

Казанский (Приволжский) федеральный университет

На правах рукописи  
УДК 004.8

Тощев Александр Сергеевич

**ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА ПОВЫШЕНИЯ  
ЭФФЕКТИВНОСТИ ИТ-СЛУЖБЫ ПРЕДПРИЯТИЯ**

Специальность 05.13.11 —  
«Математическое и программное обеспечение вычислительных машин,  
комплексов и компьютерных сетей»

Диссертация на соискание учёной степени  
кандидата технических наук

Научный руководитель:  
доктор физико-математических наук, профессор,  
заслуженный деятель науки РТ,  
зав. кафедрой дифференциальных уравнений  
Института математики и механики им. Н.И. Лобачевского  
Казанского (Приволжского) федерального университета  
Елизаров Александр Михайлович

Казань — 2016

## Оглавление

	Стр.
<b>Введение . . . . .</b>	<b>5</b>
<b>Глава 1. Интеллектуальные системы регистрации и анализа</b>	
<b>проблемных ситуаций, возникающих в ИТ-инфраструктуре</b>	
<b>предприятия . . . . .</b>	<b>13</b>
1.1 Обзор исследований в области интеллектуальных систем	
регистрации и анализа проблемных ситуаций . . . . .	13
1.2 Сравнительный анализ систем регистрации и устранения	
проблемных ситуаций . . . . .	19
1.3 Сравнительный анализ методов и комплексов обработки текстов	
на естественном языке . . . . .	22
1.3.1 Обработка эталонных текстов . . . . .	22
1.3.2 Исправление ошибок первого и второго типов . . . . .	24
1.3.3 Сравнение средств обработки русского и английского языков	25
1.4 Выводы по главе 1 . . . . .	27
<b>Глава 2. Модель интеллектуальной системы принятия решений для</b>	
<b>регистрации и анализа проблемных ситуаций в</b>	
<b>ИТ-инфраструктуре предприятия . . . . .</b>	<b>29</b>
2.1 Построение модели Menta 0.1 с использованием деревьев	
принятия решений . . . . .	29
2.1.1 База знаний на основе OWL . . . . .	30
2.1.2 Основные компоненты модели . . . . .	32
2.2 Модель Menta 0.3, построенная с использованием генетических	
алгоритмов . . . . .	33
2.2.1 Основные компоненты модели . . . . .	33
2.2.2 База знаний на основе графов . . . . .	35
2.3 Модель TU 1.0, основанная на модели мышления Марвина Мински	36
2.3.1 Особенности модели мышления . . . . .	36
2.3.2 Основные компоненты модели . . . . .	38
2.4 Выводы по главе 2 . . . . .	41

<b>Глава 3. Реализация модели TU 1.0 для системы интеллектуальной регистрации и устранения проблемных ситуаций . . . . .</b>	<b>43</b>
3.1 Архитектура системы . . . . .	43
3.1.1 Компоненты системы . . . . .	46
3.1.2 Компонент WebService . . . . .	49
3.1.3 Компонент CoreService.ThinkingLifeCycle . . . . .	51
3.1.4 Компоненты $T^3$ . . . . .	60
3.1.5 Вспомогательные компоненты . . . . .	73
3.2 Модель данных TU Knowledge . . . . .	77
3.3 Прототип системы . . . . .	82
3.4 Выводы по главе 3 . . . . .	85
<b>Глава 4. Экспериментальные исследования эффективности работы модели TU . . . . .</b>	<b>86</b>
4.1 Экспериментальные данные . . . . .	86
4.2 Оценка эффективности . . . . .	87
4.3 Результаты экспериментов . . . . .	88
4.4 Выводы по главе 4 . . . . .	90
<b>Заключение . . . . .</b>	<b>91</b>
<b>Список сокращений и условных обозначений . . . . .</b>	<b>93</b>
<b>Словарь терминов . . . . .</b>	<b>94</b>
<b>Список литературы . . . . .</b>	<b>95</b>
<b>Список иллюстраций . . . . .</b>	<b>106</b>
<b>Список таблиц . . . . .</b>	<b>108</b>
<b>Приложение А. Интерфейсная модель . . . . .</b>	<b>109</b>
<b>Приложение Б. Описание модуля Goal (Цель) и Action (Действия) . . . . .</b>	<b>111</b>
<b>Приложение В. Рецепты решений . . . . .</b>	<b>114</b>
<b>Приложение Г. Экспериментальные данные . . . . .</b>	<b>116</b>

<b>Приложение Д. Свидетельство о регистрации . . . . .</b>	<b>121</b>
<b>Приложение Е. Акт о внедрении . . . . .</b>	<b>122</b>

## Введение

В настоящее время все более популярным и распространенным становится процесс передачи функций поддержки информационной инфраструктуры (далее — ИТ-инфраструктуры) предприятия какой-либо внешней компании (см., например, [1]). Это явление стало называться «ИТ-аутсорсинг» (от анг. "out source" — вне источника). С развитием рынка информационных систем компаниям становится невыгодно держать свой штат службы поддержки, и они отдают эти функции сторонней компании (см. [2]). В некоторых случаях передаются все функции поддержки пользователей: будь-то заявка на ремонт компьютера или же информационный запрос, возникающий из-за простого незнания внутренних процессов компании. В результате создается единая точка входа для пользователей, поддерживаемая сторонней компанией [3]. Обобщая, можно сказать, что на аутсорсинг передают все, что возможно: управление персоналом, уборку помещений, обеспечение питанием, разработку программного обеспечения (далее — ПО) (см., например, [4]) и т. д.

В некоторых областях, например, в области информационных технологий (ИТ) за счет аутсорсинга экономия средств предприятия достигает 30% (по данным Gartner [5]). Из-за возросшей популярности бизнеса по аутсорсингу именно в ИТ-области и появления большого количества компаний возникла сильная конкуренция [6], что привело к снижению цен на услуги и потребовало сокращения издержек компаний. Для поиска путей оптимизации издержек было необходимо применение методов системного анализа для решения сложившихся проблем [7]. Также было отмечено падение рентабельности бизнеса как минимум для малых компаний [8], [2]. В контексте оптимизации издержек в настоящей диссертации рассматриваются модель области, модель системы и ее реализация, которая повышает эффективность работы специалиста технической поддержки (далее специалист) путем частичной (в некоторых случаях, полной) автоматизации обработки инцидентов (случаев, происшествий) [9], начиная с разбора запросов, сформулированных на естественном языке, и заканчивая применением найденного решения.

Главным требованием к системе повышения эффективности ИТ-службы предприятия является замена части функций, которые сейчас выполняют специалисты:

1. Обработка запросов на естественном языке — эта функция широко вос требована и в системах анализа проблем пользователя с построением статистики «Удовлетворенность пользователя программным продуктом» [10]. Общее понимание проблемы зависит от понимания языка, на котором общаются специалисты;
2. Возможность обучения. Такая возможность системы позволяет упростить ее эксплуатацию и расширение. По данным исследования [11], возможность обучения очень важна для любой интеллектуальной системы, включая системы управления роботами. Обучение обеспечивает системе большие гибкость и универсальность;
3. Общение со специалистом. Поддержание диалога (коммуникации) — необходимое условие для обучения. Кроме того, социальная функция — неотъемлемая часть интеллектуальных систем (см., например, [12]);
4. Проведение логических рассуждений (возможность размышлять): аналогия, дедукция, индукция — умение обобщить решение одной проблемы и, экстраполируя его, применить для решения других. Иными словами, это возможность для системы принять правильное решение. Например, принятие решений широко используется в интеллектуальных системах управления производством [13].

Интерес к области интеллектуальных систем обработки информации можно, в частности, оценить как количество публикаций за последние годы, процитированных в базе данных Scopus, — с 2004 года в среднем оно составило около 1010 в год.

На данный момент времени многие компании ведут в различных областях разработку подобных систем, обладающих свойствами, описанными выше. Системы такого класса также называются *вопросно-ответными*. Примером является набирающая популярность IBM Watson [14], [15] (которая является коммерческой и закрытой, информации о ее внутреннем устройстве мало). Другой пример — компания HP использует результаты исследования [16] для автоматического определения проблем и степени удовлетворенности пользователей из отчетов об использовании программного обеспечения. Также эта компания работает над автоматическим решением проблем (как описано выше).

В настоящей диссертации представлены результаты и апробации создания вопросно-ответной системы на основе исследования целевой области (удаленная

поддержка информационной инфраструктуры предприятия) и построения модели системы. Акцент был сделан на создании интеллектуальной системы для решения широкого круга проблем.

Следует отметить, что большинство проблем, которые решает удаленная служба поддержки информационной инфраструктуры предприятия, носит достаточно тривиальный характер (по данным компании ОАО «АйСиЭл КПО-ВС (г. Казань)»): установить приложение; переустановить приложение; решить проблему с доступом к тому или иному ресурсу. Названные проблемы решают специалисты технической поддержки, которая обычно делится на несколько линий по уровню умения специалистов. Каждая линия поддержки представлена своим классом специалистов. В среднем команда, обслуживающая одного заказчика, насчитывает около 60 человек. Как показывают исследования, решение части задач может быть автоматизировано. Если это будет сделано, специалисты получат дополнительное время для решения более сложных задач.

### **Общая характеристика диссертации**

**Целью** диссертации является разработка интеллектуальной системы повышения эффективности деятельности ИТ-службы предприятия (ИТ — информационные технологии).

**Область исследования** — разработка систем управления базами данных и знаний.

**Предметом исследования** является процесс регистрации и устранения проблемных ситуаций, возникающих в ИТ-инфраструктуре предприятия.

**Методы исследования** — теоретические методы: имитационное моделирование, теория баз знаний в ИИ; специальные методы: системное моделирование; экспериментальные методы: метод наблюдений, проведение экспериментов.

Для достижения поставленной цели были решены следующие **задачи**:

1. Провести анализ систем управления базами знаний в области поддержки информационной инфраструктуры предприятия;
2. Разработать и построить модель проблемно-ориентированной системы управления базой знаний для принятия решений и оптимизации процесса регистрации, анализа и обработки запросов пользователей в области обслуживания информационной инфраструктуры предприятия;

3. На основе построенной модели разработать архитектуру и создать прототип интеллектуальной системы повышения эффективности деятельности ИТ-службы предприятия;
4. Провести апробацию прототипа на тестовых данных.

**Основные положения, выносимые на защиту:**

1. Результаты анализа систем управления базами знаний в области поддержки ИТ-инфраструктуры предприятия;
2. Построенная модель проблемно-ориентированной системы управления базой знаний и оптимизации процессов обработки запросов пользователей в области обслуживания ИТ-инфраструктуры предприятия;
3. Созданный прототип программной реализации модели проблемно-ориентированной системы управления базой знаний и оптимизации обработки запросов пользователей в области обслуживания ИТ-инфраструктуры предприятия;
4. Результаты апробации прототипа проблемно-ориентированной системы управления на контрольных примерах.

**Научная новизна** проведенного исследования состоит в следующем:

1. На основе обобщения модели мышления, разработанной М. Мински, создана имитационная модель проблемно-ориентированной системы управления, принятия решений в области обслуживания ИТ-инфраструктуры предприятия;
2. Исследованы возможности использования моделей мышления применительно к области обслуживания информационной инфраструктуры предприятия;
3. Представлены новая схема данных и оригинальный способ хранения данных для построенной модели мышления, эффективный по сравнению со стандартными способами хранения (такими, например, как реляционные базы данных);
4. На основе построенного обобщения модели мышления Мински созданы архитектура системы обслуживания информационной инфраструктуры предприятия и программный прототип этой системы.

**Практическая значимость.** Система, разработанная в рамках данной диссертации, имеет значимый практический характер. Идея работы зародилась под влиянием производственных проблем в ИТ-отрасли, с которыми автор сталки-

вался ежедневно в процессе разрешения различных инцидентов, возникающих в деятельности службы технической поддержки ОАО «АйСиЭл КПО-ВС (г. Казань)» — одном из крупнейших системообразующих предприятий ИТ-отрасли Республики Татарстан. Поэтому было необходимо выработать глубокое понимание конкретной предметной области, чтобы выбрать приемлемое программное решение, получившее практическое применение при организации информационной поддержки ИТ-инфраструктуры конкретного предприятия.

**Достоверность** полученных научных результатов и выработанных практических рекомендаций базируется на корректной постановке общих и частных рассматриваемых задач, использовании известных фундаментальных теоретических положений, достаточном объёме данных, использованных при статистическом моделировании, и широком экспериментальном материале, использованном для численных оценок достижимых качественных показателей.

Исследования, проведенные в диссертации, соответствуют паспорту специальности 05.13.11 — Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей, сопоставление приведено в таблице 1.

Таблица 1 — Сопоставление направлений исследований предусмотренных специальностью 05.13.11, и результатов, полученных в диссертации

Направление исследования	Результат работы
Языки программирования и системы программирования, семантика программ	Разработана семантическая модель организации хранения знаний
Системы управления базами данных и знаний	Разработан прототип Thinking Understanding (TU) системы хранения знаний и принятия решений в сфере поддержки ИТ-инфраструктуры предприятия, который был испытан на модельных данных

**Таблица 1 – продолжение**

<b>Направление исследования</b>	<b>Результат работы</b>
Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования	Разработан метод параллельной обработки экспертной информации с возможностью обучения при помощи прототипа ТУ

**Апробация работы.** Основные результаты диссертационной работы докладывались на следующих конференциях:

- Десятая молодежная научная школа-конференция «Лобачевские чтения —2011». Казань, 31 октября – 4 ноября 2011 года;
- Международная конференция ”3rd World Conference on Information Technology (WCIT-2012)”. Barcelona, 14 – 16 November 2012, Spain;
- II Международная конференция «Искусственный интеллект и естественный язык (AINL-2013)». Санкт-Петербург, 17 – 18 мая 2013 года;
- VI Международная научно-практическая конференция «Электронная Казань 2014». Казань, 22 – 24 апреля 2014 года;
- XVI Всероссийская научная конференция «Электронные библиотеки: перспективные методы и технологии, электронные коллекции (RCDL-2014)». Дубна, 13 – 16 октября 2014 года;
- Семинары по программной инженерии ”All-Kazan Software Engineering Seminar (AKSES-2015)”. Kazan, 9 April 2015;
- Международная конференция ”Agents and multi-agent systems: technologies and applications (AMSTA-2015)”. Sorento, 17 – 19 June 2015, Italy.

Практическая апробация результатов работы проводилась на выгрузке инцидентов из системы регистрации запросов службы технической поддержки ИТ-инфраструктуры ОАО «АйСиЭл КПО-ВС (г. Казань)». Процент успешно обработанных запросов пользователей составил 61%. Ожидаемый результат был 70% (под ним понимались разрешенные запросы пользователя), но 61% также приемлем, так как серьезно увеличивает эффективность разрешения запросов.

**Личный вклад.** Автор провел анализ запросов пользователей и классифицировал их; построил модель процессов целевой области и выявил возможности оптимизации процессов в ней. Данные для исследования (выгрузка из систем регистрации запросов пользователей ICL) были получены при помощи А.В. Крехова. Совместно с М.О. Талановым автор создал базовую архитектуру системы. Автор разработал компоненты системы, провел испытание системы на экспериментальных данных и отладил ее работу.

**Публикации.** Основные результаты по теме диссертации изложены в 10 печатных изданиях [17–26], из которых статьи [22; 23] проиндексированы в БД Scopus и входят в перечень журналов ВАК РФ, статья [23] также проиндексирована в БД Web of Science, работа [24] опубликована в журнале из перечня ВАК РФ, статья [19] проиндексирована в БД РИНЦ, работы [17–19] опубликованы в материалах международных и всероссийских конференций, статьи [20; 21] опубликованы в международном журнале "International Journal of Synthetic Emotions", входящем в индекс ACM.

В работе [17] А.С. Тощев предложил оригинальную идею автоматического конструирования приложений. В статье [18] А.С. Тощевым был разработан программный комплекс, М.О. Таланов предложил идею, а А.В. Крехов предоставил тестовые данные из системы регистрации запросов службы технической поддержки ИТ-инфраструктуры ОАО «АйСиЭл КПО-ВС (г. Казань)». В работе [19] А.С. Тощев предложил и реализовал архитектуру интеллектуального агента, М.О. Таланов поставил задачу проверки результатов работы подхода. В статьях [20; 21] А.С. Тощев выполнил проверку модели, предложенной М.О. Талановым. В работе [22] А.С. Тощев реализовал модель. В статье [23] А.С. Тощев выполнил доработку модели мышления для универсальности, М.О. Таланов поставил задачу придания универсальности системе. В статье [24] А.С. Тощев исследовал результаты работы системы регистрации запросов службы технической поддержки ИТ-инфраструктуры ОАО «АйСиЭл КПО-ВС (г. Казань)» и выдвинул гипотезу о возможности автоматизации разрешения части запросов. В работах [25; 26] А.С. Тощев провел разработку и проверку модели, М.О. Таланов разработал основную концептуальную идею.

**Объем и структура работы.** Диссертация состоит из введения, четырех глав, заключения и пяти приложений. Полный объем диссертации составляет

122 страницы с 47 рисунками и 28 таблицами. Список литературы содержит 101 наименование.

## **Глава 1. Интеллектуальные системы регистрации и анализа проблемных ситуаций, возникающих в ИТ-инфраструктуре предприятия**

### **1.1 Обзор исследований в области интеллектуальных систем регистрации и анализа проблемных ситуаций**

В данной главе рассматриваются постановка задачи и обзор исследований в области интеллектуальных систем регистрации и анализа проблемных ситуаций.

**Постановка задачи.** Большинство проблем, которые решает удаленная служба поддержки информационной инфраструктуры предприятия, носит достаточно тривиальный характер (по данным компании ОАО «АйСиЭл КПО-ВС (г. Казань)»): установить приложение; переустановить приложение; решить проблему с доступом к тому или иному ресурсу. Названные проблемы решают специалисты технической поддержки, которая обычно делится на несколько линий по уровню умения специалистов (см. таблицу 1.1). Каждая линия поддержки представлена своим классом специалистов. В среднем команда, обслуживающая одного заказчика, насчитывает около 60 человек. Процентное соотношение специалистов разных линий отображено на рисунке 1.1.

Таблица 1.1 — Описание работы специалистов различных уровней поддержки

Уровень	Описание
Первая линия	Решение уже известных, задокументированных проблем, работа напрямую с пользователем
Вторая линия	Решение ранее неизвестных проблем
Третья линия	Решение сложных и нетривиальных проблем
Четвертая линия	Решение архитектурных проблем инфраструктуры

Работа специалистов первой линии поддержки состоит из множества рутинных и простых задач. На рисунке 1.2 показано соотношение разных типов проблем, встречающихся во время работы службы поддержки, в таблице 1.2 приведена расшифровка типов. Данные подготовлены на основе анализа работы команд ОАО «АйСиЭл КПО-ВС (г. Казань)».

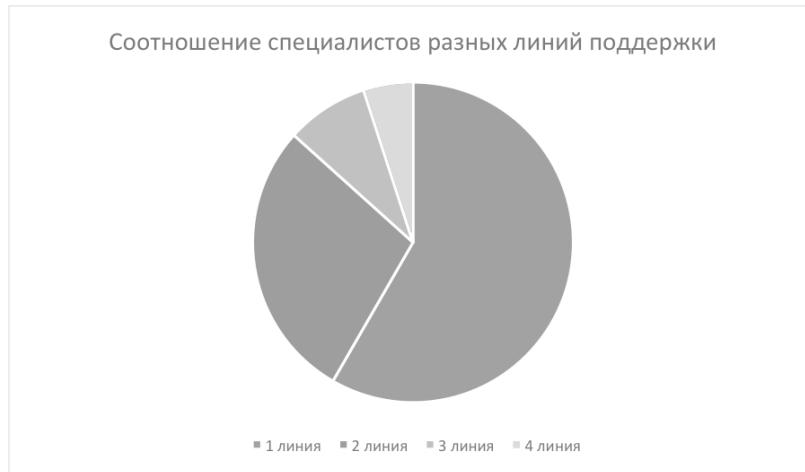


Рисунок 1.1 — Диаграмма состава команд

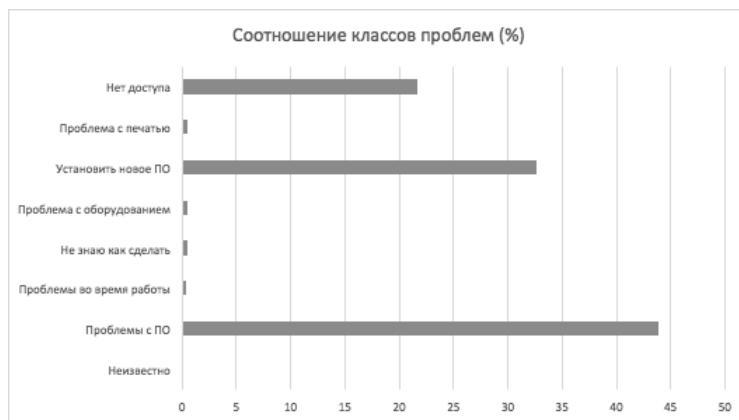


Рисунок 1.2 — Диаграмма соотношений типов проблем

Таблица 1.2 — Категории инцидентов в области удаленной поддержки инфраструктуры

Категория	Описание
Проблема с ПО	Проблема при запуске ПО на компьютере. Решается переустановкой
Проблемы во время работы	Проблема с функционированием программного обеспечения
Как сделать	Запрос на инструкцию по работе с тем или иным компонентом рабочей станции
Проблема с оборудованием	Неполадки на уровне оборудования
Установить новое ПО	Требование установки нового программного обеспечения
Проблема с печатью	Установка принтера в систему
Нет доступа	Нет доступа к общим ресурсам

Как показывают исследования, решение части задач может быть автоматизировано. Если это будет сделано, специалисты получат дополнительное время

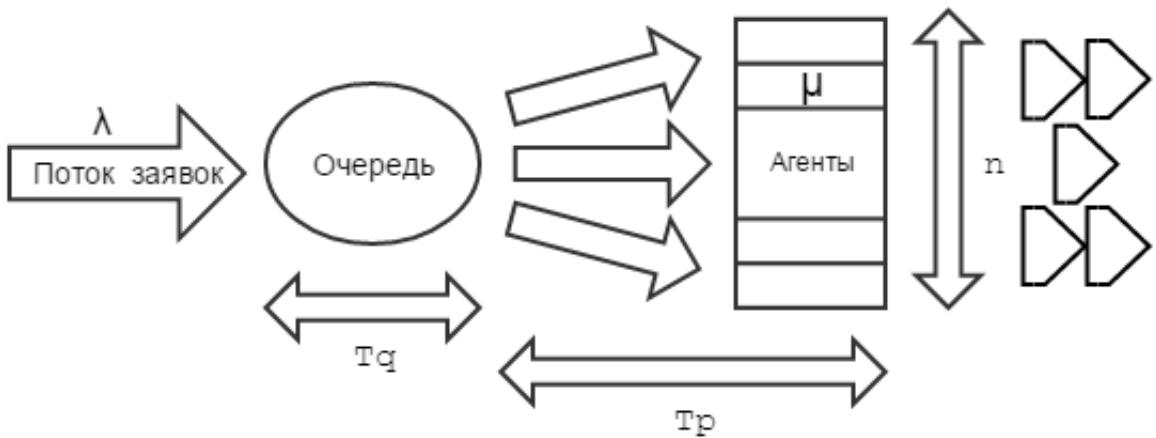


Рисунок 1.3 — Модель системы массового обслуживания в ИТ.

- Аналитическое решение для простейших систем, которое позволяет выразить  $T_q(t)$  через  $\lambda$ ,  $\mu$  и  $n$ ;
- Решение с помощью имитационного подхода, где строится гистограмма  $T_q(t)$ , по которой оценивается достаточность  $n$  для обеспечения SLA;
- Решение с помощью эконометрического подхода, которое подходит для систем с достаточно большим  $n$ . В таких системах возможно оценить  $T_q(t)$  по имеющейся статистике.

В [27] на основе комбинации формулы Эрланга, модели Энгсета и модели Полячека – Хинчина построена формула для решения задач ТМО на основе аналитического подхода путем нахождения распределения вероятностей для  $T_{qp}$ . Основной же задачей этой работы является прогнозирование необходимых ресурсов для максимизации SLA ( $SLA = 1 - \alpha$ ). В данной диссертации ставится задача минимизации  $T_{qp}$ ,  $S(\mu)$  и динамического выделения ресурсов. На основе статистики, собранной в компании ОАО «АйСиЭл КПО-ВС (г. Казань)», был подсчитан следующий коэффициент  $T_{qp} = 47,9$  при  $n = 6$ ;  $SLA = 0,82$ ;  $\alpha = 0,18$ ;  $\alpha_n = 2920$ . Для анализа потока заявок в данном случае лучше использовать имитационный подход, так как  $n$  слишком мало. На рисунке 1.4 представлен средний поток заявок по часам, посчитанный на основе собранной статистики. На нем наглядно видно, как изменяется поток с течением дня.

### Оценка стоимости работы специалиста

По данным аналитики портала SuperJob [28], в Казани средняя зарплата системного администратора с опытом работы в 2014 году составляла 30 – 35 тыс. руб. (с учетом 21 рабочего дня в месяце — 179–208 руб. в час). В соответствии с действующим российским законодательством [29] расходы компаний на одного



Рисунок 1.4 — Средний поток заявок по часам.

работника определяются по формуле

$$L = R + R * (F_1 + F_2 + F_3),$$

где  $R$  — выплата человеку в час,  $F_1$  — НДФЛ 13%,  $F_2$  — совокупность отчислений в ФБ (6%), ПФР (14%), ТФОМС (2%), ФФОМС (1,1%), ФСС (2,9%),  $F_3$  — налог на прибыль (20%). Таким образом, расходы компании на сотрудника варьируются от 285 до 314 руб. в час, а за 8-ми часовой рабочий день — от 2280 до 2512 руб. Далее, аренда выделенного сервера (такого, например, как Xeon X3, 1.7 GHz, 8GB RAM, 256GB SSD) стоит 8 900 руб./мес. (см. [30]) (53 рубля за 1 час ( $R_p = 53$  с учетом 8-ми часового рабочего дня)). Но сервер может работать 24 часа в сутки за исключением простоев на обслуживание, которые обычно составляют не более 5% времени.

Итого: сервер работает 478,8 часов в месяц. С этой точки зрения эксплуатация сервера будет стоить 18,5 руб. в час ( $R_p = 18,5$ ). Один сервер в своем быстродействии может заменить несколько специалистов при решении соответствующих задач. Чтобы решение было экономически эффективным, необходимо, чтобы оно сокращало расходы как минимум на 30% (по данным ОАО «АйСиЭл КПО-ВС (г. Казань)»).

Подсчет на основе стоимости часа и пропорции показывает, что работа специалиста — это 6% работы виртуального агента т.е. сервера (без учета работы сервера параллельно над несколькими задачами). Таким образом, уровень разре-

шения инцидентов системой в 50% выполнит требования по прибыли примерно на 186%.

### **Предпосылки развития изучаемой предметной области**

Основной тенденцией в развитии области удаленной поддержки ИТ-инфраструктуры являются попытки удешевить и улучшить стоимость предоставления услуг [2].

Компании, работающие на этом рынке, вкладывают большие средства в автоматизацию. Кроме того, современное развитие науки и техники, точнее, вычислительных мощностей [31] позволяет провести автоматизацию даже самых научкоемких процессов. Дальнейшей перспективой развития области удаленной поддержки ИТ-инфраструктуры является замена человеческих специалистов автоматизированными системами. Разработки в этом направлении ведут многие компании, например, компания HP, которая имеет свою систему регистрации различных инцидентов [32] и сейчас ведет работу над ее автоматизацией. В качестве некоторого сравнения можно провести параллель происходящего процесса с промышленной революцией XVIII–XIX веков (см., например, [33]).

### **Обзор исследований в изучаемой предметной области**

Область исследования, с которой связана диссертация, является комплексной и включает в себя различные направления работ, в частности, создания различных интеллектуальных систем. Сфера применения интеллектуальных систем обширна, например, в Институте Чиная (Индия) Е. Джубилсоном и П. Дханавантини ведутся исследования интеллектуальных систем обработки запросов пользователей в области телекоммуникаций [34], а в университете Ганновера (Германия) Р. Брунс и Дж. Данкель разрабатывают интеллектуальные системы для обработки запросов в службу спасения с целью уменьшения времени реакции на происшествие [35]. В Санкт-Петербургском государственном университете под руководством В.И. Золотарева проводится оценка эффективности службы информационной поддержки в Вычислительном центре СПбГУ [36]. В Сингапуре С. Фу и П. Леонг проведен анализ эффективности ИТ-службы поддержки крупной компании и показана возможность автоматизации ряда процессов [37].

Исследования в области интеллектуальных систем повышения эффективности ИТ-службы предприятия ведутся также лидерами отрасли: компаниями HP [32] и IBM [14]. Например, известна многоцелевая интеллектуальная система

IBM Watson, разработкой и исследованием которой занимается группа под руководством профессора А. Гоэля (США–Китай).

Еще одно из направлений исследований в области обработки естественного языка составляет подход GATE [38], который активно развивается в университете Шеффилда (Великобритания) под руководством Г. Каллаган, Л. Моффат и С. Сзаз. Другое направление — это семантический поиск, исследования в этой области также активно ведутся в университете Шеффилда, в частности, выработан подход ”Mimir”, который реализует возможности поиска по принципу «поиск и открытие» [39]. Для организации поиска решений в соответствии с запросами пользователей в таких системах используются онтологии, например, широко применяется подход, предложенный С. Дей и А. Джеймс из Калифорнийского университета (США), основанный на применении деревьев тегов в онтологии [40].

