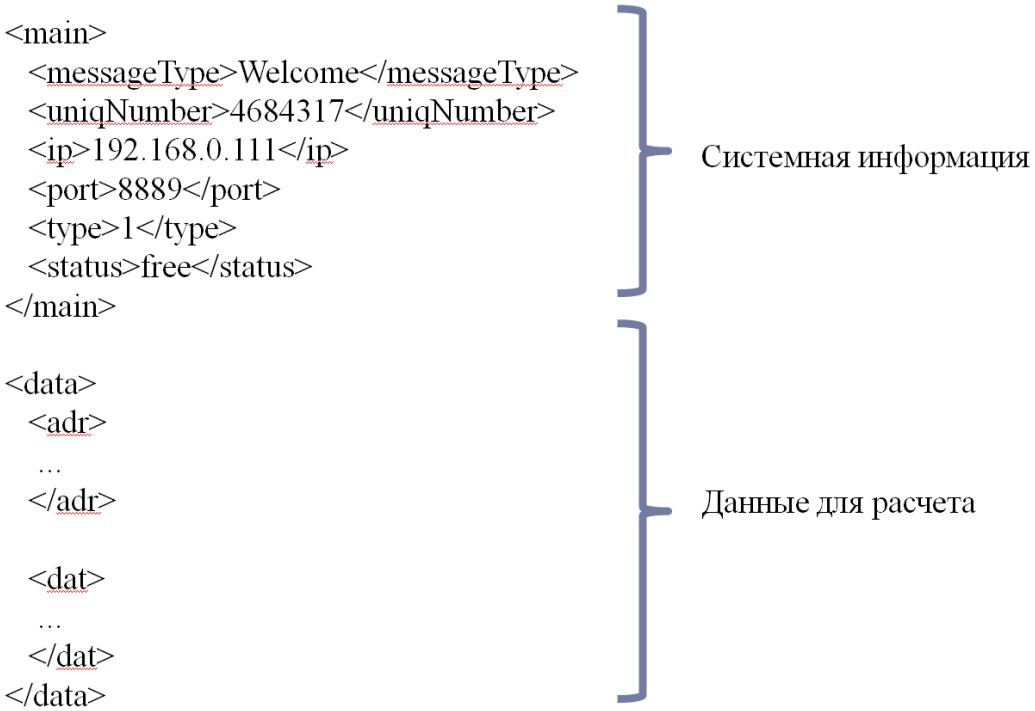


Рис. 2.21. Структура XML-файлов.

Во второй части файла хранятся данные. Это могут быть как данные для расчетов, так и данные для кооперации. XML-файлы делятся на два типа: расчетные и системные. В расчетных расположена информация, необходимая для проведения вычислений. Системные XML-файлы нужны для координации и кооперации, чтобы агенты могли договариваться, распределять задачи между собой и принимать коллективные решения [69, 70].

2.3.6 Алгоритм поведения агентов

Поскольку нам необходимо решать комплексные вычислительные задачи, то в такой ситуации агенты должны действовать сообща, чтобы выполнить задачу максимально качественно и в наиболее короткие сроки. Также во время функционирования системы в нее могут входить новые агенты, а другие – прекращать работу. Агенту необходимо получать информацию о других агентах, находящихся в системе, чтобы знать, с кем

можно взаимодействовать. Таким образом, необходим агент, который контролировал бы работу всей системы.

Для этого вся информация, касающаяся существующих в системе агентов, хранится централизованно в главном модуле. Он составляет и редактирует список активных агентов, следит за ходом выполнения работы, а так же хранит исходные данные для расчетов, промежуточные и итоговые результаты работы агентов.

Об изменениях в своем состоянии агент должен известить главный модуль, чтобы тот обладал актуальной информацией о состоянии системы. В том случае, если агент выполняет работу дольше, чем планировалось, то главный модуль отправляет ему запрос, чтобы агент передал ему отчет о ходе выполнения работы. В случае, если ответ от агента не приходит (агент «завис», отключился и т.д.), то главный модуль удаляет его из списка агентов и перепоручает его работу другому свободному в данный момент агенту, или ждет, пока таковой освободится.

На рис. 2.22 представлен общий алгоритм поведения агентов. После того, как агент включился и готов к работе, он направляет информацию о себе главному модулю, для внесения его в список активных агентов, чтобы в дальнейшем к нему могли обращаться другие агенты. Далее агент ждет поступления новой задачи и после этого начинает выполнять вычисления. После окончания вычислений агент информирует об этом главный модуль. Когда агент выполняет свою часть алгоритма, он должен передать данные другому агенту, который продолжит вычисления. До начала передачи этого агента необходимо найти. Его поисками будет заниматься тот агент, который только что закончил свою часть алгоритма. Он запрашивает у главного модуля список свободных на данный момент агентов и отправляет каждому запрос. Как только агентом получен ответ, то эта информация передается главному модулю, а данные для расчета отсылаются найденному агенту. Выход из алгоритма осуществляется после завершения вычислений.

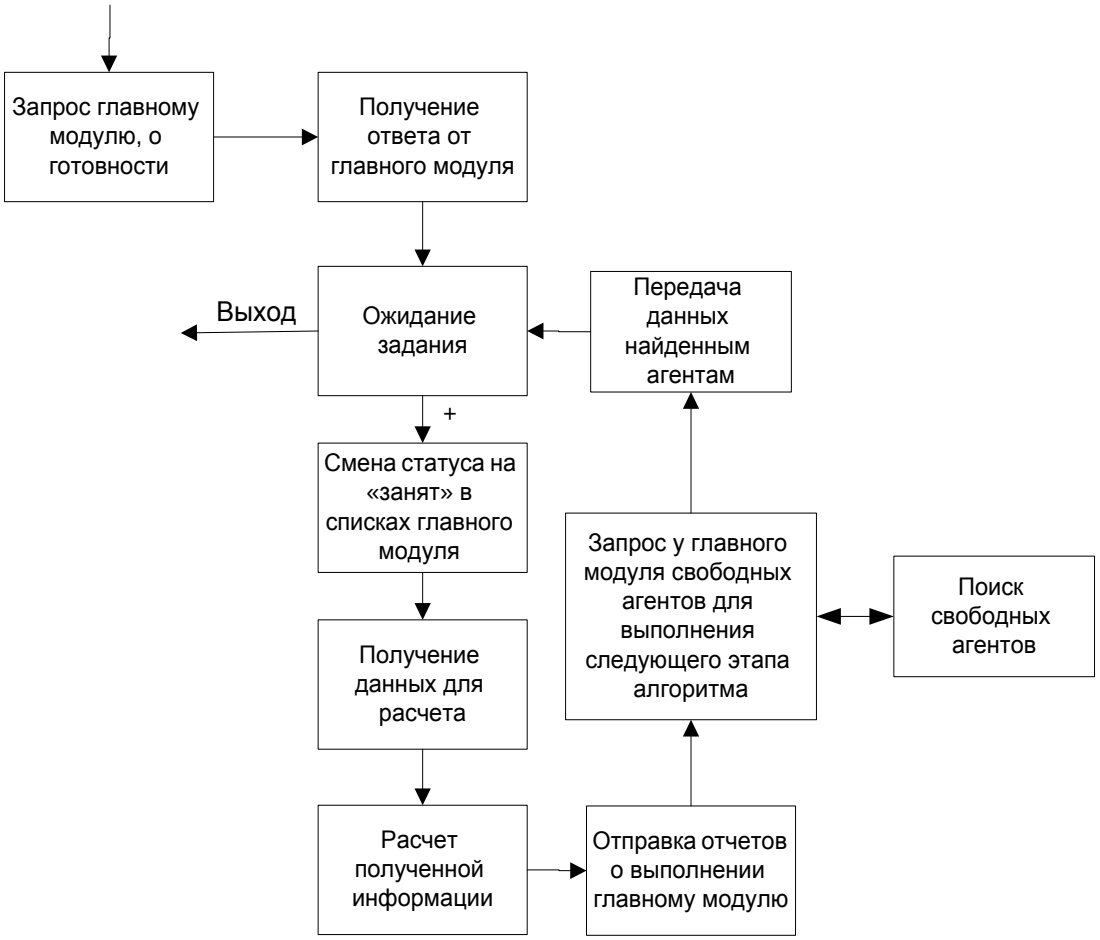
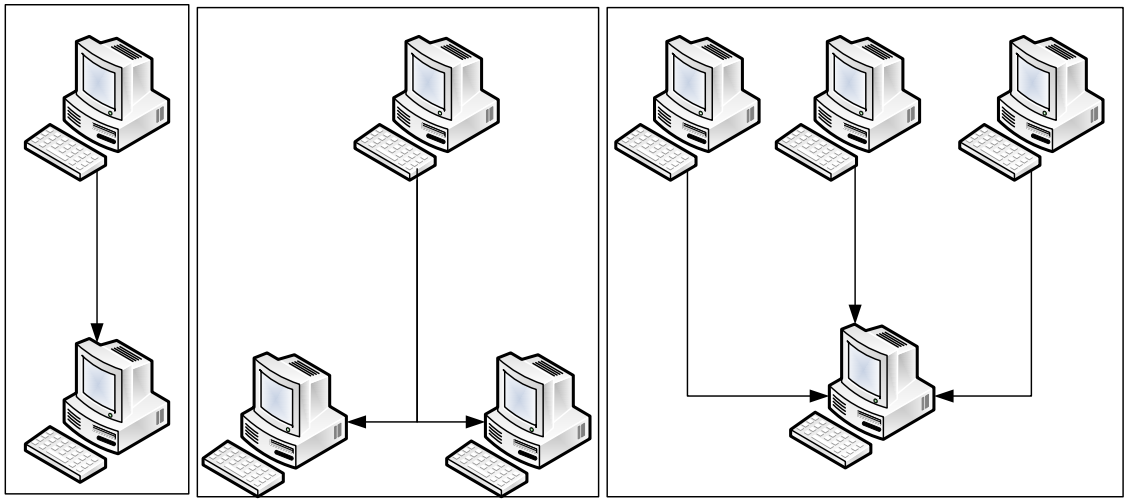


Рис. 2.22. Общий алгоритм поведения агентов.

2.3.7 Распределенное решение задач

Одно из основных преимуществ многоагентных систем — распределенная обработка данных. Нецелесообразным является использование многоагентных систем для реализации линейных алгоритмов решения задач, поскольку затраты времени на пересылку данных по сети будут лишь тормозить процесс расчета задач. Однако в случае, когда алгоритм позволяет проводить какие-либо вычисления параллельно, распределенная обработка может значительно сократить время расчетов. Поэтому при поиске агентов могут возникнуть сложные ситуации, требующие кооперации между агентами [69]. Основные возможные ситуации, которые могут возникнуть при поиске агентов, указаны на рисунке 2.23.



а) б) в)

Рис. 2.23. Ситуации, которые могут возникнуть при поиске агентов.

В первом случае (а) агенту необходимо найти другого агента.

$A_i \rightarrow A_j$, где A_i – текущий агент, A_j– следующий агент.

В данной ситуации никаких сложностей нет. Первый же ответивший на запрос агент становится искомым и приступает к выполнению задания.

Вторая ситуация (б) ненамного отличается от первой. Здесь один агент должен найти двух или более агентов, чтобы те продолжили решать задачу, выполняя параллельные вычисления.

$A_i \rightarrow A_{j,} A_k$, где A_i – текущий агент, $A_{j,} A_k$ – следующие агенты.

Важно обработать ситуацию, когда из требуемого количества агентов свободны только несколько. В таком случае следует дать задание свободным агентам, после этого запросить у главного модуля обновленный список и продолжить поиски до тех пор, пока не найдется свободный.

Самая проблематичная ситуация, когда нескольким агентам необходимо найти одного (в).

$A_i, A_{j,} A_k \rightarrow A_g$, где $A_i, A_{j,} A_k$ – текущие агенты, A_g – следующий агент.

Это может произойти, если распределенный этап алгоритма сменяется линейным этапом. В этом случае агентам необходимо взаимодействовать друг с другом, чтобы определить следующего агента. Каждый должен знать координаты остальных агентов, чтобы информировать друг друга о

результатах поиска. Все агенты начинают поиск в тот момент, когда закончат выполнять свою задачу. Тот агент, который первым нашел свободного агента, извещает остальных об этом и пересылает им адрес найденного агента, чтобы они знали, куда передавать данные. Может произойти так, что два агента одновременно обнаружили свободного. Поэтому перед тем, как передавать данные найденному агенту, агенты-поисковики должны достигнуть единого мнения в том, какого из найденных агентов они будут использовать. В зависимости от того, какой из агентов был найден раньше, тому и отдается предпочтение. Когда агенты-поисковики окончательно утвердили кандидатуру следующего агента, они передают ему данные для расчетов.

2.3.8 Системно-концептуальные соглашения

Программной платформой для разработки выбран «.Net Framework 4.5». Языком, используемым для разработки, выбран C#, поскольку он был создан специально под вышеописанную платформу и обладает наибольшей совместимостью со всеми ее функциями. Для разработки программного комплекса используется среда Microsoft Visual Studio 2013. Данная среда является официальным и наиболее полным средством разработки программ на языке C#. Для более простого и быстрого внесения изменений нужно взять за основу структуры ПК шаблон проектирования «MVC» (Model-view-controller, «Модель-представление-контроллер»). Для разделения модели и интерфейса решено использовать шаблон «наблюдатель».

2.3.9 Требования к будущей системе

Для реализации алгоритма распределенного оценивания состояния в состав технических средств должен входить многопроцессорный компьютер, выполняющий роль сервера, на котором будет осуществляться решение основных задач комплекса (деление расчетной схемы, формирование подсистем, агрегирование результатов). Для выполнения остальных задач

(оценивание состояния подсистем, решение координационных задач) в состав технических средств должны входить персональные компьютеры (2 и более), объединенные в локальную вычислительную сеть.

Агенты должны быть распределены по локальной сети для обеспечения децентрализованной обработки данных. Они должны уметь общаться между собой, обмениваться информацией и кооперироваться для выполнения задач. Эти задачи будет решать главный модуль – агент координации, отслеживающий всех находящихся в сети агентов, определяющий их функции и контролирующий их работу.

2.4 Разработка базовых компонентов для построения МАС

2.4.1 Базовые компоненты агентов

Каждый агент в системе, за исключением главного модуля, состоит из трех частей [70]:

- ***Передаваемый интерфейс.*** Поскольку пользователь взаимодействует только с главным модулем, то каждый агент не имеет своего собственного интерфейса, однако он должен каким-то образом получить параметры для проведения дальнейших вычислений. Для решения этой задачи агент делегирует свой интерфейс главному модулю, передает ему информацию о требуемых параметрах, их количество, тип данных и ограничения. На основании этой информации главный модуль генерирует интерфейс для пользователя. После того, как пользователь ввел все требуемые данные, главный модуль выполнит их проверку, в соответствии с предоставленными агентом ограничениями. Если все параметры введены корректно, то они передаются агенту вместе с данными для проведения расчетов.
- ***Модуль сетевого взаимодействия.*** Для достижения поставленных целей агентам необходимо обмениваться информацией. Это может

быть как системная информация (обеспечивающая функционирование всех объектов), так и информация, необходимая для вычислений. В системе обмен данными происходит при помощи стека протоколов TCP/IP и XML-документов с данными. Их совместное использование обеспечивает возможность взаимодействовать агентам, написанным на разных программных платформах. В каждом агенте присутствуют два сетевых компонента, один принимает сообщения из сети, другой отправляет собственные. Эти два процесса работают параллельно не только друг с другом, но и с вычислительным модулем, о котором сказано ниже. Когда агент получает сообщение из сети, он приступает к его обработке. Если в этот момент приходит новое сообщение, то оно помещается в стек. В стеке сообщение хранится до тех пор, пока агент не закончит обработку всей ранее пришедшей информации.

- ***Вычислительный модуль.*** Каждый агент в системе присутствует для того, чтобы решать определенную задачу в рамках общего алгоритма. Данный модуль отвечает за ту работу, которую агент выполняет в системе.

2.4.2 Базовые компоненты главного модуля

Для того, чтобы контролировать состояние всей системы, осуществлять мониторинг и следить за ходом решения задач, требуется специальный агент, который не будет выполнять ни одну из частей расчетного алгоритма. Эти функции выполняет главный модуль, являясь ядром всей многоагентной системы. Он так же, как и остальные компоненты, является агентом, однако обладает большим набором полномочий и функций:

- Формирование всей многоагентной подсистемы.
- Регистрация новых агентов в системе.
- Получение исходных данных от пользователя.
- Распределение задач по свободным агентам.

