Казанский (Приволжский) федеральный университет

На правах рукописи
УДК 004.8

Тощев Александр Сергеевич

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ ИТ-СЛУЖБЫ ПРЕДПРИЯТИЯ

Специальность 05.13.01 —
«Системный анализ, управление и обработка информации (информационные технологии)»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
доктор физико-математических наук, профессор,
заслуженный деятель науки Республики Татарстан
А.М. Елизаров

Казань — 2016

Оглавление

Стр.

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 востребована 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].

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 структуры предприятия) и построения модели си-

стемы. Акцент был сделан на создании интеллектуальной системы для решения широкого круга проблем.

Следует отметить, что большинство проблем, которые решает удаленная служба поддержки информационной структуры 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. Процентное соотношение специалистов разных линий поддержки отображено на рисунке 1.

Таблица 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

Рисунок 1 — Диаграмма состава команд

Работа специалистов первой линии поддержки состоит из множества рутинных и простых задач. На рисунке 2 показано соотношение разных типов проблем, встречающихся во время работы службы поддержки, в таблице 2 приведена расшифровка типов. Данные подготовлены на основе анализа работы команд ОАО «АйСиЭл КПО-ВС (г. Казань)».

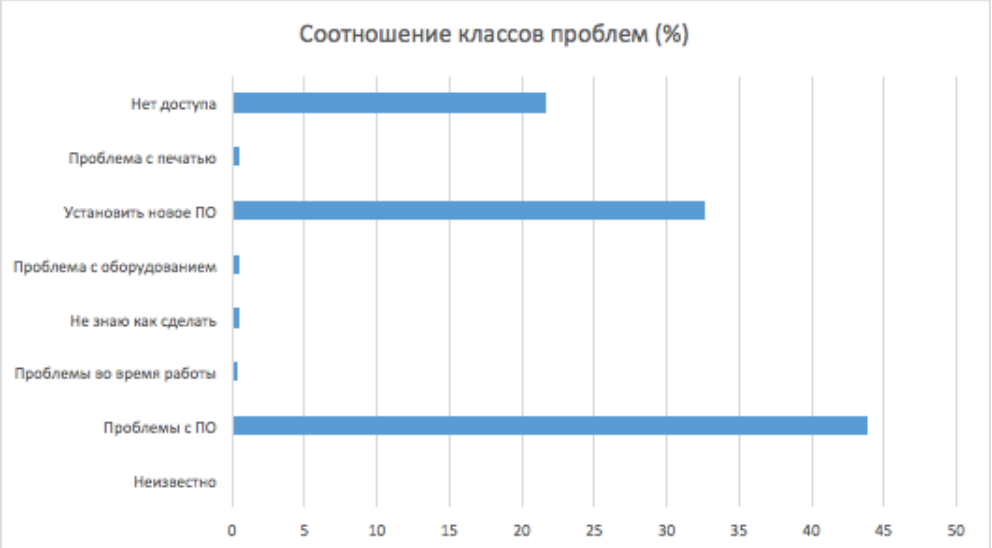

Рисунок 2 — Диаграмма соотношений типов проблем

Таблица 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

для решения более сложных задач.

Предпосылки развития изучаемой предметной области. Основной тенденцией в развитии области удаленной поддержки 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 [17] позволяет провести автоматизацию даже самых наукоемких процессов. Дальнейшей перспективой развития области является замена человеческих специалистов автоматизированными системами. Разработки в этом направлении ведут многие компании, например, компания HP, которая имеет свою систему регистрации различных инцидентов и сейчас ведет работу над ее автоматизацией. В качестве некоторого сравнения можно провести параллель происходящего процесса с промышленной революцией XVIII–XIX веков (см., например, [18]).

В главе 1 рассмотрены исследования в области интеллектуальных систем повышения эффективности ИТ-службы предприятия. Компания HP и IBM широко ведут исследования в этой области, используя технологии обработки естественного языка. В области обработки естественного языка также широко известен подход GATE [19] , исследование и развитие которого активно ведется Университетом Шеффилда. Важно отметить, что интеллектуальные системы также исследуются в сфере телекоммуникаций, подобное исследование ведется в Институте Чиная [20].

Методологии, 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) [21–23]. Наличие стандартов диктует унифицированность как постановки проблем, так и алгоритмов решения, а также способствует возможности частичной или в некоторых случаях полной автоматизации решения проблем.