Для придания интеллектуальной системе гибкости необходимо дать ей возможность проводить логические рассуждения. Одной из ведущих организаций в этом направлении исследований является консорциум OpenCog [41] (США). Этими работами руководит Бен Герцель (председатель Artificial General Intelligence Society и OpenCog Foundation) — один из мировых лидеров в области искусственного интеллекта. Исследования в области машинной логики также ведутся в рамках проекта NARS [42] под руководством профессора университета Темлпа (США) Пея Вонга.

Интерес к области интеллектуальных систем обработки информации можно, в частности, оценить как количество публикаций за последние годы, процитированных в базе данных Scopus, — с 2004 года в среднем оно составило около 1010 в год.

### **Стандарты, используемые в области ИТ-аутсорсинга: ITIL и ITSM**

В области ИТ-аутсорсинга есть несколько готовых стандартов ведения работ, одним из которых является библиотека ITIL. Этот стандарт описывает лучшие практики организации работ в области ИТ-аутсорсинга. Используемый в библиотеке подход соответствует стандартам ISO 9000 (ГОСТ Р ИСО 9000) [43–45]. Наличие стандартов диктует унифицированность как постановки проблем, так и алгоритмов решения, а также способствует возможности частичной или в некоторых случаях полной автоматизации решения проблем.

## **1.2 Сравнительный анализ систем регистрации и устранения проблемных ситуаций**

**HP OpenView** [32; 46–48] является комплексным программным решением по мониторингу ИТ-инфраструктуры предприятия и имеет множество модулей. На рисунке 1.5 представлен вид системы, которая обладает широким спектром возможностей: мониторинг [49; 50]; регистрация инцидентов; управление системами. Система не поддерживает: понимание и формализацию запросов; автоматическое устранение проблемы на основе формализации запроса.

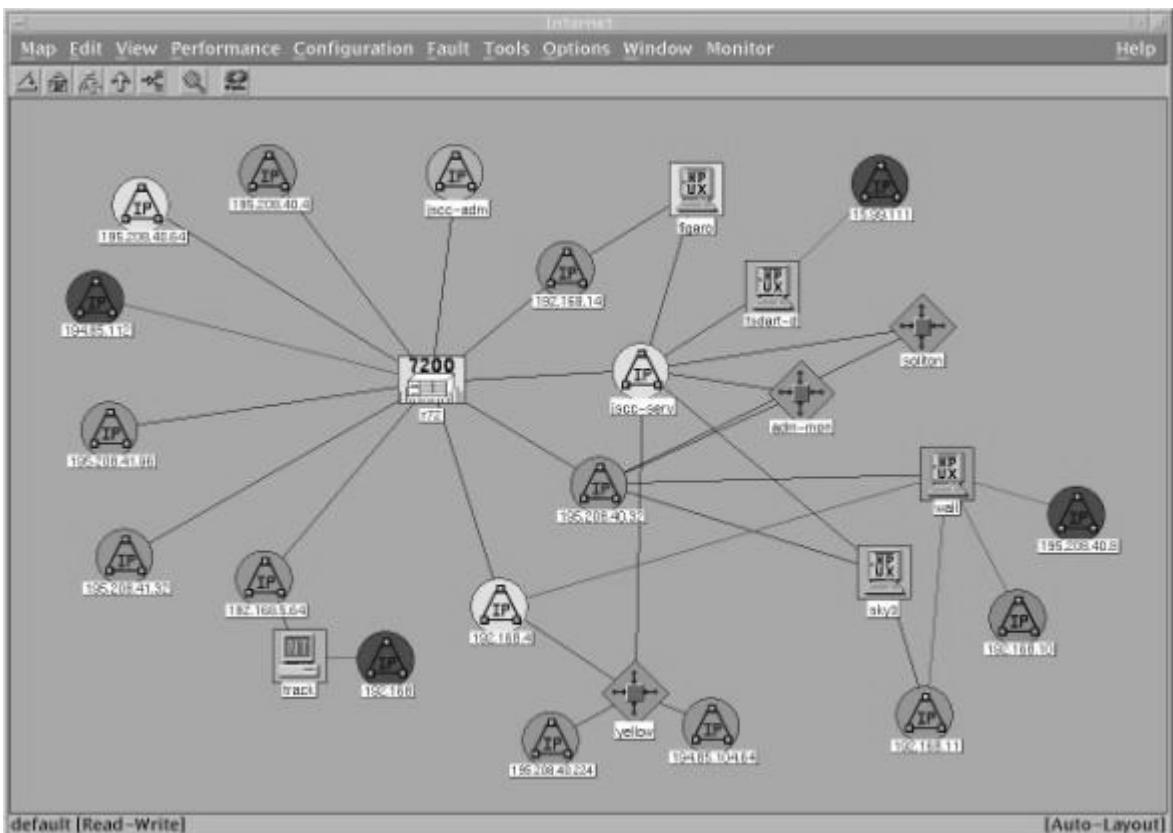


Рисунок 1.5 — HP OpenView (материал из Wikipedia)

Система ServiceNOW<sup>1</sup> — средство автоматизации сервиса. На рисунке 1.6 представлен вид этой системы, которая предоставляет следующие возможности: регистрация инцидентов и создание цепи их обработки. Система не поддерживает: понимание и формализацию запросов; автоматическое исправление проблемы на основе формализации запроса. Система широко используется в ИТ-инфраструктуре CERN [51; 52] для регистрации инцидентов и их решения.

**IBMWatson** — это вопросно-ответная система, которая поддерживает понимание и формализацию запросов и поиск решений. Система не поддерживает

<sup>1</sup>Система ServiceNOW <http://www.servicenow.com/>

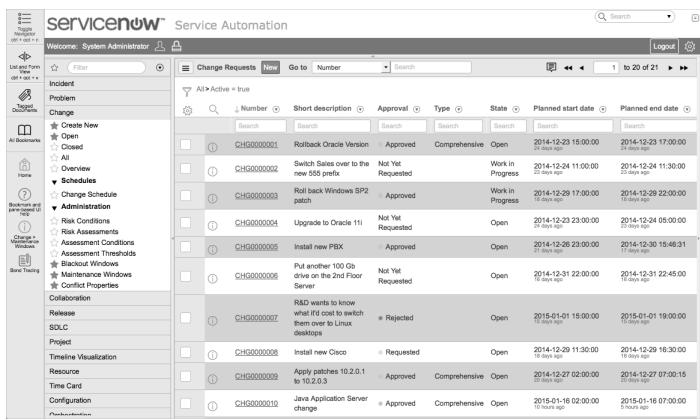


Рисунок 1.6 — Service NOW (материал из <http://wiki.servicenow.com/>)

автоматическое разрешение проблемы на основе формализации запроса. Система широко используется в медицине для постановки диагнозов болезней [53–56] и реализует базовые принципы искусственного интеллекта [57; 58]. Ее разработка велась под суперкомпьютер IBM Deep Blue [59] группой под руководством профессора А. Гоэля. На рисунке 1.7 представлен общий вид этой системы.

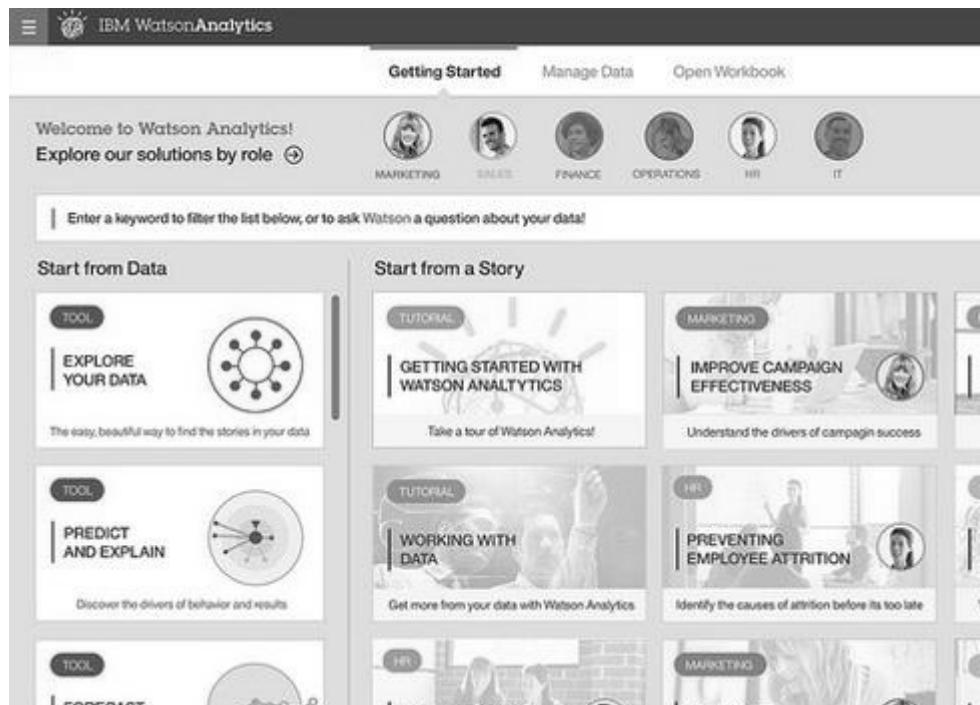


Рисунок 1.7 — Пример работы системы Watson (материал из <http://www.informationweek.com/>)

Кроме того, известны следующие дополнительные способы и системы автоматизации разрешения проблемы пользователя:

- Обработка инцидентов посредством регулярных выражений. В таком решении нет гибкости, так как обработка идет путем поиска ключевых слов вне контекста. Метод регулярных выражений частично используется для

- обработки естественного языка, поиска [60], диагностики активных систем [61], анализа поведения функций [62], обработки данных в системе eDiscovery [63], в разработке способов программирования [64];
- Обработка инцидентов при помощи скриптов — автоматизируются лишь рутинные операции.

На данный момент ни одна из описанных систем в полной мере автоматически не разрешает запросы пользователей: не фиксирует их, не проводит анализ, не ищет решение, не применяет решение и не дает обратной связи пользователю. Каждая система в той или иной мере реализует те или иные функции, но системы, которая реализует их все, нет.

Для того чтобы создать модель системы, необходимо определить требования к ней или критерии, соответствие которым будет служить одним из доказательств состоятельности системы наряду с экспериментальными результатами.

Перечисленные ниже требования сформированы, исходя из возможностей специалистов службы поддержки, а также анализа проблем, которыми они занимаются. Большинство инцидентов — тривиальные и типичные, но все они разные. Для человека проблемы "Please install Firefox" и "Please install Chrome" идентичны, но с точки зрения формализации это не так — общее в них можно найти, взглянув на обобщение различающейся части: Firefox и Chrome являются пакетами программного обеспечения.

Чтобы понять, каким требованиям должны соответствовать интеллектуальная система регистрации и анализа проблемных ситуаций в ИТ-области, нужно понять, что делает специалист службы поддержки. Такой специалист регистрирует проблему, анализирует, ищет решение, проводя логические рассуждения и фиксируя в памяти удачное решение, и решает проблему. Итак, чтобы обеспечить полностью автоматическое разрешение инцидентов, интеллектуальная система регистрации и анализа проблемных ситуаций в ИТ-области должна соответствовать следующим критериям: осуществлять мониторинг ИТ-инфраструктуры пользователя; регистрировать инциденты; создавать цепи обработки (Workflow) инцидента; понимать и формализовать запросы пользователя на естественном языке; искать решение и применять найденное решение; обучаться алгоритму разрешения инцидента; уметь проводить логические рассуждения (обобщение, специализация, синонимичный поиск).

Из этого списка требований к системе важно выделить формализацию запросов на естественном языке. Ниже приведены результаты анализа разработок в области формализации запросов на естественном языке.

### **1.3 Сравнительный анализ методов и комплексов обработки текстов на естественном языке**

#### **1.3.1 Обработка эталонных текстов**

В данном разделе проведен обзор обработчиков естественного языка. За основу были взяты инциденты, выгруженные из систем поддержки ИТ-инфраструктуры ОАО «АйСиЭл КПО-ВС (г. Казань)». В силу специфики предметной области (информационные технологии) основным языком был выбран английский язык. Был сформирован список из типичных эталонных фраз, на которых тестировались обработчики естественного языка. Фразы были выявлены путем анализа существующих отчетов об инцидентах. Примерами инцидентов являются следующие запросы.

**Инцидент 1.** *User had received wrong application. User has ordered Wordfinder Business Economical for her service tag 7Q4TC3J, there is completed order in LOT with number ITCORD-18125. However she received wrong version, she received Wordfinder Tehcnical instead of Business Economical. Please assist (Пользователю было установлено неверное приложение. Пользователь заказал приложение "Wordfinder Business Economical" для ее запроса номер 7Q4TC3J. По нашим данным запрос выполнен, номер результата ITCORD-18125. Однако, он получил неверную версию — он получил приложение "Wordfinder Tehcnical" вместо "Wordfinder Business Economical". Пожалуйста, помогите).<sup>2</sup>*

**Инцидент 2.** *Laptop — user has almost full C: but when he looks in the properties of the files and folders on C: they are only 40GB and he has a 55GB drive (Ноутбук — диск C: пользователя переполнен, но он посмотрел свойства C: и увидел, что имеется только 40GB, хотя пользователь установил накопитель на 55Gb).*

**Инцидент 3.** *User cannot find Produkt Manageron start menu. Please reinstall (Пользователь не может найти Produkt Manageron в меню "Пуск". Пожалуйста, переустановите).*

---

<sup>2</sup>Здесь и далее в переводе оригинальные синтаксические, грамматические и стилистические ошибки сохранены, дабы продемонстрировать сложность формализации запросов на естественном языке.

**Инцидент 4.** *User needs to have pdf 995 re-installed please* (Пользователю нужно переустановить "pdf 995").

При анализе этих запросов были использованы следующие обработчики естественного языка: Open NLP [65], Relex [66], StanfordParser [67]. Результат их работы оценивался при помощи метрик, представленных в таблице 1.3, а полученные результаты приведены на рисунке 1.8.

Таблица 1.3 — Таблица метрик

Метрика	Описание	Формула
Precision	Точность	$P = \frac{tp}{tp + fp},$ где $P$ — precision, $tp$ — успешно обработанные слова, $fp$ — должно успешные
Recall	Чувствительность	$R = \frac{tp}{tp + fn},$ где $R$ — recall, $tp$ — успешно обработанные слова, $fn$ — должно неуспешные
$F$	$F$ — measure (результативность)	$F = \frac{P * R}{P + R},$ где $P$ — precision, $R$ — recall.

Из диаграммы 1.8 видно, что наилучшие результаты показывает обработчик Relex [66]. После анализа необработанных инцидентов у всех обработчиков были выявлены проблемы двух типов:

- невозможность корректировки простых грамматических ошибок, связанных с пропущенными пробелами или неверным форматированием (ошибки первого типа);
- полисемия, например, слово *please* интерпретировалось как глагол, хотя является по смыслу «вводным словом» (ошибки второго типа).

Несмотря на хорошие результаты — 63% успешно разобранных предложений, ошибки первого и второго типов серьезно ухудшают результат. Эффектив-

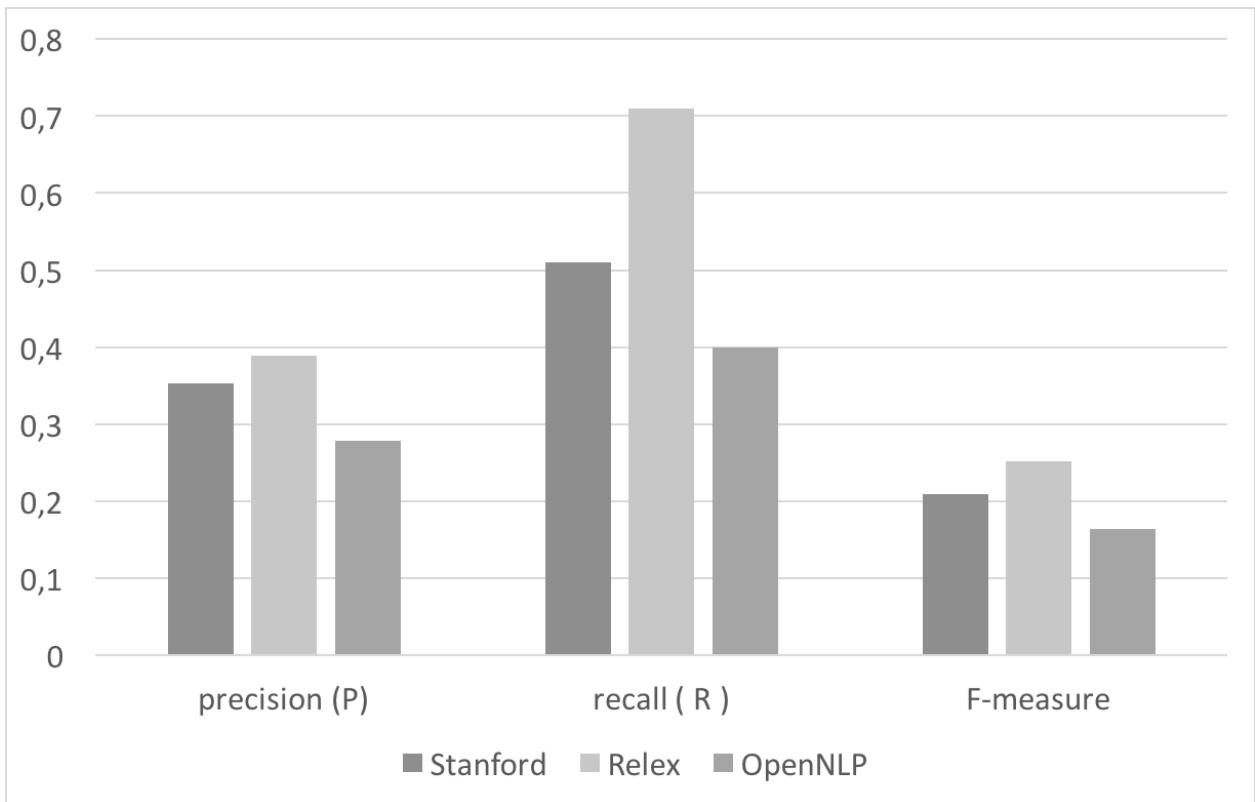


Рисунок 1.8 — Результаты обработки текстов

ность разбора предложений в 63% случаев недостаточна для успешной работы системы. Чтобы улучшить показатель, был разработан комплекс мер для устранения ошибок первого и второго типов.

### 1.3.2 Исправление ошибок первого и второго типов

Чтобы разрешить проблемы, связанные с ошибками первого и второго типов, была проведена предварительная обработка текста, состоящая из 2-х фаз: комплексная корректировка — для ошибок первого типа; обработка при помощи внутренней базы знаний — для ошибок второго типа. Чтобы избавиться от орфографических, грамматических и синтаксических ошибок, был сконструирован составной корректировщик, который имеет модульную структуру и осуществляет корректировку последовательно в рамках выбранной области знаний (например, одного проекта). В результате были сконструированы модули корректировки: Google API — модуль подключения к открытym системам Google для использования их алгоритмов корректировки; After the Deadline — модуль, использующий открытый программный продукт After the Deadline для исправления текстов.

Таким способом удалось исправить большинство ошибок, связанных с синтаксисом, грамматикой и орфографией в рамках выбранной области (корректи-

ровка ошибок в ограниченной области гораздо проще, чем универсальная корректировка). Также удалось исправить ошибки неверного написания: наличия лишних пробелов, пропуска запятых и точек ввиду лаконичности входной информации. Необходимо отметить, что входная информация представляет собой простые предложения в рамках терминологии одной области знаний (ИТ, специфические термины, которые используются только на этом проекте), что значительно упрощает корректировку ошибок. По-прежнему осталась проблема обработки неверной интерпретации слов в тексте (полисемии).

Для корректировки ошибок второго типа был сконструирован модуль для обработчика естественного языка Relex, который разбивал стандартный процесс обработки на «предобработку» и «обработку». Стадия «обработки» включает в себя такой же алгоритм работы, как был до этого в модуле Relex, а стадия «предобработки» проверяет входные данные (слово или предложение) на предмет его вхождения во внутреннюю базу знаний, и если таковое имеется, то приложение передает соответствующие корректировки обратно в модуль. Например, Relex во фразе "please install firefox" считает, что "please" — это глагол, поэтому в базе знаний нашей системы отмечено, что "please" — это форма вежливости (проблема полисемии), тем самым Relex больше не интерпретирует "please" как глагол в выбранной области знаний (то есть для всех случаев).

### 1.3.3 Сравнение средств обработки русского и английского языков

Средства обработки естественного языка принято относить к большому классу средств NLP — Natural Language Processing [68]. Для английского языка существует множество открытых средств обработки этого языка, для русского языка найти их гораздо сложнее. Рассмотрим архитектуру средств обработки естественного языка на примере популярного комплекса обработки — OpenCog Relex [66].

OpenCog Relex использует результаты работы открытого компонента для лексического анализа под названием Link Grammar [69]. Он поддерживает множество языков: английский, русский, турецкий, немецкий и т. д. В качестве формата вывода Relex использует синтаксис Link Grammar и преобразует его в формат связей, как показано в примере 1. Разбор примера приводится далее.

**Пример 1.** User is unable to start KDP web, please reinstall Java.

**Результат**

```
|_obj (start, КВР)
```

```

pos(start, verb)
inflection-TAG(start, .v)
5 tense(start, present)
pos([web], WORD)
noun_number(KBP, singular)
definite-FLAG(KBP, T)
pos(KBP, noun)
10 _advmod(reinstall, please)
pos(reinstall, verb)
inflection-TAG(reinstall, .v)
tense(reinstall, present)
pos(please, adv)
15 inflection-TAG(please, .e)
noun_number(Java, singular)
definite-FLAG(Java, T)
pos(Java, noun)
pos(., punctuation)
20 _obj(,, Java)
pos(,, verb)
tense(,, infinitive)
HYP(,, T)
_to-do(unable, ,)
25 pos(unable, adj)
inflection-TAG(unable, .a)
tense(unable, present)
pos(to, prep)
inflection-TAG(to, .r)
30 pos(be, verb)
inflection-TAG(be, .v)
_preadadj(User, unable)
noun_number(User, singular)
definite-FLAG(User, T)
35 pos(User, noun)

```

Далее проведем разбор слова start. В результате мы получим несколько отношений:

- pos(start, verb) — start глагол;
- tense(start, present) — время настоящее;
- inflection-TAG(start, .v) — метод обозначения на схеме (индекс).

Остальные обработчики пока не поддерживают русский язык. Существуют открытые проекты, но они еще недостаточно развиты. Таковым является, например, русский словарь для LinkGrammar<sup>3</sup>, но он не доступен для скачивания.

#### 1.4 Выводы по главе 1

В данной главе рассмотрены существующие на данный момент интеллектуальные системы регистрации и анализа проблемных ситуаций, возникающих в процессе функционирования ИТ-инфраструктуры предприятия. В таблице 1.4 приведены сводные данные по системам. По ним можно сказать, что ни одна из рассмотренных систем полностью не реализует все необходимые функции для интеллектуальной системы разрешения проблемных ситуаций в ИТ-инфраструктуре предприятия. В главе 1 также выработаны критерии сравнения обработчиков естественного языка и выполнен анализ средств обработки естественного языка. По полученным показателям эффективности было решено использовать OpenCog Relex.

Таблица 1.4 — Сравнительный анализ функциональности существующих решений

Сравнительный пункт	HP Open View	ServiceNOW	IBM Watson
Мониторинг	Да	Да	Да
Регистрация инцидентов	Да	Да	Да
Управление системами	Да	Нет	Нет
Создание цепи обработки (Workflow) инцидента	Да	Да	Нет
Понимания и формализация запросов на естественном языке	Нет	Нет	Да
Поиск решений	Нет	Нет	Да
Применение решений	Нет	Нет	Нет
Обучение разрешению инцидента	Нет	Нет	Да
Продолжение следует			

<sup>3</sup><http://www.abisource.com/projects/link-grammar/russian/>

**Таблица 1.4 – продолжение**

<b>Сравнительный пункт</b>	<b>HP Open View</b>	<b>ServiceNOW</b>	<b>IBM Watson</b>
Умение проводить логические рассуждения: генерализацию, специализацию, синонимичный поиск	Нет	Нет	Нет

## **Глава 2. Модель интеллектуальной системы принятия решений для регистрации и анализа проблемных ситуаций в ИТ-инфраструктуре предприятия**

В данной главе рассмотрены модели, которые были изучены и использованы при создании системы принятия решений для регистрации и анализа проблемных ситуаций в ИТ-инфраструктуре предприятия. Отметим, что работа над системой велась с 2011 года, за истекшее время было создано три рабочих версии прототипа системы, реализующих различные модели мышления.

Созданными и испытанными моделями, использованными при создании системы принятия решений для регистрации и анализа проблемных ситуаций в ИТ-инфраструктуре предприятия, являются:

- модель Menta 0.1, построенная с использованием деревьев принятия решений;
- модель Menta 0.3, построенная с использованием генетических алгоритмов [70] ;
- модель TU 1.0, основанная на модели мышления Марвина Мински [71].

Модель, построенная на базе нейронных сетей (поддерживающая обучение), была отброшена на предварительной стадии оценки, так как она предъявляет большие требования к производительности (скорости работы и требуемых ресурсов) [72], что в свою очередь порождает высокую стоимость. Далее каждая модель будет рассмотрена подробно.

### **2.1 Построение модели Menta 0.1 с использованием деревьев принятия решений**

Данная модель была одной из первых, которые были апробированы. Она основана на деревьях принятия решений [73], которые широко используются в вопросно-ответных системах [74–76]. При построении модели использованы компоненты, реализующие обработку запросов на естественном языке, поиск решения, применение найденного решения, хранение в базе знаний.

Система ориентирована на выполнение таких простых команд, как, например, «Добавить поле в форму». Основные функции модели представлены следующими потоками: получение и формализация запроса; поиск решения при помощи деревьев принятия решений; изменение приложения согласно запросу; генерация

и компиляция приложения. Рассмотрим подробнее, как устроена система. Начнем с того, как в системе представлены данные.

### 2.1.1 База знаний на основе OWL

Для представления данных в системе была использована база знаний, построенная на основе OWL-файла (см. [77]). С помощью редактора Protege [78] в базу вводились начальные данные о целевом приложении (приложении, которое будет модифицироваться системой согласно запросам пользователей) в виде семантической сети. В качестве такого приложения была создана система, которая вела учет заказов пользователя на покупку того или иного товара в интернет-магазине.

На рисунке 2.1 представлен один из программных классов (согласно терминологии ООП [79]) этой системы — Order — в формате OWL. Этот класс отвечает за обработку заказов. Слева отображены супер классы (классы-предки), к которым он привязан. Например, класс BLL относится к бизнес-логике приложения, Module — отдельный модуль в рамках системы. Справа представлены свойства класса, а их описания приведены в таблице 2.1. С помощью предикатов определяется поведение свойства: создать файл, создать новое поле. В таблице 2.2 представлено описание иерархии предикатов. На рисунке 2.2 представлен класс CreateCustomer в OWL, в который входит описание всех необходимых свойств для генерации файла исходного кода на языке Java.

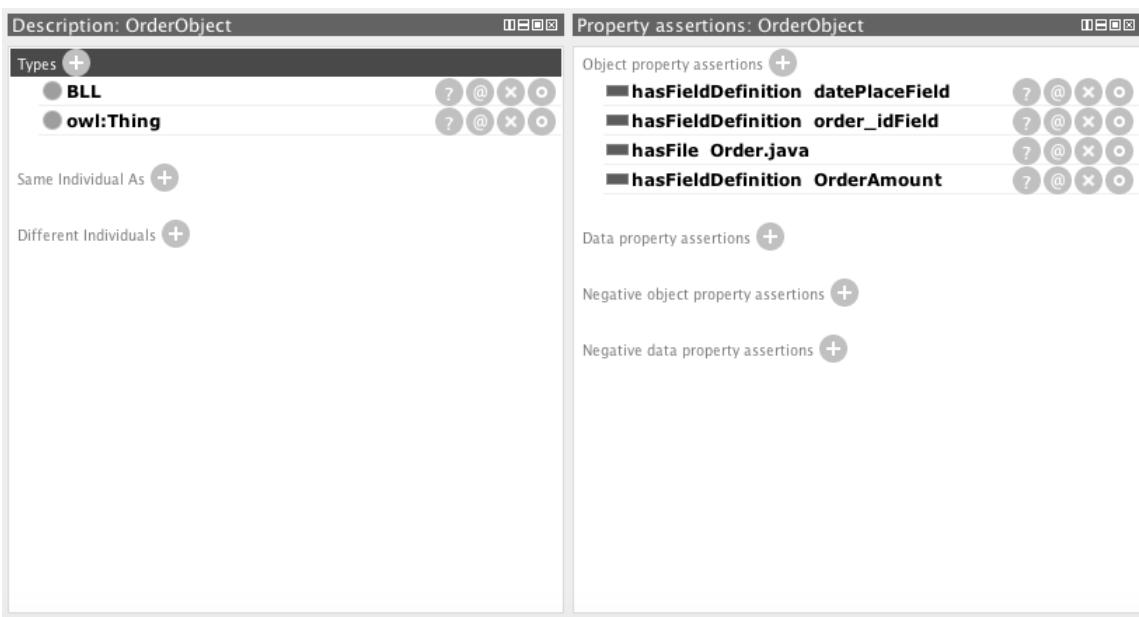


Рисунок 2.1 — Представление класса Order в OWL. Визуализация Protege

Таблица 2.1 — Описание свойств класса Order в OWL

Свойство	Предикат	Описание
OrderAmount	hasFieldDefinition	Поле: сумма заказа
orderidField	hasFieldDefinition	Поле: идентификатор заказа
Order.java	hasFile	Идентификатор имени файла для генерации
datePlaceField	hasFieldDefinition	Поле: время размещения заказа

Таблица 2.2 — Описание иерархии предикатов

Предикат	Описание
hasFieldDefinition	Предикат, обозначающий свойство класса
hasMethodDefinition	Предикат, обозначающий функцию
classDefinition	Обозначение класса
database	Обозначение базы данных

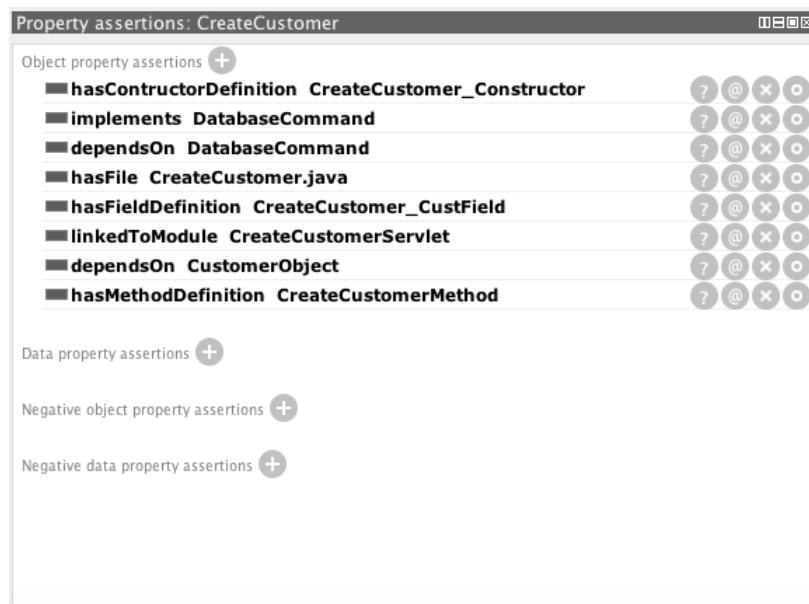


Рисунок 2.2 — Представление класса CreateCustiner в OWL. Визуализация Protege

### 2.1.2 Основные компоненты модели

Основными компонентами модели являются: Request parser (Stanford parser); Генерация Action (Action Generator); Исполнение Action (Action Applier); Генерация приложения (Application Generator).

