Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Цириков Семен Алексеевич

Построение модели и анализатора для ERP-системы

Отчёт по производственной практике

Научный руководитель:
д.ф.-м.н. Терехов А.Н.

Консультант:
Ген. директор ЗАО «Радион» Вишневецкий В.И.

Санкт-Петербург
2020

Оглавление

1. Введение

EPR (Enterprise Resource Planning, планирование ресурсов предприятия) – это организационная стратегия интеграции производства и операций, управления трудовыми ресурсами, финансового менеджмента и управления активами, ориентированная на непрерывную обработку информации о ресурсах предприятия посредством специализированного пакета прикладного программного обеспечения, предоставляющая общую модель данных и процессов для всех сфер деятельности предприятия. ERP-системой называют конкретный программный пакет, реализующий стратегию ERP.

Концепция ERP была сформулирована [10] в 1990-ом году, и к середине 1990-ых начали появляться первые тиражируемые ERP-системы. К настоящему времени внедрение ERP-системы считается фактически необходимым условием для функционирования конкурентоспособной компании, так как бизнес получает от использования системы следующие преимущества:

1. Доступность и согласованность данных
2. Мониторинг работы сотрудников
3. Снижение роли человеческого фактора
4. Единая инфраструктура инструментов учёта
5. Автоматизация рутинных задач

Хотя уже существует большое количество прикладных решений на рынке ERP-систем, исследования в этой области продолжаются с целями оптимизации и обеспечения надлежащего качества масштабируемости информационных систем при увеличении объёмов данных в обороте. В ходе данной работы планируется спроектировать модель такой системы и реализовать для неё анализатор с целью дальнейшей проверки гипотез об алгоритмах и моделях данных, представляющих интерес как потенциально превосходящие по эффективности существующие решения.

2. Постановка задачи

Цель данной работы – спроектировать модель для ERP-системы и для некотрых модулей разработать анализатор, транслирующий исходный код в представление созданной модели.

Для достижения указанной цели были поставлены следующие задачи:

- Составить обзор рынка существующих ERP-систем:
  - Глобально (в мире)
  - Локально (в России)
- Спроектировать архитектуру модели:
  - Диаграмму компонентов
  - Диаграмму прецедентов
  - Диаграмму деятельности
  - Диаграмму классов
  - Диаграмму развёртывания
- Реализовать синтаксический анализатор для модели по модулям:
  - Внутренний язык программирования
  - Внутренний язык запросов

3. Обзор

В данной секции сперва приведена статистика использования различных ERP-систем в России и в мире, а затем обосновывается выбор конкретного решения в качестве опорного при проектировании модели и реализации её анализатора.

3.1. Статистика

Согласно исследованиям (мир[1], Россия[2]), выборка сведений из которых приведена в таб. 1, мировым лидером по внедрению ERP-систем в 2019-ом году является SAP, в то время как в России наибольшую популярность имеет 1С.

Мир		Россия	
SAP	7%	1C	50%
Oracle	3%	SAP	32%
Sage	2.5%	Microsoft	6%

Таблица 1: Топ-3 ERP-поставщика по проценту их рыночной доли

Такое отличие не является аномальным, так как во многих регионах мира в силу исторического [8] развития ERP появлялись локальные системы, которые и захватили большую часть рыночной доли. Причиной этого является то, что ERP-системы в значительной мере зависят от законодательства страны, в которой их собираются применять (специфика налогообложения, требования к аудиту), что не позволило на момент зарождения ERP-систем создать единый продукт и быстро локализовать его для рынков разных стран.

3.2. 1C:Enterprise

В силу популярности 1C:Enterprise на отечественном рынке ERP-систем существует большой объём доступной документации и обучающих материалов, что облегчает понимание устройства системы. В то же время

[1] https://www.appsruntheworld.com/top-10-erp-software-vendors-and-market-forecast/

[2] https://drgroup.ru/Analiz-rynka-ERP-sistem-v-Rossii.html

создание нового продукта с отличающимся устройством привело бы к дополнительным издержкам на осуществление миграции на новую платформу со стороны пользователей и разработчиков, что, в свою очередь, стало бы препятствием к популяризации созданного решения. Исходя из приведённых соображений было решено спроектировать модель, которая была бы совместима с 1C:Enterprise, что в перспективе, после реализации анализатора, позволит использовать примеры программ, конфигураций модулей и информационных баз, находящиеся в открытом доступе, в качестве тестовых входных данных.