Оценка стоимости работы специалиста при автоматизации.

По данным аналитики портала SuperJob [24], в Казани средняя зарплата системного администратора с опытом работы в 2014 году составляла 30 – 35 тыс.

руб. 2014 года (из расчета на 1 час с учетом 21 рабочего дня в месяце — 179–208 руб. в час. В соответствии с действующим российским законодательством [25] расходы компании на одного работника определяются по формуле

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

где R — выплата человеку в час, F1 — НДФЛ 13%, F2 — совокупность отчислений в ФБ (6%), ПФР (14%), ТФОМС (2%), ФФОМС (1,1%), ФСС (2,9%), F3 — 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 (Xeon X3, 1.7 GHz, 8GB RAM, 256GB SSD) стоит 8 900 руб./мес. (см. [26]) (53 рубля за 1 час 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. 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)»). Грубый подсчет на основе стоимости часа и пропорции показывает, что работа специалиста — это 6% работы сервера (без учета работы сервера параллельно над несколькими задачами). Таким образом, уровень разрешения инцидентов системой в 50% выполнит требования по прибыли примерно 186%.

Общая характеристика диссертации

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

Область исследования — разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации в ИТ-отрасли.

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

Методы исследования — теоретические методы: метод идеализации, метод формализации; специальные методы: системное моделирование, системный

анализ; экспериментальные методы: метод наблюдений, проведение экспериментов.

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

1. Провести теоретико-множественный и теоретико-информационный анализ сложных систем в области поддержки информационной инфраструктуры;
2. Разработать модель проблемно-ориентированной системы управления, принятия решений и оптимизации процесса принятия, анализа и обработки запросов пользователей в области обслуживания информационной структуры предприятия;
3. На основе построенной модели разработать архитектуру и создать прототип интеллектуальной вопросно-ответной системы повышения эффективности деятельности ИТ-службы предприятия;
4. Провести апробацию прототипа на тестовых данных.

Основные положения, выносимые на защиту:

1. Результаты теоретико-множественного и теоретико-информационного анализа сложных систем в области поддержки ИТ-инфраструктуры предприятия;
2. Построенная модель проблемно-ориентированной системы управления, принятия решений и оптимизации процессов обработки запросов пользователей в области обслуживания ИТ-инфраструктуры предприятия;
3. Созданный прототип программной реализации модели проблемно-ориентированной системы управления, принятия решений и оптимизации обработки запросов пользователей в области обслуживания ИТ-инфраструктуры предприятия;
4. Результаты апробации прототипа проблемно-ориентированной системы управления, принятия решений и оптимизации деятельности на контрольных примерах и анализ ее результатов.

Научная новизна проведенного исследования состоит в следующем:

1. На основе модели мышления создана модель проблемно-ориентированной системы управления, принятия решений в области обслуживания ИТ-инфраструктуры предприятия;
2. Представлены новая модель данных для модели мышления и оригинальный способ хранения данных для этой модели, эффективный по сравне-

нию со стандартными способами хранения данных (такие, как реляционные базы данных);

3. Выполнено оригинальное исследование моделей мышления применительно к области обслуживания информационной структуры предприятия;
4. На основе модели мышления Мински созданы архитектура системы обслуживания информационной структуры предприятия и программный прототип этой системы.

Практическая значимость. Система, разработанная в рамках данной диссертации, носит значимый практический характер. Идея работы зародилась под влиянием производственных проблем в ИТ-отрасли, с которыми автор сталкивался каждый день 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 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 системного анализа, 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 05.13.01 — Системный анализ, управление и обработка информации, сопоставление приведено в таблице 3.

Таблица 3 — Сопоставление направлений исследований в рамках специальности 05.13.01 и исследований, проведенных в диссертации

Направление исследования	**Результат работы**

Таблица 3 – продолжение

Направление исследования	Результат работы
Разработка критериев и моделей описания и оценки эффективности решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации	Разработана модель системы принятия решения и обработки информации в сфере поддержки ИТ-инфраструктуры предприятия.
Разработка проблемно-ориентированных систем управления, принятия решений и оптимизации технических объектов	Разработан прототип Thinking Understanding (TU) системы принятия решений в сфере поддержки ИТ-инфраструктуры предприятия, который был испытан на модельных данных.
Методы получения, анализа и обработки экспертной информации	Разработан метод обработки экспертной информации с возможностью обучения при помощи прототипа TU.
Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений и обработки информации	Созданы специальные алгоритмы для анализа запросов пользователей и принятия решений.
Теоретико-множественный и теоретико-информационный анализ сложных систем	Проведен комплексный анализ области поддержки программного обеспечения крупного ИТ-предприятия, с помощью которого построена модель области и выделены направления и возможности оптимизации принятия решений.