*Request parser* формализует запрос на естественном языке. *Action Generator* генерирует объект класса Action (который содержит описание требуемых над моделью приложения действий) из результатов работы, основываясь на Деревьях принятия решений [80] и базе данных. Основной задачей данного модуля является генерация имени, действия и поля. Модуль *Action Applier* отыскивает объект в модели по данным от Action Generator и производит действие, кроме того, используя предикат dependOn, он производит модификацию всех зависимых классов.

В модели поддерживаются два типа Action: RemoveFieldAction (удаление поля), AddFieldAction (добавление поля). После завершения работы производится генерация целевого приложения на языке Java при помощи OWL-модели в модуле *Application Generator*. На рисунке 2.3 представлена UML-диаграмма последовательности для основного рабочего потока приложения: пользователь вводит в систему запрос ”Add new field to Customer (Добавить новое поле в класс Customer)”; модуль StanfordParser вычленяет из запроса связи типа dobj (связь объекта и действия); модуль ActionGenerator создает на основе связей объект, описывающий список действий (далее — действие) над моделью, для получения требуемого результата; модуль ActionApplier, используя действие, изменяет модель приложения; модуль ApplicationGenerator применяет изменения к приложению. В результате у класса Customer (данный класс содержит набор свойств, необходимых для описания клиента магазина) появляется новое поле ”New”.

После проведения экспериментов было выявлено, что приложение не может использовать ранее найденные решения, абстрагируя их. Например, система знает, как произвести операцию добавления поля, но вот аналогичную операцию — добавить метод — сделать не может. Поиск решения также потребовал специального обучения: система не могла путем перебора информации в своей базе знаний найти решение. В системе также не было возможности обучения.

После применения этой системы была предпринята попытка найти более универсальное решение. Результатом стало построение модели Menta 0.3, описанной ниже.

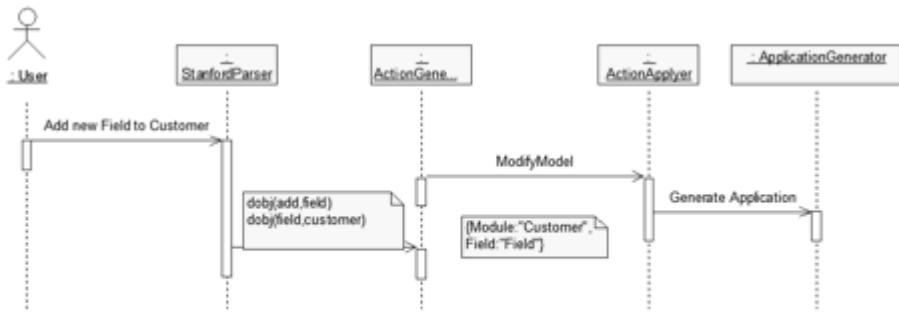


Рисунок 2.3 — UML-диаграмма последовательности для основного потока в модели Menta 0.1

## 2.2 Модель Menta 0.3, построенная с использованием генетических алгоритмов

В данную модель по сравнению с предыдущей были добавлены модуль логики для оценки решения и модуль генетических алгоритмов для генерации решения. Отметим, что генетические алгоритмы часто применяются в биологически инспирированных системах [81], [82]. Кроме того, есть примеры их использования в системах поддержки принятия решений, однако эффективность таких систем не подтверждена [83]. В рамках модели Menta 0.3 были отработаны следующие основные компоненты будущей итоговой модели: критерии приемки (Acceptance Criteria); How-To — для хранения решений проанализированных проблем; формат данных OWL; использование логических вычислений для проверки решения. Система Menta 0.3 содержала внутри себя модель целевого приложения (как и Menta 0.1) и список решений тех или иных проблем (How-To). При помощи генетического алгоритма модель строила How-To решения, проверяла его при помощи логического движка NARS [42] на соответствие входным критериям приемки. С точки зрения генетических алгоритмов это — функция отбора особей из поколения [84].

### 2.2.1 Основные компоненты модели

Модель состоит из компонентов, представленных в Таблице 2.3.

Таблица 2.3 — Компоненты модели Menta 0.3

Компонент	Описание
MentaController	Веб-служба [85], которая предоставляет интерфейс для общения с пользователем и остальными системами
SolutionGenerator	Модуль отвечает за генерацию решения. На вход он получает Acceptance Criteria. Основой является генетический алгоритм. Для него был выбран framework ejc [86]. Из всех возможных классов в базе знаний, отсеянных по классификатору, составляются паросочетания. К каждому паросочетанию применяется логическое суждение на основе AcceptanceCriteria (за это отвечает модуль ReasonerAdapter). В итоге паросочетание получает оценку в виде пары Frequency, Confidence (частота, вероятность). Таким образом находится наилучшее паросочетание. Если его показатель 1,1, то решение принимается, иначе отбрасывается (на данный момент установлен жесткий показатель). SolutionGenerator включает в себя SolutionChecker, который включает в себя ReasonerAdapter.
SolutionChecker	Проверка решения. Принимает на вход выбранные How-To, AcceptanceCriteria. Комбинирует их и передает ReasonerAdapter.
ReasonerAdapter	Транслирует How-To в термины NARS. NARS — non-axiomatic reasoning system [42] (система логических суждений, разработанная профессором Питером Вонгом). Принцип действия NARS — это всевозможная комбинация фактов. Каждый факт имеет свои частоту и вероятность. Их сочетанием получается композиция данных фактов.

**Таблица 2.3 – продолжение**

<b>Компонент</b>	<b>Описание</b>
Translator	Транслирует объекты базы знаний (знания) в отчеты. Последние бывают следующих типов: Solution Report; UML Report; Patch. В данной версии используется первый тип отчета. Он содержит описание на выбранном языке программирования решения, найденного системой.
Applicator	Данный модуль применяет решение к модели приложения, содержащейся в базе знаний. Также данная модель включает FileApplicator, который генерирует решение в виде файлов на выбранном языке программирования.
KBServer	База знаний приложения. Используется сервер non-SQL БД HypergraphDB.

В предыдущей модели в качестве хранения данных использовался файл, что было неудобно в случае, если приложение работает параллельно с несколькими запросами. В системе Menta 0.3 был использован специальный сервер баз данных, речь о котором пойдет далее.

### 2.2.2 База знаний на основе графов

При реализации базы знаний (здесь и далее — KBServer) был создан промежуточный модуль доступа к данным (здесь и далее — DAO, Data Access Object), данный подход широко используется в проектировании программного обеспечения [87]. Это позволяет максимально отделить реализацию KBServer от конкретного хранилища.

**EntityManagerFactory**. Данный класс является входной точкой и создает объект, с помощью которого приложение осуществляет работу с базой знаний. Класс автоматически выбирает необходимые настройки для объекта.

**EntityManager**. Это основной класс для загрузки и хранения объектов из базы данных.

**Configuration**. Этот класс хранит такие параметры настройки базы данных, как физическое положение БД и максимальное количество подключений.

**EntityTransaction.** Данный класс используется для управления транзакциями при доступе к объектам базы данных.

При выборе физического хранилища данных было проанализировано несколько хранилищ OWL-данных: OWLIM, SESAME и HG. Результаты их сравнения представлены в таблице 2.4.

Таблица 2.4 — Сравнение скорости доступа к данным баз знаний

	Sesame	OWLIM	HG
<b>Единицы измерения</b>	<b>мс.</b>	<b>мс.</b>	<b>мс.</b>
предварительно скомпилированные запросы	26 253	3 012	6 813
без кеша	30 545	1 122	9 045
с кешем	24 258	962	985

Несмотря на то, что OWLIM дает лучшие результаты, был выбран HGDB, который предоставляет более широкие возможности доступа к данным, такие, например, как поддержка алгоритмов работы с графами.

## 2.3 Модель TU 1.0, основанная на модели мышления Марвина Мински

Следующим этапом разработки стала модель, построенная с применением теории Марвина Мински. Эта модель сохранила следующие основные концептуальные элементы предыдущих моделей и показала свою состоятельность на контрольных примерах: Acceptance Criteria; Обучение; Поиск и применение решения; Отсутствие обработки естественного языка. Данная модель является более универсальной и представляет собой верхнеуровневую архитектуру обработки запроса (мышления), где компонентами являются лучшие части предыдущих систем.

### 2.3.1 Особенности модели мышления

В 2006 году Марвин Мински опубликовал свою книгу "The emotion machine" [71], в которой предложил свой взгляд на систему мышления и памяти человека. В основу его теории легла парадигма триплета Критик – Селектор – Образ мышления (далее  $T^3$ ), k-line (линия, которая связывает приобретенные знания,

например, огонь — горячо) для сопоставления знаний. На рисунке 2.4 представлена схематичное изображение  $T^3$ .

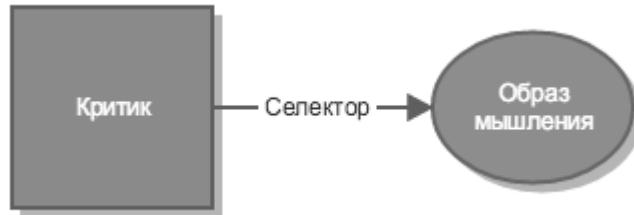


Рисунок 2.4 — Критик – Селектор – Образ мышления

**Критик** представляет собой определенный переключатель: внешние обстоятельства, события или иное воздействие. Например, «включился свет, и зрачки сузились», «обожглись и одернули руку». Критик активируется только тогда, когда для этого достаточно обстоятельств. Одновременно может активироваться несколько критиков. Например, человек решает сложную задачу, идет активация множества критиков: выполнить расчет, уточнить технические детали. Кроме того, параллельно может активироваться критик, сообщающий о необходимости отдыха.

**Селектор** занимается выбором определенных ресурсов, одним из которых является **Образ мышления**.

**Образ мышления** — это способ решения проблемы. Образ мышления может быть сложным и способен активировать других критиков. Например, размышляя над проблемой, специалист понимает, что нужно произвести полный перебор всех возможных комбинаций параметров, чтобы получить нужный результат, и тут он решает поискать готовое решение: а может кто-то уже сделал такой перебор, и можно будет использовать его результаты. Здесь «поиск готового решения» является критиком внутри образа мышления «поиск решения».

На рисунке 2.5 представлена расширенная модель работы  $T^3$ . Критик активирует Селектор, который активирует Образ мышления (овал). Последний в свою очередь может активировать нового Критика или же совершить определенные действия. Например, зажегся зеленый свет светофора, значит, можно переходить дорогу. Под ресурсами здесь понимается набор знаний из базы знаний: Критики, Селекторы, Образы мышления, готовые решения.

Если активировалось много Критиков, то проблему нужно уточнить, так как степень неопределенности слишком высока. Если проблема очень похожа на уже проанализированную, то можно действовать и судить по аналогии.

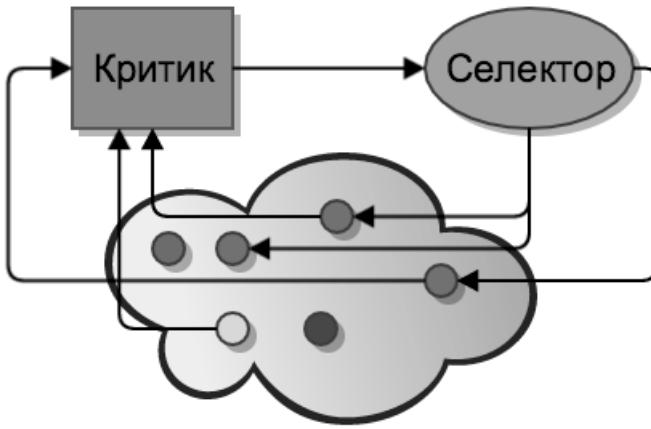


Рисунок 2.5 —  $T^3$  в разрезе ресурсов

### 2.3.2 Основные компоненты модели

#### Уровни мышления

Концепция уровней мышления представляет собой модель степени ментальной активности человека. Никто из людей не может похвастаться скоростью гепарда, гибкостью кошки или силой медведя. На наш взгляд, все это компенсируется возможностью изобретения образов мышления. Например, чтобы быть быстрыми, люди избрали различные механизмы (самолеты, машины и др.). Чтобы быть сильными, они избрали оружие.

Все изобретения являются результатом взаимодействия человека с окружающим миром. Именно данное взаимодействие заставляет людей изобретать что-то новое, создавать шедевры литературы и летать в космос. По ходу своего развития разум человека проходит путь от врожденных инстинктов до возможности создания фундаментальных трудов, таких как «Теории всего» [88]. В этом ему помогает возможность гибкого мышления: изобретение различных подходов к решению проблемы. Далее мы рассмотрим концепцию уровней мышления, следуя Марвину Мински [71].

Начнем с описания уровней мышления, которое представлено в таблице 2.5. Деление на данные уровни носит условный характер. Например, уровни 5 и 6 можно объединить, но, по словам Марвина Мински, принцип бритвы Оккама, который успешно применяется в физике, не должен также легко и однозначно применяться в психологии и теории мышления.

Таблица 2.5 — Описание уровней мышления, предложенных Марвином Мински

Уровень	Описание
Инстинктивный уровень	Происходят инстинктивные реакции (врожденные). Например, коленный рефлекс. Общую формулу для этого уровня можно выразить как «если ..., то сделать так».
Уровень обученных реакций	Используются накопленные знания, то есть те знания, которым человек обучается в течение жизни. Например, переходить дорогу на зеленый свет. Общую формулу для этого уровня можно описать как «если ..., то сделать так».
Уровень рассуждений	Мышление с использованием рассуждений. Например, если перебежать дорогу на зеленый свет, то можно успеть вовремя. На данном уровне сравниваются последствия нескольких решений и выбирается оптимальное. Общую формулу для этого уровня можно выразить как «если ..., то сделать так, тогда будет так».
Рефлексивный уровень	Рассуждения с учетом анализа прошлых событий. Например, «в прошлый раз я побежал на моргающий зеленый и чуть не попал под машину».
Саморефлексивный уровень	Построение определенной модели, с помощью которой идет оценка своих поступков. Например, «мое решение не пойти на это собрание было неверным, так как я упустил столько возможностей, я был легкомысленным».
Самосознательный уровень	Оценка своих поступков с точки зрения высших идеалов и оценок окружающих. Например, «а что подумают мои друзья? А как бы поступил мой герой?»

На рисунке 2.6 представлено схематичное изображение уровней мышления, названных выше. 1–3 уровни составляют личность человека. 2–5 представляют ЭГО человека (Человеческое Я) — осознание человека в общении с окружаю-

щими. 3–6 представляют собой сверх ЭГО человека (сверх Я) — его моральные установки.



Рисунок 2.6 — Иллюстрация концепции Уровней мышления

**Концепция k-line.** Эта концепция была первый раз упомянута Марвином Мински в 1987 году в журнале *Cognitive Science*. В книге [89] Марвин Мински раскрывает концепцию k-line. Полностью концепция описана позже в книге [71]. K-line представляет собой связь между двумя событиями, объединяющими их в знание, например, объединение Образа мышления, найденного решения и активированной проблемы.

На рисунке 2.7 показана k-line, которая объединяет образы мышления, решения и других Критиков. Данная концепция позволяет «запоминать» удачные решения.

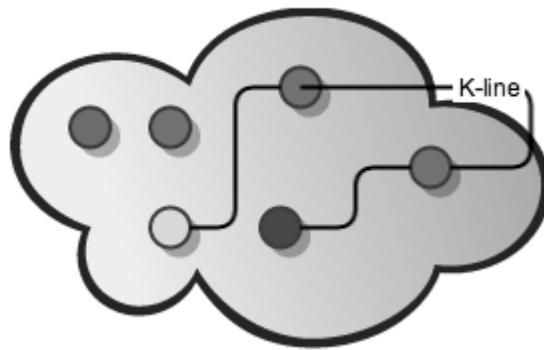


Рисунок 2.7 — Иллюстрация концепции k-line

## 2.4 Выводы по главе 2

Модель Menta 0.1 имеет следующие недостатки: отсутствие устойчивости к грамматическим и содержательным ошибкам входной информации. Например, входной файл не имел отношения к программной системе, модель которой была в базе знаний в формате OWL; система поиска решения работала только в рамках модели одной программы; отсутствовала функция обучения.

В данный момент существует новый подход, который использует леса деревьев принятия решений [80], он в рамках данной модели не рассматривался. Модель Menta 0.3 имеет следующие недостатки: отсутствие обучения; отсутствие обработки естественного языка; модуль HyperGraphDB оказался непригодным для промышленного использования; NARS в виду своих особенностей оказался непригодным для промышленного применения на значительном объеме фактов ( $>20$ ), так как содержал в себе комбинаторный взрыв<sup>1</sup>. Например, при 10 фактах количество сочетаний будет равно 45 на первом уровне, далее алгоритм NARS будет сравнивать результаты этих сочетаний. Кроме того, после апробации оказалось, что критерии приемки практически описывают необходимое решение, что является недопустимым. Данный подход был описан в статье [90].

Для программной экспертной системы очень важно обладать способностью мыслить и рассуждать, например, очень важно для системы уметь проводить аналогии между уже известными разрешениями проблем и вновь появившимися.

---

<sup>1</sup>Комбинаторный взрыв — термин, используемый для описания эффекта резкого («взрывного») роста временной сложности (до бесконечности) алгоритма при увеличении размера входных данных задачи.

Множество запросов типично и отличается лишь параметрами. Например, пожалуйста, установите Office, Antivirus и т. д.

Также для экспертной системы важно уметь абстрагировать специализированные рецепты решения. К примеру, система научилась решать инцидент "Please install Firefox" («Пожалуйста, установите Firefox»). Абстрагировав данный инцидент до степени "Please install browser" («Пожалуйста, установите браузер»), система сможет теми же способами попробовать разрешить новую проблему.

После рассмотрения нескольких моделей была выбрана модель мышления Марвина Мински, так как она может быть адаптирована на области разрешения проблемных ситуаций, возникающих в процессе эксплуатации ИТ-инфраструктуры предприятия. На основе подхода Мински была построена модель системы, которая поддерживает основные функции: обучение, понимание инцидента, поиск решения, применение решения.

## Глава 3. Реализация модели TU 1.0 для системы интеллектуальной регистрации и устранения проблемных ситуаций

В данной главе рассматривается реализация модели TU: архитектура системы и программная реализация. Архитектура была создана с учетом принципов проектирования Enterprise систем [91].

### 3.1 Архитектура системы

Данный раздел описывает основные режимы функционирования системы и концепцию ее построения. Архитектура системы представляет собой модульную систему, чтобы компоненты можно было удобно заменять [92], например, подключать различные обработчики естественного языка. Основные компоненты системы описаны в таблице 3.1.

Таблица 3.1 — Основные компоненты системы Thinking-Understanding (TU)

Компонент	Описание
TU Webservice	Основной компонент взаимодействия с внешними системами, включая пользователя
CoreService	Ядро системы, содержит основные классы
DataService	Компонент работы с данными
Reasoner	Компонент вероятностной логики
ClientAgent	Компонент выполнения скриптов на целевой машине
MessageBus	Шина данных для системы

Система может работать в 2-х режимах: режим обучения и режим запроса. Диаграмма вариантов использования для режима обучения представлена на рисунке 3.1. Главным действующим лицом (согласно терминам UML) является специалист технической поддержки (TSS) (в общем случае это Пользователь (User)). Специалист технической поддержки может выполнять следующие действия: обучать систему; предоставлять правильное решение, если идет режим обучения; ввести запрос, если система функционирует в основном режиме; отслеживать

применение исправления проблемы. Подробное описание представлено в таблице 3.2.

Таблица 3.2 — Описание ветвей в варианте использования «Режим обучения»

Ветвь	Описание
communication:Train	Обучение посредством коммуникации с системой специалиста технической поддержки
communication:ProvidesSolution	В случае коммуникации в режиме обучения специалист технической поддержки должен предоставить не только сам запрос, который будет formalизован системой, но обучить систему алгоритму разрешения проблемы, содержащейся в запросе. Система formalизует запрос, formalизует решение и создаст между ними связи
communication:ProvideRequest	Специалист технической поддержки вводит в систему запрос
communication:MonitorsSolution	Специалист технической поддержки смотрит, как применяется решение, если находится проблема, то решение корректируется посредством запроса CorrectSystemSolutions

Второй вариант использования — это основной поток. Главными действующим лицом системы является Заказчик (здесь и далее — Customer) (в общем случае это базовый класс Пользователь (здесь и далее — User)). Вариант использования имеет несколько ветвей, представленных в таблице 3.3. В данном варианте система функционирует в «боевом режиме», то есть ищет ответ на запросы пользователя. В этом режиме есть возможность обучения: если система сталкинется с проблемой, то она задаст вопрос пользователю.

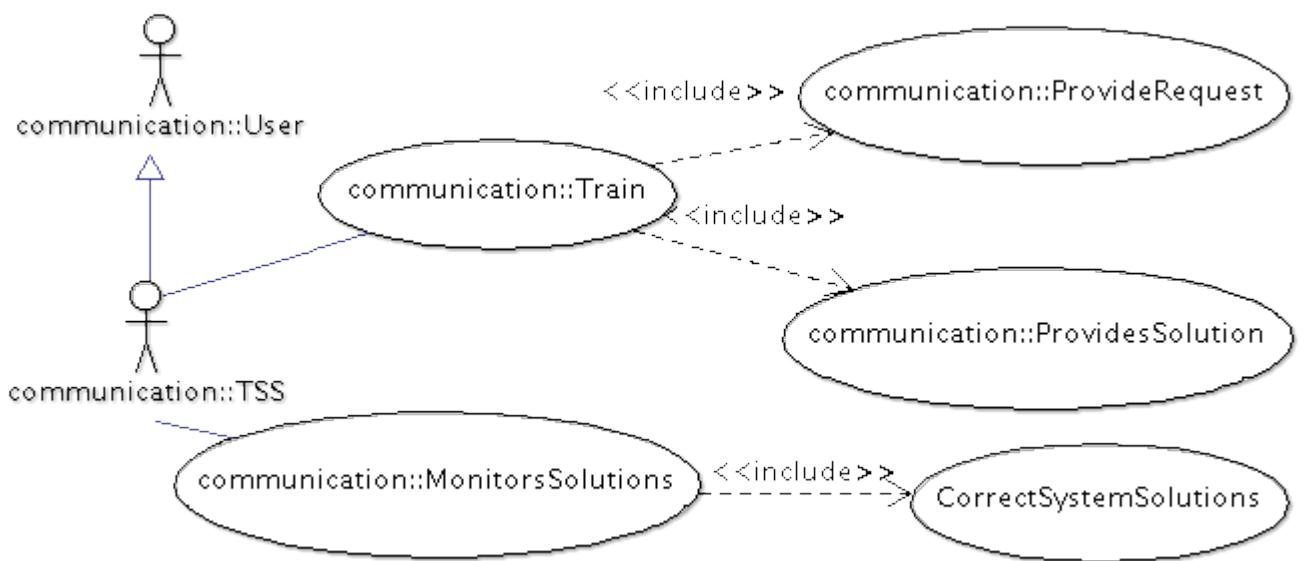


Рисунок 3.1 — Вариант использования. Обучение

Таблица 3.3 — Описание ветвей в варианте использования  
«Основной режим»

Ветвь	Описание
ProvideRequest	Заказчик вводит запрос в систему на естественном языке. Это могут быть либо прямая команда (например, Install Firefox, please), либо описание проблемы
communication:ProvideClarificationResponse	В случае, если система не может формализовать запрос либо нашлось множество решений, система запрашивает у пользователя детали
communication:ProvideConfirmationResponse	В случае, когда система нашла решение, она запрашивает у пользователя подтверждение, что искомое решение решило его проблему

### 3.1.1 Компоненты системы

На рисунке 3.2 представлено верхнеуровневое взаимодействие основных компонентов системы. В данном разделе будет дано краткое описание компонентов, последующие разделы будут посвящены отдельным компонентам, где будет представлено их подробное описание. В конце главы будет приведен подробный алгоритм взаимодействия всех компонентов системы.

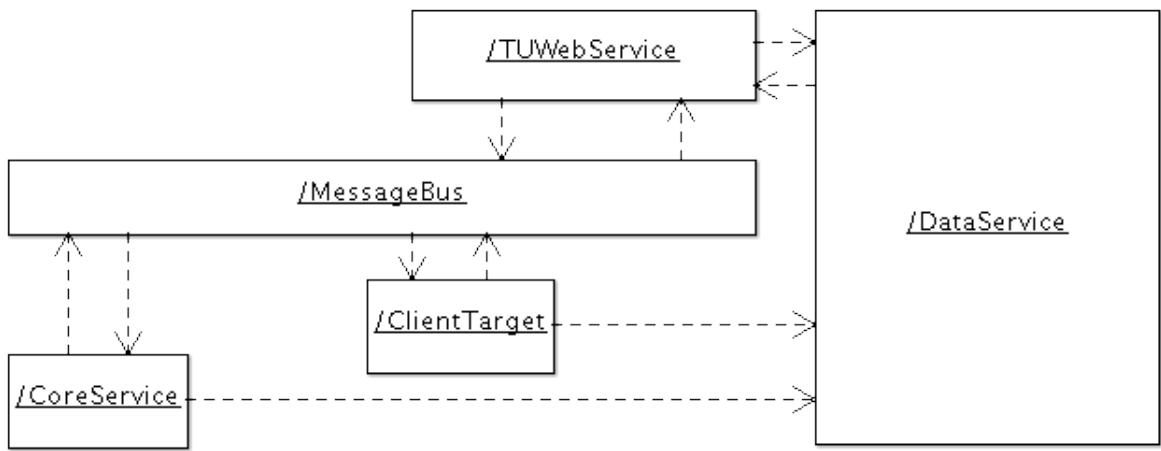


Рисунок 3.2 — Диаграмма взаимодействия компонентов

Основной точкой взаимодействия (с позиции пользователя) с системой является компонент WebService (см. рисунок 3.1.2). Данный компонент построен на стандарте WS SOAP для универсального использования в различных внешних системах [93]. Взаимодействие происходит по следующей схеме:

1. WebService получает запрос пользователя и сохраняет запрос в Базе Знаний (см. приложение 4.4);
2. WebService отправляет сообщение типа Request с информацией о запросе в компонент MessageBus (шина);
3. Один из экземпляров CoreService обрабатывает запрос;
4. Компонент CoreService обрабатывает запрос и сохраняет результаты в Базе Знаний, затем он отправляет в MessageBus сообщение RequestCompleted и сообщение ActionsToExecute с указанием действий, которые необходимо выполнить;

5. WebService получает сообщение RequestCompleted с результатами выполнения запроса и уведомляет подписчиков (конечных пользователей);
6. Компонент ClientAgent получает сообщение ActionsToExecute со списком действий, которые необходимо исполнить на целевых машинах.

Компонент MessageBus — это шина данных, которая обрабатывает сообщения и посыпает их указанным компонентам. TUWebService — компонент взаимодействия с «внешней средой», который предоставляет список функций и типов, посредством вызова которых можно обработать запрос в системе. Компонент DataService — отвечает за хранения данных, предоставляет базу заний для приложения. CoreService — ядро приложения, которое управляет основным жизненным циклом системы. ClientTarget — клиентский компонент для выполнения команд на машинах клиента. Сюда относятся как удаленное исполнение посредством WMI (Windows Management Instruments) и т. п., так и непосредственная установка клиента на машины пользователей. На рисунке 3.3 (здесь и далее для детального изучения изображения, пожалуйста, воспользуйтесь функциями увеличения в pdf) представлено детальное описание компонентов с подкомпонентами.

Каждый верхнеуровневый компонент на рисунке 3.3 включает в себе более мелкие компоненты. Например, CoreService состоит из ThinkingLifeCycle — компонента управления жизненным циклом, Selector — компонента поиска и выбора ресурсов, Way2Think — компонента, описывающего алгоритмы поиска и применения решений и Critic — вероятностных триггеров, которые срабатывают на входящие события. Более детальное описание компонентов и механизмов представлено в остальных разделах данной главы.