4. Реализация

4.1. Диаграмма компонентов

При анализе предметной области была поставлена задача учесть архитектуру существующих программ, основанных на ERP-системе 1С, в целях обеспечения совместимости проекта. Диаграмма компонентов [3], созданная для описания единого языка модели и отражающая связи объектов, которыми оперирует система, представлена на Рис.1.

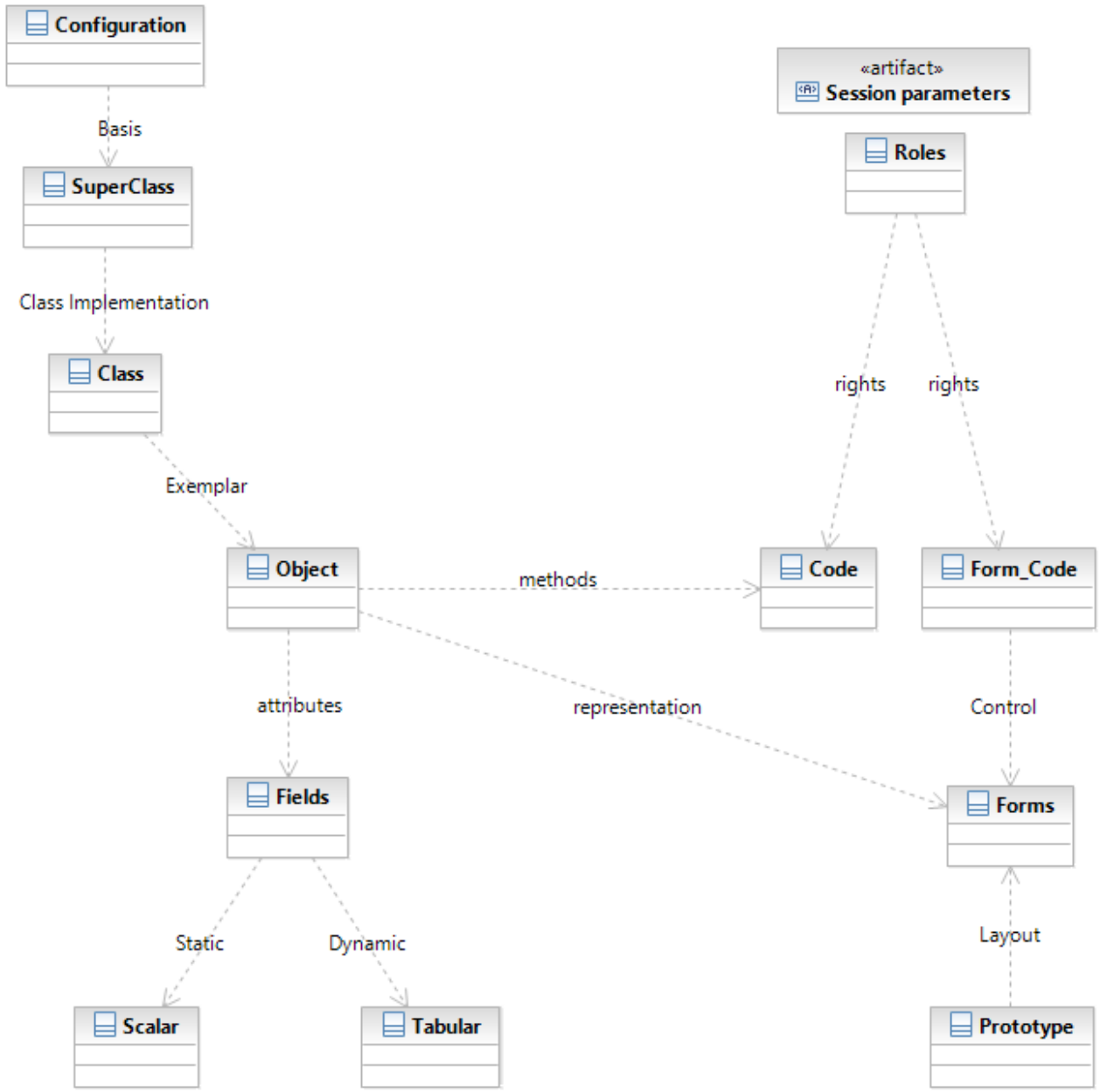

Рис. 1: Диаграмма компонентов ERP-системы

На диаграмме под конфигурацией понимается сама информационная система, которая состоит из предопределённых суперклассов. От суперклассов наследуются конкретные классы, расширяя базовую функциональность посредством декларации новых компонентов. Экземпляром класса является объект, который содержит данные (поля), разделённые на статические (скалярные) и динамические (табличные) части, представление (формы), которые в данной модели являются прикладным интерфейсом взаимодействия с пользователем системы и могут наследовать разметку от прототипов, определённых в системе или разработчиком, а также кода объекта и форм, которые определяют поведение объекта при определённых событиях, таких как инициализация объекта, изменением его полей или других триггерах. Исполнение кода регулируется механизмом ролей, реализующим мандатное управление доступом: т.е. при входе пользователя в систему, ему сопоставляется определённая роль и сохраняется в артефакте, названном параметры сессии; далее при попытке выполнить какое-либо действие, система сопоставляет уровень доступа, присвоенный его роли, с требуемым для исполнения данного кода, после чего либо отказывает в проведении действия, либо производит его.

4.2. Диаграмма прецедентов

Назначение диаграммы прецедентов [7], также известной как диаграммы вариантов использования, состоит в описании системы на концептуальном уровне, отражении отношений между акторами (внешние сущности, использующие систему, например: люди, другие программные системы) и прецедентами (возможностями моделируемой системы, её функциональностью). Диаграмма представлена на Рис. 2.