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)». Созданная система показала требуемые результаты обработки данной информации (процент успешно обработанных запросов составил более чем 30%).

Личный вклад. Автор исследовал целевую область: провел анализ запросов пользователей и классифицировал их; построил модель целевой области и выявил возможности оптимизации. Совместно с Талановым М.О. 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 11 печатных изданиях [27–37], из которых статьи [33; 34] проиндексированы в БД Scopus и входят в перечень ВАК, статья [34] проиндексирована в БД Web of Science и входит в перечень ВАК, работа [35] опубликована в журнале из списка ВАК, статья [30] проиндексирована в БД РИНЦ, работы [27], [28–30] опубликованы в материалах международных и всероссийских конференций.

Объем и структура работы. Диссертация состоит из введения, четырех глав, заключения и пяти приложений. Полный объём диссертации составляет 126 страниц с 43 рисунками и 27 таблицами. Список литературы содержит 0 наименований.

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

1.1 Сравнительный анализ систем регистрации и устранения проблемных ситуаций

В данной главе рассматриваются имеющиеся на данный момент интеллектуальные системы регистрации и анализа проблемных ситуаций.

HP OpenView [38] [39] [40] [41] является комплексным программным решением по мониторингу ИТ-инфраструктуры предприятия и имеет множество модулей. На рисунке 1.1 представлен вид системы, которая обладает широким спектром возможностей: мониторинг [42] [43]; регистрация инцидентов; управление системами. Система не поддерживает: понимание и формализацию запросов; автоматическое исправление проблемы на основе формализации запроса.