В этой главе были представлены компоненты и их декомпозиция в разрезе системы. На рисунках видны крупноблочная структура системы, а также детальная — в разрезе общих блоков. Кроме того, описано взаимодействие с системой с точки зрения конечного пользователя, а также взаимодействия с другими системами. Также даны описание крупноблочных компонентов, а также детальное описание одного из основных компонентов.

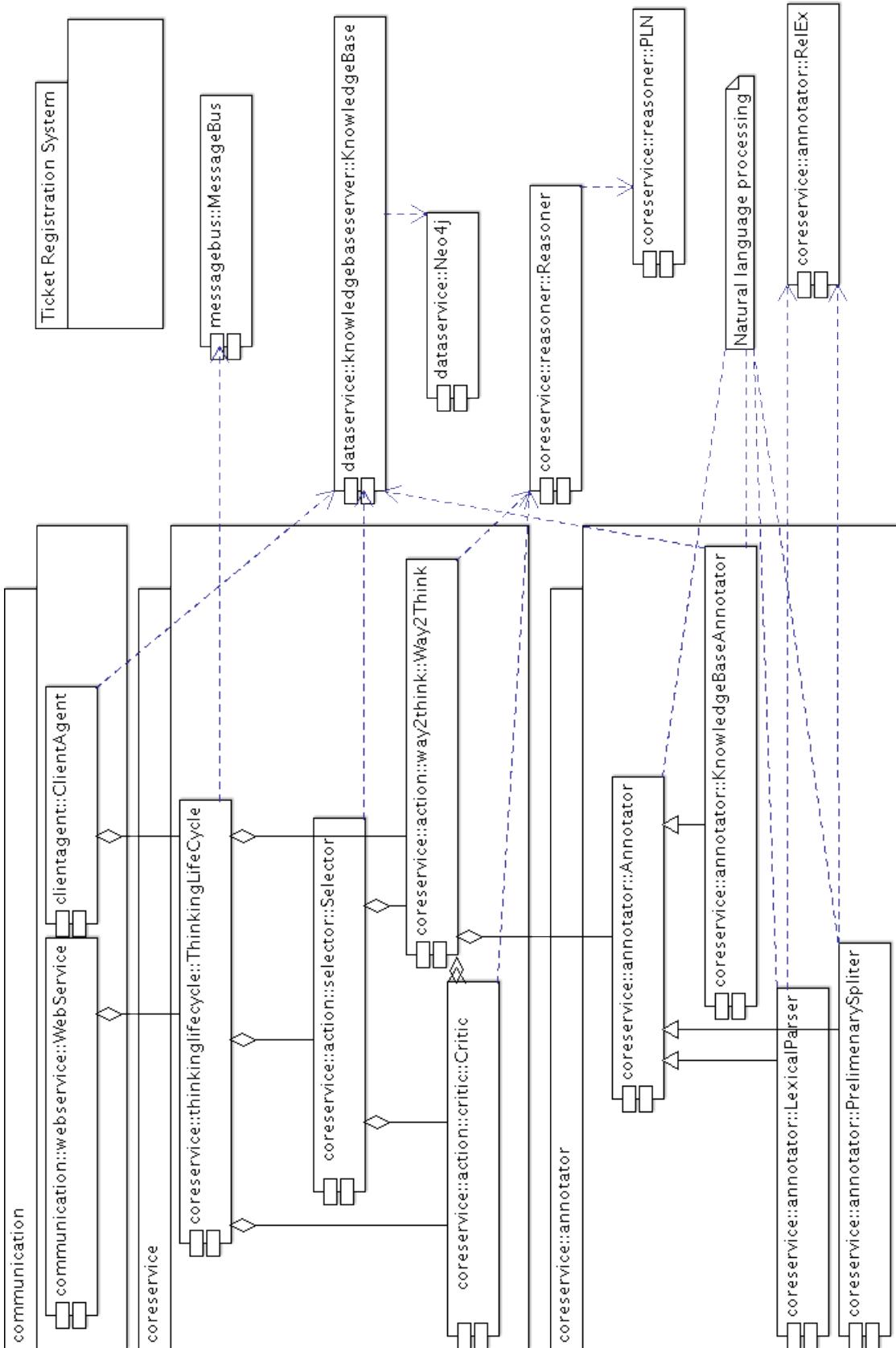


Рисунок 3.3 — Детальная диаграмма компонентов системы

### 3.1.2 Компонент WebService

Данный компонент обрабатывает запросы пользователей, а также внешних систем. Запрос пользователя представляется посредством объекта Request, который содержит информацию о пользователе, а также ссылку на сервис пользователя, который будет вызван, когда запрос будет обработан. Вся работа происходит в компоненте CoreService. На рисунке 3.4 представлен интерфейс компонента. В таблице 3.4 представлено описание методов. Подробное описание классов пред-

<<interface>>
WebService
createRequest(request : Request) : RefObject
subscribe(user : User, subscription : Subscription)
unsubscribe(user : RefObject, subscription : RefObject)
updateSubscription(user : RefObject, subscription : Subscription)
getSubscription(subscriptionID : RefObject) : Subscription
findRequests(user : RefObject) : List[Request]
createUser(user : User) : RefObject
updateUser(user : User)
removeUser(user : RefObject)
findRequest(request : RefObject) : Request

Рисунок 3.4 — Интерфейс компонента WebService

ставлено в приложении А. Основной поток работы компонента состоит из следующих шагов:

1. Пользователь создает запрос, используя метод WebService.createRequest;
2. Система сохраняет запрос в Базе Знаний и начинает его обработку;
3. Когда изменяется статус запроса (request.state), система оповещает подписчиков путем вызова ссылки на сервис пользователя, которая хранится в объекте Request.

В общем виде данный компонент представляет фасад системы, ее внешнюю часть. Задача данного компонента — обеспечить легкое взаимодействие с системой и инкапсулировать основную логику системы. Соответствующие методы были подобраны так, чтобы внешним системам не нужно было реализовывать модель данной системы, а достаточно было использовать базовые облегченные объекты для приема и передачи информации. Данный подход подробно описан в книге Мартина Фаулера [94] и имеет название Data Transfer Object. В данном

компоненте также широко используется подход Фасад, который также описан в названной книге. Архитектура компонента была проверена на предмет наличия антипаттернов проектирования, описанных в книге Вильяма Брауна [95].

В данном разделе был описан компонент взаимодействия с пользователям и внешними системами. Описаны архитектура компонента, основные методы, а также приведен пример основного потока.

Таблица 3.4 — Описание методов компонента WebService

<b>Метод</b>	<b>Описание</b>
createRequest(request:Request): [RefObject]	Регистрирует запрос от пользователя. В качестве параметра в метод передается SubscriptionID, по которому идет проверка запроса
subscribe(user:User, subscription:Subscription)	Создает подписку пользователя
unsubscribe(user:RefObject, subscription:RefObject)	Убирает подписку пользователя
updateSubscription(user:RefObject, subscription:Subscription)	Обновляет подписку пользователя
getSubscription(subscriptionID: RefObject): List<Request>	Возвращает подписку
findRequests(user:RefObject)	Возвращает запросы пользователя
createUser(user:User): RefObject	Создает пользователя
updateUser(user:User)	Обновляет информацию о пользователе
removeUser(user:RefObject)	Удаляет информацию о пользователе
findRequest(request:RefObject): Request	Возвращает запрос по ссылке

### 3.1.3 Компонент CoreService.ThinkingLifeCycle

Данный компонент системы отвечает за управление жизненным циклом системы: потоками, событиями приложения. Он запускает исполнение Критиков (Critic), Селекторов (Selector), Образов мышления (WayToThink), осуществляя обмен данных между компонентами. Компонент построен на фреймворке Akka Concurrency, который позволяет разрабатывать приложения, работающие параллельно [96]. Архитектура модуля построена с учетом модели TU.

В данном компоненте реализовано шесть уровней мышления: Instinctive — инстинктивный уровень; Learned — уровень обученных реакций; Seliberative — уровень рассуждений; Reflective — рефлексивный уровень; Self-Reflective Thinking — саморефлексивный уровень; Self-Conscious Reflection — самосознательный уровень.

На уровне Instinctive идет обработка инцидентов, генерированных по шаблону. Объект, который используется для обработки, применяет паттерн Akka [96]. На рисунке 3.5 представлена диаграмма классов компонента. В таблице 3.5 представлено подробное описание методов компонента с иллюстрациями.

На уровне Learned работают Критики классификации проблемы, они анализируют тип инцидента и активируют необходимые ресурсы. Более подробное описание работы Критиков приведено в следующих разделах.

На уровне Deliverative работает постановщик целей (который также является Критиком), который задает основные цели системы, тем самым запуская работу. Например, главная цель системы — решить проблему пользователя. Она активирует подцели, которые способствуют достижению основной цели.

На уровне Reflective работает Критик контроля времени, который отслеживает время выполнения запроса пользователя.

На уровне Self-Reflective Thinking осуществляется коммуникация с пользователем для уточнения запросов, проверки и применения найденного решения.

На уровне Self-Conscious работает Критик эмоционального состояния системы, который контролирует общее состояние системы и ее ресурсы. Критик контроля времени также обращается к этому критику для запроса ресурсов.

Данное расположение компонентов не случайно, оно реализует модель TU в привязке к различным уровням мышления. Проводя аналогию, можно сказать, что на более низких уровнях идет решение простых тактических задач, а на более высоких выстраивается общая стратегия поведения системы.

Таблица 3.5 — Описание методов класса (компоненты) ThinkingLifeCycle

Метод	Описание
onMessage(message : Message)	Данный метод вызывается при получении сообщения от шины. После этого происходит обработка запроса, формируется список действий, которые нужно выполнить. После этого запускается исполнение этих действий. На рисунке 3.6 представлена диаграмма действий этого метода
apply(request : Request) : List[Action]	Данный метод используется для запуска обработки входящего запроса. Для запроса создается контекст, если такой уже не был создан. После этого вызывается следующий компонент системы Selector, который выбирает необходимые ресурсы из Базы Знаний. На рисунке 3.8 представлена диаграмма действий этого метода
apply(actions : List[Action]) : TransFrame	Данный метод запускает обработку действий. Все действия разделяются на Critic (триггеры действий, которые в итоге должны перейти в WayToThink через Selector) и WayToThink (образа мышления, непосредственные обработчики данных, классы, которые производят изменения данных). На рисунке 3.9 представлена диаграмма действий этого метода
processWay2Think(inputContext: Context, outputContext: Context): TransFrame	Данный метод запускает обработку WayToThink. Он создает входной контекст (InputContext), заполняет его параметрами, создает выходной контекст OutputContext. Затем он запускает обработку данных во входном контексте. На рисунке 3.10 представлена диаграмма действий этого метода

**Таблица 3.5 – продолжение**

<b>Метод</b>	<b>Описание</b>
processCritic(context: Context): List[SelectorRequestRulePair]	Данный метод запускает обработку Critic. На рисунке 3.11 представлена диаграмма действий этого метода
init(): Boolean	Данный метод инициализирует экземпляр класса ThinkingLifeCycle. Во время инициализации происходят создание или подключение Базы Знаний (см. Словарь терминов 4.4). На рисунке 3.12 представлена диаграмма действий этого метода
start(): Boolean	Данный метод является необходимым для поддержки технологии Akka Concurrency, фактически он вызывает метод init
stop(): Boolean	Данный метод является необходимым для поддержки технологии Akka Concurrency, фактически он останавливает работу экземпляра класса: останавливается сессия к шине данных, останавливается подключение к Базе Знаний
registerProcess(process : Process, level : Level) : Process	Данный метод ставит задачу в очередь исполнения. В качестве параметра принимается Level (уровень приоритета процесса)
stop(processLevel : Level) : List[Process]	Данный метод останавливает процесс. В качестве параметра принимается ссылка на процесс. На рисунке 3.13 представлена диаграмма действий этого метода

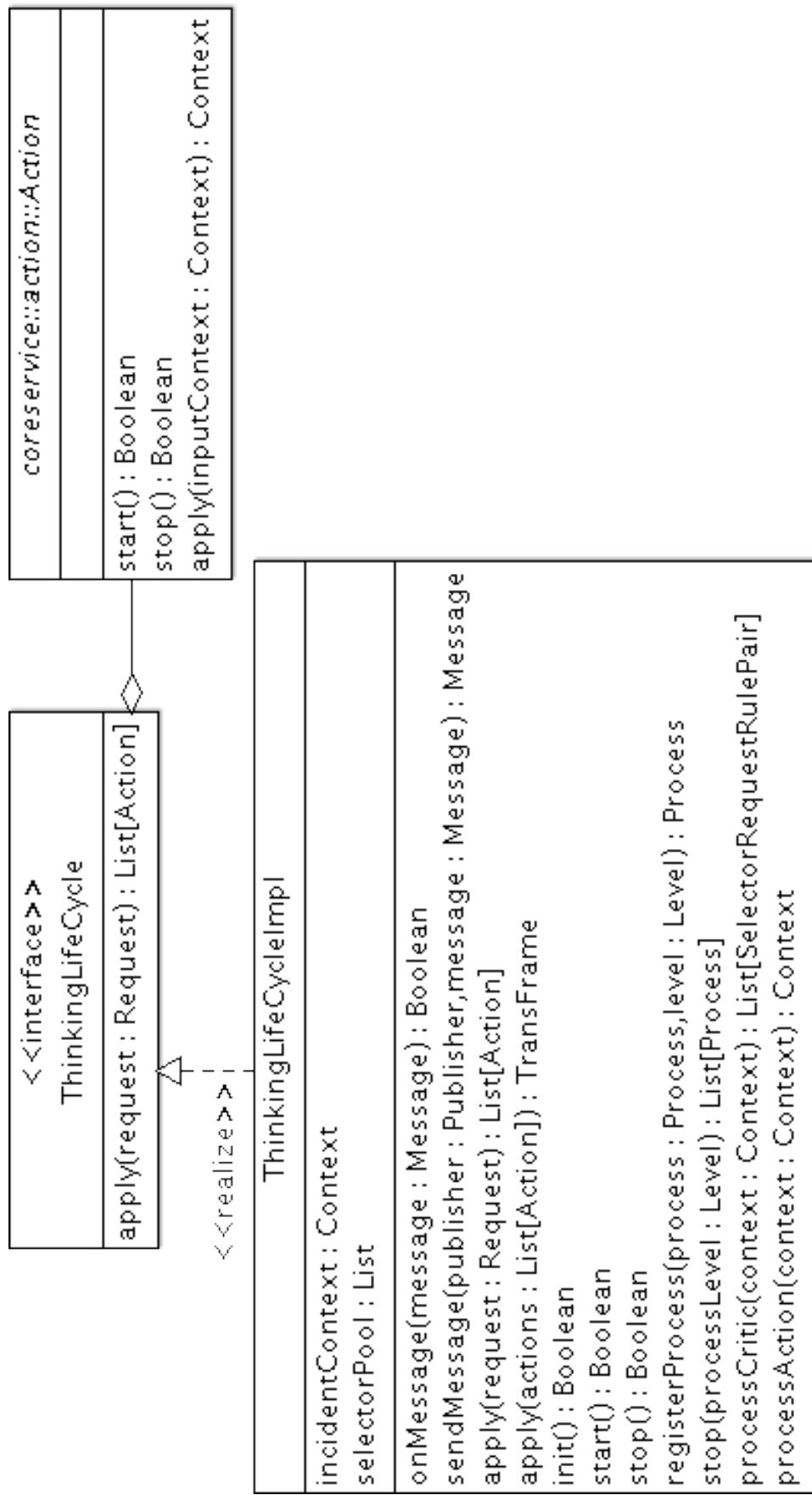


Рисунок 3.5 — Диаграмма классов ThinkingLifeCycle

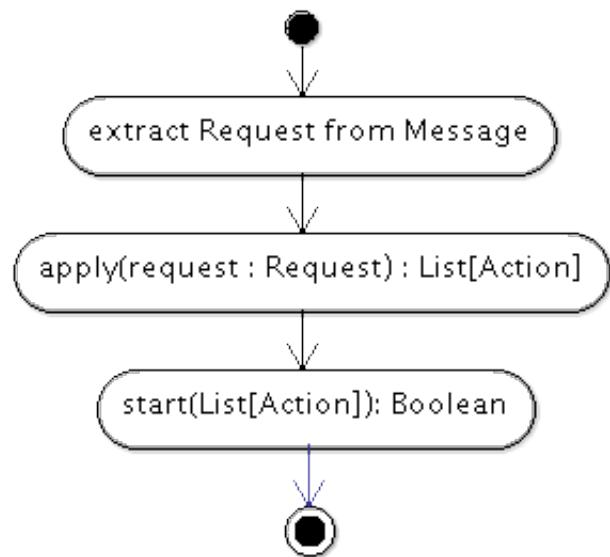


Рисунок 3.6 — Диаграмма действий метода onMessage компонента ThinkingLifeCycle

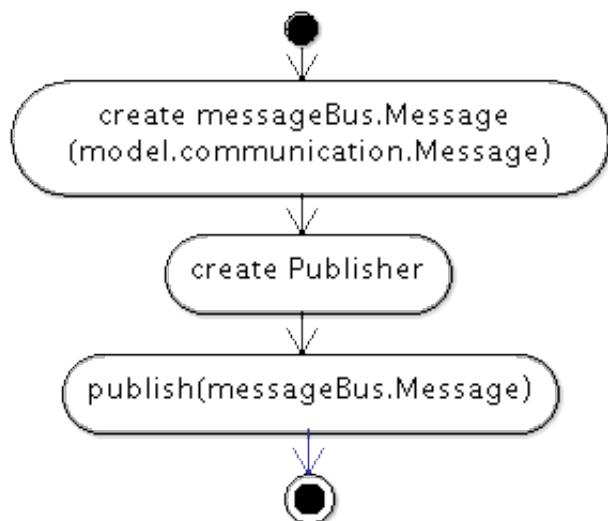


Рисунок 3.7 — Диаграмма действий метода sendMessage компонента ThinkingLifeCycle

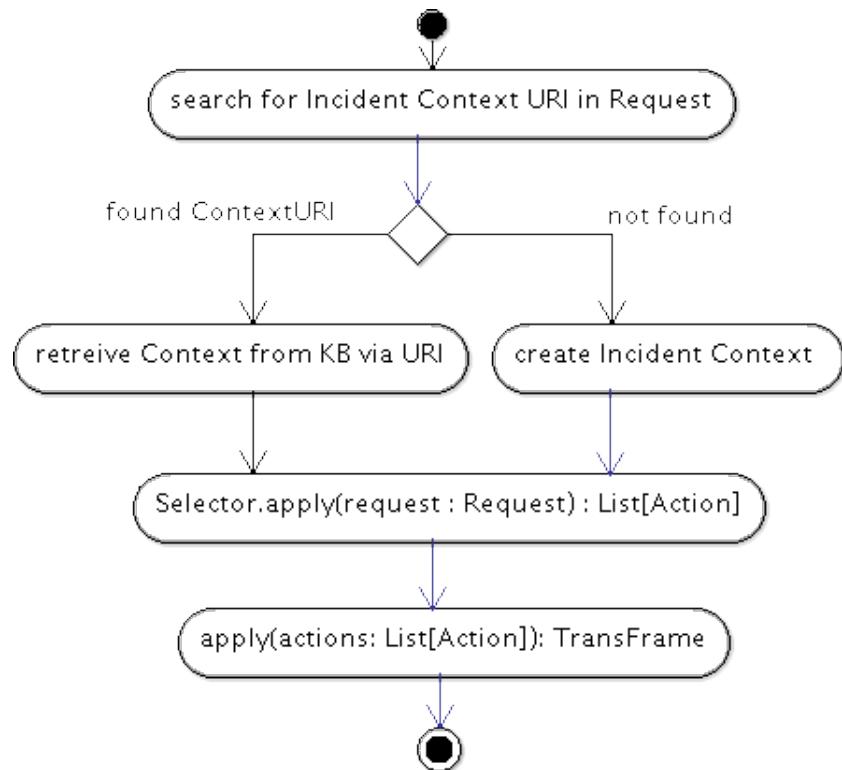


Рисунок 3.8 — Диаграмма действий метода `apply` компонента `ThinkingLifeCycle`

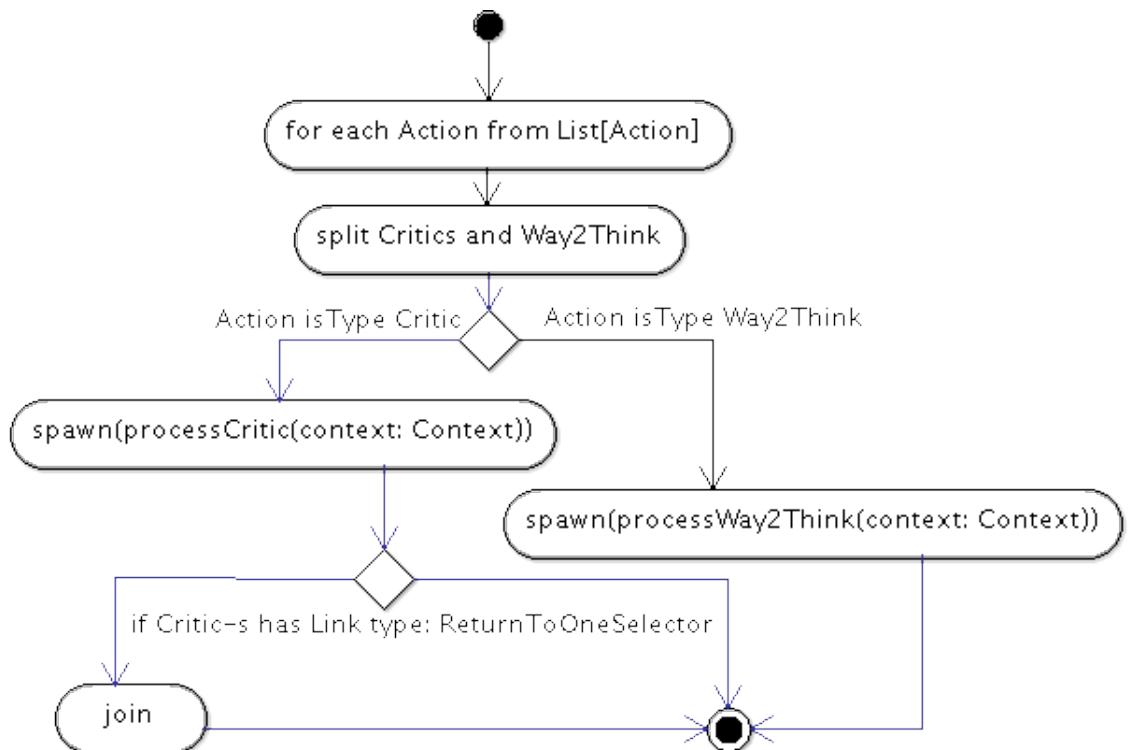


Рисунок 3.9 — Диаграмма действий метода `apply` компонента `ThinkingLifeCycle`

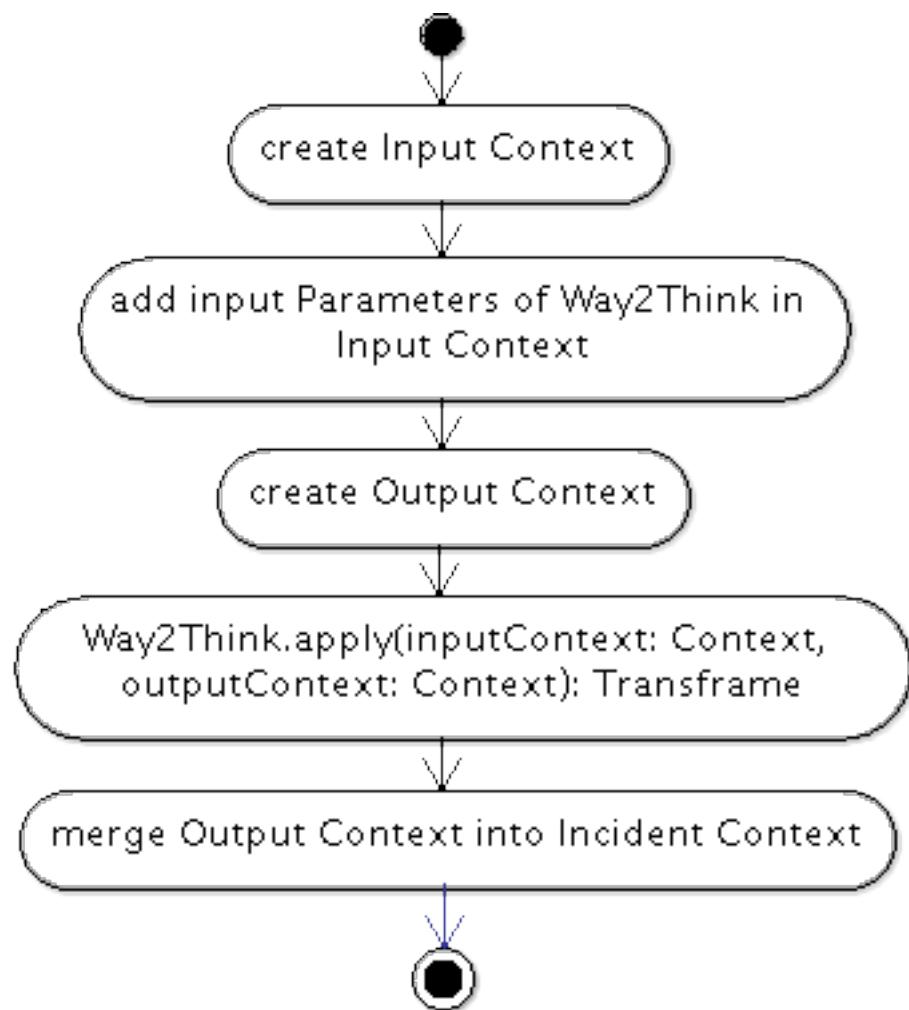


Рисунок 3.10 — Диаграмма действий метода `processWay2Think` компонента `ThinkingLifeCycle`

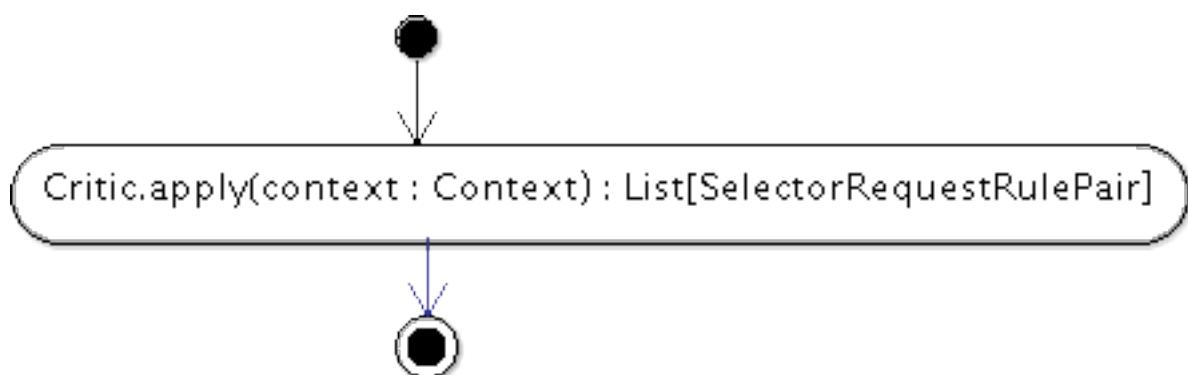


Рисунок 3.11 — Диаграмма действий метода `processCritic` компонента `ThinkingLifeCycle`

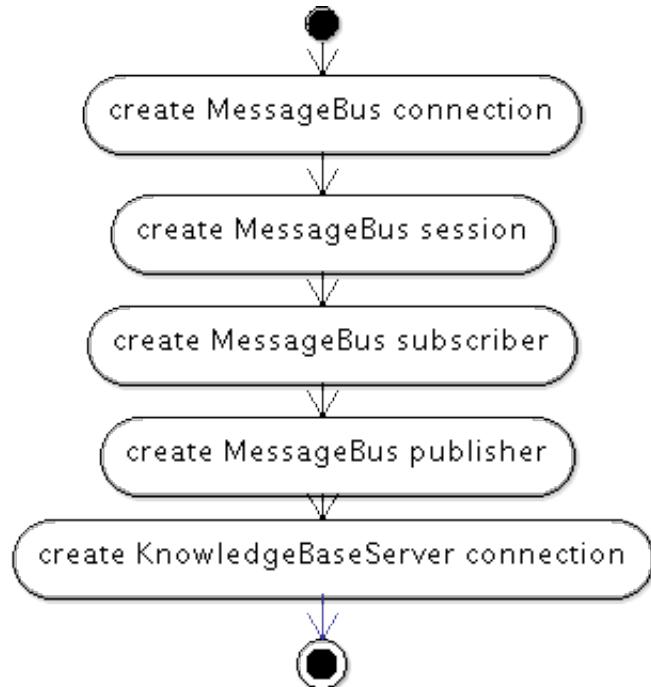


Рисунок 3.12 — Диаграмма действий метода `init` компонента `ThinkingLifeCycle`

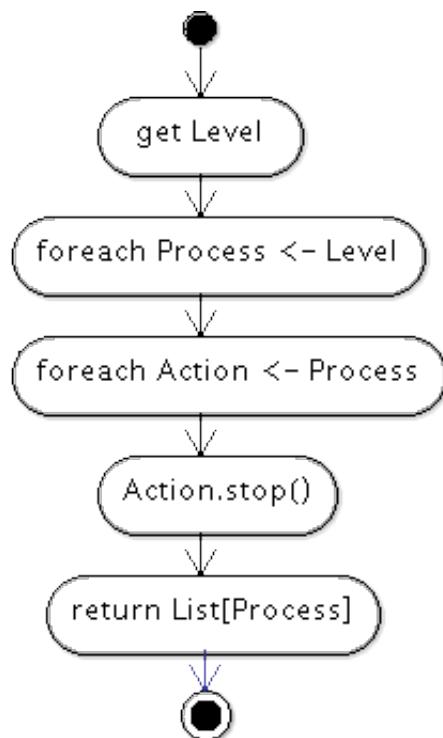


Рисунок 3.13 — Диаграмма действий метода `stop` компонента `ThinkingLifeCycle`

**Описание работы компонента.** Чтобы понять работу компонента, далее рассмотрим описание алгоритма его работы.

*Запуск и остановка.* Когда приложение запускается, оно инициализирует компонент ThinkingLifeCycle (далее TLC), который активирует набор критиков, базируясь на текущей цели системы. Например, если цель — классифицировать инцидент, то активируется набор критиков: разобрать, проверить, классифицировать. Когда приложение останавливается, оно останавливает все объекты класса и подклассов Actions (Critics, WayToThink), Selectors и ThinkingLifeCycle.