На приведённой диаграмме определены два основных класса пользователей, которые предусматриваются в системе. Первый из них – это администратор сети, чья роль предполагает права доступа, связанные не с оперированием объектами предметной области, а топологией распределённой сети узлов, которые обмениваются данными и кодом между собой. Так для него предусмотрены возможности изменения прав доступа для

Рис. 2: Диаграмма прецедентов

различных ролей, а также переопределение действующей схемы данных, которыми будут пользоваться другие субъекты.

Второй класс пользователей – операторы – это сущности, которые в рамках определённой схемы и их роли могут вносить, изменять или удалять данные из системы, запрашивать отчёты по операциям, произведённым за выбранный период или создавать и изменять автоматизированные задачи, которые в ходе своей работы могут взаимодействовать с теми же данными и совершать те же операции, которые доступны по роли пользователя.

Оба класса пользователей перед взаимодействием с системой в обязательном порядке должны пройти процедуру авторизации для определения доступных им ролей. Следует учитывать, что сами роли являются частью предметной области, из-за чего возможно существование ролей, которые представляют собой неполный набор возможностей из перечисленных для одного класса или сочетают в себе функции обоих классов.

4.3. Диаграмма деятельности

Диаграмма прецедентов определяет цели, которых хочет добиться актор при использовании системы. Способы достижения этих целей раскрываются в набор сценариев или бизнес-процессов, которые можно модели-

ровать с помощью диаграммы деятельности [4]. Диаграмма для модели нашей системы представлена на Рис.3.