- Создание и редактирование агентных сценариев.
- Мониторинг работы агентов.

Так же, как и у остальных агентов, у него есть модуль для сетевого взаимодействия, с помощью него он раздает команды вычислительным агентам, а так же получает от них информацию о ходе решения задачи. В остальном главный модуль отличается от других агентов по наличию компонентов:

- ***Генератор пользовательского интерфейса.*** Как было сказано выше, каждый агент в системе передает главному модулю список параметров, которые необходимо указать. Чтобы пользователь имел возможность в удобной форме работать с этими данными, главному модулю необходимо на основании пришедшей от агента информации сгенерировать интерфейс.
- ***Список активных агентов.*** Количество агентов в системе в каждый момент времени может быть различным. Главный модуль должен знать, какие сейчас агенты в сети, по какому адресу они находятся, и какие задачи могут выполнять. Для этих целей он составляет список из активных агентов и постоянно его корректирует в случае появления новых агентов либо отключения старых.
- ***Редактор агентных сценариев.*** Агентные сценарии применяются для того, чтобы дать пользователям возможность самостоятельно менять алгоритм решения задач без привлечения программиста. Когда пользователю нужно провести расчеты, он может собственноручно определить порядок, в котором будут вызваны те или иные агенты. При необходимости, он сможет отредактировать существующий алгоритм, добавить новые этапы расчетов или убрать существующие, чтобы получить промежуточные результаты. Поэтому после включения агент должен передать главному модулю не только информацию о своем местонахождении в сети, но и о тех задачах, которые он умеет решать.

- ***Модуль мониторинга.*** Когда агент выключается, он должен известить об этом главный модуль, чтобы тот удалил его из списка. Однако существует вероятность, что какой-нибудь из агентов «зависнет». Чтобы отслеживать такие случаи, главный модуль с определенной периодичностью посылает всем агентам сообщение, на которое они должны ответить и сообщить свой статус. Если какой-то из агентов не отвечает на сообщения, то главный модуль исключает его из списка активных агентов и при необходимости перенаправляет его задачи другому агенту аналогичного типа.

2.5. Метод оценки надежности многоагентной системы

Для функционирования многоагентной системы необходимо, чтобы все ее составляющие надежно работали. Выход из строя одного или нескольких агентов может как повлиять на быстродействие системы, так и полностью вывести ее из строя. Решением данной проблемы может быть введение в систему резервных агентов, которые смогут заменить вышедших из строя. Подобное решение эффективно, но требует дополнительных мощностей, что может пагубно повлиять на быстродействие системы. Таким образом, необходимо выявить критически важных агентов, выход из строя которых приведет к максимальному ущербу, и принять меры по повышению надежности их работы.

Существующие методики обеспечения отказоустойчивости многоагентных систем (МАС) Brokered MAS [71], DARX [72], Meta-Agent [73] и работа [74] основаны на избыточности агентов или избыточности агентов и некоторых задач. Такие методики подтверждают возможность повышения уровня надежности многоагентных систем, то есть снижения количества отказов, только экспериментально [75]. Автором предлагается **численный метод оценки надежности МАС, основанный на теории графов,** позволяющий в краткие сроки определить критически важных для

функционирования МАС агентов, без необходимости проводить натурные эксперименты.

Для оценки надежности МАС разработана методика, реализующая предложенный метод:

1. Построение графа агентных взаимодействий.
2. Анализ структуры графа.
3. Использование методов описания графа (например, построение матрицы смежности для этого графа).
4. Определение степеней вершин на основе определения степеней захода и исхода. $P_i^+(a) = \sum_{j=1}^{m} q_{ij}$; $P_j^-(a) = \sum_{i=1}^{m} q_{ij}$; где m – количество вершин, $P_i^+(a)$ – полустепень захода (показатель входящих связей), $P_j^-(a)$ – полустепень исхода.
5. Выделение вершин с наибольшей степенью: $P(a) = P_i^+(a) + P_j^-(a)$.
6. Выявление критически важных для функционирования МАС агентов, на основе ранжирования степеней вершин графа.
7. Формулирование рекомендаций по повышению надежности этих агентов посредством резервирования агентов данного типа.

Применение предложенного метода будет проиллюстрировано для конкретной конфигурации МАС в разделе 3.

2.6 Выводы по главе 2

В рамках проделанной работы автором были получены следующие результаты:

- Предложен методический подход к созданию многоагентных систем в энергетике, на примере задачи оценивания состояния ЭЭС. Детализирован каждый этап, а также представлены графические схемы бизнес-процессов для каждого этапа.
- Предложен вычислительный метод управления взаимодействием агентов на основе алгебраических сетей. Построены событийные модели агентных

сценариев с использованием аппарата Joiner-сетей, позволяющие пользователям самостоятельно вносить изменения в алгоритм работы системы без участия программистов. Использование агентных сценариев обеспечивает гибкость при составлении пользователями алгоритма решения задачи.

- Для редактирования и создания новых алгоритмов в системе предложено использовать агентные сценарии, составленные пользователями. Приведены и детализированы все возможные шаблоны частей сценария.
- Описан предлагаемый автором алгоритм взаимодействия агентов через TCP/IP и состав XML-сообщений, используемых в системе. Это позволяет обмениваться информацией агентам, написанным на разных программных платформах. Описана структура XML-файлов и тех данных, которые в них хранятся.
- Разработан алгоритм поведения агентов при выполнении многоэтапных расчетов. Определены возможные ситуации, которые могут произойти при поиске агентов.
- Определены системно-концептуальные соглашения, принятые при разработке многоагентной системы.
- Выделены и описаны базовые компоненты агентов.
- Определены основные требования к многоагентной системе.
- Предложен метод оценки надежности многоагентных систем, с использованием теории графов.

3 Реализация, апробация и оценка надежности многоагентной системы оценивания состояний ЭЭС EstateMAS

3.1 Разработка МАС ОС ЭЭС «EstateMAS» на основе типовой МАС

Средства разработки и исполнения распределенных приложений, которыми, как правило, являются многоагентные системы, опирающиеся на статический подход (позволяют передавать только данные приложений) или динамический подход (обеспечивают возможности передачи исполняемого кода). Для решения данной задачи достаточным является использование статического подхода, поскольку все этапы алгоритма заранее определены и не требуют изменения во время функционирования многоагентной системы.

Система должна обеспечивать возможность выполнения следующих функций:

- Деление расчетной схемы ЭЭС на подсистемы;
- Оценивание состояния подсистем;
- Решение координационных задач между подсистемами;
- Агрегирование результатов расчетов.
- Обеспечение возможности работы агентов в локальной сети.

На основании типовой МАС (рис. 3.1) автором была выполнена реализация МАС «EstateMAS». Разработанная система EstateMAS состоит из 4-х вычислительных агентов и одного главного модуля. Она включает в себя: агент «Главный модуль», агент декомпозиции, агентов оценивания состояния, агент координации и агент агрегирования. Для реализации агента оценивания состояния используется вычислительное ядро, в основе которого лежит ПВК «Оценка-ПК», который сейчас относится к категории унаследованного ПО.

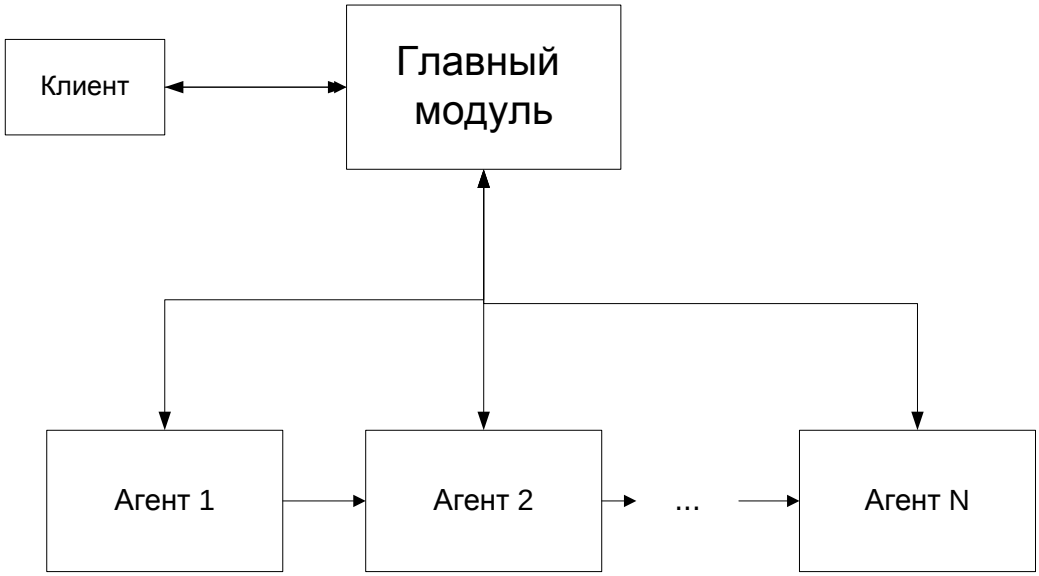


Рис. 3.1. Архитектура типовой МАС.

Для использования ПВК «Оценка-ПК» автором была разработана «оболочка» (middle-ware), позволяющая агенту использовать все функции данного программного комплекса, без необходимости проводить его реинжиниринг.

На рис. 3.2 приведена общая архитектура многоагентной системы для оценивания состояния ЭЭС [68].

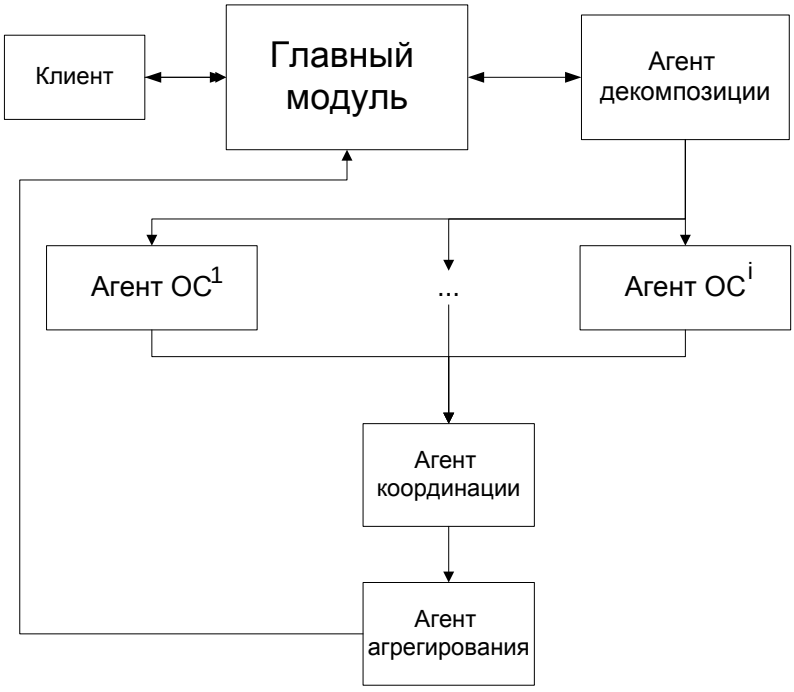


Рис. 3.2. Архитектура многоагентной системы оценивания состояния ЭЭС.

3.1.1 Агент декомпозиции

В функции данного агента входит декомпозиция расчетной схемы на подсистемы по граничным узлам или ветвям и уровням напряжения. В качестве исходных данных выступают данные о схеме замещения рассчитываемой сети и о составе имеющихся в ней измерений. Исходные данные для каждой ЭЭС хранятся в нескольких файлах, имеющих для каждой ЭЭС одинаковое уникальное имя и различные расширения:

1. dat – Данные по узлам и ветвям базовой схемы.
2. adr – данные о базовом составе телеизмерений.
3. psv – Данные о псевдоизмерениях.
4. con – Программные константы.

Алгоритм работы агента выглядит следующим образом:

1. Получение файлов с исходными данными от главного модуля.
2. Агент начинает процесс декомпозиции. Необходимо разбить узлы и ветви на группы по уровням напряжения. Однако пользователь может самостоятельно поменять граничные значения, в случае, если это необходимо.
3. После того как агент распределил узлы, он начинает проводить ту же операцию с ветвями. Ветви между узлами, находящимися в одной группе, заносятся в эту же группу. Граничные ветви, которые соединяют узлы, находящиеся в разных группах, заносятся в обе эти группы.
4. Агент проверяет все граничные связи и дополняет каждую группу граничными узлами из соседних групп.
5. Формируются 12 выходных файлов. Для каждой группы создаются свои файлы форматов dat, adr, psv и con.
6. Агент передает получившиеся файлы главному модулю, после чего переходит в режим ожидания новой задачи.

Модель данных в агенте представлена следующими классами: AdrItem222, AdrItem333, AdrItem444, DatItem200, DatItem300 и Psv. Классы AdrItem222, AdrItem333 и AdrItem444 предназначены для хранения строк из файла с базовым составом ТИ, с кодами строк 222, 333 и 444 соответственно. Классы DatItem200 и DatItem300 предназначены для хранения строк из файла с данными по узлам и ветвям базовой схемы. Объект класса DatItem200 предназначен для записи строк с индексами 102, 201, 202. Объект класса DatItem300 предназначен для записи строк с индексом 301. Все описанные выше классы являются расширением класса ItemDatAbs, в котором заданы обязательные атрибуты для всех классов, хранящих строки.

В объектах класса Psv хранится информация о псевдоизмерениях. Основным классом данного агента является controller. В нем содержится информация из всех исходных файлов, граничные значения для декомпозиции, а так же присутствуют атрибуты, которые после работы агента будут содержать подсистемы, полученные в результате разбиения исходных данных на группы.

На рис. 3.3 изображена диаграмма классов агента декомпозиции. На рисунке не присутствуют классы, реализующие сетевой компонент каждого агента, ввиду большого размера диаграммы. Эти элементы идентичны тем, что присутствуют в других агентах, поэтому они будут описаны ниже по тексту.

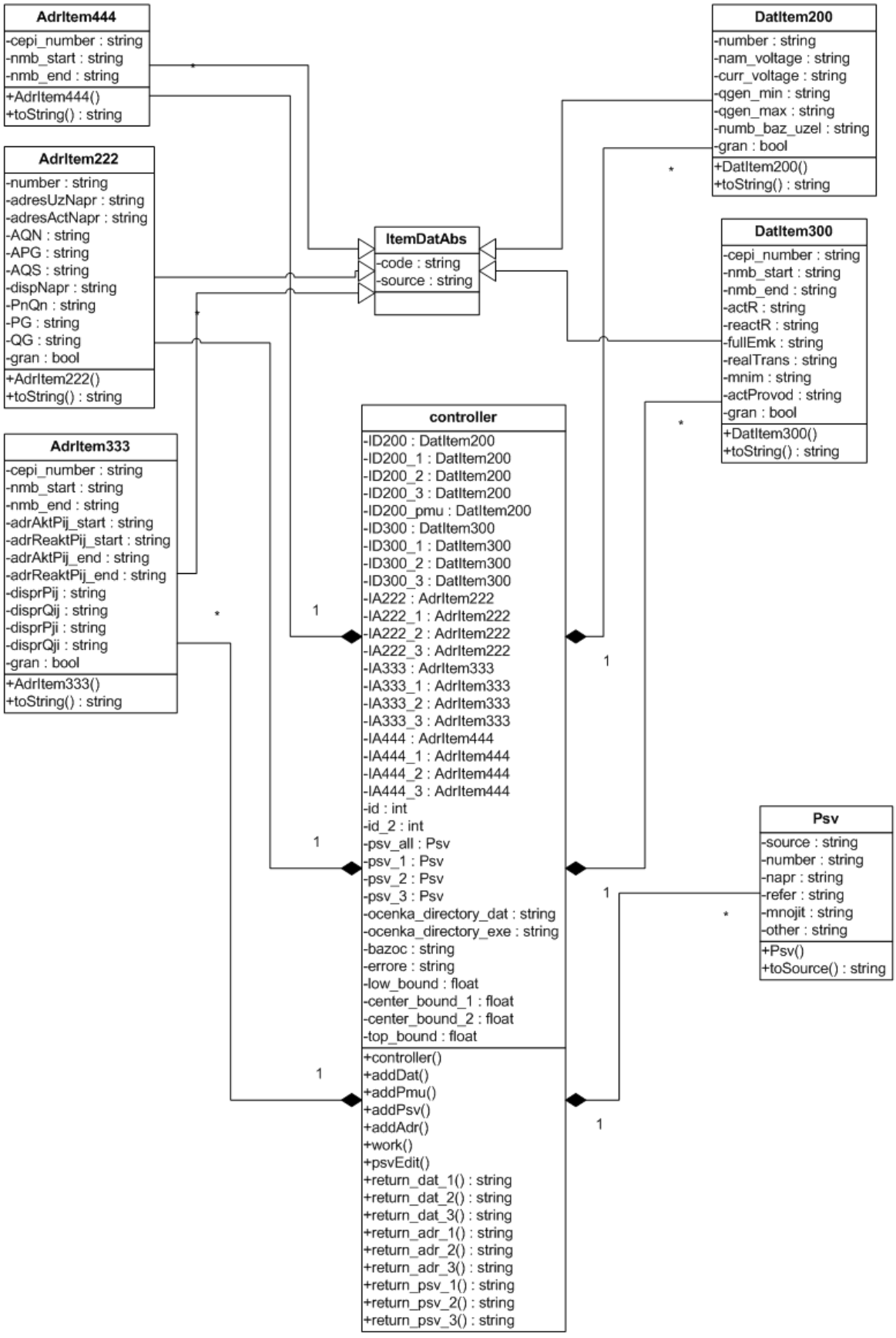


Рис. 3.3. Диаграмма классов агента декомпозиции.