Коммуникация с остальными компонентами системы происходит посредством сообщений, отправленных через MessageBus (Шину Данных) JMS [97]. Далее рассмотрим подробнее взаимодействие с остальными компонентами системы.

1. Критик возвращает компоненту ThinkingLifeCycle (далее TLC) список Селекторов (SelectorRequestRule);
  - (a) TLC запускает обработку компонента Selector;
  - (b) Selector возвращает TLC список Action (см. приложение Б) из Базы Знаний;
  - (c) TLC параллельно запускает возвращенные Action.
    - i. Если Action — это Critic;
    - ii. TLC создает InputContext (входной контекст приложения) и копирует туда все данные из Context (контекста) инцидента, созданного пользователем;
    - iii. Если Action — это Critic с ссылками ReturnToSameSelector, то TLC ждет результаты и отправляет компоненту Selector список SelectorRequestRule, которые были возвращены в качестве результата работы Critic. Иными словами, Critic может вернуть новый Selector. В данном случае нам нужно провести операцию Join для всех потоков [98]. В иных же случаях все Action запускаются в параллельных потоках.
    - i. Если Action — это WayToThink;
    - ii. TLC создает InputContext (входной контекст приложения) и копирует туда все данные из Context (контекста), возвращенного Selector;

- iii. TLC (см. таблицу 4.4) запускает WayToThink;
- iv. TLC сохраняет параметры в OutputContext;
- v. TLC сохраняет итоговый результат работы и возвращает его.

В данном разделе было приведено описание основного цикла приложения с примерами работы. В следующих разделах содержится подробное описание работы каждого компонента на более низком уровне. В конце главы приведены развернутое описание стандартного цикла приложения и диаграмма расположения компонентов в разрезе уровней мышления.

### 3.1.4 Компоненты $T^3$

**Селектор (Selector)** — это компонент, который ответственен за получение списка действий и ресурсов из базы знаний, согласно входным параметрам.

**Входной критерий.** TLC запускает Selector с параметрами в виде контекста инцидента, который создал пользователь.

**Выходной критерий.** Selector получает список Action: WayToThink или Critic.

<<interface>>
Selector
apply(request : Request)
apply(goal : Goal)
apply(criticResult : SelectorRequest)
apply(criticResult : SelectorRequestRulePair)
apply(criticResults : List[SelectorRequestRulePair])
apply(criteria : SemanticNetwork)
start()
stop()

Рисунок 3.14 — Интерфейс компонента Selector

На рисунке 3.14 показан интерфейс компонент. В таблице 3.6 приведено описание методов компонента.

Таблица 3.6 — Описание методов класса (компоненты) Selector

Метод	Описание
apply(request : Request) : Action	Данный метод на основе запроса пользователя получает из Базы знаний необходимые Critic 3.1.4. На рисунке 3.15 представлена диаграмма действий этого метода
apply(goal: Goal) : Action	Данный метод на основе цели системы получает из Базы знаний необходимые Critic 3.1.4. На рисунке 3.16 представлена диаграмма действий этого метода
apply(criticResult : ActionProbabilityRule) : Action	Данный метод на основе работы Critic получает из Базы знаний необходимые Action. На рисунке 3.17 представлена диаграмма действий этого метода

Чтобы лучше понять работу компонента и его назначение, далее рассмотрим сценарии его использования и взаимодействия с остальными компонентами на примере работы в режиме классификации входящего запроса.

#### Действия при классификации входящего запроса

1. TLC (см. секцию 3.1.3) запускает входящие Critic (см. секцию 3.1.4) параллельно;
2. Когда Critic возвращает результат работы в виде ActionProbabilityRuleTriple, TLC запускает Selector с этим параметром;
3. Selector запускает GetMostProbableWay2Think, который возвращает наиболее вероятный WayToThink;
4. В некоторых случаях Selector может вернуть менее вероятный вариант, если на уровне мышления Reflective после проверки решения, оно было признано некорректным, или же пользователь признал его таким.

На рисунке 3.18 представлена диаграмма действий классификации инцидента. TLC (см. секцию 3.1.3) получает цель классифицировать инцидент, за-

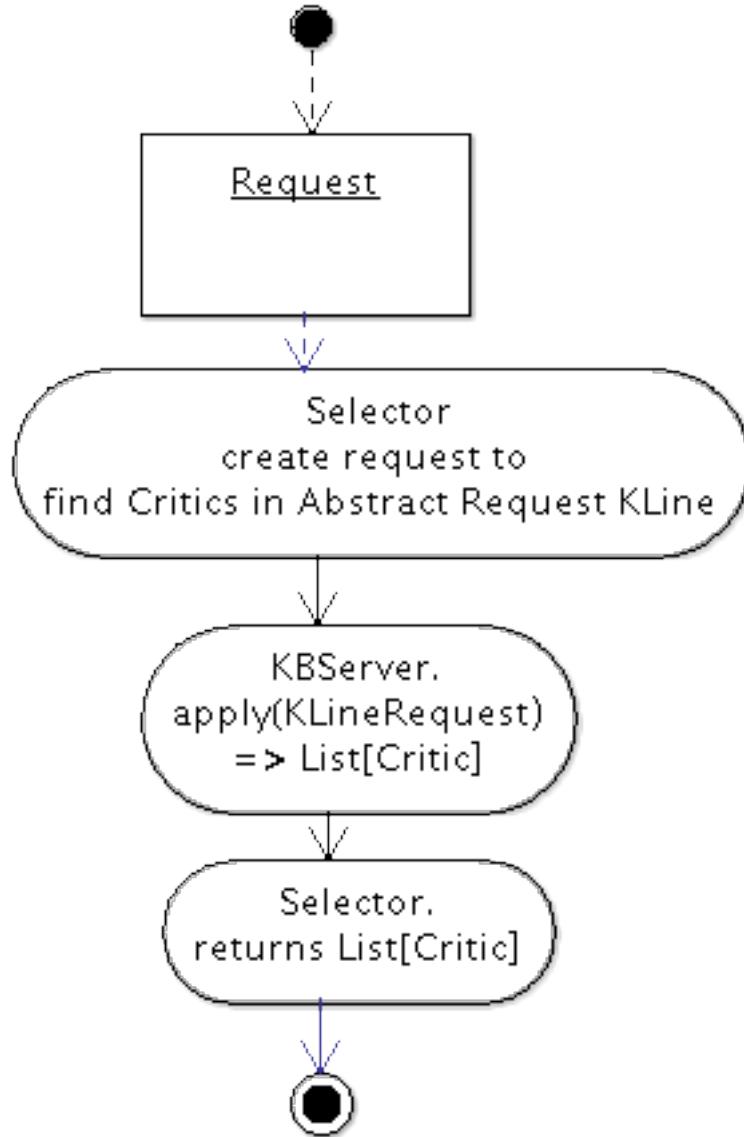


Рисунок 3.15 — Диаграмма действий метода `Selector.apply(request : Request)` компонента `Selector`

тем `Selector` по этой цели возвращает `Critic`. После чего TLC запускает обработку `Critic` в разных потоках (параллельно). В данном случае рассматривается три `Critic`: `DirectInstruction` — прямые инструкции, данный `Critic` возвращает `WayToThink Simulate` (см. секцию 3.1.4), который ищет связь между концепциями в запросе и концепциями в Базе Знаний; `ProblemWithDesiredState` — проблема с ожидаемым результатом, данный `Critic` возвращает `Simulate` и `Reformulate WayToThink`, которые ищут сопоставление концепциями в Базе Знаний и пытаются преобразовать запрос к `DirectInstruction` запросу (прямым инструкциям); `ProblemWithoutDesiredState` — проблема без ожидаемого результата, данный

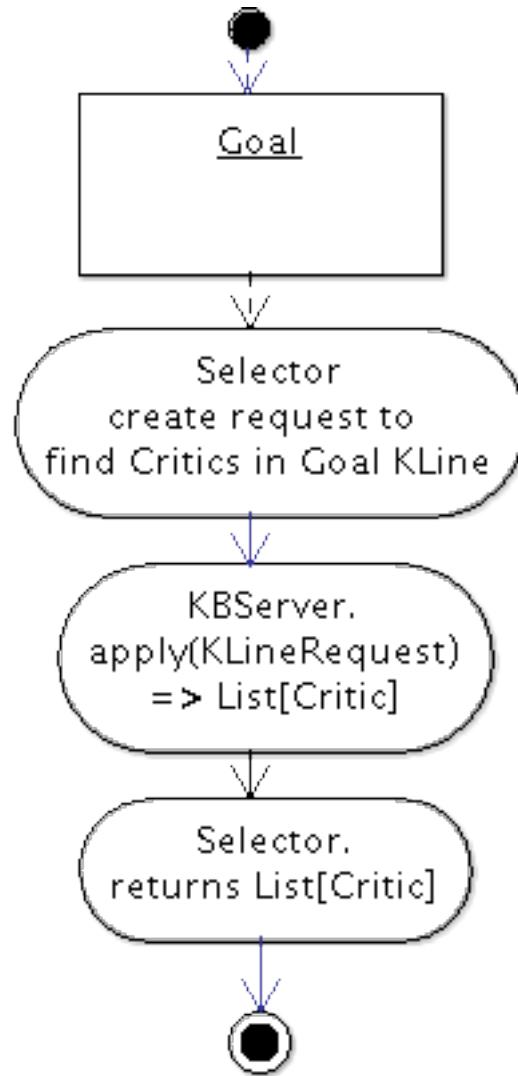


Рисунок 3.16 — Диаграмма действий метода `Selector.apply(goal: Goal)` компонента `Selector`

`Critic` возвращает `Simulate`, `Reformulate`, `InferDesiredState`, который пытается преобразовать проблему к `ProblemWithDesiredState`.

После работы компонента TLC (см. секцию 3.1.3) собирает результаты выполнения всех `Critic` и запускает их, пока не будет достигнута изначальная цель.

В данном разделе была описана работа компонента `Selector` с примерами работы и демонстрацией интеграции с остальными компонентами. Данный компонент тесно связан с компонентом TLC 3.1.3. Особенностью работы компонента является то, что список ресурсов, который он возвращает, можно задавать динамически, он также может формироваться во время работы системы.

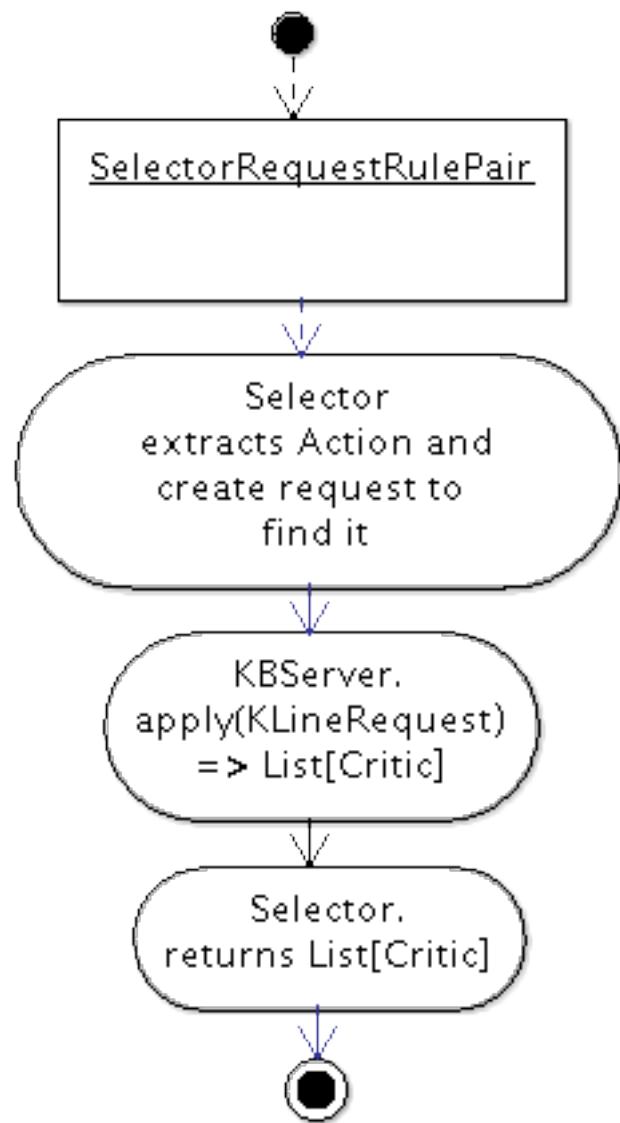


Рисунок 3.17 — Диаграмма действий метода `Selector.apply(criticResult : ActionProbabilityRule)` компонента `Selector`

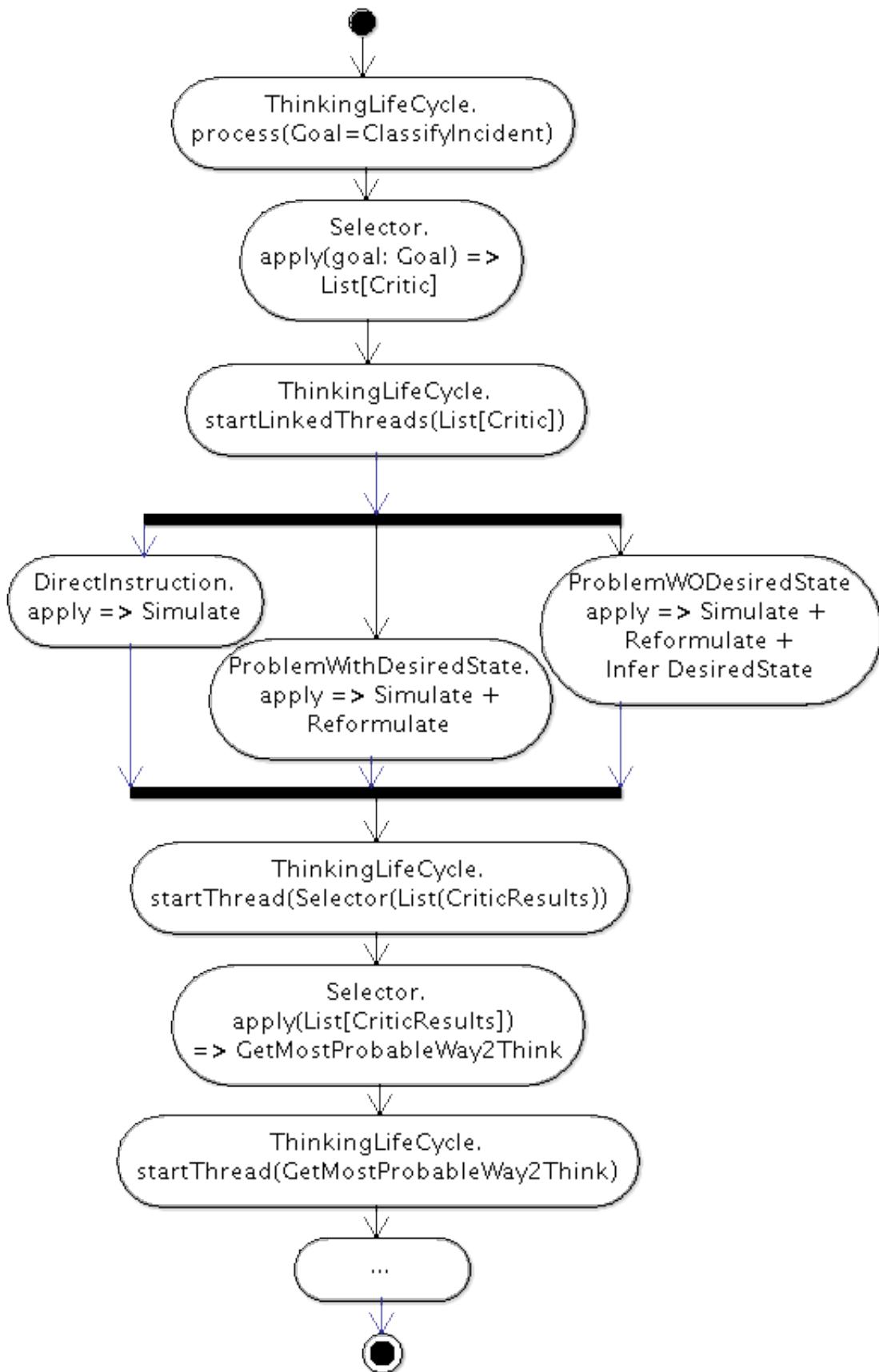


Рисунок 3.18 — Диаграмма действий классификации инцидента

**Critic** является основным компонентом для анализа в  $T^3$ . Critic используеться для классификации входной информации, рефлексии, само-анализа и служит определенным вероятностным переключателем. Например, компоненты контроля времени, контроля эмоционального состояния системы — это тоже Critic.

**Входной критерий.** TLC 3.1.3 запускает Critic согласно Goal (Цель) (см. Приложение Б) или входящему запросу от пользователя.

**Выходной критерий.** Critic генерирует SelectorRequest 3.1.4. На входе Critic принимает: загруженные из базы правила для работы Critic (CriticRules); DomainModel:SemanticNetwork (см. 4.4) — доменная модель, представляющая собой семантическую сеть; описание инцидента, представляющее собой семантическую сеть.

На выходе Critic предоставляет: SelectorRequest 3.1.4 — запрос на выбор Selector из базы знаний; CriticRule — правило, которое сработало для активации. Данное правило является логическим предикатом, т. е. содержит в себе определенную формулу для вычисления вероятности.

На рисунке 3.19 представлена диаграмма действий Critic, описание диаграммы приведено в таблице 3.8. В системе существуют разные типы Critic, их описание доступно в таблице 3.7.

Таблица 3.7 — Описание основных типов Critic, используемых в системе

Critic	Описание
Manager	простой тип критика, который работает как триггер, например, Goal (см. приложение Б), который запускает необходимый WayToThink
Control	контролирующий Critic, который ждет определенного события (срабатывает на определенное событие). Например, заканчивается отведенное на решение время
Analyser	анализатор, обрабатывает и выявляет тип инцидента. Например, прямые инструкции, проблема с желаемым состоянием, выбор наиболее вероятного действия

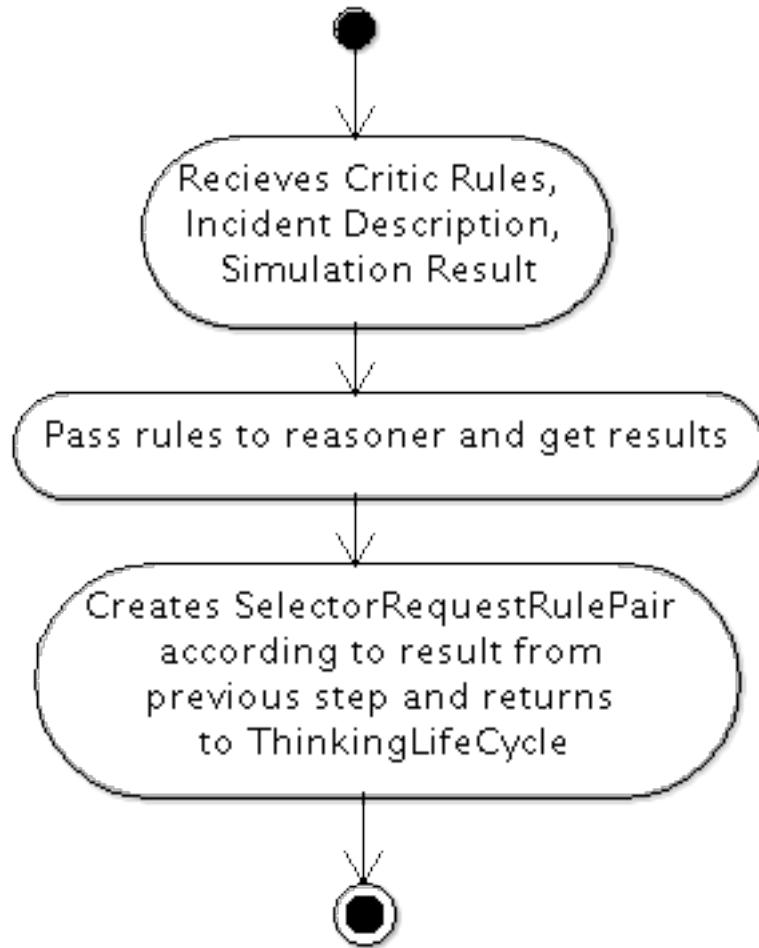


Рисунок 3.19 — Диаграмма действий компонента Critic

Таблица 3.8 — Описание методов компонента Critic

Метод	Описание
exclude():List[CriticLink]	Данный метод возвращает список CriticLink, которые при срабатывание данного Critic будут игнорироваться с определенной вероятностью, после срабатывания Critic будет посчитана суммарная вероятность активации. После чего система решит, какой Critic был вероятнее всего активирован.

**Таблица 3.8 – продолжение**

<b>Метод</b>	<b>Описание</b>
include():List[Critic]	Данный метод возвращает список объектов класса CriticLink, которые при срабатывание данного Critic будут включаться с определенной вероятностью.
apply(currentSituation: SemanticNetwork, domainModel:SemanticNetwork): List[SelectorRequestRulePair]	Данный метод запускает Critic, после чего вернется список Selector (см. секцию <a href="#">3.1.4</a> ) с определенной вероятностью, после чего TLC (см. секцию <a href="#">3.1.3</a> ) их активирует.

Как сказано выше, Critic действуют на разных уровнях мышления. Далее приведено описание Critic с привязкой к уровням мышления.

1. Уровень обученных реакций;
  - (a) PreprocessManager — предобработка информации;
  - (b) Классификаторы инцидентов: Прямые инструкции, Проблема с желаемым состоянием, Проблема без желаемого состояния;
  - (c) SolutionCompletenessManager — связывается с пользователем и проверяет устраивает ли его найденное решение.
2. Уровень рассуждений;
  - (a) Выбор наиболее вероятного Selector по Rule. Данный Critic после проверки правил, выбирает из них правило с большей вероятностью.
3. Рефлексивный уровень;
  - (a) Менеджер целей. Установка целей.
4. Саморефлексивный уровень;
  - (a) ProcessingManager — запускает выполнение запроса;
  - (b) TimeControl — контроль времени исполнения запроса;
  - (c) DoNotUnderstandManager — активируется, когда необходимо уточнение пользователя для продолжения работы.
5. Самосознательный уровень.
  - (a) EmotionalStateManager — контроль общего состояния системы.

Основным примером работы компонента может служить классификация инцидентов (подробнее рассматривалось ранее). Например, у нас есть Critic для прямых инструкций DirectInstruction, есть для ситуации с желаемым состоянием DesiredState. Пусть на входе будет запрос вида: install antivirus. DesiredState найдет здесь действие — install, но не найдет желаемого состояния, то есть вероятность его выполнения будет 60%. DirectInstruction будет искать действие и объект, которые присутствуют в запросе, его вероятность будет 100%, его TLC (см. секцию 3.1.3) и активируют как наиболее вероятный.

Это был простой пример работы компонента, на самом деле механизм работы гораздо гибче: он поддерживает включения, исключения, составные правила и логику. Компонент также является динамически формируемым.

В данном разделе был описан важный компонент системы, который определяет алгоритм ее работы, в этом компоненте скрыта основная возможность системы думать и решать, согласно набором правил и внешним обстоятельствам.

**WayToThink** является основным операционным компонентом  $T^3$ . Основными задачами данного компонента являются: обновление, преобразование, сохранение данных и коммуникация с пользователем, иными словами, все, что в той или иной форме изменяет операционный контекст данных системы.

**Входной критерий.** Запуск осуществляется из компонента ThinkingLifeCycle (см. секцию 3.1.3). Входными данными является InputContext, который содержит параметры WayToThink.

**Выходной критерий.** WayToThink завершил работу. На выходе возвращают измененные в ходе работы данные. В общем виде компонент описывает последовательность действий. В системе используется два больших класса WayToThink — простой и составной (сложный). Простые WayToThink являются встроенными в систему, остальные являются комбинацией компонентов: Critic 3.1.4, Selector 3.1.4, WayToThink 3.1.4. В таблице 3.9 приведено описание встроенных в систему WayToThink. На рисунке 3.20 представлен интерфейс компонента. В таблице 3.10 представлено описание методов WayToThink.

Таблица 3.9 — Описание встроенных в систему WayToThink

WayToThink	Описание
Создать контекст	Данный WayToThink создает объект Context для аккумуляции данных запроса

**Таблица 3.9 – продолжение**

<b>WayToThink</b>	<b>Описание</b>
Установить общий статус системы	Данный WayToThink устанавливает состояние системы в глобальном контексте
Установить цель системы	Данный WayToThink устанавливает цель запроса в текущем контексте <b>Б</b>
Разделить фразу на слова и предложения	Данный WayToThink разбивает фразу на слова и возвращает список слов
Найти связи между входной информацией и базой знаний	Данный WayToThink ищет связь между входной информацией и базой знаний
Извлечь связи	Данный WayToThink возвращает список связей из фразы
Сохранить наиболее вероятное решение	Данный WayToThink сохраняет наиболее вероятное решение
Перефразировать (Reformulate)	Данный WayToThink ищет связь между текущим контекстом и известными проблемами, если есть неизвестные концепции, то он пытается их переформулировать при помощи пользователя
Смоделировать (Simulate)	Данный WayToThink ищет связь между текущим контекстом и проблемами уже сохраненными в базе
Найти решение	Данный WayToThink производит поиск решения, которое прикреплено к проблеме, которая была найдена при помощи моделирования и перефразирования
Остановить работу	Данный WayToThink останавливает работу системы

Таблица 3.10 — Описание методов компонента WayToThink

<b>Метод</b>	<b>Описание</b>
start()	Запустить обработку информации

**Таблица 3.10 – продолжение**

<b>Метод</b>	<b>Описание</b>
stop()	Остановить обработку, например, если выполнение идет слишком долго
apply(inputContext:Context): Context	Применить WayToThink. Исполнение начнется только после вызова метода start

WayToThink также используется как описание алгоритма разрешения проблемы (см. приложение **B HowTo**), то есть описывает последовательность действий, необходимых для устранения проблемной ситуации.

В зависимости от типа WayToThink активируется та или иная последовательность действий. В общем виде последовательность имеет следующий вид: получить данные, обработать, вернуть данные. В случае, например, WayToThink Simulate второй шаг имеет вид «найти связь между текущим контекстом и проблемами, уже сохраненными в Базе Знаний». Если же WayToThink является описанием решения, то второй шаг может быть набором вызова системных утилит с параметрами из первого шага.

В данном разделе был описан основной компонент модификации данных WayToThink. Нужно отметить, что данный компонент также является важной частью модели TU. Проектировался он для универсального применения во всех случаях, когда необходимо действие над данными. Например, если нужно использовать скрипт, который был написан на языке интерпретации Bash, и ввести его в систему, можно разбить каждый шаг и вызов на отдельную часть и сделать сложный WayToThink с ветвлениями и циклами.

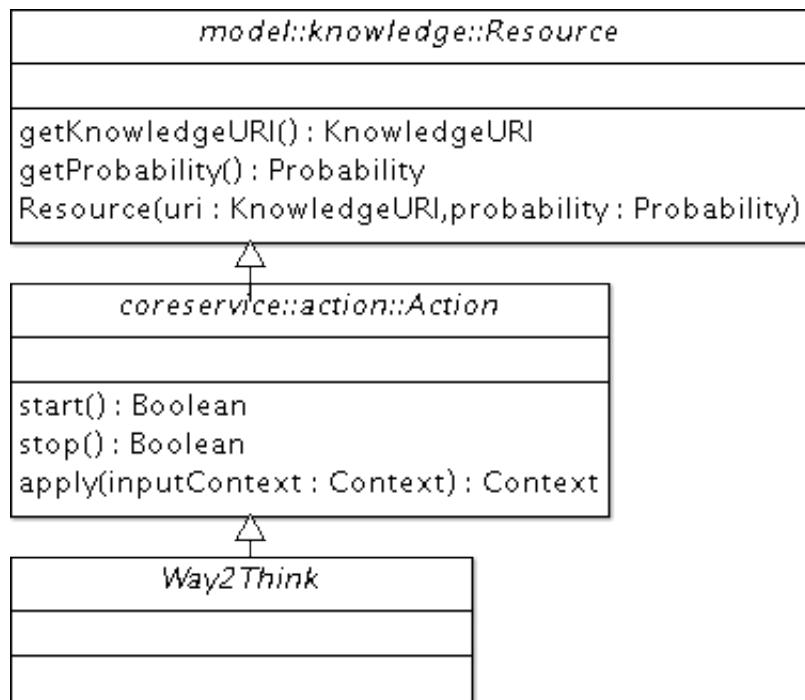


Рисунок 3.20 — Интерфейс компонента WayToThink

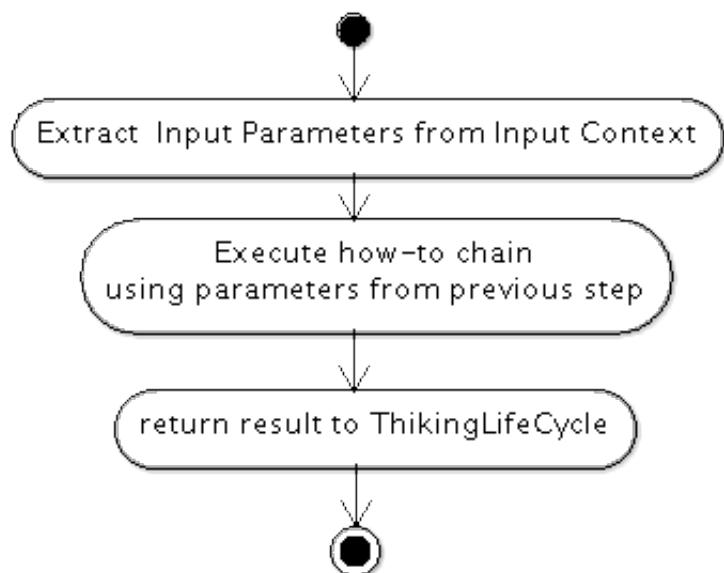


Рисунок 3.21 — Работа компонента WayToThink в режиме описания решения проблемы (HowTo)

### 3.1.5 Вспомогательные компоненты

Данный компонент проводит предварительную подготовку текста: грамматическую и орфографическую коррекции текста, а также разделение на предложения. На рисунке 3.22 представлен интерфейс компонента. Компонент также является WayToThink, так как он производит модификацию данных контекста. В таблице 3.11 приведено описание методов класса.