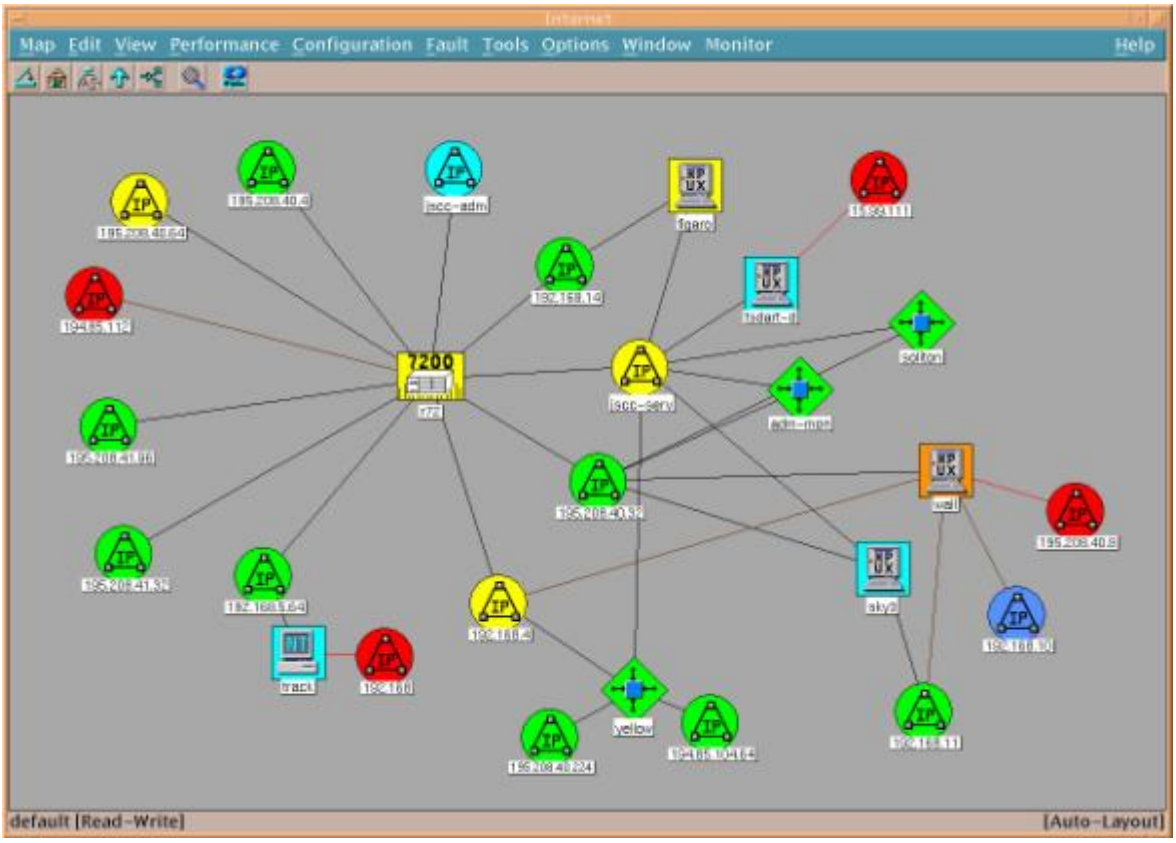

Рисунок 1.1 — HP OpenView

Система **ServiceNOW** — средства автоматизации сервиса. На рисунке 1.2 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 [44] [45] для регистрации инцидентов и их решения.

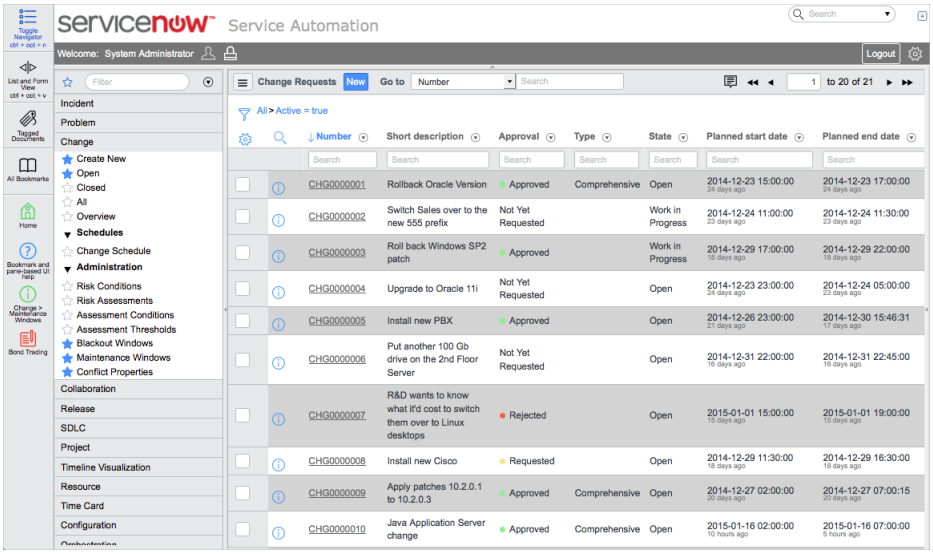

Рисунок 1.2 — Service NOW

IBMWatson — это вопросно-ответная система, которая поддерживает: понимание и формализацию запросов и поиск решений. Система не поддерживает автоматическое исправление проблемы на основе формализации запроса. Система широко используется в медицине для постановки диагнозов болезней [46] [47] [48] [49] и реализует базовые принципы искусственного интеллекта [50] [51]. Ее разработка велась под суперкомпьютер IBM Deep Blue [52]. На рисунке 1.3 представлен общий вид этой системы.

Кроме того, известны следующие дополнительные способы и системы автоматизации:

– 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 [53], диагностики активных систем [54], анализа поведения функций [55], обработки данных в системе eDiscovery [56], в разработке способах программирования [57];
– Обработка инцидентов при помощи скриптов — автоматизируются лишь рутинные операции.

Таким образом, был сформирован список требований к системе, которые в следующем разделе будут уточнены и формализованы. На данный момент ни