3.1.2 Агент оценивания состояния

Задачей данного агента является выполнение процесса оценки состояний ЭЭС. Агент оценивания состояния был реализован в качестве «обертки» для ПВК «Оценка - ПК». Агент осуществляет все сетевые взаимодействия с главным модулем, получает исходные данные и подготавливает файлы для расчета. После того как исходные данные подготовлены, агент запускает ПВК, который выполняет расчет и затем возвращает все результаты главному модулю.

ПВК «Оценка - ПК» разработан в ИСЭМ СО РАН и предназначен для использования в оперативном диспетчерском управлении для получения расчетной модели текущего режима электроэнергетической системы по данным телеизмерений параметров режима и телесигналов о положении коммутационной аппаратуры (задача оценивания состояния) [76]. ПВК разработан в объектно-ориентированной среде Delphi, которая позволяет: осуществлять высокопроизводительный обмен данными с различными базами данных, организовать взаимодействие с системами сбора данных и средствами визуализации системы SCADA, создать платформонезависимую версию комплекса ОС, работоспособную не только в различных версиях операционной системы Windows, но и в некоторых клонах Unix (Linux, Aix). ПВК «Оценка - ПК» позволяет решать следующие технологические задачи:

1. Анализ ошибок в задании пассивных параметров и топологии сети.
2. Достоверизация обобщенных ТС по согласованию их показаний со значениями соответствующих ТИ.
3. Формирование контрольных уравнений по измеренным параметрам, достоверизация ТИ по анализу невязок КУ и их ранжировка на достоверные, сомнительные, плохие и непроверенные.
4. Уточнение состояния обобщенных ТС с помощью КУ.
5. Фильтрация случайных погрешностей измерений.
6. Дорасчет по полученным оценкам неизмеренных параметров режима.

7. Идентификация дисперсии ТИ и формирование архива признаков достоверности ТИ.
8. Сервисные функции, осуществляющие ввод и редактирование исходной информации из файлов ПВК или из базы данных, отображение результатов расчетов в табличной и графической форме.

На рис. 3.4 изображена диаграмма классов агента оценки состояний.

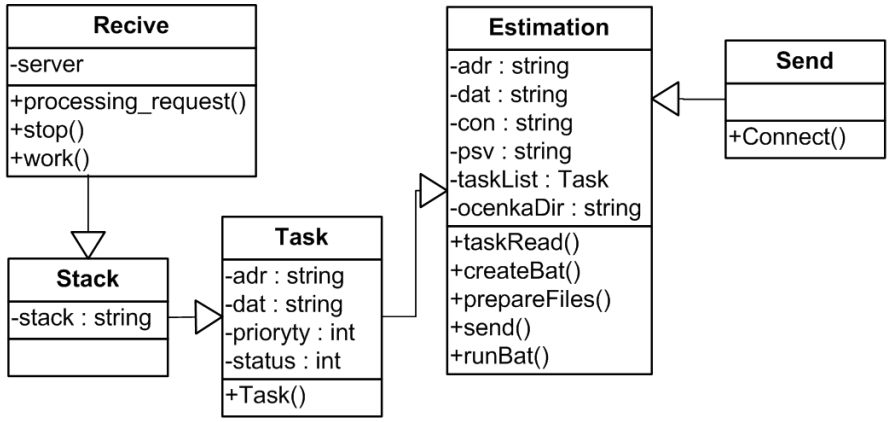


Рис. 3.4. Диаграмма классов агента оценки состояний.

Классы Recive и Send предназначены для сетевого взаимодействия с другими агентами. Они работают параллельно основному процессу. Как только приходит новое сообщение, приемник помещает новое сообщение в класс Stack, затем все сообщения из стека обрабатываются и на основании этого формируется новая задача, которая записывается в объект класса Task.

Далее расчетный компонент агента, представленный классом Estimation, записывает файлы с исходными данными в предназначенные для них папки ПВК «Оценка- ПК». Агентом создается файл с расширением .bat, в котором указан скрипт для запуска расчетов подготовленных файлов с исходными данными в ПВК «Оценка- ПК».

После завершения расчетов полученные файлы передаются следующему агенту через объект класса Send.

3.1.3 Агент координации

Основной задачей агента координации является согласование между собой подсистем, которые в дальнейшем будут объединены в одну. Агент проверяет значения оценок в граничных узлах, расположенных в разных схемах, но имеющих один номер. В случае, если оценки одинаковые (с учетом незначительной погрешности измерений), агент координации возвращает подсистемы назад главному модулю, либо передает их агенту агрегации. Если в оценках есть значительное различие, то в значения измерений вносятся корректировки и для этих подсистем вновь производится процесс оценки состояний. После этого весь процесс повторяется до тех пор, пока согласование не будет достигнуто.

Помимо этого, агент координации выполняет и другие функции:

- Получение из главного модуля данных всех подсистем первого уровня декомпозиции.
- Расчет активных и реактивных мощностей в граничных узлах либо перетоков мощности в граничных ветвях.
- Передача агенту агрегирования значений активных и реактивных мощностей граничных узлов или перетоков мощности граничных ветвей.
- Получение данных граничных узлов областей уровня напряжения из модулей области уровня напряжения.
- Расчет активных и реактивных мощностей во всех граничных узлах областей подсистемы.
- Передача рассчитанных значений в модуль агрегирования.

На рис. 3.5 изображена диаграмма классов агента координации.

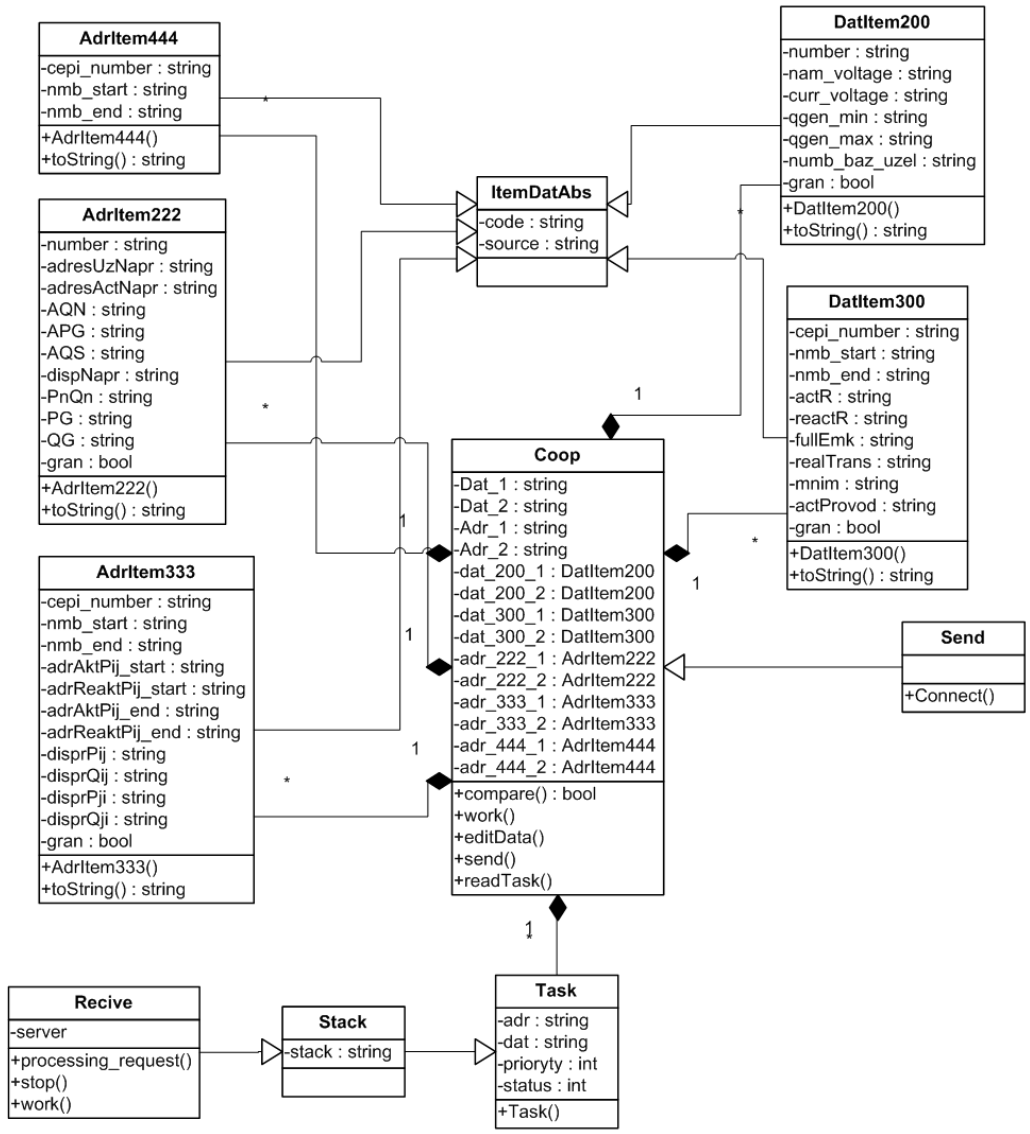


Рис. 3.5. Диаграмма классов агента координации.

Алгоритм сетевого взаимодействия с другими агентами идентичен тому, что описан в агенте оценки состояний, а модель данных такая же как и в агенте декомпозиции, поэтому их описание опускается.

Исходные данные этого агента – это две подсистемы, которые имеют одинаковые граничные узлы. Поскольку расчеты каждой подсистемы выполнялись отдельно друг от друга, то могут возникнуть проблемы при объединении схем. Этот агент устраняет возникшие расхождения в рассчитанных показателях и передает данные следующему агенту.

3.1.4 Агент агрегирования

Агент агрегирования выполняет заключительный этап в процессе оценки состояний. Основной и главной его задачей является объединение всех подсистем в исходную. Для этого необходимо удалить дублирующиеся записи и проверить правильность адресов узлов, которые могли быть изменены на этапе декомпозиции. В случае необходимости адреса корректируются. На выходе агент получает файлы, содержащие в себе исходную схему с удаленными плохими данными и рассчитанными параметрами. Функции агента:

- Получение от модуля координации значений активных и реактивных мощностей граничных узлов областей и активных и реактивных мощностей граничных узлов либо перетоков мощности граничных ветвей крупных подсистем.
- Агрегирование этих данных в выходные файлы.
- Передача выходных данных в главный модуль.

На рис. 3.6 изображена диаграмма классов агента агрегирования. Алгоритм сетевого взаимодействия с другими агентами идентичен тем, что описаны выше. В классе Agreg хранится информация о трех подсистемах, которые он объединяет вновь в единую схему.

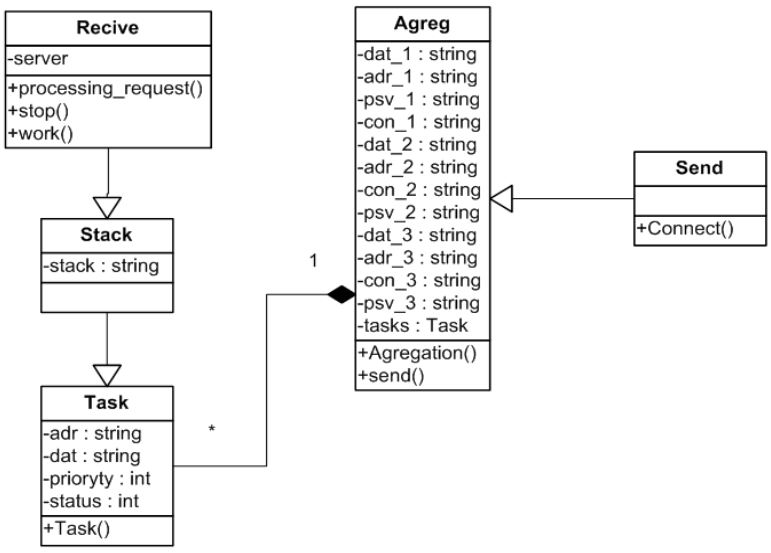


Рис. 3.6. Диаграмма классов агента агрегирования.

3.1.5. Усовершенствованная многоагентная технология ОС

Таким образом, мы можем усовершенствовать технологию оценивания состояния, разработав для каждого этапа собственные программные средства поддержки этого этапа. В будущем допускается возможность интеграции разработанной системы EstateMAS с другими интеллектуальными инструментальными средствами [77, 78]. Для перехода от этапа качественной оценки к количественной с использованием ПК, разработано, при участии автора, специальное программное средство, которое может быть использовано в дальнейших исследованиях, предполагающих развитие методов оценивания состояния ЭЭС [79].

Автором предложена усовершенствованная многоагентная технология численного решения задачи оценивания состояния (табл. 2), заключающаяся в использовании на каждом этапе собственных программных средств поддержки предлагаемой технологии [80].

Таблица 2. Усовершенствованная многоагентная технология ОС

Этап	**Программное средство**	**Результат этапа**
Получение задания от пользователя	Главный модуль	Исходные данные получены, запущен процесс решения задачи
Декомпозиция исходных данных	Агент декомпозиции	Расчетная схема разделена на *n* отдельных подсистем
Оценивание состояния	Агенты ОС	В полученных подсистемах проведено оценивание состояния
Проверка значений в граничных узлах	Агент координации	Проведено сравнение значений напряжения и фазового угла в граничных узлах в разных подсистемах, в случае, если согласование значений в граничных узлах не достигнуто, в данные внесены коррективы
Объединение данных	Агент агрегирования	Полученные ранее подсистемы вновь объединены в одну с результатами ОС

3.2 Апробация работы системы на реальных данных

Предложенный автором методический подход был применен для разработки многоагентной системы оценивания состояний ЭЭС. На данный момент разработан агент декомпозиции исходных данных, содержащих полученную от датчиков информацию о текущем режиме по уровням напряжения. Также были реализованы главный модуль и агенты оценивания состояния, кооперации и агрегирования. Для тестирования правильности работы агентов к ним был добавлен пользовательский интерфейс. Пример интерфейса агента декомпозиции изображен на рис. 3.7, в дальнейшем он был добавлен в главный модуль [81].