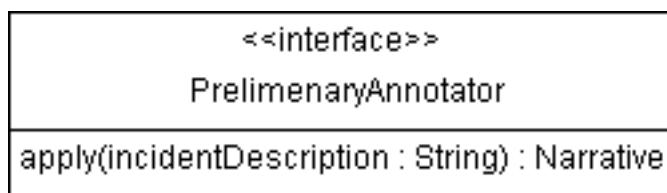


Рисунок 3.22 — Интерфейс компонента PreliminaryAnnotator

Таблица 3.11 — Описание методов компонента PreliminaryAnnotator

Метод	Описание
apply(incidentDescription:String): Narrative	Данный метод запускает обработку входного текста и его корректировку

С точки зрения корректировки текст подвергается обработке средствами проверки языка, например, открытый комплекс After the deadline [99], а также Google API. В части разбиения текста на слова используется алгоритм из открытого комплекса Link Grammar [100]. В целом компонент содержит в себе также составную системы разбора, что отличает его от прямого использования алгоритма. Он манипулирует результатом работы из нескольких подсистем, для увеличения степени точности.

Одной из особенностей компонента является использование внутренней базы знаний для предобработки текста, чтобы убрать неточности, которые будут мешать работе средств NLP. Например, часто средства NLP не понимают концепцию слова *please*, поэтому в базе изначально хранится эта концепция с правильным значением. Таким образом, на вход средствам NLP поступает уже аннотированный текст, что позволяет на 20% (согласно экспериментальным данным) увеличить точность обработки.

## **Компонент CoreService.KnowledgeBaseAnnotator**

Данный компонент устанавливает связи между терминами во входной фразе и базой знаний и также является WayToThink 3.1.4.

**Входные критерии.** Список подготовленных фраз в виде объектов.

**Выходные критерии.** Список ссылок на внутренние знания.

### **Описание работы компонента**

1. Получен Термин;
2. Поиск в локальной базе знаний;
3. Если совпадение не найдено идет запрос во внешнюю базу знаний;
4. Внешняя база возвращает список синонимов;
5. Компонент ищет по синонимам во внутренний базе знаний;
6. Если поиск успешен, то создается связь между входящем термином, синонимом и концепцией в базе знаний.

Например, входящий запрос содержит термин "program", база знаний содержит термин "computer software". Идет запрос во внешние базы знаний, найдено "computer software, program". Будет добавлена связь-аналогия в база знаний "program — computer software".

## **Компонент DataService**

Данный компонент отвечает за хранение данных в системе. База знаний построена на графах. На рисунке 3.23 представлен интерфейс компонента. В базе знаний используется два типа объектов Object — объект базы знаний, BusinessObject — объект для Web Service (User, Request). BusinessObject является кортежем для интеграции с внешними системами. У объекта есть ID, который уникально удостоверяет его в рамках системы. В таблице 3.12 приведено описание методов компонента.

Таблица 3.12 — Описание методов компонента DataService

<b>Метод</b>	<b>Описание</b>
save(obj:Resource): Resource	Данный метод позволяет сохранить ресурс в базу знаний
remove(obj:Resource)	Данный метод позволяет удалить объект
select(obj:Resource): Resource	Данный метод позволяет выбрать объект
link(obj>List<Resource>, linkName:String)	Данный метод позволяет сделать ссылку между 2-мя объектами

**Таблица 3.12 – продолжение**

<b>Метод</b>	<b>Описание</b>
selectLinkedObject(obj:Resource, linkName:String): Link<Resource>	Данный метод позволяет выбрать все объекты, которые имеют связь под названием linkName с объектом obj
addLinkedObject(parent:Resource, toLink:Resource, linkName:String)	Данный метод позволяет создать ссылку linkName с объектом
saveRequest(obj:Request)	Данный метод позволяет получить запрос из Базы Знаний
selectRequest(obj:RefObject)	Данный метод позволяет получить запрос из Базы Знаний
saveBusinessObject(obj:RefObject): RefObject	Данный метод позволяет сохранить объект в базу
selectBusinessObject(obj:RefObject): RefObject	Данный метод позволяет получить объект из Базы Знаний

### Компонент Reasoner

Данный компонент осуществляет логические вычисления для системы, например, для обработки правил в компоненте Critic 3.1.4. На рисунке 3.24 представлен интерфейс компонента. В таблице 3.13 приведено описание методов компонента.

Таблица 3.13 — Описание методов компонента Reasoner

<b>Метод</b>	<b>Описание</b>
apply(request: ReasonerRequestNode): ReasonerRequestNode	Данный метод проводит обработку правил и считает вероятность (Probability) и уверенность (Confidence)

На данный момент в качестве реализации в системе используется два движка логический вычисление PLN [101] и NARS [42]. Основным применением данного компонента являются правила для Critic. Логика правил обрабатывается при помощи этого компонента. Этот результат используется для определения вероятности активации данного Critic. Подобное использование дает гибкость в построении свода правил.

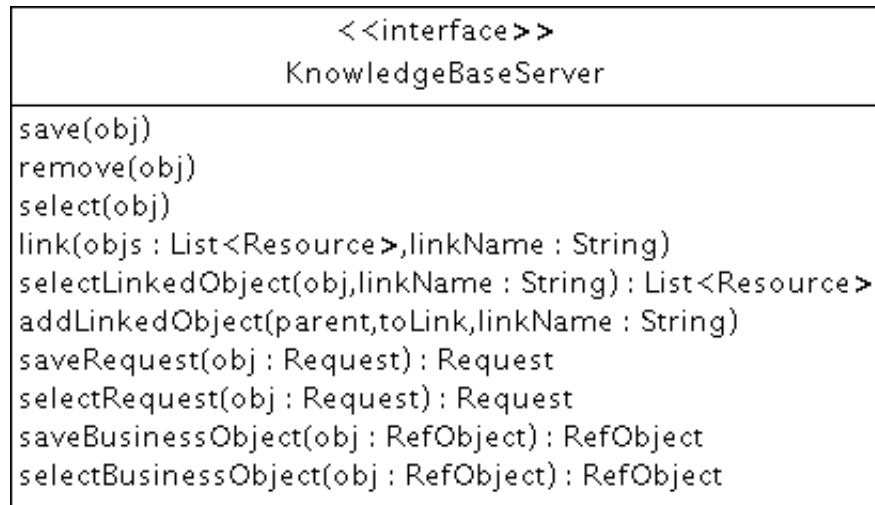


Рисунок 3.23 — Интерфейс компонента KnowledgeBaseServer

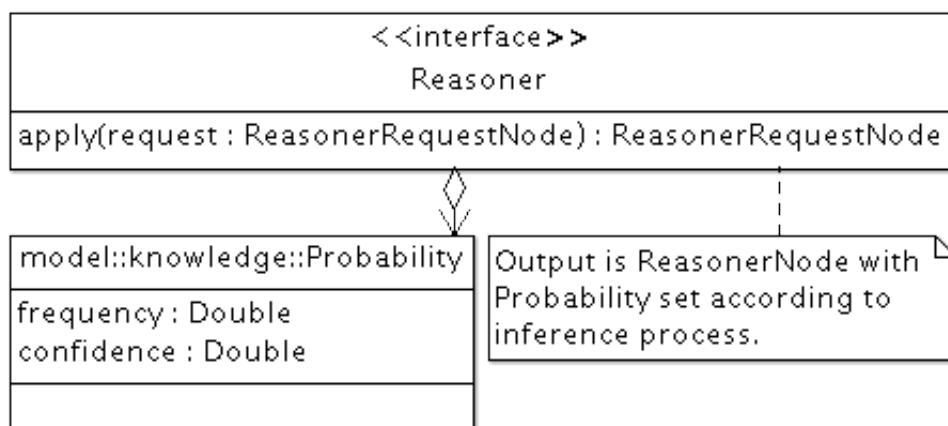


Рисунок 3.24 — Интерфейс компонента Reasoner

### 3.2 Модель данных TU Knowledge

Одной из важных частей системы является реализация хранения данных на основе модели TU. Для работы системы была разработана уникальная схема данных — TU Knowledge, которая сочетает в себе OWL и графовую базу данных. Язык OWL, традиционно использующийся для структурирования информации в Вебе [77], обрел широкое использование во многих схемах данных, так как давал возможность дополнительного расширенного описания взаимосвязи между данными. На рисунке 3.25 представлена схема данных TU Knowledge. В таблице 3.14 представлено описание схемы TU Knowledge.

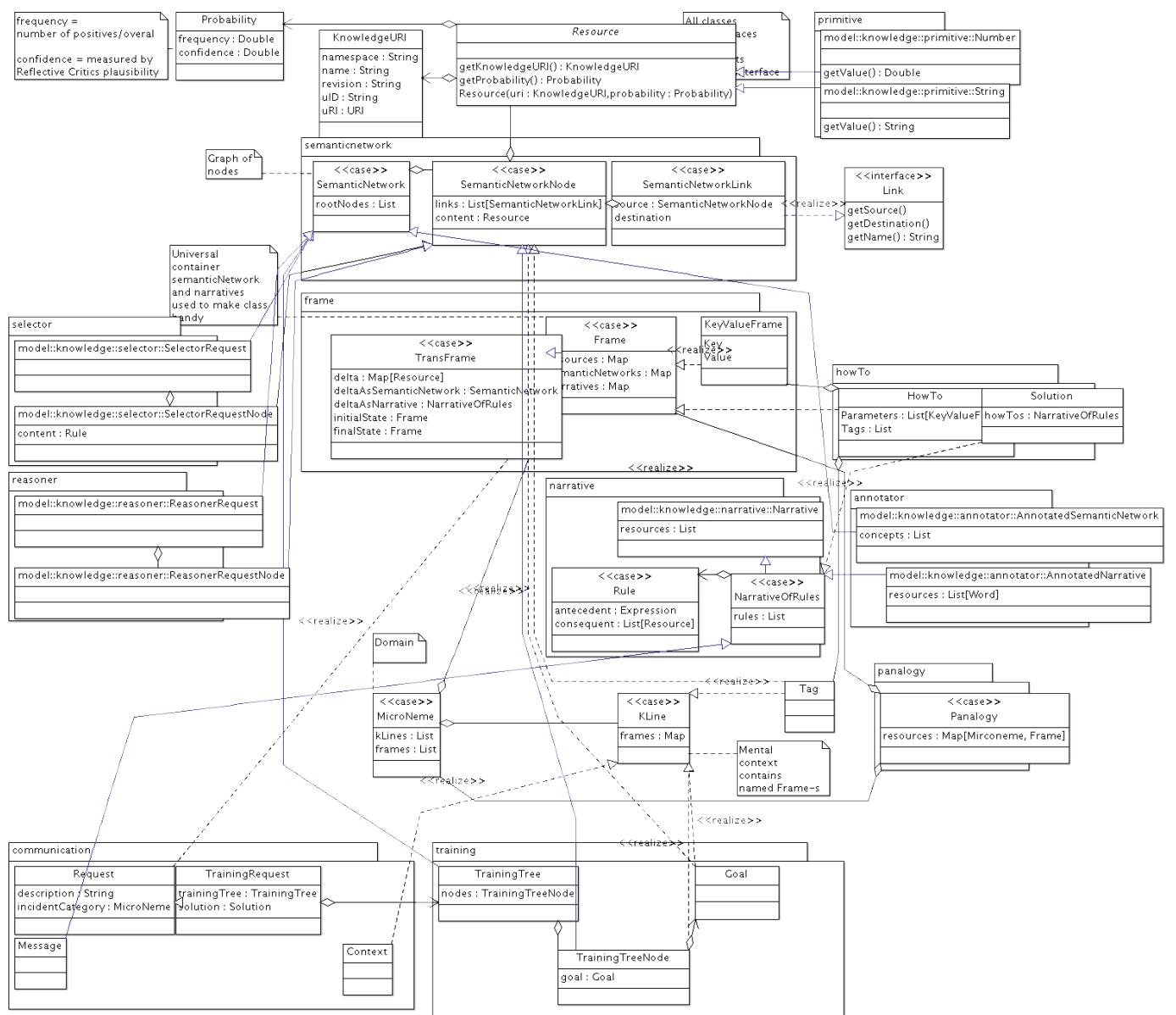


Рисунок 3.25 — Схема данных TU Knowledge в формате UML

Таблица 3.14 — Описание классов TUKnowledge

Класс	Описание
Knowledge	Базовый класс всех объектов модели. Содержит в себе URI, по которому уникально идентифицируется. Поддерживает версионность. Свойствами данного объекта обладают все объекты системы. Также содержит Probability (Вероятность) и Confidence (Уверенность) поля. Например, когда в результате работы WayToThink получается Knowledge он имеет Confidence 0, так как он только что был сгенерирован, когда его проверит Critic на его состоятельность при помощи определенных в Critic правил, то он поставит ему не 0 Confidence
Narrative	Список слов исходного запроса
Rule	Правило. Класс описывающий правила в системы. Например, правило по которому сработает Critic <a href="#">3.1.4</a>
AnnotatedNarrative	Слова исходного запроса и их сопоставление на концепции в Базе Знаний
SemanticNetwork	Граф из SemanticNetworkNode и SemanticNetworkLink
SemanticNetworkNode	Узел графа SemanticNetwork, содержит в себе ссылки на другие узлы, а также ссылку на Knowledge
SemanticNetworkLink	Ссылка в графе SemanticNetworkLink
Frame	Коллекция объектов Knowledge, с возможностью назначения специального атрибута (тега) для семантической группировки
TransFrame	Коллекция Frame, содержащая два состояния одного фрейма: до и после
Goal	Цель. Приложение <a href="#">Б</a>
Tag	То же, что и цель, но используется для меток
Preliminary annotation	SemanticNetwork входного запроса
KnowledgeBase annotation	SemanticNetwork с сопоставлением концепциям Базы Знаний
Domain model	SemanticNetwork доменной модели

**Таблица 3.14 – продолжение**

<b>Класс</b>	<b>Описание</b>
Situation model	SemanticNetwork, часть DomainModel, созданной для обработки текущего запроса. Приложение <b>Б</b>
Incident	SemanticNetwork входного запроса к системе
K-Line	Связь между объектами. Например, когда в систему поступает запрос она создает K-Line между Conversation, Narrative
Conversation	SemanticNetwork, контекст инцидента
InboundRequest	SemanticNetwork входного запроса
Training Request	SemanticNetwork входного запроса для обучений

### **Описание запросов в рамках TU Knowledge**

В рамках модели данных TU Knowledge проблема имеет следующее описание:

- Область (Микронема);
- Дата обращения;
- Автор;
- Приоритет;
- Категория;
- Теги;
- Описание.

Запрос на обучение включает следующие части:

- Область (Микронема);
- Дерево обучения;
- Ограничения (например, время на решение).

**Микронема** описывает контекст работы системы. В разрезе человеческого мышления это область работы мозга с нейронами и связями. Сочетание разных микронем может привести к изменению взглядов человека, характера, например, когда кто-то узнает новую идею и это заменяет его предыдущие представления о том или ином явлении.

**Дерево обучения** базируется на структуре цель–подцель (см. приложение **Б**). Получение системой целей происходит из подцелей или при получении запро-

са пользователя. Например, при запросе пользователя устанавливается цель "Help User (Помочь пользователю)".

Во время процедуры обучения возможно, что на одном уровне окажется несколько целей, тогда необходимо провести дополнительное уточнение, если такое невозможно, то будет выбрана первая цель.

Во время работы MostProbableWay2Think может использовать несколько WayToThink, в таком случае он возьмет первый (наиболее вероятный путь). Если в результате его использования цель достигнута не будет, то будет выбран менее вероятный.

### Пример

```

1. SubGoal = Resolve incident
2. SubGoal = ParseIncidentDescription, Way2Think = ProcessText:
   KnowingHow, SemanticNetWorkWithKLines =
{
5 nsubj(received-3, User-1)
 aux(received-3, had-2)
 root(ROOT-0, received-3)
 amod(application-5, wrong-4)
 dobj(received-3, application-5)
10
 advmod(received-3, However-1)
 nsubj(received-3, user-2)
 ccomp(received-8, received-3)
 amod(version-5, wrong-4)
15 dobj(received-3, version-5)
 nsubj(received-8, user-7)
 root(ROOT-0, received-8)
 nn(Tehcnical-10, Wordfinder-9)
 dobj(received-8, Tehcnical-10)
20 advmod(of-12, instead-11)
 prep(Tehcnical-10, of-12)
 nn(Economical-14, Business-13)
 pobj(of-12, Economical-14)
}
25 2. SubGoal = UnderstandIncidentType, Critics = Deliberative,
    Type = ProblemDescription with DesiredState
3. SubGoal = ModelCurrentSituation using ProjectDomain Model,
   Way2Think = Simulate, Model =
{
```

```

User Desired(ordered) Soft(Wordfinder Business Economical)
Operator Installed Soft(Wordfinder Tehcnical) - wrongly
30 }
    3. SubGoal = FormalizeProblemDescription using ProblemModel(
        Wrong state, Desired state), Way2Think = Reformulate,
        Model=
    {
WrongState = Soft.installed(Wordfinder Tehcnical), Soft.
    notInstalled(Wordfinder Business Economical)
DesiredState = Soft.installed(Wordfinder Business Economical),
    Soft.unInstalled(Wordfinder Tehcnical)
35 }
    3. SubGoal = Find solution, Way2Think = ExtendedSearch,
        Solution =
    { Install(Wordfinder Business Economical), UnInstall(
        Wordfinder Tehcnical) }

```

### 3.3 Прототип системы

В прототипе были реализованы 4 уровня мышления. Ниже описан стандартный поток системы, который дает возможность понять основной принцип работы.

1. Поступает запрос пользователя: "User had received wrong application. User has ordered Wordfinder Business Economical. However she received wrong version, she received Wordfinder Tehcnical instead of Business Economical. Please assist." («Пользователь получил неверное приложение. Пользователь заказал приложение "Wordfinder. Бизнес версия", но получил неверную версию, — "Wordfinder. Техническая версия". Пожалуйста, помогите»);
2. Компонент GoalManger (Менеджер целей) устанавливает цель системы HelpUser (Помочь пользователю);
3. Главный компонент Thinking Life Cycle (далее TLC) активирует набор компонентов Critic (Критик), привязанный к данной цели (HelpUser);
4. Активируется компонент PreliminaryAnnorator (Предварительный обработчик), который разбирает запрос, проводя орфографическую коррекцию и предварительный разбор;
5. Компонент KnowledgeBaseAnnotator (разбор при помощи накопленных знаний) создает семантическую сеть и ссылки на нее;
6. Компонент Critic (Критик), привязанный к цели HelpUser на Рефлексивном уровне, запускает WayToThink (Образ мышления) ProblemSolving (Разрешить проблемную ситуацию) с целью: ResolveIncident;
7. Компонент Critic на Рефлексивном уровне выбирает WayToThink KnowingHow (Поиск рецепта решения);
  - (a) Запускаются параллельно все компоненты класса Critic, которые привязаны к цели ResolveIncident (Решить проблему), в данном случае это DirectInstruction (прямые инструкции), ProblemWithDesiredState (проблемы с желаемым состоянием), ProblemWithoutDesiredState (проблема без желаемого состояния);
  - (b) Компонент Selector (Селектор) выбирает среди всех результатов наиболее вероятный результат работы. В данном случае им будет Problem Description with desired state (Проблема с желаемым состоянием);

- (c) Компонент KnowingHow сохраняет варианты выбора Selector;
  - (d) Компонент Simulation (Моделирование) WayToThink с параметрами «создать модель текущий ситуации» создает: концепцию существующей ситуации (CurrentState), концепцию пользователя, концепцию программного обеспечения;
  - (e) Компонент Reformulation WayToThink (Компонент дополнения), используя результаты предыдущего шага, синтезирует артефакты, которых не хватает, чтобы получить из CurrentState DesiredState (Желаемое состояние), так как он не указан явно. WayToThink запускает Critic размышления, чтобы найти корень проблемы. Он находит CurrentState (настоящее состояние) — Wordfinder Technical и DesiredState (состояние, которое нужно пользователю) — Wordfinder Business Economical;
  - (f) Рефлексивные Critic оценивают состояние системы — на каком шаге она находится, и если цель не достигнута, то запускают другой WayToThink, например, DirectInstruction;
  - (g) Компонент Critic Solution Generator (Компонент генерации решения) запускает KnowingHow WayToThink, ExtensiveSearch (Поиск решения);
  - (h) Компонент Selector выбирает наиболее вероятный образ мышления. В данном случае это будет ExtensiveSearch, который будет находить решения, позволяющие привести систему в необходимое пользователю состояние (DesiredState), если сделать это невозможно, то система инициирует коммуникацию с пользователем.
8. Рефлексивный Critic проверяет состояние системы. Если Цель достигнута, то пользователю посыпается ответ.
9. На данном шаге активируются компоненты класса Critic на самосознательном уровне, которые сохраняют информацию о затратах на решение.
- На рисунке 3.26 представлена UML-диаграмма действий системы, согласно алгоритму, описанному выше и с привязкой к уровням мышления.

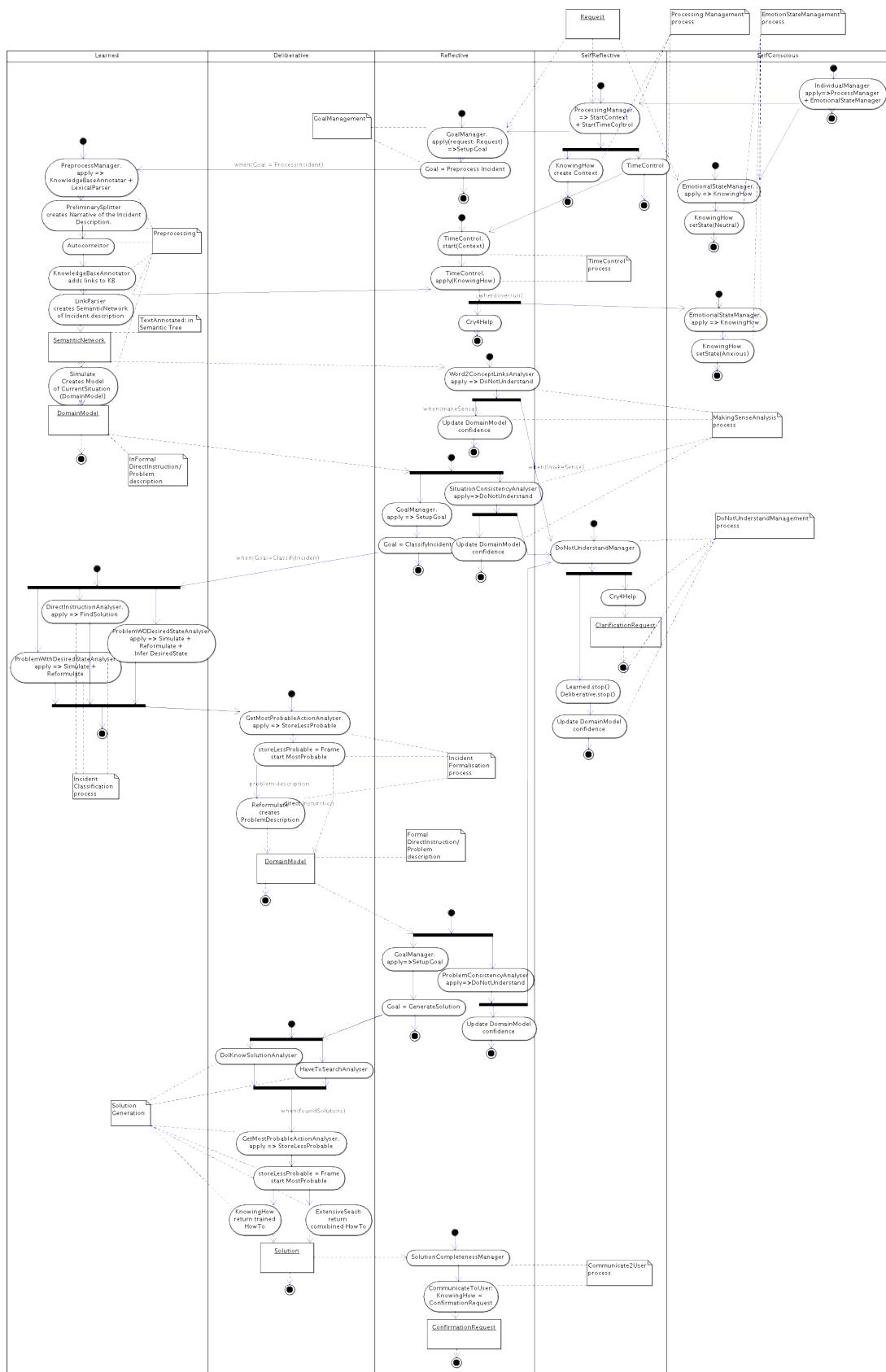


Рисунок 3.26 — Диаграмма действий LifecycleActivity

### 3.4 Выводы по главе 3

В данной главе были рассмотрены:

- архитектура системы по модели TU;
- модель данных TU Knowledge;
- реализация системы;
- состав прототипа;
- основной поток действий системы.

Кроме того, приведены алгоритмы и методы, использованные при создании системы; рассмотрены технологии, использованные при создании прототипа. Для удобства основные диаграммы выполнены с использованием универсального формата UML 2.0. В главе продемонстрированы основные потоки работы как для каждого компонента, так и для всех компонентов в целом.

По данной архитектуре была выполнена программная реализация с использованием функционального языка Scala. Основной платформой для эксплуатации системы был выбран Debian дистрибутив системы Linux, точнее, Ubuntu 12 (и выше). Связано это, прежде всего, с тем, что ряд компонентов был написан на C++ и использует библиотеки, доступные только на Linux.

Система была протестирована на экспериментальных данных, которые представлены в Приложении Г и предоставлены ОАО «АйСиЭл КПО-ВС (г. Казань)», о чем свидетельствует акт о внедрении (см. Приложение Е).

## Глава 4. Экспериментальные исследования эффективности работы модели ТУ

### 4.1 Экспериментальные данные

В качестве экспериментальных данных были взяты выгрузки проблем из информационных систем ОАО «АйСиЭл КПО-ВС (г. Казань)». Для начального обучения в систему заложено две базовые концепции: Object — объект, который является базовой концепцией для всех объектов и Action — действие, которое является базовой концепцией для всех действий. В таблице 4.1 представлен список основных тренировочных данных.

Таблица 4.1 — Описание экспериментальных данных

Входное предложение	Описание
Tense is kind of concept (Время — это концепция).	Обучающий запрос. Создает связь между концепцией Tense и Concept
Please install Firefox (Установите Firefox).	Запрос. Пользователь просит установить Firefox. Результатом должен быть найдено решение по установки Firefox
Browser is an object (Браузер — это объект).	Обучающий запрос. Создает связь между концепцией Browser и object
Firefox is a browser (Firefox — это браузер).	Обучающий запрос. Создает связь между концепцией Firefox и browser
Install is an action (Установить — это действие).	Обучающий запрос. Создает связь между концепцией Install и action

**Таблица 4.1 – продолжение**

<b>Входное предложение</b>	<b>Описание</b>
User miss Internet Explorer 8 (У пользователя нет Internet Explorer 8).	Запрос. Проблема с желаемым состоянием (DesiredState)
User needs document portal update (Пользователю требуется обновление документов).	Запрос. Проблема с желаемым состоянием
Add new alias Host name on host that alias is wanted to: hrportal.lalala.biz IP address on host that alias is wanted to: 322.223.333.22 Wanted Alias: webadviser.lalala.net (Добавьте, пожалуйста, новую ссылку на hrportal.lalala.biz через 322.223.333.22).	Запрос. Сложная проблема
Outlook Web Access (CCC) — 403 — Forbidden: Access is denied (Нет доступа к Outlook Web Access (CCC)). Сложная проблема	
PP2C — Cisco IP communicator. Please see if you can fix the problem with the ip phone, it's stuck on configuring ip + sometimes Server error rejected: Security etc (PP2C — коммуникатор Cisco IP. Пожалуйста, помогите исправить проблему с ИП-телефоном, он застревает во время конфигурирования и иногда показывает ошибку «Безопасность»).	Запрос. Сложная проблема

Полный список информации об экспериментальных данных представлен в приложении Г.