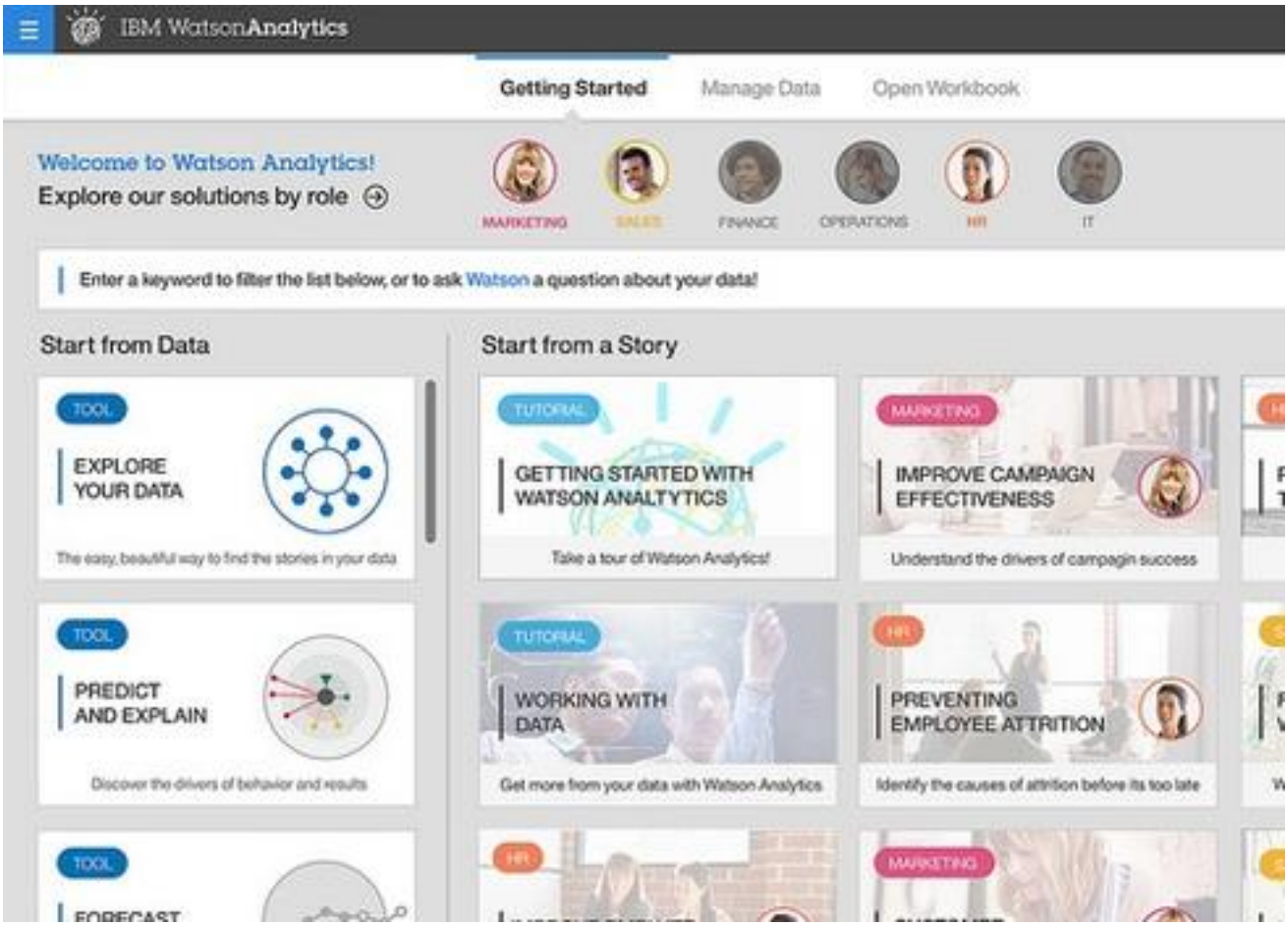

Рисунок 1.3 — Пример работы системы Watson

одна из исследованных систем в полной мере автоматически не решает запросы пользователей: фиксирует их, проводит анализ, ищет решение, применяет решение и дает обратную связь пользователю. Каждая система в той или иной мере реализует те или иные критерии, но системы, которая отвечает всем критериям нет.

1.2 Основные требования к интеллектуальным системам регистрации и анализа проблемных ситуаций в ИТ-области

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 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 генерализацию различающейся части: Firefox и Chrome являются пакетами программного обеспечения.

Итак, основными требованиями к интеллектуальным системам регистрации и анализа проблемных ситуаций в ИТ-области являются возможности реализации этими системами: мониторинга ИТ-инфраструктуры пользователя; регистрации инцидентов; создания цепи обработки (Workflow) инцидента; понимания и формализации запросов на естественном языке; поиска решений и применения найденных решений; обучения решению инцидента; умения проводить логические рассуждения (генерализацию, специализацию, синонимичный поиск).

Из всего списка требований к системе важно выделить формализацию запросов на естественном языке. Это отдельная и обширная область исследования. Ниже приведены результаты анализа разработок в данной области.