Основными исходными данными для работы МАС ОС являются данные о схеме замещения рассчитываемой сети и о составе имеющихся в ней измерений. Характеристики узлов и связей хранятся в файлах DAT и ADR. Эти данные готовятся в унифицированных форматах, разработанных в ЦДУ ЕЭС. Для работы программы необходимо задать следующие массивы:

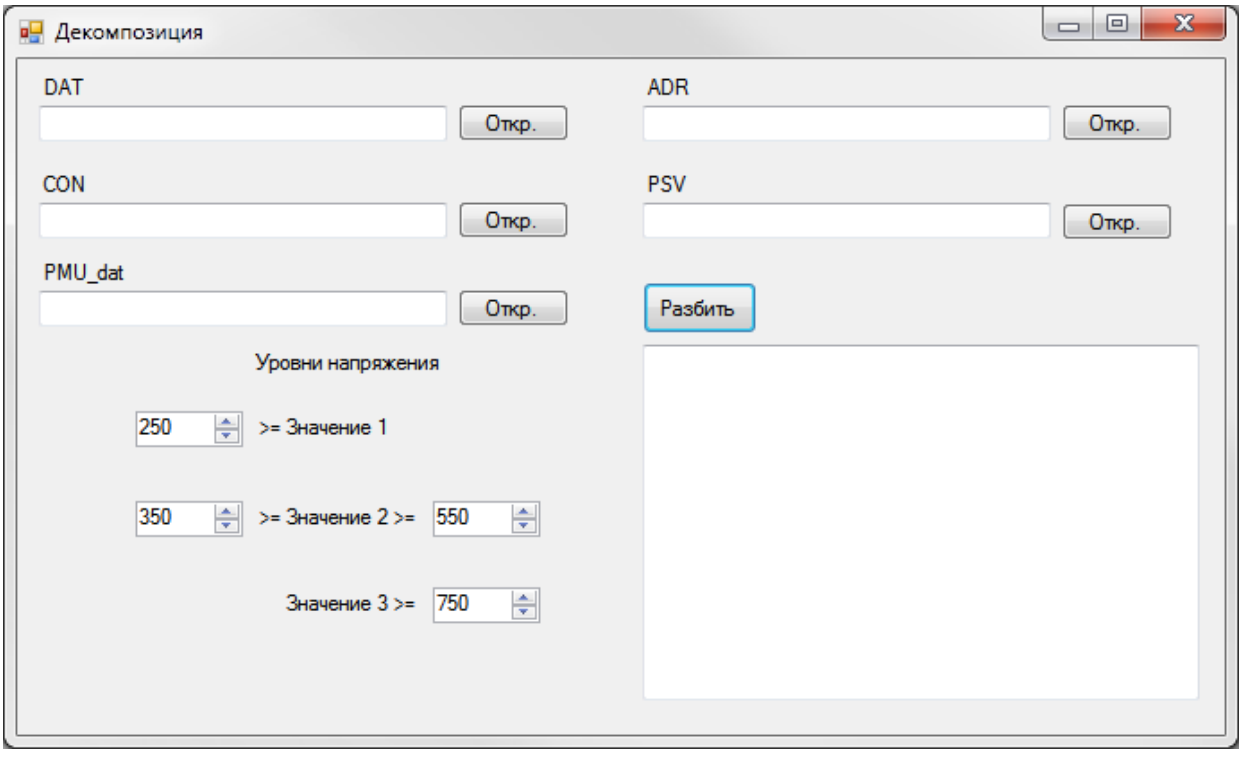


Рис. 3.7. Интерфейс агента декомпозиции

Массив 02 – информация по узлам. В строки с кодом 0201 заносится следующая информация (табл. 3):

Таблица 3. Информация по узлам

Позиция в строке	Параметр
9-16	номер узла
17-24	номинальное напряжение в КВ
57-64	текущее значение напряжения
65-72	минимальное значение Qген
73-80	максимальное значение Qген

В строке с кодом 0202 содержится информация о базисном узле. Информация о базисном узле может быть задана также в строке с кодом 0102. Если строки 0202 или 0102 в файле отсутствуют, то в качестве базисного выбирается первый по порядку узел.

Массив 03 – информация о ветвях. В строках с кодом 0301 должна содержаться следующая информация (табл. 4):

Таблица 4. Информация о ветвях

Позиция в строке	Параметр
5-6	параллельной цепи (0 или " " для одноцепных линий)
9-16	номер узла "начало связи"
17-24	номер узла "конец связи"
25-32	активное сопротивление ветви в Ом
33-40	реактивное сопротивление ветви в Ом
41-48	полная емкостная проводимость ветви в МКсим или реактивная проводимость шунта
49-56	действительная составляющая комплексного коэффициента трансформации
57-64	мнимая составляющая
65-72	активная проводимость ветви или узлового шунта в мкСм

Данные по составу ТИ, ТС и их адресам(файл *.adr).

Адреса измерений в узлах и их дисперсии хранятся в массиве 0222 в следующем порядке (табл. 5):

Таблица 5. Массив 0222

Позиция в строке	Параметр
9-16	номер узла
17-24	адрес ТИ узлового напряжения
25-32	адрес ТИ активной нагрузки в узле APN
33-40	адрес ТИ реактивной нагрузки в узле AQN
41-48	адрес ТИ активной генерации в узле APG
49-56	адрес ТИ реактивной генераци в узле AQG
57-64	дисперсия ТИ напряжения
65-72	дисперсия ТИ PN, QN
73-80	дисперсия ТИ PG
81-88	дисперсия ТИ QG

Адреса ТИ в связях и их дисперсии хранятся в массиве 0333 в следующем порядке (табл. 6):

Таблица 6. Массив 0333

Позиция в строке	Параметр
5-6	номер параллельной цепи (0 или " " для одноцепной линии)
9-16	номер узла i "начало связи"
17-24	номер узла j "конец связи"
25-32	адрес ТИ перетока активной мощности в начале связи Pij(если ТИ отсутствует, то " ")
33-40	адрес ТИ перетока реактивной мощности в начале связи Qij
41-48	адрес ТИ перетока активной мощности в конце связи Pji
49-56	адрес ТИ перетока реактивной мощности в конце связи Qji
57-64	дисперсия ТИ Pij
65-72	дисперсия ТИ Qij
73-80	дисперсия ТИ Pji
81-88	дисперсия ТИ Qji

Если в файле *.adr задать дисперсию ТИ равной 1, то ее значение будет определяться соответствующей данному виду ТИ групповой дисперсией из файла настроечных констант. Задача вычислительного эксперимента заключается в том, чтобы провести декомпозицию файлов схемы по уровням напряжения. Её выполняет агент декомпозиции. Его апробация была выполнена на примере схемы "Московского кольца" [82, 83]. Файлы ADR и DAT этой схемы представлены в приложении 1, графическое представление схемы изображено на рис. 3.8.

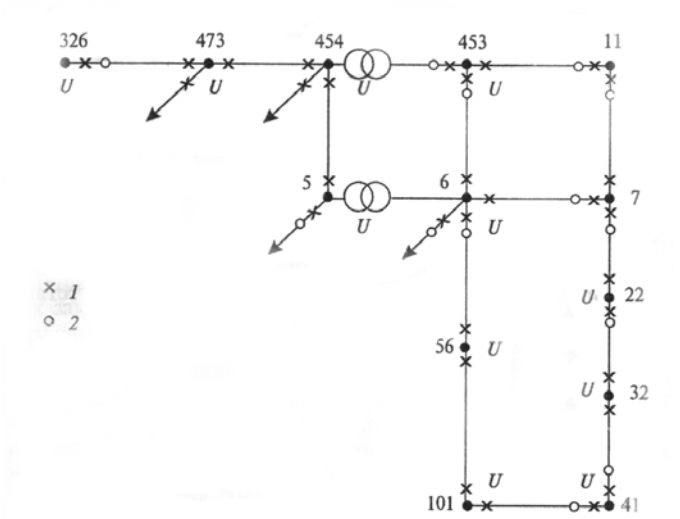


Рис. 3.8. Схема «Московское кольцо»

Была проведена декомпозиция схемы по уровням напряжения. Первая группа включала в себя узлы с напряжением от 750 кВ и выше, вторая группа состояла из узлов с напряжением от 500 кВ и ниже. В каждую группу также попадают граничные узлы и связи. Результаты декомпозиции представлены графически на рис 3.9.

Для сравнения выполнен расчет полной схемы и получившихся подсистем. Результаты расчетов в ПВК «Оценка-ПК» показаны в таблице 7 (выделены значения в граничных узлах), где U – напряжение (кВ), δ – фазовый угол (в градусах). Была решена задача координации значений в граничных узлах, после чего повторно проведено оценивание состояний с корректированными значениями, поэтому разница между показателями

минимальна, не превышает установленную в агенте координации погрешность.

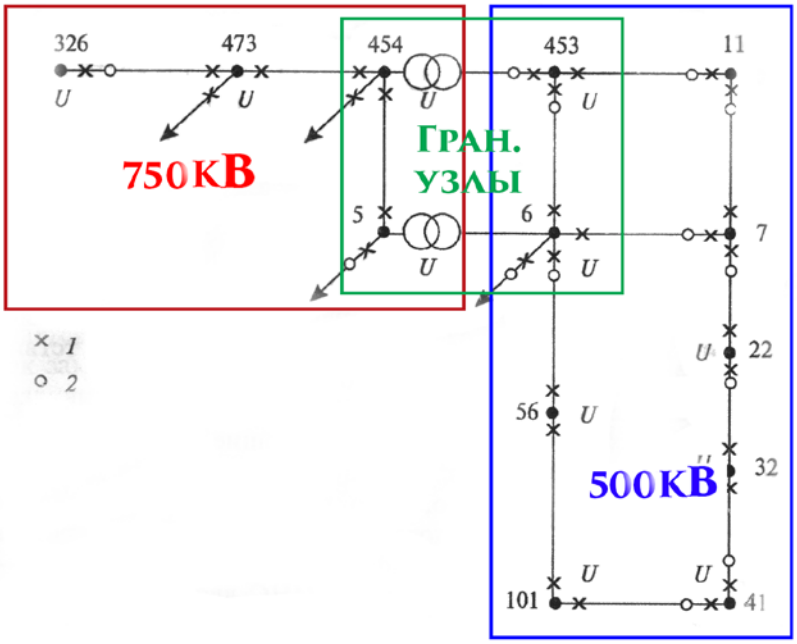


Рис. 3.9. Схема «Московское кольцо» после декомпозиции.

Таблица 7. Результаты расчетов.

№ узла	Измеренное напряжение кВ	Оцененные значения					
	Полная схема	Полная схема		Первая подсистема		Вторая подсистема	
		U	δ	U	δ	U	δ
5	**752,7**	**747**	**-5,85**	**746,63**	**-5,42**	**746,65**	**-5,44**
6	**509**	**517,5**	**-7,03**	**517,72**	**-7,71**	**517,8**	**-7,73**
7	500	502	-9,39	-	-	502,41	-8,77
11	500	505	-10,14	-	-	505,34	-9,24
22	500	498	-9,03	-	-	498,055	-9,79
32	500	497	-6,71	-	-	496,775	-6,47
41	500	512	-6,16	-	-	511,595	-6,08
56	500	515	-6	-	-	515,125	-5,36
101	500	515	-6,08	-	-	514,98	-5,84
326	750	741	-11,08	740,64	-10,29	-	-
453	**510,3**	**512**	**-7,06**	**512,045**	**-6,92**	**512,029**	**-6,92**
454	**740**	**740**	**-4,8**	**740,35**	**-4,36**	**740,35**	**-4,36**
473	750	753	-0,01	753,26	-0,04	-	-

3.3 Оценка надежности различных конфигураций МАС «EstateMAS»

Рассмотрим предлагаемую методику на примере описанного выше эксперимента [84]. Был построен граф агентных взаимодействий, в котором агенты сопоставлены с вершинами: Amm – главный модуль, Ad – агент декомпозиции, Aesi – агенты оценивания состояния, Acoop – агент координации, Aagr – агент агрегирования. Вершины соединяются дугами, причем дуга из Ai идет в Aj только в том случае, если агент Ai обращается к агенту Aj. Полученный граф представлен на рисунке 3.10.

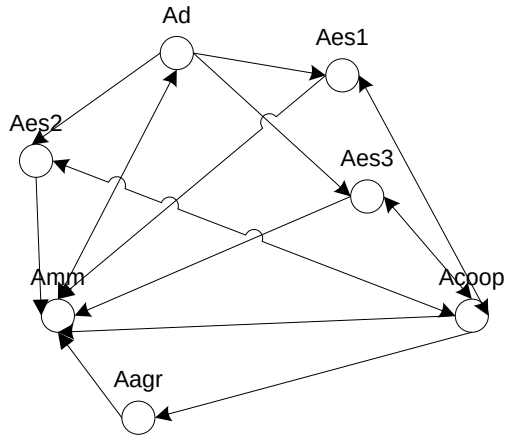


Рис. 3.10. Граф агентных взаимодействий.

Построим матрицу смежности для этого графа (табл. 8):

Таблица 8. Матрица смежности для графа

	Amm	Ad	Aes1	Aes2	Aes3	Acoop	Aagr	$P_j^-(a)$
Amm	0	1	0	0	0	0	0	**1**
Ad	1	0	1	1	1	0	0	**4**
Aes1	1	0	0	0	0	1	0	**2**
Aes2	1	0	0	0	0	1	0	**2**
Aes3	1	0	0	0	0	1	0	**2**
Acoop	1	0	1	1	1	0	1	**5**
Aagr	1	0	0	0	0	0	0	**1**
$P_i^+(a)$	**6**	**1**	**2**	**2**	**2**	**3**	**1**	

Пользуясь свойствами матриц смежности, можно найти степень вершин $P(a)= P_i^+(a) + P_j^-(a)$, где $P_i^+(a)$ – полустепень захода (показатель входящих связей), $P_j^-(a)$– полустепень исхода.

$P_i^+(a) = \sum_{j=1}^{m} q_{ij}$; $P_j^-(a) = \sum_{i=1}^{m} q_{ij}$; где м – количество вершин.

$$q_{ij} = \begin{cases} 1, \text{ если из вершины i идет дуга к вершине j;} \\ 0, \text{ если из вершины i нет дуги к вершине j;} \end{cases}$$

Степени вершин указаны в таблице 9:

Таблица 9. Степени вершин.

a	Amm	Ad	Aes1	Aes2	Aes3	Acoop	Aagr
P(a)	7	5	4	4	4	8	1

Отсюда следует, что наибольшую степень имеют (наиболее часто взаимодействуют с другими агентами) агенты: Acoop, Amm, Ad, что необходимо учесть при оценках надежности и отказоустойчивости системы, а так же при запуске резервных агентов.

Проведен также более обширный эксперимент, основанный на данных по системе Красноярскэнерго (рис. 3.12), состоящей из 111 узлов и 176 связей. Исходная схема была разделена на 2 подсистемы по 27 (включая 11 граничных узлов из другой подсистемы) и 95 узлов (включая 12 граничных узлов из другой подсистемы). Всего было выделено 23 граничных узла, в которые были установлены (программно) датчики PMU. Поскольку значения узловых напряжений (данные от PMU) в граничных узлах принимаются за точные, расчет идет относительно этого условия. Таким образом, мы можем исключить агента координации из алгоритма ОС. В итоге расчетные значения обеих подсистем и общей схемы получились идентичными, без необходимости вносить корректировки в значения в граничных узлах. Численные результаты данного эксперимента описаны в приложении 1, граничные узлы выделены жирным шрифтом.

Для этого примера так же были построены граф агентных взаимодействий (рис. 3.11), матрица смежности (табл. 10) и определены степени вершин (табл. 11). Было выявлено, что наибольшую степень имеют агенты: Amm и Ad.

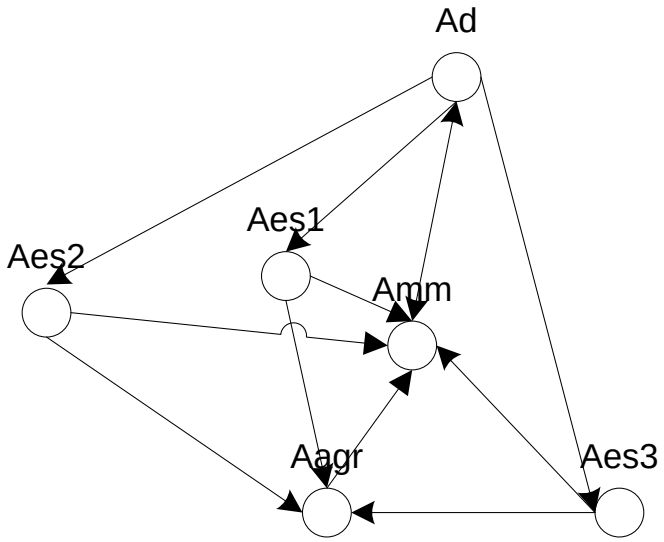


Рис. 3.11. Граф агентных взаимодействий.

Таблица 10. Матрица смежности для графа.