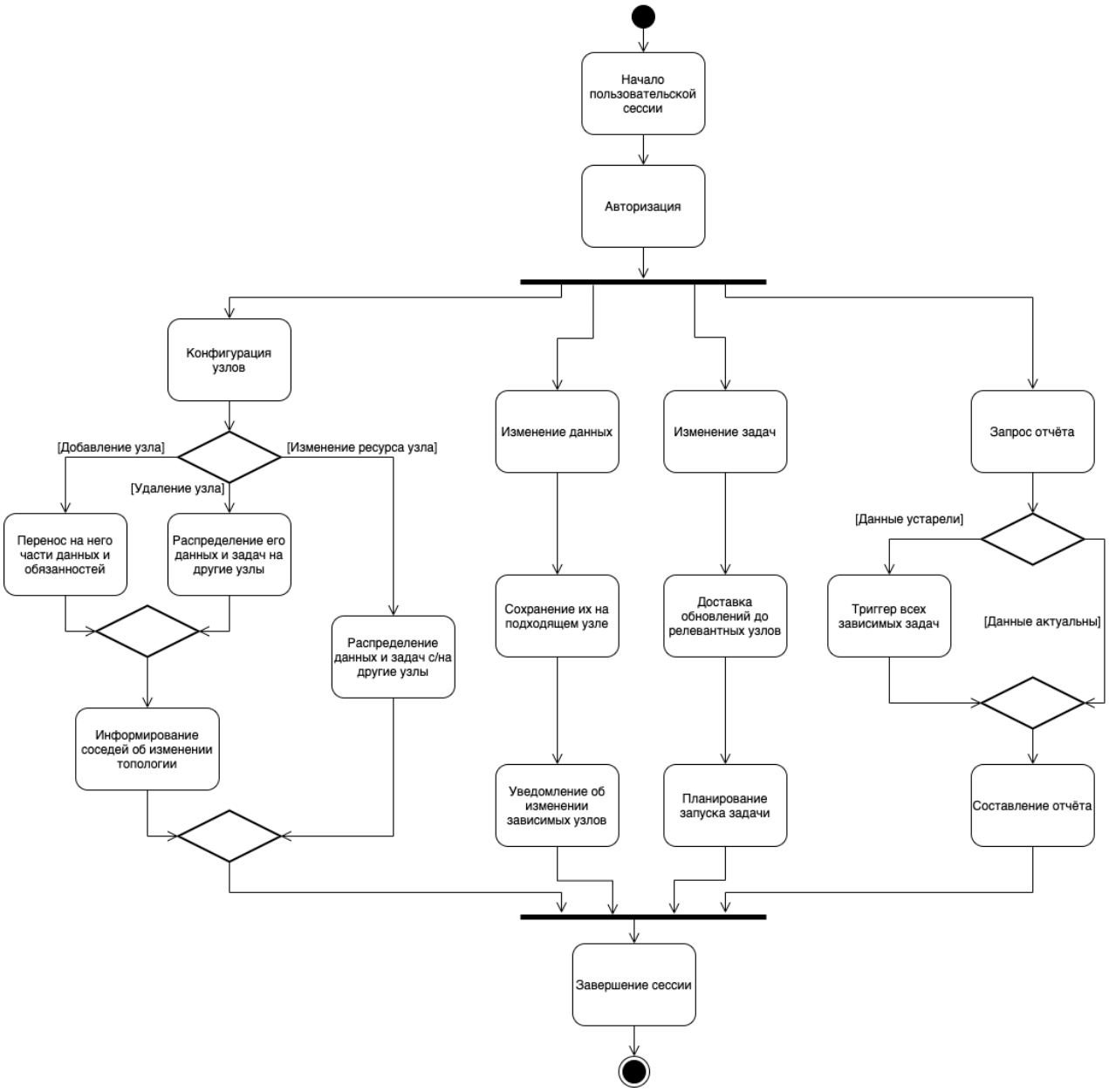

Рис. 3: Диаграмма деятельности

Как было замечено ещё в диаграмме прецедентов, каждая сессия начинается с авторизации, чтобы сопоставить пользователя с доступными ему ролями. Затем в левой колонке описан один из возможных сценариев для класса администратора сети, когда он изменяет конфигурацию узлов, что приводит к перераспределению нагрузки между ним и его соседями. Сценарии изменения схемы данных и ролей не представлены в силу их тривиальности.

Следующие три ветви диаграммы моделируют сценарии для класса оператора. Рассмотрим самую правую из них, в которой рассмотрен запрос отчёта за произвольный период времени. На момент обращения к отчёту он может быть не сформирован, если очередь задач ещё не дошла до этого и данные, необходимые для него, устарели на данном узле – в таком случае будет запущен триггер всех зависимых задач с повышенным приоритетом, что актуализировать данные. Затем отчёт будет отправлен тому узлу, который запросил его.

Сессия заканчивается по воле пользователя или с принуждения администратора сети, в том числе с помощью возможных лимитов времени сессии, определённых им в виде автоматизированных задач.