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)». Ввиду 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 ITCOORD-18125. However she received wrong version, she received Wordfinder Tehcnical instead of Business Economical. Please assist.*

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.*

Инцидент 3. *User cannot find Produkt Manageron start menu. Please reinstall.*

Инцидент 4. *User needs to have pdf 995 re-installed please.*

При анализе были использованы следующие обработчики естественного языка: Open NLP [58], Relex [59], StanfordParser [60]. Результат их работы оценивался при помощи метрик, представленных в таблице 1.1, а полученные результаты приведены на рисунке 1.4.

Из диаграммы видно, что наилучшее результаты показывает обработчик Relex [59]. После анализа необработанных инцидентов у всех обработчиков было выявлены проблемы двух типов:

1. невозможность корректировки простых грамматических ошибок, связанных с пропущенными пробелами или неверным форматированием (ошибки первого типа);

Таблица 1.1 — Таблица метрик

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

2. неверная интерпретация слов в предложении, например, слово please интерпретировалось как глагол, хотя является по смыслу ”формой вежливости”(ошибки второго типа).

Несмотря на хорошие результаты — 63% успешно разобранных предложений, ошибки первого и второго типа серьезно ухудшают результат. Эффективность разбора предложений в 63% недостаточна для успешной работы системы. Чтобы улучшить показатель, был создан комплекс мер для устранения ошибок первого и второго типа.