	Amm	Ad	Aes1	Aes2	Aes3	Aagr	$P_j^-(a)$
Amm	0	1	0	0	0	0	**1**
Ad	1	0	1	1	1	0	**4**
Aes1	1	0	0	0	0	1	**2**
Aes2	1	0	0	0	0	1	**2**
Aes3	1	0	0	0	0	1	**2**
Aagr	1	0	0	0	0	0	**1**
$P_i^+(a)$	**5**	**1**	**1**	**1**	**1**	**3**	

Таблица 11. Степени вершин.

a	Amm	Ad	Aes1	Aes2	Aes3	Aagr
P(a)	6	5	3	3	3	4

Следует заметить, что могут быть использованы и другие подходы к описанию и анализу графа. Например, в [85] рассматривается степень близости вершин. Это может быть учтено при дальнейшем развитии предлагаемого подхода.

После выявления наиболее критичных агентов могут быть предложены меры по повышению их надежности. На данный момент наиболее доступное решение – резервирование (дублирование) критичных агентов. В принципе, для повышения надежности работы агентов могут быть использованы и

другие рекомендации по повышению надежности программного обеспечения [86].

3.4. Применение результатов диссертационной работы в научных проектах, поддержанных грантами

Результаты диссертационной работы были применены в ряде научных проектов, в том числе поддержанных грантами РФФИ:

- Базовый проект ИСЭМ СО РАН, № гос.регистрации 01201361373 «IV.35.1.3. Методы, технологии и инструментальные средства интеллектуализации поддержки принятия решений в интегрированных интеллектуальных энергетических системах»
- Грант РФФИ №13-07-00140 (2013-2015) “Методология создания и интеграции интеллектуальных, агентных и облачных вычислений в Smart Grid (умных энергетических системах) – руководитель д.т.н. Массель Л.В.
- Грант РФФИ №16-07-00474 (2016-2018): «Методология построения интеллектуальных систем семиотического типа для стратегического ситуационного управления в критических инфраструктурах» – руководитель д.т.н. Массель Л.В.
- Проект Программы Президиума РАН №229 (2012-2014), «Методы и инструментальные средства поддержки принятия решений в исследованиях и обеспечении энергетической безопасности на основе интеллектуальных вычислений» (2012-2014) Программы №15 Президиума РАН «Информационные, управляющие и интеллектуальные технологии и системы» - руководитель д.т.н. Л.В. Массель.
- Грант РФФИ №15-07-01284 (2015-2017): «Методы ситуационного управления и семантического моделирования в энергетике» – руководитель к.т.н. А.Г. Массель.
- Грант РФФИ и Белорусского республиканского фонда фундаментальных исследований (БРФФИ) для молодых ученых №15-07-04074 Бел_мол_а

«Методы интеллектуальной поддержки принятия решений в энергетике России и Беларуси при реализации угроз энергетической безопасности» (2015-2016) – руководитель к.т.н. А.Г. Массель. В рамках последнего проекта результаты диссертационной работы переданы в Институт энергетики НАН Беларуси.

- Интеграционный проект фундаментальных исследований СО РАН по конкурсу проектов фундаментальных исследований НАН Беларуси и СО РАН № 18 «Методы построения интеллектуальной инструментальной среды для поддержки принятия решений при определении стратегии развития энергетики России и Беларуси с позиций энергетической безопасности» (2012-2014) - руководитель д.т.н. Массель Л.В.

  В рамках двух последних проектов результаты диссертационной работы переданы в Институт энергетики НАН Беларуси и применены при подготовке материалов для правительства Беларуси.

3.4. Выводы по главе 3

Автором, на основе предложенного методического подхода, выполнено проектирование многоагентной системы оценивания состояний ЭЭС. Выделены и описаны основные компоненты для каждого типа агентов. Также были построены диаграммы классов и определены функции каждого агента и и требования к нему. Разработаны главный модуль (агент-менеджер), агенты декомпозиции, координации и агрегирования. Агент оценки состояний был реализован на основе проведенного автором реинжиниринга ПВК «Оценка-ПК». Проведены вычислительные эксперименты по ОС схемы «Московское кольцо» и ЭЭС Красноярска. Выполнено сравнение результатов расчета основной схемы и схем, полученных после декомпозиции, разницы в полученных показателях не было, что доказало корректность алгоритма работы МАС EstateMAS [80-83, 87, 88].

Использование МАС при декомпозиции задачи ОС обеспечивает возможность **проводить расчет отдельных подсистем параллельно друг от друга**, что на системах с большим количеством узлов **позволяет ускорить процесс обработки данных**, однако это требует **решать координационную задачу**, что может привести к дополнительным итерациям расчета. Благодаря агентным сценариям пользователь может редактировать алгоритм, что обеспечивает более **гибкий подход к решению задачи**.

До разработки МАС «EstateMAS» задача оценивания состояния решалась с помощью ПВК «Оценка» только для отдельных подсистем. Не предусматривались задание и смена сценариев и операции декомпозиции и координации, которые в МАС «EstateMAS» реализованы и автоматизированы. Получены 6 свидетельств о государственной регистрации программ для ЭВМ для отдельных агентов и для системы «EstateMAS» в целом [89-94].

Метод оценки надежности МАС применен для конфигураций МАС, использованных для примеров расчетов: для каждого примера построены графы агентных взаимодействий, получены матрицы смежности, определены степени вершин, выявлены критичные агенты и сформулированы рекомендации по повышению их надежности.

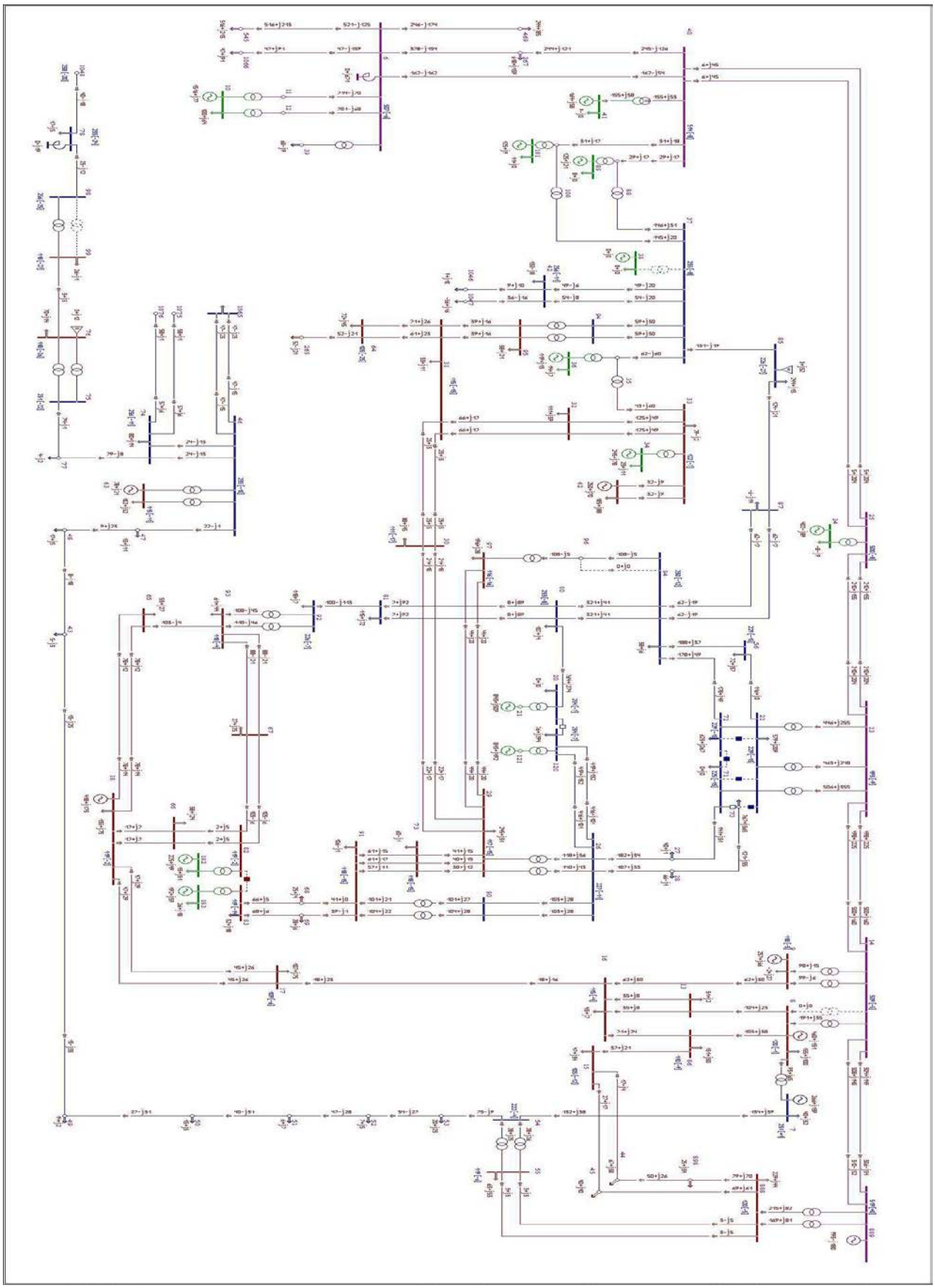


Рис. 3.12. Схема ЭЭС Красноярска.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

В работе получены следующие основные результаты:

1. Проведен анализ предметной области, связанной с разработкой концепции интеллектуальных энергетических систем и решением задач оценивания состояния ЭЭС.
2. Выполнен анализ существующих подходов к разработке многоагентных систем и возможностей их применения при реализации концепции интеллектуальных энергетических систем на примере задачи оценивания состояния ЭЭС.
3. Разработан методический подход к построению типовых многоагентных систем и предложена методика, включающая пять основных этапов: проектирование и реализацию МАС, разработку агентных сценариев, алгоритм межплатформенного взаимодействия и поведения агентов, разработку базовых компонентов МАС. Апробация методического подхода проведена на примере разработки МАС для решения задачи оценивания состояния ЭЭС. Рассмотрены и реализованы основные этапы разработки системы, представлены графические схемы бизнес-процессов для каждого этапа. Описан предлагаемый автором алгоритм взаимодействия агентов через TCP/IP и состав XML-сообщений, используемых в системе, который позволит обмениваться информацией агентам, написанным на разных программных платформах. Описана структура XML-файлов и тех данных, которые в них хранятся.
4. Предложен вычислительный метод управления взаимодействием агентов на основе алгебраических сетей. Построены событийные модели агентных сценариев с использованием аппарата Joiner-сетей, позволяющие пользователям самостоятельно вносить изменения в алгоритм работы системы без участия программистов. Использование агентных сценариев

обеспечивает гибкость при составлении пользователями алгоритма решения задачи.

5. Разработан алгоритм поведения агентов при выполнении многоэтапных расчетов. Определены возможные ситуации, которые могут произойти при поиске агентов. Описан алгоритм декомпозиции задачи оценивания состояния ЭЭС. Определены основные требования к многоагентной системе, реализующей этот алгоритм. Определены системно-концептуальные соглашения, принятые при разработке многоагентной системы.
6. Предложен численный метод оценки надежности многоагентной системы. Выполнена оценка надежности конфигураций МАС для конкретных задач с использованием разработанного метода. На основе предлагаемого методического подхода выполнено проектирование многоагентной системы оценивания состояний ЭЭС «EstateMAS». Выделены и описаны основные базовые компоненты для каждого типа агентов. Построены диаграммы классов и выделены функции каждого агента и требования к нему. Были разработаны агенты декомпозиции, координации и агрегирования, а так же главный модуль. Агент оценки состояний реализован на основе ПВК «Оценка-ПК».
7. Разработана усовершенствованная многоагентная технология численного решения задачи оценивания состояния с использованием предложенного методического подхода и разработанного программного обеспечения.
8. Проведены вычислительные эксперименты по ОС схем «Московское кольцо» и ЭЭС Красноярска. Выполнено сравнение результатов расчета основной схемы и схем, полученных после декомпозиции, разницы в полученных показателях не было, что подтверждает корректность работы системы.
9. Результаты диссертационной работы применены в базовом проекте ИСЭМ СО РАН №01201361373 и в проектах по грантам РФФИ №13-07-140 (2013-2015), №16-07-00474 (2016-2018), №15-07-01284 (2015-2017), гранту Программы Президиума РАН №229 (2012-2014), РФФИ №15-07-04074

Бел_мол_а (2015-2016), а так же гранту СО РАН и Института энергетики НАН Беларуси №18Б (2012-2014). В рамках двух последних проектов результаты диссертационной работы переданы и использованы в Институте энергетики НАН Беларуси.

Список литературы

1. Grid 2030: A national version for electricity's second 100 years. Office of Electric Transmission and Distribution, United States Department of Energy, July 2003, pp. 89.
2. Amin S.M., Wollenberg B.F. Toward a Smart Grid: power delivery for the 21 st century // IEEE Power and Energy Magazine, 2005, Vol. 3, No. 5, pp. 34-41.
3. European Smart Grids technology platform: Vision and strategy for Europe's electricity networks of the future. European Commission, 2006, pp. 38.
4. Shahidehpour M. Smart Grid: A new paradigm for power delivery// IEEE Bucharest Power Tech., Bucharest, Romania, June 28 – July 2, 2009, pp. 7.
5. Глушко С., Пикин С. Технологическая концепция Smart Grid – облик электроэнергетики будущего // Энергорынок, 2009, №11(71). – С. 68-72.
6. Кобец Б.Б., Волкова И.О. Smart Grid в электроэнергетике // Энергетическая политика, 2009, вып. 6. – С. 54-56.
7. Дорофеев В.В., Макаров А.А. Активно-адаптивная сеть – новое качество ЕЭС России // Энергоэксперт. 2009, № 4. – С. 28-34.
8. Chuand A., McGranaghan M. Function of a local controller to coordinate distributed resources in a Smart Grid // IEEE PES General Meeting, Pittsburg, USA, July 20-24, 2004, pp. 6.
9. McDonald J. Adaptive intelligent power systems: Active distribution networks // Energy Policy, 2008, Vol. 36, pp. 4346-4351.

10. Mamo X., Mallet S., Coste Th., Grenard S. Distribution automation: The cornerstone for Smart Grid development strategy // IEEE PES General Meeting, Calgary, Canada, July 26-30, 2009, pp. 6.
11. Simard G., Chartrand D., Christophe P. Distribution automation: Applications to move from today's distribution system to tomorrow's Smartgrid // IEEE PES General Meeting, Calgary, Canada, July 26-30, 2009, pp. 5.
12. Гамм А.З., Герасимов Л.Н., Голуб И.И., Гришин Ю.А., Колосок И.Н. Оценивание состояния в электроэнергетике – М.: Наука. – 1983. – 302 с.
13. Воропай Н.И., Стенников В.А. Интегрированные интеллектуальные энергетические системы // Известия РАН. Энергетика. – 2014. – №1. – С.64-71.
14. Scada, доступно по: http://www.scada.ru (информация от 18 сентября 2013).
15. G. Phadke. Synchronized Phasor Measurements. A Historical Overview. // IEEE/PES Transmission and Distribution Conference, 2002, №.1. – С.476-479.
16. Концепция интеллектуальной электроэнергетической системы России с активно-адаптивной сетью // под ред. академиков Фортова В.Е. и Макарова А.А. – М: ОАО «НТЦ ФСК ЕЭС», 2012. – С. 235.
17. Массель Л.В. Проблема построения интеллектуальных и программных компонентов Smart Grid и подход к ее решению на основе агентной технологии / Материалы XL Международной конференции «Информационные технологии в науке, образовании, телекоммуникации и бизнесе» // Приложение к журналу «Открытое образование», Украина, Крым, 2012. – С. 22-25.
18. Рассел С., Норвиг П. Искусственный интеллект: современный подход, 2-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2006. – С. 1408.
19. Maes P. Artificial Life Meets Entertainment: Life Like Autonomous Agents// Communication of the ACM. – 1995. –Vol. 38, №11, pp. 108-114.
20. Тарасов В.Б. Агенты, многоагентные системы, виртуальные сообщества: стратегическое направление в информатике и искусственном интеллекте. // Новости искусственного интеллекта. – 1998. – №3. – С. 5-54.