4.4. Диаграмма классов

Диаграмма классов [6] призвана продемонстрировать статическую структуру декларативных элементов в системе, их отношения, поля и методы, в более приближенном к реализации понимании, чем к предметной области. Этот важный артифакт проектирования вносит ясность для будущих разработчиков системы, как программировать модель, чтобы обеспечить связи между необходимыми объектами и что следует изолировать от внешнего доступа. Он также расширяет понимание общего языка модели, который начал вводиться с диаграммы компонентов. Полученная диаграмма классов приведена на Рис.4.

Диаграмма проясняет, что абстракция Роли – это множество объектов, которые определяют, разрешён ли Доступ к объекту системы для заданного Пользователя. Доводится до сведения общая природа таких классов сущностей, как Оператор и Администратор, которые расширяют общее понятие Пользователя, обладающего Ролями, а Маршрутизатор и Исполнитель – понятие Узла, обладающего Ресурсами. Явно обозначается, что между Объектом и Схемой имеется логическая зависимость, также как и у Отчёта с Представлением и Задачей.

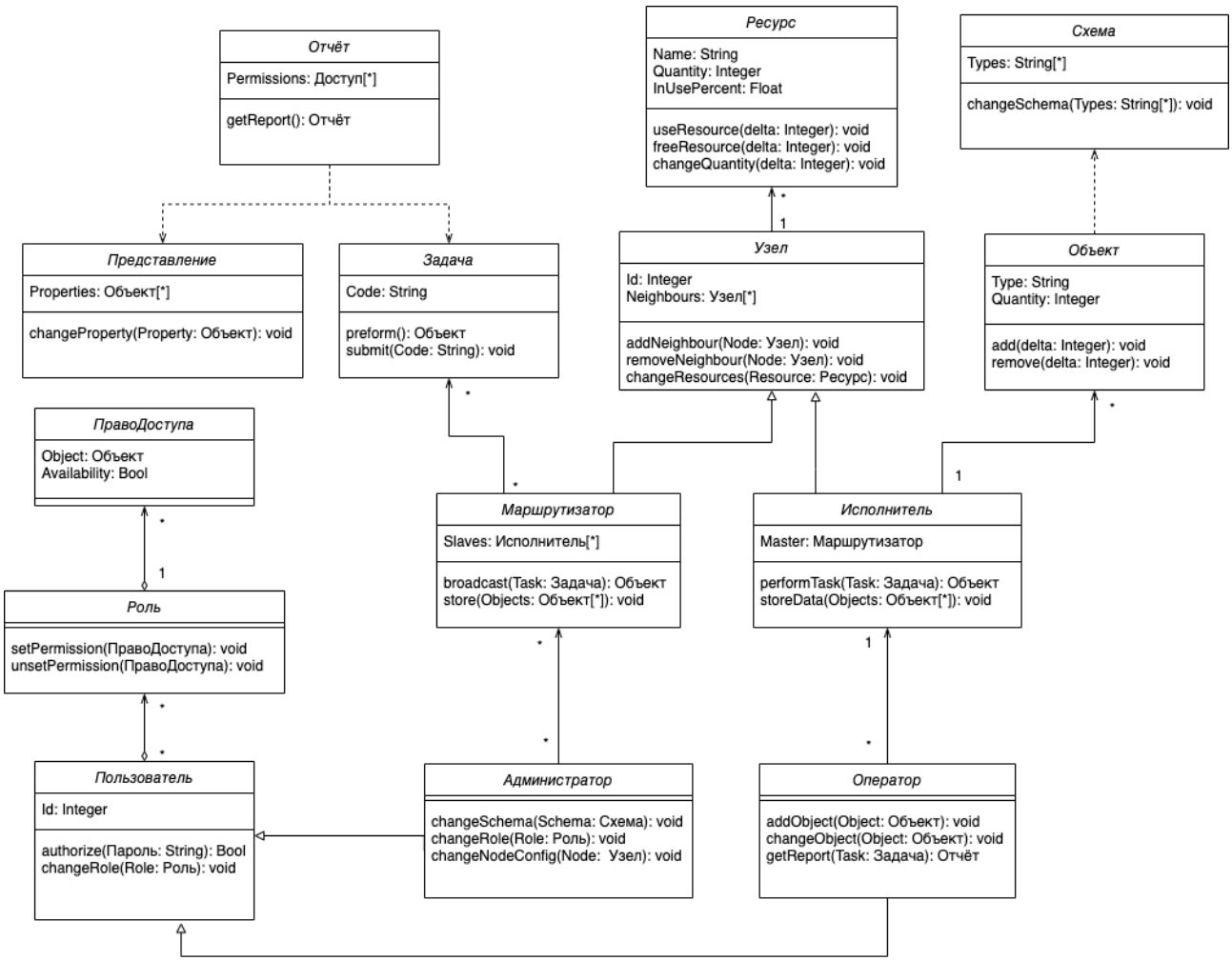

Рис. 4: Диаграмма классов

4.5. Диаграмма развёртывания

Для понимания идеи распределения нагрузки вычислений между независимыми вычислителями, которая лежит в основе этого проекта, можно моделировать распределение программных сущностей между устройствами, задействованными в системе. Для проектирования этого распределения в кластере существует диаграмма развёртывания [5]. На ней отдельным устройством (возможно, виртуальным) обозначены логически зависимые модули, а также указаны их обязанности. Диаграмма развёртывания для нашей модели изображена на Рис.5.

В данном случае подразумевается кластер, где каждая группа узлов-рабочих соединена только с одним узлом-диспетчером (прежде названный маршрутизатором), в то время как диспетчер, помимо своих рабочих, со-