## 4.2 Оценка эффективности

Для верификации экспертной системы поддержки принятия решений ТУ была выбрана область поддержки информационной инфраструктуры предприятия, которая в рамках работы исследована и смоделирована в Главе 1. Для доказательства жизнеспособности решения производилась верификация в 2 этапа:

- Этап 1. Разбор входящего запроса на естественном языке и вычленение концепции;

- Этап 2. Обработка по разработанной архитектуре и реализации модели мышления.

Для Этапа 1 использовалась отфильтрованная выгрузка инцидентов. Были выявлены уникальные инциденты — 1000. На данном этапе удалось добиться качества разбора на уровне 67%. Успешным считался разбор, когда правильно были определены концепции, например, существительное определялось как существительное, глагол как глагол.

Для Этапа 2 использовалась часть инцидентов, которая представлена в предыдущий главе. На них запускался программный комплекс и анализировались результаты. Удалось добиться 95% успешных инцидентов. Успешным считался инцидент, который был успешно сопоставлен концепциям в базе знаний.

Результатом успешной обработки инцидента считалось найденное решение, если же решения не было, то проверялось правильное понимание системой всех концепций, так как решения не было в базе знаний. Запуск работы системы производился при помощи автоматизированных тестов. Проверка данных также осуществлялась при помощи этой технологии. Система также может функционировать в режиме диалога и в консольном варианте, в этом режиме видно взаимодействие с пользователем.

### 4.3 Результаты экспериментов

Система показала свою жизнеспособность на модельных данных. Были проведены тесты в сравнении с работой человеческого специалиста. Был выбран контрольный список инцидентов. Сравнивался поиск решения для инцидентов. Основное время при опросе специалиста тратилось на коммуникацию. В Таблице 4.2 приведены результаты сравнения. Тесты были выполнены на компьютере Intel Core i7 1700 MHz, 8GB RAM, 256 GB SSD, FreeBSD.

Таблица 4.2 — Результаты сравнения с работой специалиста

Инцидент	TSS1 (.мс)	TU (.мс)
Tense is kind of concept (Время — это концепция)	15000	385
Please install Firefox (Установите Firefox)	9000	859
Browser is an object (Браузер — это объект)	20000	400
Firefox is a browser (Firefox — это браузер)	5000	659
Install is an action (Установить — это действие)	8000	486

**Таблица 4.2 – продолжение**

<b>Инцидент</b>	<b>TSS1 (.мс)</b>	<b>TU (.мс)</b>
User miss Internet Explorer 8 (У пользователя нет Internet Explorer 8)	10000	10589
User needs document portal update (Пользователю требуется обновление документов)	15000	16543
Add new alias Host name on host that alias is wanted to: hrportal.lalala.biz IP address on host that alias is wanted to: 322.223.333.22 Wanted Alias: webadviser.lalala.net (Добавьте, пожалуйста, новую ссылку на hrportal.lalala.biz через 322.223.333.22)	10000	18432
Outlook Web Access (CCC) — 403 — Forbidden: Access is denied (Нет доступа к Outlook Web Access (CCC))	15000	10342
PP2C — Cisco IP communicator. Please see if you can fix the problem with the ip phone, it's stuck on configuring ip + sometimes Server error rejected: Security etc (PP2C — коммуникатор Cisco IP. Пожалуйста, помогите исправить проблему с ИП-телефоном, он застревает во время конфигурирования и иногда показывает ошибку «Безопасность»)	13000	12343

Основной проблемой для системы составляют инциденты с большой неоднозначностью, например, "I should have Internet Explorer, but Firefox was installed". Здесь непонятно, нужен ли пользователю браузер Firefox или нет. В этом случае система должна выявить проблему о необходимости пользователю Internet Explorer.

Другой пример, который трудно однозначно решить, используя классические подходы: I install Internet Explorer previously, but i need Chrome. Здесь есть следующие наборы концепций: i, install, Internet Explorer; i, need, Chrome. Используя регулярные выражения, однозначно решить не удастся, но, используя интеллектуальное решение, эту проблему решить можно. В рамках TU сработает более вероятный Critic, который определит проблему "need Chrome", базируясь на наличии концепции "previously". В таблице 4.3 приведены результаты работы

системы в разрезе категорий инцидентов.

Таблица 4.3 — Описание экспериментальных данных

Класс проблемы	% успешных
Проблема с ПО	64%
Проблемы во время работы	10%
Как сделать	10%
Проблема с оборудованием	0%
Установить новое ПО	100%
Проблема с печатью	80%
Нет доступа	100%

Показатели, приведенные в главе 1,  $\alpha$  — доля заявок, для которых время в очереди превышает  $\max(T_q)$ ;  $\mu$  — величина, обратная среднему времени нахождения заявки у агента;  $n$  — число агентов;  $T_q$  — время нахождение заявки в очереди в часах;  $SLA$  — уровень обслуживания ( $1-\alpha$ ), доля заявок, для которых время в очереди не превышает  $\max(T_q)$ .  $T_p$  — время удовлетворения заявки;  $\alpha_n$  — количество заявок;  $T_{qp} = T_q + T_p$  — время прохождения заявки через систему, приобрели следующие значения  $T_{qp} = 32,9$  при  $n = 8$ ;  $SLA = 0,96$ ;  $\alpha = 0,04$ ;  $\alpha_n = 2920$ . Предыдущие значение было  $T_{qp} = 47,9$  при  $n = 6$ ;  $SLA = 0,82$ ;  $\alpha = 0,18$ ;  $\alpha_n = 2920$ , но увеличение  $n$  не привело к увеличению задействованных в работе специалистов. Согласно таблице 4.3 средний процент обработанных заявок составил 52%, что составляет более половины всех заявок и требуемых 30% (см. Главу 1).

#### 4.4 Выводы по главе 4

В главе были рассмотрены экспериментальные данные, которые были использованы для верификации системы, также дается обоснование, почему были выбраны именно эти данные. На основе экспериментов была посчитана скорость работы системы в сравнении со специалистом технической поддержки. Были приведены сложные для решения примеры входных запросов пользователя и дан их разбор.

## Заключение

Решены следующие задачи и достигнуты следующие результаты.

1. Создана модель проблемно-ориентированной системы управления знаниями в области обслуживания информационной инфраструктуры предприятия на основе обобщения модели мышления;
2. Представлены новая модель данных для модели мышления и оригинальный способ их хранения, более эффективный по сравнению с классическими базами данных, использующими реляционный подход;
3. Выполнено оригинальное исследование моделей мышления в области обслуживания информационной инфраструктуры предприятия;
4. На основе модели, разработанной в диссертации, созданы архитектура системы и ее прототип;
5. Система, разработанная в рамках данной работы, включает в себя инновационные методы и алгоритмы поддержки принятия решений, использует обобщенную модель мышления Мински;
6. Представлена визуализация структуры области удаленной поддержки инфраструктуры.

Представленные в диссертации модель мышления, ее архитектура и реализация являются уникальными — на данный момент времени это единственная реализация модели мышления Мински.

Система, разработанная в диссертации, не является узкоспециализированной и подходит для других областей, где требуется организация базы знаний, например, при постановке медицинского диагноза, чтобы отбросить ложные диагнозы.

В области диагностики проблем можно обучить систему сведениям об узлах автомобиля и проблемах, с ними связанных, признаках этих проблем и способах их устранения.

Работа велась с использованием открытых технологий, без использования проприетарного программного обеспечения. Работа была презентована автору книги Object-Oriented Software Construction [79] Берtrandу Мейеру в рамках серии лекций, проведенных при содействии Университета Иннополис в Казани в 2015 году в рамках AKSES-2015<sup>1</sup>, и была им отмечена. Работа выполнялась при помощи компании ОАО «АйСиЭл КПО-ВС (г. Казань)», в рамках работы

---

<sup>1</sup><http://university.innopolis.ru/en/research/selab/events/akses>

использовались и обрабатывались данные, собранные во время работы команд ОАО «АйСиЭл КПО-ВС (г. Казань)» над поддержкой информационной структуры предприятий-заказчиков.

## Список сокращений и условных обозначений

**selectLinkedObject(obj:Resource, linkName:String): Link<Resource>** —

Описание метода. selectLinkedObject — название метода. (obj:Resource, linkName:String) — параметры метода. linkName — имя параметра. String тип данных. Link<Resource> — тип возвращаемых данных. Если метод данных не возвращает, то ничего не указывается.

**DomainModel:SemanticNetwork** — Описание класса, где DomainModel — сам класс, а SemanticNetwork — класс-родитель.

**TU** — Сокращение от ThinkingUnderstanding.

**TLC** — Thinking Life Cycle.

**НДФЛ** — Налог на доходы физически лиц.

**ПО** — Программное обеспечение.

**ФБ** — Федеральный бюджет.

**ПФР** — Пенсионный фонд России.

**ТФОМС** — Территориальный фонд обязательного медицинского страхования.

**ФФОМС** — Федеральный фонд обязательного медицинского страхования.

**ФСС** — Фонд социального страхования.

**БД** — База данных.

**мс.** — Миллисекунды.

## Словарь терминов

**База Знаний** — База данных приложения, представленная в виде онтологии знаний.

**WayToThink** — Путь мышления. Основан на определении Марвина Мински [71]. Класс объектов, которые модифицируют данные.

**Critic** — Основан на определении Марвина Мински [71]. Класс объектов, которые выступают триггерами при наступление определенного события.

**ThinkingLifeCycle** — Основан на определении Марвина Мински [71]. Класс объектов, которые выступают основными объектами для запуска в приложении — рабочими процессами.

**Selector** — Компонент, отвечающий за выборку данных из Базы Знаний.

**Instinctive** — Инстинктивный уровень.

**Learned** — Уровень обученных реакций.

**Deliberative** — Уровень рассуждений.

**Reflective** — Рефлексивный уровень.

**Self-Reflective Thinking** — Саморефлексивный уровень.

**Self-Conscious Reflection** — Самосознательный уровень.

**ThinkingUnderstanding** — Система, созданная в рамках работы. Дословный перевод «Мышление-Понимание».

**Вариант использования** — Термин из стандарта UML, который описывает возможные способы функционирования системы.

**Диаграмма действий** — Термин из стандарта UML, который описывает последовательность действий пользователя.

## Список литературы

1. Коптелов, А. Вывод ИТ-подразделений на аутсорсинг: проблемы и решения [Текст] / А. Коптелов // Информационные технологии. — 2006. — Т. 1311. — С. 22 – 23.
2. Коптелов, А. Анализ эффективности аутсорсинга [Текст] / А. Коптелов, О. Вишняков // IT news. — 2007. — Т. 7(80). — С. 12 – 15.
3. Коптелов, А. Аутсорсинг центра технической поддержки пользователей [Текст] / А. Коптелов, С. Уштей // IT news. — 2007. — Т. 2(75). — С. 5 – 10.
4. Коптелов, А. Аутсорсинг разработки программного обеспечения [Текст] / А. Коптелов, Н. Елманова // Информационные технологии. — 2006. — Т. 16. — С. 5 – 10.
5. Коптелов, А. ИТ-служба передается на аутсорсинг [Текст] / А. Коптелов, И. Караваев // ИКС. — 2007. — Т. 8. — С. 22 – 24.
6. Statista. Global market size of outsourced services from 2000 to 2014 (in billion U.S. dollars) [Electronic resource]. — [S. l. : s. n.], 2015. — URL: <http://www.statista.com/statistics/189788/global-outsourcing-market-size/> (online; accessed: 25.01.2015).
7. Hartshorne, R. Outsourcing of information and knowledge services: A supplier's view [Text] / R. Hartshorne // Business Information Review. — 2015. — Vol. 32. — P. 103 – 109.
8. Зацепа, С. Рентабельность малого бизнеса и ИТ-аутсорсинг [Текст] / С. Зацепа // Управление компанией. — 2006. — Т. 7. — С. 90 – 98.
9. Коптелов, А. Автоматизация центра поддержки пользователей [Текст] / А. Коптелов // Мобильные телекоммуникации. — 2006. — Т. 9. — С. 103 – 109.
10. Tutubalina, E. Target-based topic model for problem phrase extraction (Conference Paper) [Text] / E. Tutubalina // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). — 2015. — Vol. 9022. — P. 271 – 277.

11. Bello-Orgaz, G.A. Social big data: Recent achievements and new challenges (Article) [Text] / G.A Bello-Orgaz, J.J. Jung, D.A. Camacho // Information Fusion. — 2015. — Vol. 28. — P. 45 – 59.
12. Baddoura, R. This Robot is Sociable: Close-up on the Gestures and Measured Motion of a Human Responding to a Proactive Robot (Article) [Text] / R Baddoura, G. Venture // International Journal of Social Robotics. — 2015. — Vol. 7. — P. 489 – 496.
13. Michalos, G. Decision making logic for flexible assembly lines reconfiguration [Text] / G. Michalos, P. Sipsas, S. Makris // Robotics and Computer-Integrated Manufacturing. — 2016. — Vol. 37. — P. 233 – 250.
14. Devarakonda, M. Problem-oriented patient record summary: An early report on a Watson application [Text] / M. Devarakonda, D. Zhang, C.H. Tsou // 2014 IEEE 16th International Conference on e-Health Networking, Applications and Services. — 2014. — Vol. 1. — P. 281 – 286.
15. Wagner, J. TOP 10 MACHINE LEARNING APIS: AT T SPEECH, IBM WATSON, GOOGLE PREDICTION [Electronic resource]. — [S. l. : s. n.], 2015. — URL: <http://goo.gl/FQ9G6G> (online; accessed: 01.01.2016).
16. Ivanov, V. Clause-based approach to extracting problem phrases from user reviews of products [Text] / V. Ivanov, E. Tutubalina // Communications in Computer and Information Science. — 2014. — Vol. 436. — P. 229 – 236.
17. Тощев, А. С. К новой концепции автоматизации программного обеспечения [Текст] / А. С. Тощев // Труды Математического центра имени Н.И. Лобачевского. Материалы Десятой молодежной научной школы-конференции «Лобачевские чтения - – 2011. Казань, 31 октября – 4 ноября 2011». — 2011. — Т. 44. — С. 279 – 282.
18. Toshchev, A. Thinking-Understanding approach in IT maintenance domain automation [Text] / A. Toshchev, M. Talanov, A. Krehov // Global Journal on Technology: 3rd World Conference on Information Technology (WCIT-2012). — 2013. — Vol. 3. — P. 879 – 894.

19. Тощев, А.С. Архитектура и реализация интеллектуального агента для автоматической обработки входящих заявок с помощью искусственного интеллекта и семантических сетей [Текст] / А.С. Тощев, М.О. Таланов // Ученые записки Института социально-гуманитарных знаний. — 2014. — Т. Вып. № 1(12), Ч. II. — С. 288 – 292.
20. Toshchev, A. Computational emotional thinking and virtual neurotransmitters [Text] / A. Toshchev, M. Talanov // International Journal of Synthetic Emotions (IJSE). — 2014. — Vol. 5 (1). — P. 30 – 35.
21. Toshchev, A. Appraisal, coping and high level emotions aspects of computational emotional thinking [Text] / A. Toshchev, M. Talanov // International Journal of Synthetic Emotions (IJSE). — 2015. — Vol. 6 (1). — P. 65 – 72.
22. Тощев, А.С. Модель мышления и понимания в автоматической обработке запросов пользователя [Текст] / А.С. Тощев // Труды 16-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». — 2014. — С. 425 – 427.
23. Toshchev, A. Thinking lifecycle as an implementation of machine understanding in software maintenance automation domain [Text] / A. Toshchev, M. Talanov // Agent and Multi-Agent Systems: Technologies and Applications: 9th KES International Conference, KES-AMSTA, 2015 Sorrento, Italy, June 2015, Proceedings (Smart Innovation, Systems and Technologies). — 2015. — Vol. 38. — P. 301 – 310.
24. Тощев, А.С. Возможности автоматизации разрешения инцидентов для области удаленной поддержки информационной инфраструктуры предприятия [Текст] / А.С. Тощев // Экономика и менеджмент систем управления. — 2015. — Т. 4.2 (18). — С. 293 – 295.
25. Тощев, А.С. Вычислительная модель эмоций в интеллектуальных информационных системах [Текст] / А.С. Тощев, М.О. Таланов // Электронные библиотеки. — 2015. — Т. 18 №5. — С. 231 – 241.
26. Тощев, А.С. Применение моделей мышления в интеллектуальных вопросно-ответных системах [Текст] / А.С. Тощев // Электронные библиотеки. — 2015. — Т. 18 №5. — С. 222 – 230.

27. Петреченко, В.А. Прогнозирование эксплуатационных расходов на информационные технологии с применением теории массового обслуживания: дис. Петреченко В.А. канд. эконом. наук: 08.00.13 [Текст] / В.А. Петреченко. — М. : [б. и.], 2011. — 174 с.
28. Super Job. Уровень зарплат IT специалистов [Электронный ресурс]. — [Б. м. : б. и.], 2014. — URL: <http://www.it-analytics.ru/analytics/trends/66314.html0.11-2011> (дата обращения: 2015.05.01).
29. Налоговый кодекс Российской Федерации. Части 1 и 2 [Текст]. — Москва : Эксмо, 2015. — 1344 с.
30. TimeWeb. Стоимость аренды серверов [Электронный ресурс]. — [Б. м. : б. и.], 2015. — URL: <http://timeweb.com/ru/services/dedicated-server/> (дата обращения: 2015.11.01).
31. Садовничей, В. Суперкомпьютерные технологии в науке, образовании и промышленности [Текст] / В. Садовничей, Г. Савина. — Москва : Издательство Московского университета, 2009. — 232 с. — ISBN: [978-5-211-05719-7](#).
32. Sokolov, A. HP OpenView System Administration Handbook: Network Node Manager, Customer Views, Service Information Portal, HP OpenView Operations [Text] / A. Sokolov, K. Serdobincev ; Ed. by H. Shootze. — Astrahan : Astrahan, 2004. — 688 p.
33. Хикс, Дж. Теория экономической истории [Текст] / Дж. Хикс // Вопросы экономики. — 2003. — Т. 8. — С. 184 – 188.
34. Intelligent telecommunication system using semantic-based information retrieval [Text] / E. Jubilson, P. Dhanavanthini, P. Victer [et al.] // Advances in Soft Computing. — 2015. — Vol. 381. — P. 137 – 143.
35. Intelligent event processing for emergency medical assistance [Text] / H. Billhardt, M. Lujak, S. Ossowski [et al.] // Proceedings of the ACM Symposium on Applied Computing. — 2014. — Vol. 1. — P. 200 – 206.
36. Development of the configuration management system of the computer center and evaluation of its impact on the provision of IT services [Text] / N. Iuzhanin,

- T. Ezhakova, V. Zolotarev, V. Gaiduchok // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). — 2015. — Vol. 9158. — P. 249 – 258.
37. Schubert, F. A Web-based Intelligent Help Desk Support Environment [Text] / F. Schubert, C. Siu, P. Chor // School of Computer Engineering, Nanyang Technological University. — 2015. — Vol. 9158. — P. 249 – 258.
38. Yang, C. Developing and evaluating an IT specification extraction system [Text] / C. Yang, C. Chen, C. Peng // Electronic Library. — 2006. — Vol. 24. — P. 832 – 846.
39. Mimir: An open-source semantic search framework for interactive information seeking and discovery [Text] / V. Tablan, K. Bontcheva, I. Roberts, H. Cunningham // Journal of Web Semantics. — 2015. — Vol. 30. — P. 52 – 68.
40. Construction and evaluation of ontological tag trees [Text] / C. Verma, V. Mahadevan, N. Rasiwasia [et al.] // Expert Systems with Applications. — 2015. — Vol. 42. — P. 9587 – 9602.
41. Goetzel, B. OpenCog [Electronic resource]. — [S. l. : s. n.], 2012. — URL: <http://opencog.org/> (online; accessed: 01.01.2016).
42. Wang, P. Non-Axiomatic Logic A Model of Intelligent Reasoning. [Text] / P. Wang. — California, USA : World Scientific Publishing Company, 2013. — 276 p.
43. Ингланд, Р. Введение в реальный ITSM [Текст] / Р. Ингланд ; Под ред. О. Скрынник. — Москва : Лайвбук, 2010. — 131 с. — ISBN: 5904584059, 9785904584054.
44. Ингланд, Р. Овладевая ITIL [Текст] / Р. Ингланд ; Под ред. О. Скрынник. — Москва : Лайвбук, 2011. — 200 с.
45. Будкова, Л. Методическое руководство для подготовки к профессиональным экзаменам ISO 20000 Foundation и ISO 20000 Foundation Bridge [Текст] / Л. Будкова, Р. Журавлёв ; Под ред. О. Скрынник. — Москва : Лайвбук, 2011. — 124 с.

46. Tsvetkov, A. Automation of incidents' recording process in the network of the mobile radio communication of standard GSM-900/1800 (Conference Paper) [Text] / A. Tsvetkov, O. Ponomareva, M. Yurina // 24th International Crimean Conference Microwave and Telecommunication Technology, CriMiCo 2014. Sevastopol, Crimea. 7 September 2014 through 13 September 2014. — 2014. — Vol. 1. — P. 401 – 402.
47. Trustworthy and resilient monitoring system for cloud infrastructures (Conference Paper) [Text] / S. Padhy, D. Kreutz, A. Casimiro, M. Pasin // Proceedings of the Workshop on Posters and Demos Track, PDT'11 - 12th International Middleware Conference, Middleware'11. — 2011. — Vol. 1. — P. 87 – 95.
48. Gentschen, F. IT service management across organizational boundaries [Text] / F. Gentschen, H. Hegering, M. Schiffers // Managing Development and Application of Digital Technologies: Research Insights in the Munich Center for Digital Technology and Management. — 2006. — Vol. 1. — 341 p.
49. Catania, N. Web Services Management Framework [Electronic resource]. — [S. l. : s. n.], 2003. — URL: <http://devresource.hp.com/drc/specifications/wsmf/WSMF-Overview.jsp> (online; accessed: 01.01.2016).
50. Catania, N. Web Services Events (WS-Events) [Electronic resource]. — [S. l. : s. n.], 2003. — URL: <http://devresource.hp.com/drc/specifications/wsmf/WS-Events.pdf> (online; accessed: 01.01.2016).
51. Migration of the CERN IT data centre support system to servicenow (Conference Paper) [Text] / R.A. Alonso, G. Arneodo, O. Barrig [et al.] // Journal of Physics: Conference Series. — 2013. — Vol. 513. — P. 1 – 80.
52. Open Science Grid (OSG) ticket synchronization: Keeping your home field advantage in a distributed environment (Conference Paper) [Text] / K. Gross, S. Hayashi, S. Teige, R. Quick // Journal of Physics: Conference Series. — 2012. — Vol. 396. — P. 1 – 80.
53. Relation extraction and scoring in DeepQA [Text] / C. Wang, A. Kalyanpur, J. Fan [et al.] // IBM journal of Research and Development. — 2012. — Vol. 56. — P. 1 – 12.

54. Watson: Beyond jeopardy! [Text] / D. Ferrucci, A. Levas, S. Bagchi [et al.] // Artificial Intelligence. — 2013. — Vol. 10.1016. — P. 93 – 105.
55. Alterman, R. Understanding and summarization [Text] / R. Alterman // Artificial Intelligence Review. — 1991. — Vol. 5. — P. 239 – 254.
56. Chandrasekar, R. Elementary? Question answering, IBM’s Watson, and the Jeopardy! challenge [Text] / R. Chandrasekar // Resonance. — 2014. — Vol. 19. — P. 222 – 241.
57. Rajaraman, V. John McCarthy — Father of artificial intelligence [Text] / V. Rajaraman // Resonance. — 2014. — Vol. 19(3). — P. 198 – 207.
58. Jurafsky, D. Detecting friendly, flirtatious, awkward, and assertive speech in speed-dates [Text] / D. Jurafsky, J. Martin // Computer Speech and Language. — 2013. — Vol. 27. — P. 89 – 115.
59. Campbell, M. Deep Blue [Text] / M. Campbell, Hoane Jr., F.-H. A. Joseph, Hsu // Artificial Intelligence. — 2002. — Vol. 134. — P. 57 – 83.
60. Mahdi, A. Utilizing WordNet and regular expressions for instance-based schema matching [Text] / A. Mahdi, S. Tiun // Research Journal of Applied Sciences, Engineering and Technology. — 2014. — Vol. 8. — P. 460 – 470.
61. Lamperti, G. Diagnosis of active systems by semantic patterns [Text] / G. Lamperti, X. Zhao // IEEE Transactions on Systems, Man, and Cybernetics: Systems. — 2014. — Vol. 8. — P. 1028 – 1043.
62. Automatic extraction of function-behaviour-state information from patents [Text] / G. Fantoni, R. Apreda, F. Dell’Orletta, M. Monge // Advanced Engineering Informatics. — 2013. — Vol. 27. — P. 317 – 334.
63. Krasner, D. Flexible processing and classification for eDiscovery [Text] / D. Krasner, I. Langmore // Frontiers in Artificial Intelligence and Applications. — 2013. — Vol. 259. — P. 87 – 96.
64. Manshadi M., Gildea-D. Allen J. Integrating programming by example and natural language programming [Text] / Gildea-D. Allen J. Manshadi, M. // Proceedings of the 27th AAAI Conference on Artificial Intelligence. — 2013. — P. 661 – 667.

65. Foundation, Apache Software. Apache OpenNLP [Electronic resource]. — [S. l. : s. n.], 2012. — URL: <https://opennlp.apache.org/> (online; accessed: 01.01.2016).
66. Wikipedia. OpenCog [Electronic resource]. — [S. l. : s. n.], 2012. — URL: <http://wiki.opencog.org/w/RelEx> (online; accessed: 01.01.2016).
67. Veber, P. Introduction to information processing [Text] / P. Veber, D. Willams ; Ed. by T. Zitello. — New Jersey, USA : Prentis Hall, 2009. — 581 p.
68. Implementing an online help desk system based on conversational agent [Text] / A. Kongthon, C. Sangkeettrakarn, S. Kongyoung, C. Haruechaiyasak // Proceeding, MEDES '09 Proceedings of the International Conference on Management of Emergent Digital EcoSystems. — 2009. — Vol. 1. — P. 450 – 451.
69. Гринберг, Д. Надежный алгоритм обработки для грамматики [Текст] / Д. Гринберг // Технический отчет Университета Карнеги Мелон CMU-CS-95-125. — 1995. — Т. 9. — С. 30 – 35.
70. Russel, S. Artificial Intelligence. A Modern approach [Text] / S. Russel, P. Norvig. — New York, USA : Pearson, 2010. — 1152 p.
71. Minsky, M. The Emotion Machine [Text] / M. Minsky. — New York, USA : Simon & Schuster, 2007. — 400 p.
72. Katidiotis, A. Performance evaluation of artificial neural network-based learning schemes for cognitive radio systems [Text] / A. Katidiotis, K. Tsagkaris // Computers & Electrical Engineering. — 2010. — Vol. 36. — P. 518 – 535.
73. Deng, H. Bias of importance measures for multi-valued attributes and solutions [Text] / H. Deng, G. Runger, E. Tuv // Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN). — 2011. — P. 293 – 300.
74. Anaya, A. A visual recommender tool in a collaborative learning experience [Text] / A. Anaya, M. Luque, M. Peinado // Expert Systems with Applications. — 2016. — Vol. 45. — P. 248 – 259.
75. Sinha, Y. Comparative study of preprocessing and classification methods in character recognition of natural scene images [Text] / Y. Sinha, P. Jain, N. Kasliwal //

- Advances in Intelligent Systems and Computing. — 2016. — Vol. 45. — P. 119 – 129.
76. Trujillo-Rasua, R. K-Metric antidimension: A privacy measure for social graphs [Text] / R. Trujillo-Rasua, I. Yero // Information Sciences. — 2015. — Vol. 328. — P. 403 – 417.
  77. Lesly, Л. Owl: Representing Information Using the Web Ontology Language [Text] / Л. Lesly. — 47403, Blumington, Liberty drive 1663 : Tredford Publishing, 2005. — 302 p.
  78. Noy, N. Ontology Development 101: A Guide to Creating Your First Ontology [Text] / N. Noy, D. McGuiness // Ontology Development 101: A Guide to Creating Your First Ontology. — 2010. — P. 12 – 35.
  79. Meyer, B. Object-Oriented Software Construction 2nd Edition [Text] / B. Meyer. — Upper Sadle River, USA : Prentis Hall, 1997. — 1296 p.
  80. Rokach, L. Decision forest: Twenty years of research [Text] / L. Rokach // Information Fusion. — 2015. — Vol. 27,29. — P. 111 – 125.
  81. Designing virtual bots for optimizing strategy-game groups [Text] / M. Bedia, L. Castillo, C. Lopez [et al.] // Neurocomputing. — 2015. — Vol. 172. — P. 453 – 458.
  82. Cheng, M. Nature-inspired metaheuristic multivariate adaptive regression splines for predicting refrigeration system performance [Text] / M. Cheng, J. Chou, M. Cao // Soft Computing. — 2015. — P. 13.
  83. Bukharov, O. Development of a decision support system based on neural networks and a genetic algorithm [Text] / O. Bukharov, D. Bogolyubov // Expert Systems with Applications. — 2015. — Vol. 42. — P. 6177 – 6183.
  84. High-Speed General Purpose Genetic Algorithm Processor [Text] / S. P. Hosseini Alinodehi, S. Moshfe, M. Saber Zaeimian [et al.] // IEEE Transactions on Cybernetics. — 2015. — P. 2 – 3.
  85. Дергачев, А. М. Проблемы эффективного использования сетевых сервисов [Текст] / А. М. Дергачев // Научно-технический вестник СПбГУ ИТМО. — 2011. — Т. 71. — С. 83 – 87.