21. Городецкий В.И. Многоагентные системы: современное состояние исследований и перспективы применения// Новости искусственного интеллекта. – 1996. – №1. – С. 44-59.
22. Городецкий В.И. Многоагентные системы: основные свойства и модели координации поведения// Информационные технологии и вычислительные системы. – 1998. – №1. – С. 22-34.
23. Franklin S., Graesser A. Is It an Agent, or Just a Program?// Intelligent Agents III. Proc. of ECAI-96 Workshop on Agent Theories, Architectures and Languages (ATAL, Budapest, The Hungary, August 12-13, 1996)/Ed.by J.-P.Muller, M.Wooldridge, N.Jennings.-Berlin: Springer Verlag, 1996, pp. 21-35.
24. Decker K. Distributed Problem Solving Techniques: A Survey// IEEE Transactions on Systems, Man, and Cybernetics. – 1987. – Vol.17, №5.
25. De Champeaux D., Lea D., Faure P. Object-oriented system development. Addison-Wesley, 1993, pp. 532.
26. Meyer B. Object-oriented software construction. N. Y., NY: Prentice-Hall, 1988, pp. 1296.
27. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++.–М.: Бином; СПб: Невский диалект, 1998.– С. 560.
28. Gartner. Top Strategic Predictions for 2016 and Beyound: The Future Is a Digital Thing. https://www.gartner.com/doc/3142020?refval=&pcp=mpe (по состоянию на 19 января 2016 г.)
29. DeLoach S.A. Moving multi-agent systems from research to practice // Int. J. Agent-Oriented Software Engineering Vol. 3 No.4. 2009, pp. 378-382.
30. Luck M., et al. Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing) AgentLink. 2005. http://www.agentlink.org/roadmap/ (по состоянию на 19 января 2016г.)
31. Muller J., Fisher K. Application Impact of Multiagent Systems and Technologies: A Survey // In Agent-Oriented Software Engineering book series. Springer. 2013, pp. 1-26.

32. Woldridge M. An Introduction to MultiAgent Systems. John Wiley & Sons. 2009, pp. 368.
33. Cohen P., Levesque H. Teamwork// Nous. V. 25. 1991, pp. 487-515
34. AI Portal (2014), доступно по: http://www.aiportal.ru/articles/multiagent-systems/multiagent-systems.html (информация от 15 апреля 2015).
35. Городецкий В.И., Скобелев П.О., Бухвалов О.Л. Промышленные применения многоагентных систем: прогнозы и реалии // Труды XVIII Международной конференции «Проблемы управления и моделирования в сложных системах». Самара: ООО «Офорт», 2016. С. 137-162.
36. Многоподходное имитационное моделирование, доступно по: http://www.anylogic.ru/ (информация от 15 апреля 2015).
37. Odell. J., Giorgini P., Muller J.P. Agent-Oriented Software Engineering // 5th International Workshop, AOSE 2004, New York, NY, USA, July 2004, Revised Selected Papers (Lecture Notes in Computer Science).
38. Bresciani P., Giorgini P., Henderson-sellers B., Low G. Agent-Oriented Information Systems II // 6th International Bi-Conference Workshop, AOIS 2004,
39. Riga, Latvia, June 8, 2004 and New York, NY, USA, July 20 (Lecture Notes in Artificial Intelligence).
40. Cavedon L., Maamar Z., Martin D., Benatallah B. Extending Web Services Technologies : The Use of Multi-Agent Approaches // Multiagent Systems, Artificial Societies, and Simulated Organizations – ISBN: 0387233431
41. Фартышев Д.А., Черноусова Е.С., Черноусов А.В. Подход к разработке многоагентной распределенной интеллектуальной информационной системы для исследований в энергетике // Вычислительные технологии. Том 13. Специальный выпуск №1. г. Иркутск, 2007 – С. 108-115.
42. Фартышев Д.А., Черноусов А.В. Методы использования Web-сервисов для построения вычислительной ИТ-инфраструктуры системных исследований в энергетике // Информационные и математические технологии в науке и

управлении: Тр. XII Байк. Всерос. конф. Ч. 2. Иркутск: ИСЭМ СО РАН, 2007.– С. 46-54.

43. FIPA Abstract Architecture Specification // FIPA TC Architecture [SC00001L].– 2002.– 12 March
44. JAVA Agent DEvelopment Framework (2015) доступно по: http://jade.tilab.com/ (информация от 15 апреля 2015).
45. Poslad S., Buckle P., Hadingham R. The FIPA-OS agent platform: Open Source for Open Standards. // Proceedings of PAAM 2000, Manchester, UK, pp. 355-368.
46. Новик К.В. Сеть автоматов для моделирования асинхронного взаимодействия процессов / Автореф. дисс. на соискание степени канд. физ.-мат. наук: 05.13.18. – М.: МФТИ, 2006. – С. 22.
47. Столяров Л.Н., Новик К.В. Joiner-сеть для моделирования взаимодействующих параллельных процессов // Моделирование процессов управления: Сб. научных трудов / Моск. физ.-тех.. ин-т. – М., 2004. – С. 81-97.
48. Столяров Л.Н., Новик К.В. Реализация параллельных процессов с помощью сетей Joiner-net // Информационные и математические технологии: Сб. научных трудов / ИСЭМ СО РАН – Иркутск, 2004. – С. 11-14.
49. Норенков И.П., Кузьмик П.К. Информационная поддержка наукоемких изделий. CALS-технологии. – М.: Изд-во МГТУ им. Баумана, 2002, – С. 320: ил. ISBN 5-7038-1962-8.
50. Столяров Л.Н. Философия событийного моделирования на примере сценария энергетической катастрофы / Столяров Л.Н. // Труды Международной конференции «Информационные технологии в науке, образовании, телекоммуникации и бизнесе», Гурзуф, – 2010. – С. 197-200.
51. Гамм А.З., Колосок И.Н. Усовершенствованные алгоритмы оценивания состояния электроэнергетических систем. // Электричество. – 1987. – № 11. – С. 25-29.

52. Гамм А.З., Колосок И.Н. Обнаружение грубых ошибок телеизмерений в электроэнергетических системах. – Новосибирск: Наука, 2000. – С. 152.
53. Monticelly. “Electric power system state estimation”. Proceedings of the IEEE, 88(2): 2000, pp. 262-282.
54. Holton L., Gjelsvik A., Wu F.F., Liu W.H. Comparison of different methods for state estimation. // IEEE Trans. Power Appar. And Syst.- 1998. – Vol. 3, № 4, pp. 1798-1806.
55. Конторович А.М., Тараканов А.А. Выдерживание точных измерений при оценивании состояния электрических систем. //Информационное обеспечение диспетчерского управления в электроэнергетике. Новосибирск: Наука. – 1985. – С. 63-68.
56. Clements K. A., Krumpholz G.R., Davis P.W. Power System State Estimation with Measurement Deficiency: An Observability/Measurement Placement Algorithm. // IEEE Trans. On Power Systems. – July 1983. – Vol. PAS – 102, № 7, pp. 2012-2020.
57. Гамм А.З. Статистические методы оценивания состояния электроэнергетических систем. – М.: Наука. – 1976. – С. 220.
58. Гамм А.З., Кучеров Ю.Н., Паламарчук С.И. и др. Методы решения задач реального времени в электроэнергетике. – Новосибирск: Наука, 1991. – 293 с.
59. Гамм А.З., Голуб И.И. Наблюдаемость электроэнергетических систем. – М.: Наука. – 1990. – С. 220.
60. И.Н.Колосок, А.С.Пальцев. Двухуровневый иерархический алгоритм оценивания состояния ЭЭС и его реализация на основе мультиагентного подхода // Сб. докладов III Международной научно-практической конференции «ЭНЕРГОСИСТЕМА: управление, конкуренция, образование». Т.1. Екатеринбург, УГТУ-УПИ, 2008. – С. 354-359.
61. Пальцев А.С. Распределенная обработка телеинформации при оценивании состояния ЭЭС на основе мультиагентных технологий // автореферат — Иркутск, 2010. — С. 20.

62. Курбацкий В.Г., Томин Н.В. Применение новых информационных технологий в решении электроэнергетических задач / Системы. Методы. Технологии. №1 / 2009. – С. 113-119.
63. Панасецкий Д.А., Воропай Н.И. Развитие принципов противоаварийного управления для обеспечения устойчивости по напряжению электроэнергетических систем / Электричество №8 2011. – С. 6-14.
64. Массель Л.В., Гальперов В.И. Разработка многоагентных систем распределенного решения энергетических задач с использованием агентных сценариев / Известия Томского политехнического университета Т. 326. № 5, 2015. – С. 45-53.
65. Гальперов В.И., Колосок И.Н., Массель Л.В., Пальцев А.С. Постановка задачи разработки мультиагентной системы для оценивания состояний ЭЭС с учетом структурной и функциональной декомпозиции. – Информационные и математические технологии в науке и управлении / Труды XVIII Байкальской Всероссийской конференции "Информационные и математические технологии в науке и управлении". Часть III. – Иркутск: ИСЭМ СО РАН, 2013. – С. 231-234.
66. Гальперов В.И. Агентные сценарии для разработки многоагентных систем оценивания состояний ЭЭС // Информационные и математические технологии в науке и управлении / Труды XIX Байкальской Всероссийской конференции «Информационные и математические технологии в науке и управлении». Часть III. – Иркутск: ИСЭМ СО РАН, 2014. – С. 76-80.
67. Гальперов В.И. Методика построения многоагентных систем с использованием Joiner-сетей для описания сценариев взаимодействия агентов // Системные исследования в энергетике: Труды молодых ученых ИСЭМ СО РАН. Вып. 45. – Иркутск: ИСЭМ СО РАН, 2015. – С. 153-160.
68. Гальперов. В.И. Применение многоагентного подхода для разработки программных систем оценивания состояния ЭЭС. – Системные исследования в энергетике/ Труды молодых ученых ИСЭМ СО РАН, Вып. 44. – Иркутск ИСЭМ СО РАН, 2014.- С. 165-170.

69. Гальперов В.И. Проектирование многоагентной системы оценивания состояния ЭЭС // Труды XX Байкальской Всероссийской конференции "Информационные и математические технологии в науке и управлении". Часть II. – Иркутск: ИСЭМ СО РАН, 2015. – С. 7-13.
70. Массель Л.В., Гальперов В.И. Проектирование и разработка многоагентной системы оценивания состояний ЭЭС / Вестник ИрГТУ.- 2015.- №10 – С. 27-33.
71. Kumar S., Cohen P.R. Towards a fault-tolerant multi-agent system architecture. In: Proceedings of the Fourth International Conference on Autonomous Agents. ACM, 2000, pp. 459-466. DOI:10.1145/336595.337570
72. Guessoum Z., Briot J.P., Faci N. Towards Fault-Tolerant Massively Multiagent Systems. In:Massively Multi-Agent Systems I. Springer Berlin Heidelberg, 2005, pp. 55-69. (Ser. Lecture Notes in Computer Science; vol. 3446). DOI: 10.1007/11512073_5
73. Serugendo G.D.M., Romanovsky A. Designing Fault-Tolerant Mobile Systems. In: Scientific Engineering for Distributed Java Applications. Springer Berlin Heidelberg, 2003, pp. 185-201. (Ser. Lecture Notes in Computer Science; vol. 2604). DOI: 10.1007/3-540-36520-6_17
74. Mellouli S. A Reorganization Strategy to Build Fault-Tolerant Multi-Agent Systems. In: Advances in Artificial Intelligence. Springer Berlin Heidelberg, 2007, pp. 61-72. (Ser. Lecture Notes in Computer Science; vol. 4509). DOI: 10.1007/978-3-540-72665-4_6
75. Игумнов А. В., Сараджишвили С. Э. Оценка надежности резервированных многоагентных систем / Наука и образование.- №1.-2014.- М.: МГТУ им. Баумана. Эл. №ФС77-48211. ISSN 1994-0448. DOI: 10.7463/0114.0696290
76. Гришин Ю.А., Колосок И.Н., Коркина Е.С., Эм Л.В., Орнов В.Г, Шелухин Н.Н. Программно-вычислительный комплекс «Оценка» оценивания состояния ЭЭС в реальном времени. // Электричество. 1999. №2. – С. 8-16.

77. Массель Л.В., Курганская О.В., Гальперов В.И. Настольное приложение для интеллектного контроля и преобразования данных для вычислительного эксперимента в исследованиях энергетической безопасности IntDataTransformer Lite. Свидетельство о государственной регистрации программы для ЭВМ № 2015612394.
78. Массель Л.В., Массель А.Г. Интеллектуальные вычисления в исследованиях направлений развития энергетики // Известия Томского политехнического университета. – 2012. – Т. 321. – № 5. Управление, вычислительная техника и информатика. – С. 135-141.
79. Массель Л.В. Интеллектуализация поддержки принятия решений при моделировании и управлении режимами в Smart Grid // Интеллектуализация обработки информации: Труды 9-й Международной конференции. – Черногория, Будва, 2012 . – С. 692-695.
80. Массель Л.В., Гальперов В.И. Разработка многоагентной системы оценивания состояний электроэнергетических систем с использованием событийных моделей/ Наука и образование.- №9.-2015.- М.: МГТУ им. Баумана. Эл. №ФС77-4211. ISSN 1994-0448. DOI: 10.7463/ 0915. 0811180.
81. Galperov V. Agent-based scenarios and cross-platform communication // Proceeding of International Workshop “Intelligent, agent-based, cloud computing and cyber security in energy sector’ (IACCCSES-2014) – Mongolia-Russia, Irkutsk: MESI, 2014, pp. 16.
82. Galperov V. The development of multi-agent software for states estimation of energy power systems // Proceeding of International Workshop "Contingency management, intelligent, agent-based computing and cyber security in energy sector" (CM/IAC/CS/ES-2015) – Mongolia-Russia, Irkutsk: MESI, 2015, pp. 46.
83. Гальперов В.И. Применение Joiner-сетей при разработке многоагентных систем оценивания состояния ЭЭС / Материалы Всерос. конф. мол. ученых по математическому моделированию и информационным технологиям

(Красноярск, 28-30 октября 2015 г.) – Новосибирск: ИВТ СО РАН, 2015. – С. 65.

84. Массель Л.В., Гальперов В.И. Анализ надежности работы многоагентных систем с использованием графовой модели / Вестник ИрГТУ. – 2017. №1– С. 72-80.
85. Лифшиц Ю. Структура сложных сетей, курс "Алгоритмы для Интернета", http://logic.pdmi.ras.ru/~yura/internet/04ianote.pdf (информация от 13 ноября 2016).
86. Мальков М.В. О надежности информационных систем / Труды Кольского научного центра РАН, № 4/том 3,г. Апатиты КНЦ РАН, 2012. – С. 49-58.
87. Galperov V. The use of agent-based scripting and event models for the development of multi-agent system for state estimation of Energy Power Systems / Proceeding of International Workshop «Contingency management, intelligent, agent-based computing and cyber security in critical infrastructures» (CM/IAC/CS/CI – 2016) – Russia, Irkutsk: MESI, 2016, pp. 54.
88. Massel L.V., Galperov V.I. The development of multi-agent system state estimation of electric power systems using joiner-networks. / Proceedings of the Workshop on Computer Science and Information Technologies (CSIT'2015), Rome, Italy, September 22−26, 2015, Volume 2, Ufa State Aviation Technical University, 2015, pp. 1-6
89. Массель Л.В., Гальперов В.И. Многоагентная система для оценивания состояния ЭЭС «EstateMAS». Свидетельство о государственной регистрации программы для ЭВМ № 2016661113.
90. Массель Л.В., Гальперов В.И. Агент мониторинга и контроля функционирования многоагентной системы. Свидетельство о государственной регистрации программы для ЭВМ № 2016661065 .
91. Массель Л.В., Гальперов В.И. Агент агрегирования. Свидетельство о государственной регистрации программы для ЭВМ № 2016661106 .

92. Массель Л.В., Гальперов В.И. Агент координации. Свидетельство о государственной регистрации программы для ЭВМ № 2016661105 .
93. Массель Л.В., Гальперов В.И. Агент оценивания состояния ЭЭС. Свидетельство о государственной регистрации программы для ЭВМ № 2016661107 .
94. Массель Л.В., Гальперов В.И. Агент декомпозиции схемы ЭЭС. Свидетельство о государственной регистрации программы для ЭВМ № 2016661005 .