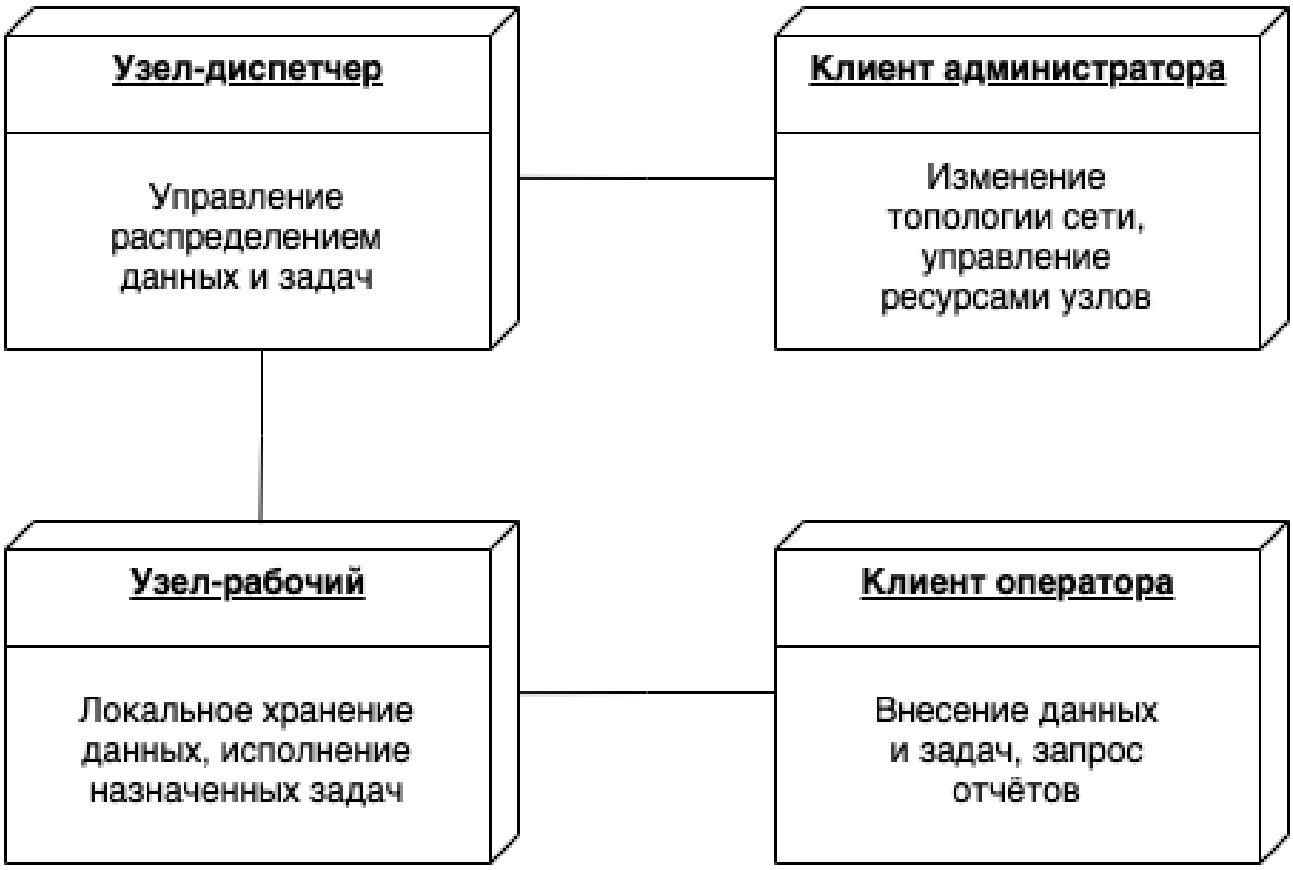

Рис. 5: Диаграмма развёртывания

единён ещё с другими диспетчерами. Клиент оператора представляет собой виртуальный клиент для узла-рабочего, а клиент администратора – виртуальный или физический узел, который может передавать сообщения напрямую диспетчерам или через них. Рабочие узлы хранят данные и вычисляют задачи, которые получают от диспетчеров, обмениваются данными только с диспетчером. Диспетчер может как хранить задачи у себя, так и получать их в процессе вещания от других диспетчеров, распространять дальше и применять к своим данным, если они соответствуют спецификации.

4.6. Синтаксический анализатор

Реализацию анализатора для модели было решено начать с синтаксического анализатора для внутреннего языка программирования системы. Для этого в соответствие с документацией крайней актуальной версии

1C:Enterprise [3] был выписан список ключевых слов языка и описана грамматика в формате ANTLR [2], с чем можно ознакомиться в открытом GitHub-репозитории [1]. При обзоре существующих решений для подобных задач было обнаружено, что для языка запросов уже существует синтаксический анализатор с открытым исходным кодом, также сделанным в формате ANTLR и опубликованный в Github-репозитории [9] под лицензией GPL-3.0.

Реализация анализатора через ANTLR представляется в данный момент прототипом, однако в дальнейшем из соображений производительности планируется выбрать конкретную архитектуру и написать компилятор по аналогии с компилятором RuC на платформу MIPS [11].

[3] `https://its.1c.ru/db/v838doc/browse/13/-1/5`

5. Заключение