86. White, D. Software review: the ECJ toolkit [Text] / D. White // Genetic Programming and Evolvable Machines. — 2011. — Vol. 13. — P. 65 – 67.
87. Layered Ensemble Architecture for Time Series Forecasting [Text] / M. M. Rahaman, M. M. Islam, K. Murase, X. Yao // IEEE Transactions on Cybernetics. — 2015. — P. 1 – 5.
88. Хокинг, С. Теория всего [Текст] / С. Хокинг. — Москва : Амфора, 2009. — 160 с.
89. Minsky, M. The Society of Mind [Text] / M. Minsky. — NY, USA : Simon & Schuster, 2007. — 336 p.
90. Talanov, M. Automating programming via concept mining, probabilistic reasoning over semantic knowledge base of SE domain [Text] / M. Talanov, A. Krekhov, A. Makhmutov // 2010 6th Central and Eastern European Software Engineering Conference, CEE-SECR 2010. — 2010. — Vol. 9. — P. 30 – 35.
91. Giachetti, R. Design of Enterprise Systems: Theory, Architecture, and Methods [Text] / R. Giachetti. — USA : CRC Press, 2010. — 448 p.
92. An auto-tuning PID control system based on genetic algorithms to provide delay guarantees in Passive Optical Networks [Text] / T. Jiménez, N. Merayo, A. Andrés [et al.] // Expert Systems with Applications. — 2015. — Vol. 42. — P. 9211 – 9220.
93. An aggregated technique for optimization of SOAP performance in communication in Web services [Text] / K. Senagi, G. Okeyo, W. Cheruiyot, M. Kimwele // Service Oriented Computing and Applications. — 2015. — P. 6 – 7.
94. Fowler, M. Patterns of Enterprise Application Architecture [Text] / M. Fowler. — USA : Addison-Wesley Professional, 2010. — 448 p.
95. Brown, W. AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis [Text] / W. Brown. — USA : Wiley, 2010. — 336 p.
96. White, D. Akka Concurrency [Text] / D. White ; Ed. by K. Rolland. — [S. l.] : Artima, 2013. — 521 p.

97. Robinson, S. WebSphere Application Server 7.0 Administration Guide [Text] / S. Robinson. — [S. l.] : PACKT publishing, 2009. — 344 p.
98. Goetzel, B. Java Concurrency in Practice [Text] / B. Goetzel, T. Payrels, D. Bloch. — [S. l.] : Addison-Wesley Professional; 1 edition, 2006. — 384 p.
99. Deadline, After The. After The Deadline [Electronic resource]. — [S. l. : s. n.]. — URL: <http://www.afterthedeadline.com/> (online; accessed: 01.01.2016).
100. Richards, D. A controlled language to assist conversion of use case descriptions into concept lattices [Text] / D. Richards, K. Boettger, O. Aguilera // 15th Australian Joint Conference on Artificial Intelligence, AI 2002; Canberra; Australia; 2 December 2002 through 6 December 2002; Code 141829. — 2002. — Vol. 2557. — P. 1 – 11.
101. Goetzel, B. Probabilistic Logic Networks: A Comprehensive Conceptual, Mathematical and Computational Framework for Uncertain Inference [Text] / B. Goetzel, I. Mathew. — New York, USA : Springer, 2008. — 333 p. — ISBN: [0-387-76871-8](#).

## Список иллюстраций

1.1	Диаграмма состава команд . . . . .	14
1.2	Диаграмма соотношений типов проблем . . . . .	14
1.3	Модель системы массового обслуживания в ИТ . . . . .	15
1.4	Средний поток заявок по часам. . . . .	16
1.5	HP OpenView (материал из Wikipedia) . . . . .	19
1.6	Service NOW (материал из <a href="http://wiki.servicenow.com/">http://wiki.servicenow.com/</a> ) . . . . .	20
1.7	Пример работы системы Watson (материал из <a href="http://www.informationweek.com/">http://www.informationweek.com/</a> ) . . . . .	20
1.8	Результаты обработки текстов . . . . .	24
2.1	Представление класса Order в OWL. Визуализация Protege . . . . .	30
2.2	Представление класса CreateCustiner в OWL. Визуализация Protege	31
2.3	UML-диаграмма последовательности для основного потока в модели Menta 0.1 . . . . .	33
2.4	Критик – Селектор – Образ мышления . . . . .	37
2.5	$T^3$ в разрезе ресурсов . . . . .	38
2.6	Иллюстрация концепции Уровней мышления . . . . .	40
2.7	Иллюстрация концепции k-line . . . . .	41
3.1	Вариант использования. Обучение . . . . .	45
3.2	Диаграмма взаимодействия компонентов . . . . .	46
3.3	Детальная диаграмма компонентов системы . . . . .	48
3.4	Интерфейс компонента WebService . . . . .	49
3.5	Диаграмма классов ThinkingLifeCycle . . . . .	54
3.6	Диаграмма действий метода onMessage компонента ThinkingLifeCycle . . . . .	55
3.7	Диаграмма действий метода sendMessage компонента ThinkingLifeCycle . . . . .	55
3.8	Диаграмма действий метода apply компонента ThinkingLifeCycle . .	56
3.9	Диаграмма действий метода apply компонента ThinkingLifeCycle . .	56
3.10	Диаграмма действий метода processWay2Think компонента ThinkingLifeCycle . . . . .	57
3.11	Диаграмма действий метода processCritic компонента ThinkingLifeCycle . . . . .	57

3.12	Диаграмма действий метода init компонента ThinkingLifeCycle . . . . .	58
3.13	Диаграмма действий метода stop компонента ThinkingLifeCycle . . . . .	58
3.14	Интерфейс компонента Selector . . . . .	60
3.15	Диаграмма действий метода Selector.apply(request : Request) компоненты Selector . . . . .	62
3.16	Диаграмма действий метода Selector.apply(goal: Goal) компонента Selector . . . . .	63
3.17	Диаграмма действий метода Selector.apply(criticResult : ActionProbabilityRule) компонента Selector . . . . .	64
3.18	Диаграмма действий классификации инцидента . . . . .	65
3.19	Диаграмма действий компонента Critic . . . . .	67
3.20	Интерфейс компонента WayToThink . . . . .	72
3.21	Работа компонента WayToThink в режиме описания решения проблемы (HowTo) . . . . .	72
3.22	Интерфейс компонента PreliminaryAnnotator . . . . .	73
3.23	Интерфейс компонента KnowledgeBaseServer . . . . .	76
3.24	Интерфейс компонента Reasoner . . . . .	76
3.25	Схема данных TU Knowledge в формате UML . . . . .	77
3.26	Диаграмма действий LifecycleActivity . . . . .	84
A.1	Диаграмма классов интерфейсной модели . . . . .	110
B.1	Диаграмма классов Action . . . . .	111
B.2	Диаграмма классов Goal . . . . .	112
B.3	Диаграмма места Goal в SemanticNetwork (Семантической сети) . .	113
D.1	Свидетельство о регистрации . . . . .	121
E.1	Акт о внедрении . . . . .	122

## Список таблиц

1	Сопоставление направлений исследований, предусмотренных специальностью 05.13.11, и результатов, полученных в диссертации	9
1.1	Описание работы специалистов различных уровней поддержки . . . . .	13
1.2	Категории инцидентов в области удаленной поддержки инфраструктуры . . . . .	14
1.3	Таблица метрик . . . . .	23
1.4	Сравнительный анализ функциональности существующих решений . . . . .	27
2.1	Описание свойств класса Order в OWL . . . . .	31
2.2	Описание иерархии предикатов . . . . .	31
2.3	Компоненты модели Menta 0.3 . . . . .	34
2.4	Сравнение скорости доступа к данным баз знаний . . . . .	36
2.5	Описание уровней мышления, предложенных Марвином Мински . . . . .	39
3.1	Основные компоненты системы Thinking-Understanding (TU) . . . . .	43
3.2	Описание ветвей в варианте использования «Режим обучения» . . . . .	44
3.3	Описание ветвей в варианте использования «Основной режим» . . . . .	45
3.4	Описание методов компонента WebService . . . . .	50
3.5	Описание методов класса (компоненты) ThinkingLifeCycle . . . . .	52
3.6	Описание методов класса (компоненты) Selector . . . . .	61
3.7	Описание основных типов Critic, используемых в системе . . . . .	66
3.8	Описание методов компонента Critic . . . . .	67
3.9	Описание встроенных в систему WayToThink . . . . .	69
3.10	Описание методов компонента WayToThink . . . . .	70
3.11	Описание методов компонента PreliminaryAnnotator . . . . .	73
3.12	Описание методов компонента DataService . . . . .	74
3.13	Описание методов компонента Reasoner . . . . .	75
3.14	Описание классов TUKnowledge . . . . .	78
4.1	Описание экспериментальных данных . . . . .	86
4.2	Результаты сравнения с работой специалиста . . . . .	88
4.3	Описание экспериментальных данных . . . . .	90
Г.1	Описание экспериментальных данных . . . . .	116

## Приложение А

### Интерфейсная модель

Интерфейсная модель содержит классы и интерфейсы для взаимодействия с пользователем. На рисунке A.1 представлена диаграмма классов модели данных. В ее основе лежит базовый объект RefObject, который объединяет в себе свойства, присущие всем объектам системы:

- ObjectID — уникальный идентификатор объектах в пределах класса объекта;
- Reference — уникальный идентификатор объектах в пределах всей Базы Знаний;
- Name — имя объекта.

Рассмотрим несколько объектов, например, Request (Запрос). Он предназначен для хранения запроса пользователя:

- SubscriptionID — идентификатор подписки, ссылка на сущность Subscription (подписка);
- RequestText — запрос пользователя в виде текста;
- Solution — описание решения запроса пользователя;
- State — статус запроса (например, «поиск решения»);
- FormalizedRequest — ссылка на формализованный запрос.

Subscription (подписка) содержит информацию о подписке пользователя на информирование о событиях. Список точек оповещения (Endpoints), благодаря которым система будет предоставлять информацию пользователю о состоянии его запроса. Точка оповещения описывается набором параметров из Type (тип) — тип точки связи с пользователем (например, веб-сервис) и Address (адрес) — адрес точки связи с пользователем.

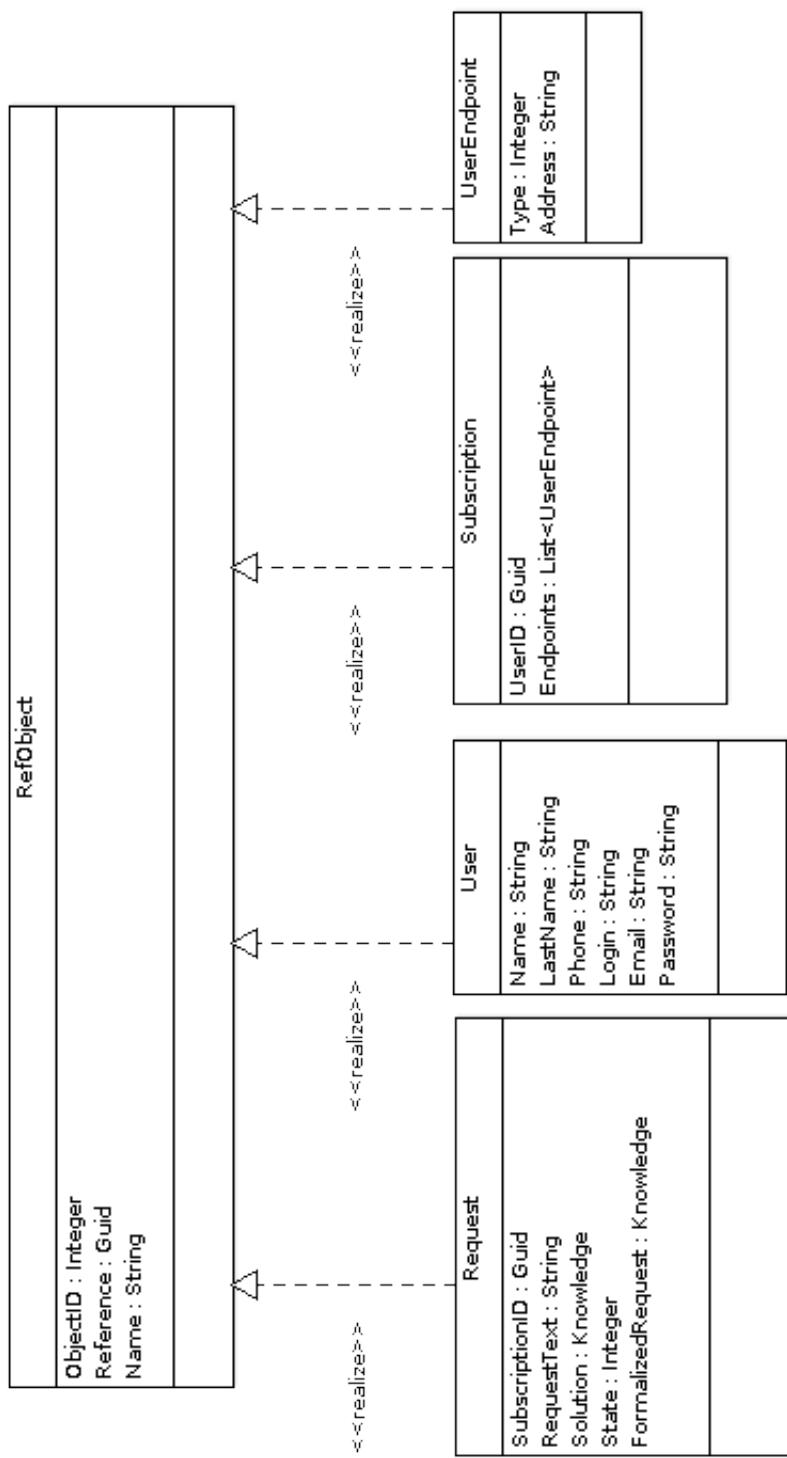


Рис. А.1 — Диаграмма классов интерфейсной модели

## Приложение Б

### Описание модуля Goal (Цель) и Action (Действия)

Action (Действие) является базовым классом для WayToThink или Critic. Он описывает общие методы этих классов: start (запуск); stop (остановка) и apply (применение) — метод, который производит работу над контекстом и изменяет данные. На рисунку Б.1 представлена диаграмма классов Action, где отображена связь с Critic и WayToThink.

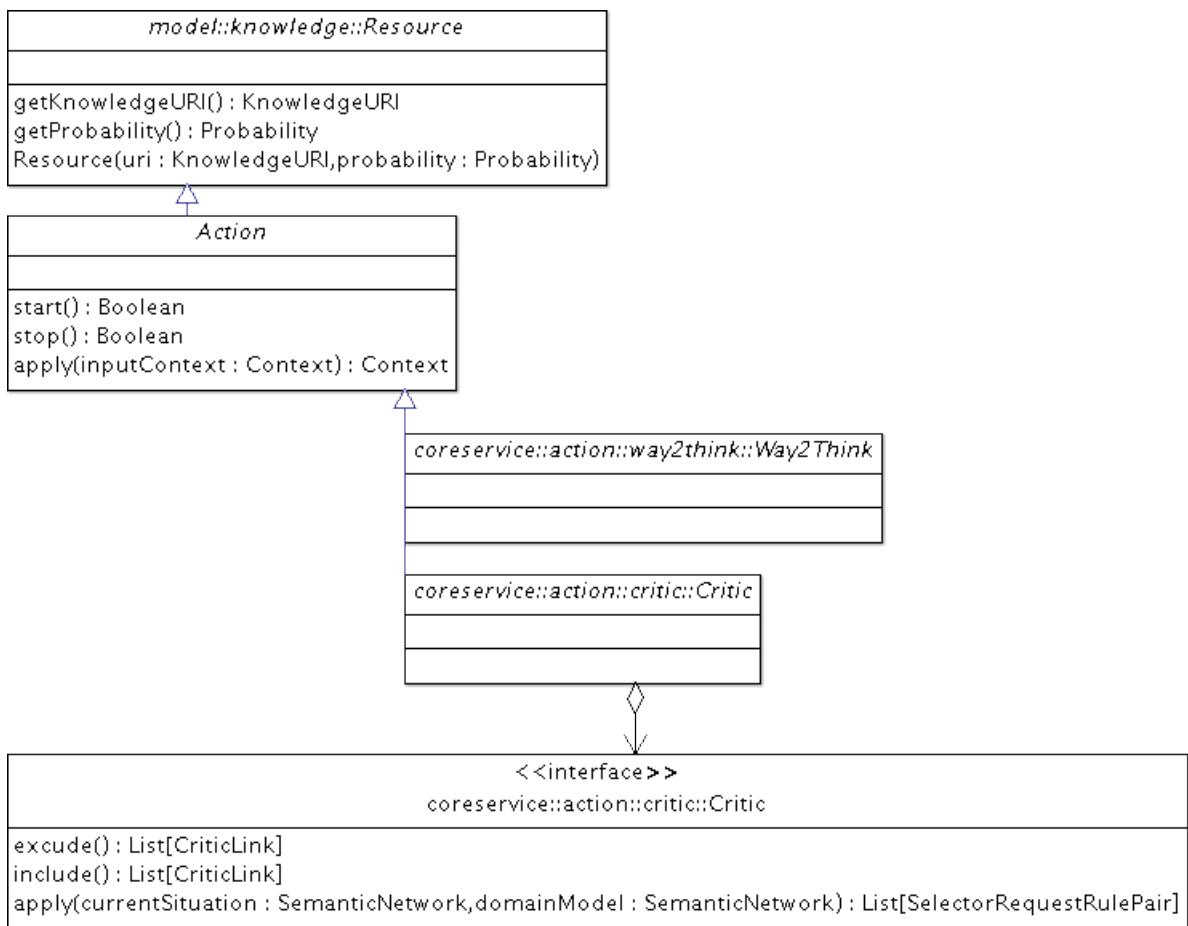


Рис. Б.1 — Диаграмма классов Action

Goal (Цель) является набором вероятностных предикатов и последовательностью How-To необходимых для того, чтобы достичь цель. Goal и How-To тесно связаны. На рисунке Б.2 показан состав Goal. Goal состоит из:

1. Parameters — параметры, которые используются предикатами для выполнения;
2. Precondition — условия, которые должны быть выполнены до начала основной проверки (Exit criteria);

3. Entry criteria — входной критерий, предикат, который определяет, что цель активировалась;
4. Exit criteria — условия, когда цель считается выполненной;
5. PostCondition — дополнительные условия для выхода;
6. HowTo — набор решения. Список путей решения.

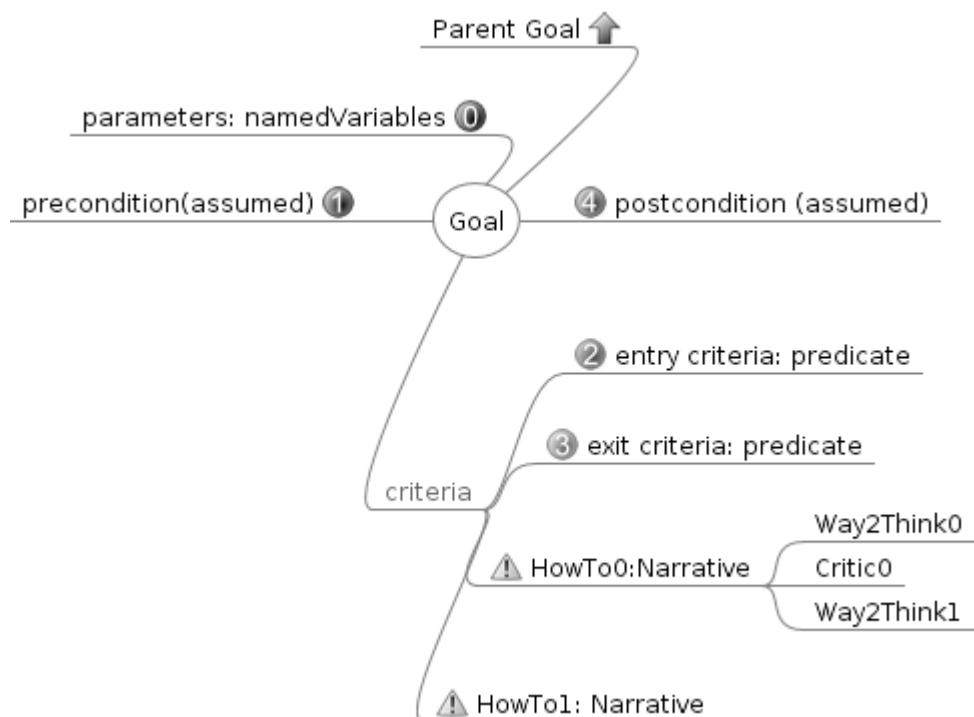


Рис. Б.2 — Диаграмма классов Goal

**Типы предикатов.** В решение используется 3 типа логических предикатов: and (и), or (или), not (отрицание). Представление Goal в SemanticNetwork показано на диаграмме [Б.3](#).

Основная цель системы — помочь пользователю. Все остальные цели являются подцелями основной: разрешить инцидент, понять тип инцидента, найти решение инцидента и т. д..

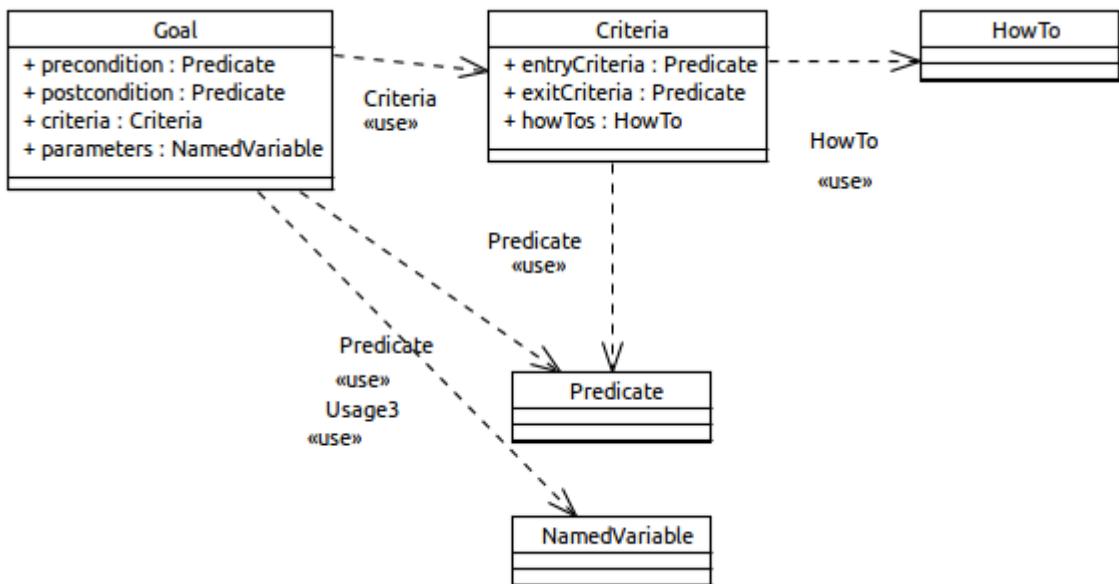


Рис. Б.3 — Диаграмма места Goal в SemanticNetwork (Семантической сети)

## Приложение В

### Рецепты решений

Рецепты решений представляют собой последовательность действий выполняемых для разрешения проблемы, описанной в инциденте. Было разработано два типа HowTo (рецепт решения): ValueHowTo — содержит в себе простое значение; FunctionalHowTo — содержит в себе функцию.

FunctionalHowTo состоит из следующих частей:

1. FunctionalBody — тело функции, описывающий содержание функции;
2. InputParameters — входные параметры функции;
3. OutputParameters — выходные параметры.

Комбинация FunctionaHowTo и ValueHowTo является рецептом решения. Например, решение проблемы неработающего сегмента кластера в формате для специалиста технической поддержки.

- Войти на сервер U1;
- Запустить утилиту 12 для Windows Servers;
- Открыть вкладку 1;
- Перейти на All Managed Server, найти нужный Server из правой панели, открыть свойства сервера;
- Нажать на Backup Exec Services;
- Выберите проблемный сегмент кластера;
- Нажмите Restart all Services;
- Подождите и проверьте статус.

Преобразованный в формат HowTo данный рецепт решения будет выглядеть как показано ниже.

```

login:howto{
Parameters: [
5   {Key:'ScriptName',
Value:'LogonScript.bat'},
{Key:'Description',
Value:'Logon to server'}
]
10 InputParameters:[
  {Key:'ServerName',

```

```
    Value:'U1'},
    {Key:'UserName',
     Value:'MyUser'}
]

OutputParameters:[
    {Key:'SessionID',
     Value:'SSSE12'},

]
}

25 launch:howto{
    Parameters:[
        {Key:'ScriptName',
         Value:'LaunchScript.bat'},
        {Key:'Description',
         Value:'Launch the application'}
    ]

InputParameters:[
    {Key:'ExecName',
     Value:'Utility12.exe'},
]

OutputParameters:[
    {Key:'SessionID',
     Value:'SSSE12'},

]
}
}
```

## Приложение Г

### Экспериментальные данные

Здесь приведена лишь часть экспериментальных данных (Общая длина файла примерно 10000 инцидентов).

Таблица Г.1 — Описание экспериментальных данных

Класс проблемы	Описание проблемы	% успешных
Проблема с ПО		64%
Проблемы во время работы		10%
Как сделать		10%
	Hi NAS Admin,Please connect following groups to the shared disk listed below and configure security permissions.	Разрешена, уточнены выявлены концепции: connect, following groups, shared disks.
	Failed LOT OrderReciever.	Не решено, не достаточно информации.
	Delivery of request VCC395244Service: Network Services - LAN - LAN Order of AliasService manager.	Не разрешено, не выявлено ключевых концепций проблемы, например, failed, error.
	User needs latest version of Catia v5 Teamcenter.reinstalled installed.He has been instructed by the design support to ask for an installation.	Разрешена, выявлены концепции: user, needs, Catia v5.

**Таблица Г.1 – продолжение**

<b>Входное предложение</b>	<b>Описание</b>	
	User should have gotten WordFinder(9, En-Sv/ v-En Affärsekonomisk) and WordFinder(9, Ne-Sv Sv-Ne, Nederländska) installed but that is not the case. please install the applciation for the userName: DELVA.	Не разрешено, непонятно, что делать с первым пользователем.
	What do you want to do?: Add new aliasHost name on host that alias is wanted to.	Разрешена, выявлены концепции: add, aliashost.
... Проблема с оборудованием	...	... 0%
	VCC ParamServicesFailingService The status of the lcfd service is Degraded. Critical.	Не разрешено, проблема с аппаратной частью.
	HeartBeat EndpointUnreachable Tivoli lcf endpoint is unreachable.	Не разрешено, проблема с аппаратной частью.
	TMW ProcessorBusy.	Не разрешено, не достаточное описание проблемы.
...	...	...
Установить новое ПО		100%
	HyperionFM SmartView 9.3.3I need a new version of a software. HyperionFM SmartView 9.3.3 is the new version I need..	Разрешено, выявлены концепции: i, need, SmartView 9.3.3.

**Таблица Г.1 – продолжение**

<b>Входное предложение</b>	<b>Описание</b>	
	User requests internet explorer 8 to be installed.	Разрешено, выявлены концепции: user, requests, internet explorer 8.
	The installation of Winrar that I got this afternoon did go wrong. During installation nothing else was running. When I tried to start Winrar I got the fault message that is attached here.	Разрешено, выявлены концепции: winrar, installation, did wrong. Решение — переустановить.
	User got wrong homepage. He want to have vcc-intranet as homepage, but got something other "email-related".	Разрешено, выявлены концепции: user, got wrong, homepage, he, want, vcc-intranet.
...	...	...
Проблема с печатью		80%
	User is not able to print from TIE. He is using compability view in IE8Name.	Разрешено, выявлены концепции: user, not able, print, TIE, compability, IE8.
	User is not able to print with PDF995. he gets a error saying Could not load.	Разрешено, выявлены концепции: user, not able, print, PDF995.
	According to IM3548717 user couldt print from SAP, now she can print from SAP, but other programs such as Outlook and excel wont work, getting errormessage "unable to connect.	Разрешено, выявлены концепции: user, getting, error message, unable, connect.

**Таблица Г.1 – продолжение**

<b>Входное предложение</b>	<b>Описание</b>	
	User is trying to print to PR40378 from Adobe Reader and Powerpoint but gets "printing failed - no pages selected".	Не разрешено, выявлены концепции: user, trying, print, Adobe Reader, PowerPoint, printing, failed. Неоднозначность запроса для системы.
...	...	...
Нет доступа		100%
	User gets Invalid Login when trying to logon to Teamcenter. He has got a new password and changed it on the new CDSID page but it still does not work.	Разрешено, создан новый логин.
	Problems with 3G modems after migration: "Mobile connection not possible. Ensure that no other programs are using your selected device, and try again in a short while" The user is a agent working at the UK.	Разрешено, система уточнила проблему в диалоге с пользователем.
	User had received wrong application. User has ordered Wordfinder Business Economical. However she received wrong version, she received Wordfinder Tehcnical instead of Business Economical.	Разрешена, уточнены выявлены концепции: Wordfinder Tehcnical, Business Economical

**Таблица Г.1 – продолжение**

<b>Входное предложение</b>	<b>Описание</b>	
	User needs to have pdf995 re-installed please.	Разрешено. Выявлены концепции pdf, need и install.
	User doesn't have the Intel proset wireless application installed on his computer.	Разрешено, выявлены концепции: doesn't have, Intel proset.
	"Teamcenter & tc vis" have been uninstalled somehow from users clientPlease reinstall.	Разрешено, выявлены концепции: reinstall, users, "Teamcenter & tc vis"
	User can no longer access any wireless networksproblem with user profile, needs to be reconfigured, installed.	Разрешено. Найдены концепции: wireless networksproblem, user, user profile, reconfigured.
...	...	...

**Приложение Д****Свидетельство о регистрации**

Рис. Д.1 — Свидетельство о регистрации

## Приложение Е

### Акт о внедрении