Приложение 1

Результаты ОС схемы ЭЭС Красноярска

Сравнение общей схемы с первой подсистемой

№ узла	Измеренное напряжение в кВ	Оцененные значения			
	Полная схема	Полная схема		Первая подсистема	
		U	δ	U	δ
6	**514,81**	**501,82**	**-3,49**	**501,82**	**-3,49**
7	228,00	223,00	0,00	223,00	0,00
8	**119,00**	**119,79**	**-11,87**	**119,79**	**-11,87**
9	**118,80**	**119,75**	**-11,92**	**119,75**	**-11,92**
10	**23,90**	**23,19**	**1,46**	**23,19**	**1,46**
11	**515,37**	**502,33**	**-2,91**	**502,33**	**-2,91**
12	**515,37**	**502,33**	**-2,92**	**502,33**	**-2,92**
13	114,88	118,31	-14,67	118,31	-14,67
14	**503,29**	**501,65**	**-14,97**	**501,65**	**-14,97**
15	105,65	116,57	-13,57	116,57	-13,57
16	114,23	117,98	-15,35	117,98	-15,35
17	109,87	113,25	-28,51	113,25	-28,51
18	118,50	120,00	-31,63	120,00	-31,63
20	236,58	237,13	-31,88	237,13	-31,88
21	15,20	15,35	-29,67	15,35	-29,67
22	**226,69**	**225,51**	**-33,65**	**225,51**	**-33,65**
23	**487,87**	**481,93**	**-24,65**	**481,93**	**-24,65**
24	**15,60**	**14,62**	**-18,11**	**14,62**	**-18,11**
25	**492,80**	**486,49**	**-20,40**	**486,49**	**-20,40**
26	229,10	229,26	-36,88	229,26	-36,88
27	226,48	231,44	-43,02	231,44	-43,02
28	226,75	228,86	-38,79	228,86	-38,79
29	118,33	114,93	-35,99	114,93	-35,99
30	111,15	103,68	-29,71	103,68	-29,71
31	115,34	108,85	-15,45	108,85	-15,45
32	116,38	109,36	-13,85	109,36	-13,85
33	120,80	113,54	-8,33	113,54	-8,33
34	18,10	16,85	-4,45	16,85	-4,45
35	229,27	215,17	-8,35	215,17	-8,35
36	18,20	16,74	-4,30	16,74	-4,30
37	**234,48**	**225,65**	**-10,49**	**225,65**	**-10,49**
38	17,80	16,79	-7,78	16,79	-7,78
39	**227,85**	**220,84**	**-3,84**	**220,84**	**-3,84**
40	**509,08**	**496,42**	**-10,86**	**496,42**	**-10,86**

41	**20,00**	**18,99**	**-6,94**	**18,99**	**-6,94**
42	236,71	228,91	-11,18	228,91	-11,18
43	231,44	229,69	9,56	229,69	9,56
44	104,82	117,35	-12,39	117,35	-12,39
45	116,05	123,71	-6,34	123,71	-6,34
46	235,46	234,53	14,60	234,53	14,60
47	234,99	233,31	13,68	233,31	13,68
48	232,64	230,77	10,78	230,77	10,78
49	228,52	227,43	7,09	227,43	7,09
50	226,89	226,22	5,78	226,22	5,78
51	225,35	225,10	4,56	225,10	4,56
52	223,66	223,84	3,16	223,84	3,16
53	221,34	222,17	1,51	222,17	1,51
54	220,08	221,95	0,24	221,95	0,24
55	120,31	123,12	-1,98	123,12	-1,98
56	227,23	227,89	-33,61	227,89	-33,61
62	120,82	113,56	-6,69	113,56	-6,69
63	114,00	113,27	13,72	113,27	13,72
64	103,39	101,48	-17,26	101,48	-17,26
65	118,89	119,49	-33,02	119,49	-33,02
66	119,14	118,62	-32,46	118,62	-32,46
67	119,13	117,77	-33,10	117,77	-33,10
68	118,46	116,27	-34,47	116,27	-34,47
69	118,46	116,28	-34,48	116,28	-34,48
70	**225,96**	**232,06**	**-44,60**	**232,06**	**-44,60**
71	225,97	227,75	-39,20	227,75	-39,20
72	**227,82**	**240,43**	**-33,15**	**240,43**	**-33,15**
73	118,41	114,93	-35,89	114,93	-35,89
74	235,95	233,95	14,46	233,95	14,46
75	231,58	230,63	3,41	230,63	3,41
76	117,85	117,85	0,00	117,85	0,00
77	234,03	232,56	5,90	232,56	5,90
78	235,40	221,14	-50,56	221,14	-50,56
80	234,15	234,78	-32,27	234,78	-32,27
81	225,83	225,47	-33,07	225,47	-33,07
82	119,61	117,88	-33,03	117,88	-33,03
83	118,73	117,86	-33,10	117,86	-33,10
84	229,11	234,18	-33,37	234,18	-33,37
85	226,93	226,00	-41,13	226,00	-41,13
86	109,01	116,18	-15,70	116,18	-15,70
87	229,34	234,18	-33,62	234,18	-33,62
88	**508,69**	**489,33**	**-10,58**	**489,33**	**-10,58**
89	**17,80**	**16,82**	**-6,07**	**16,82**	**-6,07**
90	228,91	228,88	-36,82	228,88	-36,82
91	118,52	114,93	-35,77	114,93	-35,77

92	223,62	223,00	-33,18	223,00	-33,18
93	118,82	119,00	-33,20	119,00	-33,20
94	231,65	221,71	-11,75	221,71	-11,75
95	117,16	110,91	-14,55	110,91	-14,55
96	229,28	233,69	-33,50	233,69	-33,50
97	117,29	114,67	-36,19	114,67	-36,19
98	235,53	217,35	-48,38	217,35	-48,38
99	118,08	103,29	-44,03	103,29	-44,03
100	**508,53**	**489,09**	**-10,54**	**489,09**	**-10,54**
101	**17,80**	**16,82**	**-8,69**	**16,82**	**-8,69**
120	236,58	237,14	-31,88	237,14	-31,88
121	15,00	15,27	-27,30	15,27	-27,30
182	10,50	10,42	-27,90	10,42	-27,90
183	10,50	10,38	-30,99	10,38	-30,99
265	105,39	105,39	-12,59	105,39	-12,59
888	**119,76**	**126,27**	**-4,15**	**126,27**	**-4,15**
889	533,00	535,96	0,00	535,96	0,00
898	116,46	124,19	-5,64	124,19	-5,64
1046	237,18	229,55	-10,38	229,55	-10,38
1047	233,83	226,12	-9,60	226,12	-9,60
1048	240,00	238,00	-55,42	238,00	-55,42
1065	234,46	236,53	16,92	236,53	16,92
1075	236,90	234,36	15,96	234,36	15,96
1076	236,97	234,44	16,02	234,44	16,02

Сравнение общей схемы со второй подсистемой

№ узла	Измеренное напряжение кВ	Оцененные значения			
	Полная схема	Полная схема		Вторая подсистема	
		U	δ	U	δ
6	**514.81**	**501.835**	**-3.487**	**501.835**	**-3.487**
8	**119.00**	**119.793**	**-11.872**	**119.793**	**-11.872**
9	**118.80**	**119.753**	**-11.923**	**119.753**	**-11.923**
10	**23.90**	**23.191**	**-1.458**	**23.191**	**-1.458**
11	**515.37**	**502.345**	**-2.914**	**502.345**	**-2.914**
12	**515.37**	**502.345**	**-2.924**	**502.345**	**-2.924**
14	**503.29**	**501.665**	**-14.971**	**501.665**	**-14.971**
22	**226.69**	**225.517**	**-33.645**	**225.517**	**-33.645**
23	**487.87**	**481.945**	**-24.646**	**481.945**	**-24.646**
24	**15.60**	**14.620**	**-18.113**	**14.620**	**-18.113**
25	**492.80**	**486.505**	**-20.401**	**486.505**	**-20.401**
37	**234.48**	**225.627**	**-10.489**	**225.627**	**-10.489**
39	**227.85**	**220.847**	**-3.838**	**220.847**	**-3.838**
40	**509.08**	**496.435**	**-10.857**	**496.435**	**-10.857**

41	**20.00**	**18.991**	**-6.941**	**18.991**	**-6.941**
70	**225.96**	**232.067**	**-44.602**	**232.067**	**-44.602**
72	**227.82**	**240.437**	**-33.148**	**240.437**	**-33.148**
88	**508.69**	**489.275**	**-10.583**	**489.275**	**-10.583**
89	**17.80**	**16.821**	**-6.070**	**16.821**	**-6.070**
100	**508.53**	**489.035**	**-10.537**	**489.035**	**-10.537**
101	**17.80**	**16.821**	**-8.689**	**16.821**	**-8.689**
267	515.91	510.645	-4.976	510.645	-4.976
469	516.58	514.015	-4.396	514.015	-4.396
545	496.06	496.060	-0.528	496.060	-0.528
888	**119.76**	**126.274**	**-4.145**	**126.274**	**-4.145**
889	533.00	535.975	0.000	535.975	0.000
1066	517.82	495.595	-4.464	495.595	-4.464

Приложение 2

Справка о практическом использовании результатов диссертационного исследования:

Рэспубліканскае навукова-вытворчае
унітарнае прадпрыемства
«ІНСТЫТУТ ЭНЕРГЕТЫКІ
НАЦЫЯНАЛЬНАЙ АКАДЭМІ
НАВУК БЕЛАРУСІ»
вул. Акадэмічная, 15, корп.2, 220072, г. Мінск
тэл. (017) 294 94 72, тэл./факс (017) 284 13 26
e-mail: ipe@bas-net.by
Р/р 3012081810018 у філіяле № 529 "Белсувязь"
ААТ "АСБ Беларусбанк",
г. Мінск код 720, УНП 101292548,
АКПА 37465805

Республиканское научно-производственное
унитарное предприятие
«ИНСТИТУТ ЭНЕРГЕТИКИ
НАЦИОНАЛЬНОЙ АКАДЕМИИ
НАУК БЕЛАРУСИ»
ул. Академическая, 15, корп. 2, 220072, г.Минск
тел. (017) 294 94 72, тел./факс (017) 284 13 26
e-mail: ipe@bas-net.by
Р/с 3012081810018 в филиале № 529 "Белсвязь"
ОАО "АСБ Беларусбанк",
г. Минск код 720, УНП 101292548,
ОКПО 37465805

18.11.2016 № 122-128
На № _____ ад _____

СПРАВКА

о практическом использовании результатов диссертационного исследования Гальперова Василия Ильича, выполненного на тему: «Методы, модели и алгоритмы построения многоагентных систем для оценивания состояний ЭЭС»

Отдельные результаты диссертационного исследования Гальперова Василия Ильича приняты к использованию в деятельности Института энергетики Национальной академии наук Беларуси в рамках проектов: «Методы построения интеллектуальной инструментальной среды для поддержки принятия решений при определении стратегии развития энергетики России и Беларуси с позиций энергетической безопасности» (интеграционный проект СО РАН и НАН Беларуси, 2012-2014 гг.) и «Методы интеллектуальной поддержки принятия решений в энергетике России и Беларуси при реализации угроз энергетической безопасности» (проект коллективов молодых ученых при поддержке Российского фонда фундаментальных исследований и Белорусского республиканского фонда фундаментальных исследований, 2015-2016 гг.), а именно:

- методика построения многоагентных систем (МАС) для управления в энергетике, включающая этапы построения агентных сценариев, моделирования их взаимодействия с помощью Joiner-сетей и проектирования конкретных МАС с использованием базовых программных компонентов;

- базовые программные компоненты для разработки конкретных МАС в энергетике, включающие модули интерфейса, сетевого взаимодействия агентов, редактирования агентных сценариев и мониторинга состояния агентов.

Научный руководитель института, академик А.А. Михалевич

Приложение 3

Свидетельства о государственной регистрации программ для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016661113

Многоагентная система для оценивания состояния ЭЭС «EstateMAS»

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт систем энергетики им. Л.А. Мелентьева Сибирского отделения Российской академии наук (RU)***

Авторы: ***Гальперов Василий Ильич (RU), Массель Людмила Васильевна (RU)***

Заявка № **2016618473**
Дата поступления **03 августа 2016 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***30 сентября 2016 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016661005

Агент декомпозиции схемы ЭЭС

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт систем энергетики им. Л.А. Мелентьева Сибирского отделения Российской академии наук (RU)***

Авторы: ***Гальперов Василий Ильич (RU),***
Массель Людмила Васильевна (RU)

Заявка № **2016618480**
Дата поступления **03 августа 2016 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***28 сентября 2016 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016661065

Агент мониторинга и контроля функционирования многоагентной системы

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт систем энергетики им. Л.А. Мелентьева Сибирского отделения Российской академии наук (RU)***

Авторы: ***Гальперов Василий Ильич (RU),***
Массель Людмила Васильевна (RU)

Заявка № **2016618454**
Дата поступления **03 августа 2016 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***28 сентября 2016 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016661106

Агент агрегирования

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт систем энергетики им. Л.А. Мелентьева Сибирского отделения Российской академии наук (RU)***

Авторы: ***Гальперов Василий Ильич (RU),***
Массель Людмила Васильевна (RU)

Заявка № **2016618504**
Дата поступления **03 августа 2016 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***30 сентября 2016 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016661107

Агент оценивания состояния ЭЭС

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт систем энергетики им. Л.А. Мелентьева Сибирского отделения Российской академии наук (RU)***

Авторы: ***Гальперов Василий Ильич (RU),***
Массель Людмила Васильевна (RU)

Заявка № **2016618505**
Дата поступления **03 августа 2016 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***30 сентября 2016 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016661105

Агент координации

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт систем энергетики им. Л.А. Мелентьева Сибирского отделения Российской академии наук (RU)***

Авторы: ***Гальперов Василий Ильич (RU),***
Массель Людмила Васильевна (RU)

Заявка № **2016618503**
Дата поступления **03 августа 2016 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***30 сентября 2016 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

Приложение 4

Листинг агента декомпозиции

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Agent_Decomp
{
   public class controller
   {
      public List<DatItem200> ID200;
      public List<DatItem200> ID200_1;
      public List<DatItem200> ID200_2;
      public List<DatItem200> ID200_3;
      public List<DatItem200> ID200_pmu;

      public List<DatItem300> ID300;
      public List<DatItem300> ID300_1;
      public List<DatItem300> ID300_2;
      public List<DatItem300> ID300_3;

      public List<AdrItem222> IA222;
      public List<AdrItem222> IA222_1;
      public List<AdrItem222> IA222_2;
      public List<AdrItem222> IA222_3;

      public List<AdrItem333> IA333;
      public List<AdrItem333> IA333_1;
      public List<AdrItem333> IA333_2;
      public List<AdrItem333> IA333_3;

      public List<AdrItem444> IA444;
      public List<AdrItem444> IA444_1;
      public List<AdrItem444> IA444_2;
      public List<AdrItem444> IA444_3;
      int id = 7000;
      int id2 = 5000;

      public List<Psv> psv_all;
      public List<Psv> psv_1;
      public List<Psv> psv_2;
      public List<Psv> psv_3;

      public string ocenka_directory_dat = "";
```

```
public string ocenka_directory_exe = "";
public string bazoc = "";
public string errore = "";

public float low_bound;
public float center_bound_1;
public float center_bound_2;
public float top_bound;


public controller()
{
   ID200 = new List<DatItem200>();
   ID200_1 = new List<DatItem200>();
   ID200_2 = new List<DatItem200>();
   ID200_3 = new List<DatItem200>();
   ID200_pmu = new List<DatItem200>();

   ID300 = new List<DatItem300>();
   ID300_1 = new List<DatItem300>();
   ID300_2 = new List<DatItem300>();
   ID300_3 = new List<DatItem300>();

   IA222 = new List<AdrItem222>();
   IA222_1 = new List<AdrItem222>();
   IA222_2 = new List<AdrItem222>();
   IA222_3 = new List<AdrItem222>();

   IA333 = new List<AdrItem333>();
   IA333_1 = new List<AdrItem333>();
   IA333_2 = new List<AdrItem333>();
   IA333_3 = new List<AdrItem333>();

   IA444 = new List<AdrItem444>();
   IA444_1 = new List<AdrItem444>();
   IA444_2 = new List<AdrItem444>();
   IA444_3 = new List<AdrItem444>(); ;

   psv_all = new List<Psv>();
   psv_1 = new List<Psv>();
   psv_2 = new List<Psv>();
   psv_3 = new List<Psv>();

}


//Добавление новой записи и ее разбор
public void addDat(string data)
{
   try
```