Достигнуты следующие результаты:

1. Проведён и описан обзор рынка существующих ERP-систем
2. Архитектура модели спроектирована в виде UML-диаграмм
3. Разработан парсер языка программирования и запросов 1С

Список литературы

[1] 1C_analyser. — https://github.com/SimonTsirikov/1C_analyser.

[2] ANTLR v4. — https://github.com/antlr/antlr4.

[3] Bell Donald. Uml basics: The component diagram // IBM Global Services. — 2004.

[4] Geambaşu Cristina Venera. BPMN vs. UML activity diagram for business process modeling // Accounting and Management Information Systems. — 2012. — Vol. 11, no. 4. — P. 637–651.

[5] Lee Sunguk. Unified Modeling Language (UML) for Database Systems and Computer Applications // International Journal of Database Theory and Application. — 2012. — Vol. 5, no. 1. — P. 157–164.

[6] McGill Matthew John. UML Class Diagram Syntax: An Empirical Study of Comprehension. — 2001.

[7] On well-formedness rules for UML use case diagram / Noraini Ibrahim, Rosziati Ibrahim, Mohd Zainuri Saringat et al. // International Conference on Web Information Systems and Mining / Springer. — 2010. — P. 432–439.

[8] Robert Jacobs F., 'Ted' Weston F.C. Enterprise resource planning (ERP)—A brief history // Journal of Operations Management. — 2007. — Vol. 25, no. 2. — P. 357 – 363. — Access mode: http://www.sciencedirect.com/science/article/pii/S0272696306001355.

[9] SDBL parser. — https://github.com/1c-syntax/bsl-parser.

[10] Wylie Lee. A vision of next generation MRP II // Gartner Group. — 1990. — Vol. 40.

[11] Болотов Сергей Сергеевич. Разработка компилятора для языка РуСи на платформу MIPS. — 2016.