```
    {
        if (int.Parse(data.Substring(0. 4)) < 210)
        {
            ID200.Add(new DatItem200(data));
        }
        if ((int.Parse(data.Substring(0. 4)) > 210) && (int.Parse(data.Substring(0. 4)) < 310))
        {
            ID300.Add(new DatItem300(data));
        }
    }
    catch (Exception)
    {
        errore += "Парсинг 200 b 300    " + data + Environment.NewLine;
    }
}


public void addPmu(string data)
{
    try
    {
        if (int.Parse(data.Substring(0. 4)) < 210)
        {
            ID200_pmu.Add(new DatItem200(data));
        }
    }
    catch (Exception)
    {
        errore += "Парсинг PMU    " + data + Environment.NewLine;
    }
}


public void addPsv(string data)
{
    psv_all.Add(new Psv(data));
}


public void addAdr(string data)
{
    try
    {
        if (int.Parse(data.Substring(0. 4)) == 222)
        {
            IA222.Add(new AdrItem222(data));
        }
        if (int.Parse(data.Substring(0. 4))==333)
        {
            IA333.Add(new AdrItem333(data));
```

```
            }
            if (int.Parse(data.Substring(0. 4)) == 444)
            {
               IA444.Add(new AdrItem444(data));
            }

         }
         catch (Exception)
         {
            errore += "Парсинг 222. 333. 444    " + data + Environment.NewLine;
         }
      }

      public void work()
      {

         //Распределение 2** записей
         foreach (var item in ID200)
         {
            try
            {
               var t = item.nam_voltage.Trim();
               if (float.Parse(t) <= low_bound)
               {
                  ID200_1.Add(item);
               }

               if ((float.Parse(item.nam_voltage.Trim()) > center_bound_1) &&
(float.Parse(item.nam_voltage.Trim()) <= center_bound_2))
               {
                  ID200_2.Add(item);
               }

               if (float.Parse(item.nam_voltage.Trim()) >= top_bound)
               {
                  ID200_3.Add(item);
               }
            }
            catch (Exception)
            {
               errore += "Распределение 200   " + item.source  + Environment.NewLine;
            }
         }


         List<DatItem200> temp3 = new List<DatItem200>();
         List<DatItem200> temp2 = new List<DatItem200>();
         List<DatItem200> temp1 = new List<DatItem200>();
```

```
        foreach (var item in ID300)
        {

            try
            {
                if ((ID200_1.Where(c => c.number == item.nmb_strat).Count() > 0) &&
((ID200_1.Where(c => c.number == item.nmb_end).Count() > 0) || (int.Parse(item.nmb_end) ==
0)))
                {
                    ID300_1.Add(item);
                }

                if ((ID200_2.Where(c => c.number == item.nmb_strat).Count() > 0) &&
(ID200_2.Where(c => c.number == item.nmb_end).Count() > 0 || (int.Parse(item.nmb_end) ==
0)))
                {
                    ID300_2.Add(item);
                }

                if ((ID200_3.Where(c => c.number == item.nmb_strat).Count() > 0) &&
(ID200_3.Where(c => c.number == item.nmb_end).Count() > 0 || (int.Parse(item.nmb_end) ==
0)))
                {
                    ID300_3.Add(item);
                }
            }
            catch
            {
                errore += "Распределение 300   " + item.source + Environment.NewLine;
            }


            if ((ID200_3.Where(c => c.number == item.nmb_strat).Count() > 0) &&
(ID200_2.Where(c => c.number == item.nmb_end).Count() > 0))
            {
                ID300_2.Add(item);
                ID300_3.Add(item);

                var d3 = new DatItem200(ID200.Where(c => c.number ==
item.nmb_strat).First().source);
                d3.gran = true;
                temp2.Add(d3);
                temp2.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_strat)).First());
                temp2.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_end)).First());
                bazoc += item.nmb_strat + Environment.NewLine; ;
                bazoc += item.nmb_end + Environment.NewLine; ;
```

```
                var d2 =new DatItem200(ID200.Where(c => c.number ==
item.nmb_end).First().source);
                d2.gran = true;
                temp3.Add(d2);
                temp3.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_strat)).First());
                temp3.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_end)).First());

            }

            if ((ID200_2.Where(c => c.number == item.nmb_strat).Count() > 0) &&
(ID200_3.Where(c => c.number == item.nmb_end).Count() > 0))
            {
                ID300_2.Add(item);
                ID300_2.Add(item);

                var d3 = new DatItem200(ID200.Where(c => c.number ==
item.nmb_end).First().source);
                d3.gran = true;
                temp2.Add(d3);
                temp2.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_strat)).First());
                temp2.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_end)).First());

                bazoc += item.nmb_strat + Environment.NewLine; ;
                bazoc += item.nmb_end + Environment.NewLine; ;


                var d2 = new DatItem200(ID200.Where(c => c.number ==
item.nmb_strat).First().source);
                d2.gran = true;
                temp3.Add(d2);
                temp3.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_strat)).First());
                temp3.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_end)).First());

            }

            if ((ID200_2.Where(c => c.number == item.nmb_strat).Count() > 0) &&
(ID200_1.Where(c => c.number == item.nmb_end).Count() > 0))
            {
                ID300_1.Add(item);
                ID300_2.Add(item);

                var d1 = ID200.Where(c => c.number == item.nmb_strat).First();
                d1.gran = true;
```

```
                temp2.Add(d1);
                temp2.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_strat)).First());
                temp2.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_end)).First());

                bazoc += item.nmb_strat + Environment.NewLine; ;
                bazoc += item.nmb_end + Environment.NewLine; ;


                var d2 = ID200.Where(c => c.number == item.nmb_end).First();
                d2.gran = true;
                temp1.Add(d2);
                temp1.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_strat)).First());
                temp1.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_end)).First());

            }

            if ((ID200_1.Where(c => c.number == item.nmb_strat).Count() > 0) &&
(ID200_2.Where(c => c.number == item.nmb_end).Count() > 0))
            {
                ID300_1.Add(item);
                ID300_2.Add(item);

                var d1 = ID200.Where(c => c.number == item.nmb_end).First();
                d1.gran = true;
                temp2.Add(d1);
                temp2.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_strat)).First());
                temp2.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_end)).First());

                bazoc += item.nmb_strat + Environment.NewLine; ;
                bazoc += item.nmb_end + Environment.NewLine; ;


                var d2 = ID200.Where(c => c.number == item.nmb_strat).First();
                d2.gran = true;
                temp1.Add(d2);
                temp1.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_strat)).First());
                temp1.Add(ID200_pmu.Where(c => (c.code == "0202") && (c.number ==
item.nmb_end)).First());

            }
        }

        foreach (var item in temp1)
```

```
        {
            if (ID200_1.Where(c => (c.number == item.number)&&(c.code ==
item.code)).Count() > 0)
            { continue; }
            ID200_1.Add(item);
        }

        foreach (var item in temp2)
        {
            if (ID200_2.Where(c => (c.number == item.number) && (c.code ==
item.code)).Count() > 0)
            { continue; }
            ID200_2.Add(item);
        }

        foreach (var item in temp3)
        {
            if (ID200_3.Where(c => (c.number == item.number) && (c.code ==
item.code)).Count() > 0)
            { continue; }
            ID200_3.Add(item);
        }



        foreach (var item in IA222)
        {
            if (ID200_1.Where(c => c.number == item.number).Count() > 0)
            {
                var tmp = new AdrItem222(item.source);
                if (ID200_1.Where(c => c.number == item.number).First().gran)
                {
                    tmp.gran = true;
                }
                IA222_1.Add(tmp);
            }

            if (ID200_2.Where(c => c.number == item.number).Count() > 0)
            {
                var tmp = new AdrItem222(item.source);
                if (ID200_2.Where(c => c.number == item.number).First().gran)
                {
                    tmp.gran = true;
                }
                IA222_2.Add(tmp);
            }

            if (ID200_3.Where(c => c.number == item.number).Count() > 0)
            {
                var tmp = new AdrItem222(item.source);
```

```
            if (ID200_3.Where(c => c.number == item.number).First().gran)
            {
               tmp.gran = true;
            }
            IA222_3.Add(tmp);
         }
      }



      foreach (var item in IA333)
      {
         try
         {
            int str = int.Parse(item.nmb_strat);
            int end = int.Parse(item.nmb_end);

            if (ID300_1.Where(c => (int.Parse(c.nmb_strat) == str) && (int.Parse(c.nmb_end)
== end)).Count() > 0)
            {
               if (ID300_1.Where(c => (int.Parse(c.nmb_strat) == str) &&
(int.Parse(c.nmb_end) == end)).First().gran)
               { item.gran = true; }
               IA333_1.Add(item);
            }

            if (ID300_1.Where(c => (int.Parse(c.nmb_strat) == end) && (int.Parse(c.nmb_end)
== str)).Count() > 0)
            {
               if (ID300_1.Where(c => (int.Parse(c.nmb_strat) == end) &&
(int.Parse(c.nmb_end) == str)).First().gran)
               { item.gran = true; }
               IA333_1.Add(item);
            }

            if (ID300_2.Where(c => (int.Parse(c.nmb_strat) == str) && (int.Parse(c.nmb_end)
== end)).Count() > 0)
            {
               if (ID300_2.Where(c => (int.Parse(c.nmb_strat) == str) &&
(int.Parse(c.nmb_end) == end)).First().gran)
               { item.gran = true; }
               IA333_2.Add(item);
            }

            if (ID300_2.Where(c => (int.Parse(c.nmb_strat) == end) && (int.Parse(c.nmb_end)
== str)).Count() > 0)
            {
               if (ID300_2.Where(c => (int.Parse(c.nmb_strat) == end) &&
(int.Parse(c.nmb_end) == str)).First().gran)
               { item.gran = true; }
```

```
                IA333_2.Add(item);
            }

            if (ID300_3.Where(c => (int.Parse(c.nmb_strat) == str) && (int.Parse(c.nmb_end)
== end)).Count() > 0)
            {
                if (ID300_3.Where(c => (int.Parse(c.nmb_strat) == str) &&
(int.Parse(c.nmb_end) == end)).First().gran)
                { item.gran = true; }
                IA333_3.Add(item);
            }

            if (ID300_3.Where(c => (int.Parse(c.nmb_strat) == end) && (int.Parse(c.nmb_end)
== str)).Count() > 0)
            {
                if (ID300_3.Where(c => (int.Parse(c.nmb_strat) == end) &&
(int.Parse(c.nmb_end) == str)).First().gran)
                { item.gran = true; }
                IA333_3.Add(item);
            }
        }
        catch
        {
            errore += "Распределение 333   " + item.source + Environment.NewLine;
        }
    }




    foreach (var item in IA444)
    {
        try
        {
            int str = int.Parse(item.nmb_strat);
            int end = int.Parse(item.nmb_end);

            if (ID300_1.Where(c => (int.Parse(c.nmb_strat) == str) && (int.Parse(c.nmb_end)
== end)).Count() > 0)
            {

                IA444_1.Add(item);
            }

            if (ID300_1.Where(c => (int.Parse(c.nmb_strat) == end) && (int.Parse(c.nmb_end)
== str)).Count() > 0)
            {

                IA444_1.Add(item);
            }
```

```
                if (ID300_2.Where(c => (int.Parse(c.nmb_strat) == str) && (int.Parse(c.nmb_end)
== end)).Count() > 0)
                {

                    IA444_2.Add(item);
                }

                if (ID300_2.Where(c => (int.Parse(c.nmb_strat) == end) && (int.Parse(c.nmb_end)
== str)).Count() > 0)
                {

                    IA444_2.Add(item);
                }

                if (ID300_3.Where(c => (int.Parse(c.nmb_strat) == str) && (int.Parse(c.nmb_end)
== end)).Count() > 0)
                {
                    IA444_3.Add(item);
                }

                if (ID300_3.Where(c => (int.Parse(c.nmb_strat) == end) && (int.Parse(c.nmb_end)
== str)).Count() > 0)
                {

                    IA444_3.Add(item);
                }
            }
            catch { errore += "Распределение 444 " + item.source + Environment.NewLine; }
        }


        foreach (var item in ID200_1)
        {
            if(item.code=="0202")
            {
                psvEdit(item. 1);
            }
        }

        foreach (var item in ID200_2)
        {
            if (item.code == "0202")
            {
                psvEdit(item. 2);
            }
        }
        foreach (var item in ID200_3)
        {
            if (item.code == "0202")
```

```
            {
               psvEdit(item. 3);
            }
         }

      }



      public void psvEdit(DatItem200 dt. int to)
      {

         Psv tmp = new Psv();

        //  Psv fake_tmp = new Psv(psv_all.Where(c => c.number ==
adr_tmp.adresUzNapr).First());

         Psv fake_tmp = new Psv();

            fake_tmp.number = " " + id.ToString();
            fake_tmp.napr = "    0";
            fake_tmp.mnojit = "    0.01";
            fake_tmp.refer = " "+id2.ToString();
            fake_tmp.other = "    0    0.00    0     .00    0     .00";
            fake_tmp.toSource();

            tmp.number = " " + id2.ToString();
            tmp.refer = "    0";
            tmp.mnojit = "    0.00";
            tmp.other = fake_tmp.other;
            var vol = float.Parse(dt.nam_voltage);
            vol = vol * 100;
            tmp.napr = vol.ToString().Substring(0.5);
            tmp.toSource();




         switch (to)
         {
            case 1: psv_1.Add(fake_tmp);
                  psv_1.Add(tmp);
                  IA222_1.Where(c => c.number == dt.number).First().adresUzNapr="
"+id.ToString();

               break;
            case 2: psv_2.Add(fake_tmp);
                  psv_2.Add(tmp);
```

```
                IA222_2.Where(c => c.number == dt.number).First().adresUzNapr = "    " +
id.ToString();

                break;
            case 3: psv_3.Add(fake_tmp);
                psv_3.Add(tmp);
                IA222_3.Where(c => c.number == dt.number).First().adresUzNapr = "    " +
id.ToString();

                break;
        }
          id++;
         id2++;

    }



    public string return_dat_1()
    {
        string ret="";
        foreach (var item in ID200_1)
        {
            ret += item.toString() + Environment.NewLine;
        }
        foreach (var item in ID300_1)
        {
            ret += item.toString() + Environment.NewLine;
        }
        return ret;
    }


    public string return_dat_2()
    {
        string ret = "";
        foreach (var item in ID200_2)
        {
            ret += item.toString() +Environment.NewLine;
        }
        foreach (var item in ID300_2)
        {
            ret += item.toString() + Environment.NewLine;
        }
        return ret;
    }


    public string return_dat_3()
    {
```

```
    string ret = "";
    foreach (var item in ID200_3)
    {
        ret += item.toString() + Environment.NewLine;
    }
    foreach (var item in ID300_3)
    {
        ret += item.toString() + Environment.NewLine;
    }
    return ret;
}


public string return_adr_1()
{
    string ret = "";
    foreach (var item in IA222_1)
    {
        ret += item.toString() + Environment.NewLine;
    }
    foreach (var item in IA333_1)
    {
        ret += item.toString() + Environment.NewLine;
    }
    foreach (var item in IA444_1)
    {
        ret += item.toString() + Environment.NewLine;
    }
    return ret;
}

public string return_adr_2()
{
    string ret = "";
    foreach (var item in IA222_2)
    {
        ret += item.toString() + Environment.NewLine;
    }
    foreach (var item in IA333_2)
    {
        ret += item.toString() + Environment.NewLine;
    }
    foreach (var item in IA444_2)
    {
        ret += item.toString() + Environment.NewLine;
    }
    return ret;
}

public string return_adr_3()
```

```
{
    string ret = "";
    foreach (var item in IA222_3)
    {
        ret += item.toString() + Environment.NewLine;
    }
    foreach (var item in IA333_3)
    {
        ret += item.toString() + Environment.NewLine;
    }
    foreach (var item in IA444_3)
    {
        ret += item.toString() + Environment.NewLine;
    }

    return ret;
}


public string return_psv_1()
{
    string ret = "";
    foreach (var item in psv_all)
    {
        ret += item.source + Environment.NewLine;
    }
    foreach (var item in psv_1)
    {
        ret += item.source + Environment.NewLine;
    }
    return ret;
}

public string return_psv_2()
{
    string ret = "";
    foreach (var item in psv_all)
    {
        ret += item.source + Environment.NewLine;
    }
    foreach (var item in psv_2)
    {
        ret += item.source + Environment.NewLine;
    }
    return ret;
}

public string return_psv_3()
{
    string ret = "";
```

```
        foreach (var item in psv_all)
        {
            ret+=item.source + Environment.NewLine;
        }
        foreach (var item in psv_3)
        {
            ret += item.source + Environment.NewLine;
        }
        return ret;
    }

    public static string getStr(string st. int character)
    {
        string ret="";
        for (int i = 0; i < character – st.Length ; i++)
        {
            ret += " ";
        }
        ret += st;

        return ret;

    }
}
                                                    }
```