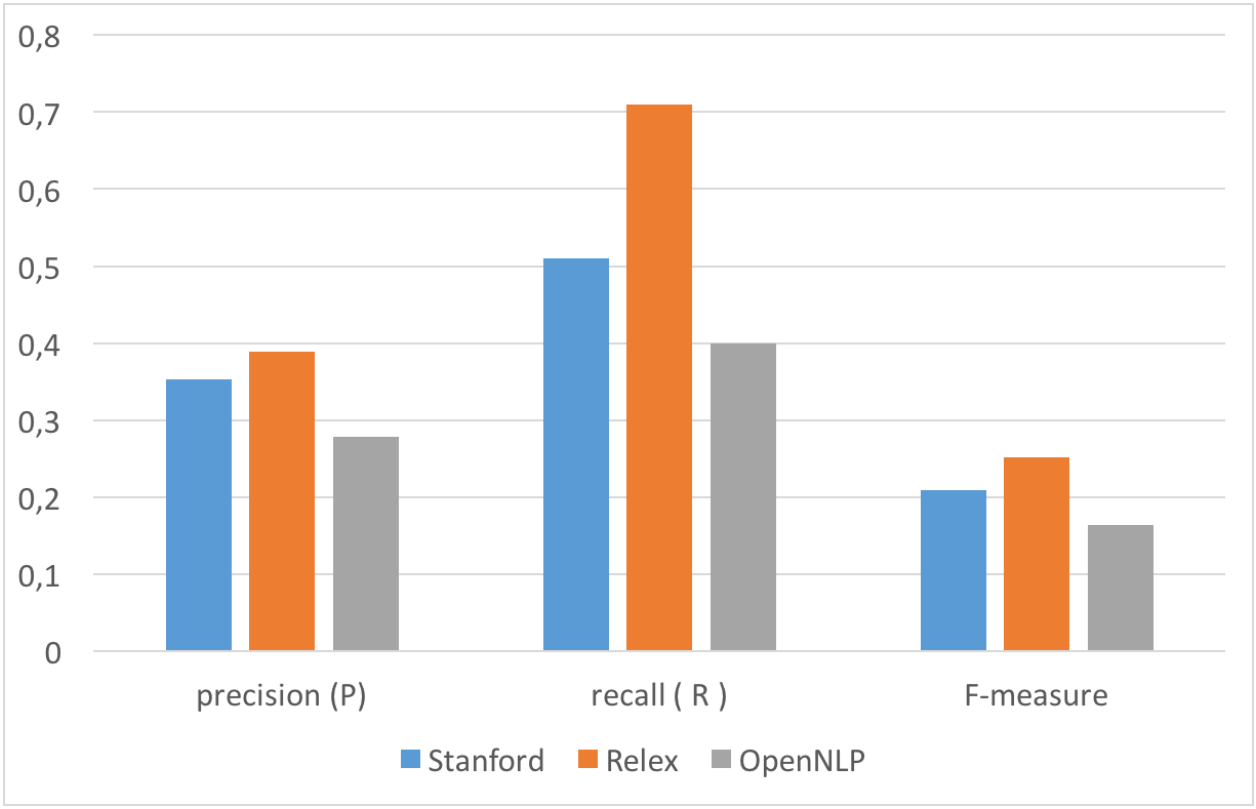

Рисунок 1.4 — Результаты обработки текстов

1.3.2 Исправление ошибок первого и второго типа

Чтобы исправить проблемы, связанные с ошибками первого и второго типа, была введена 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 (рис. 1.5). В результаты были сконструированы модули корректировки: Google API — модуль подключения к открытым 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, связанный с синтаксисом, грамматикой и орфографией. Также удалось исправить ошибки неверного написания: наличия лишних пробелов, пропуска запятых и точек. Но по-прежнему осталась проблема обработки неверной интерпретации слов в тексте.

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 алгоритм работы такой же как был до этого в модули, 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 базе знаний нашей системы отмечено, что ”please-— это форма вежливости, тем самым Relex больше не интерпретирует ”please”как глагол.

1.3.3 Сравнение средств обработки русского и английского языков

Средства обработки естественного языка принято относить к большому классу средств NLP – Natural Language Processing [61]. Для английского языка существует множество открытых средств обработки естественного языка, для русского языка найти их гораздо сложнее. Рассмотрим архитектуру средств обработки естественного языка на примере OpenCog Relex.

OpenCog Relex использует результаты работы открытого компонента для лексического анализа под названием Link Grammar [62]. 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, KBP)
pos(start, verb)
inflection-TAG(start, .v)
tense(start, present)
pos([web], WORD)
noun_number(KBP, singular)
definite-FLAG(KBP, T)
pos(KBP, noun)
_advmod(reinstall, please)
pos(reinstall, verb)
inflection-TAG(reinstall, .v)
tense(reinstall, present)
pos(please, adv)
inflection-TAG(please, .e)
noun_number(Java, singular)
definite-FLAG(Java, T)
pos(Java, noun)
pos(., punctuation)
_obj(,, Java)
```

```
pos(,, verb)
tense(,, infinitive)
HYP(,, T)
_to-do(unable, ,)
pos(unable, adj)
inflection-TAG(unable, .a)
tense(unable, present)
pos(to, prep)
inflection-TAG(to, .r)
pos(be, verb)
inflection-TAG(be, .v)
_predadj(User, unable)
noun_number(User, singular)
definite-FLAG(User, T)
pos(User, noun)
```

Далее возьмем разбор слова start. В результате мы получаем 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.

1.4 Выводы

В данной главе были рассмотрены существующие на данный момент интеллектуальные системы регистрации и анализа проблемных ситуаций, возникающих в ИТ-инфраструктуре предприятия. В таблице 1.2 приведены сводные данные по системам. В главе также выработаны критерии сравнения обработчиков естественного языка и выполнен основной анализ обработчиков естественного языка. По показателям эффективности было решено использовать OpenCog Relex.

Таблица 1.2 — Сравнительный анализ 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 формализацию запросов на естественном языке	Нет	Нет	Да
Поиск решений	Нет	Нет	Да
Применение решений	Нет	Нет	Нет
Обучение решению инцидента	Нет	Нет	Да
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 рассматриваются 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 [63] ;
- модель TU 1.0, основанная на модели мышления Марвина Мински [64].

Модель, построенная на базе нейронных сетей (поддерживающая обучение) была отброшена на предварительной стадии оценки, так как она предъявляет большие требования к производительности [65], 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 являлась одной из первых, которые были опробована. Она основана на деревьях принятия решений [66], которые широко используются в вопросно-ответных системах [67], [68], [69]. При построении модели использовались следующие компоненты: обработка запросов на естественном языке; поиск решения; применение найденного решения; база знаний.

Система ориентирована на выполнение таких простых команд, как, например, ”Добавить поле на форму”. Основные функции модели представлены следу-

ющими 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 представлены данные в системе.

2.1.1 База знаний на основе OWL

В качестве представления данных в системе использовалась база знаний на основе OWL-файла (см. [70]). С помощью редактора Protege [71] в базу вводились начальные данные о целевом приложении (приложение, которое будет модифицироваться системой согласно запросам от пользователей) в виде семантической сети. В качестве такого приложения была создана простая программа, которая вела учет заказов на покупку книг от пользователя.

На рисунке 2.1 представлен один из классов этой программы — 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. Сюда входит описание всех необходимых свойств для генерации файла исходного кода на языке Java.

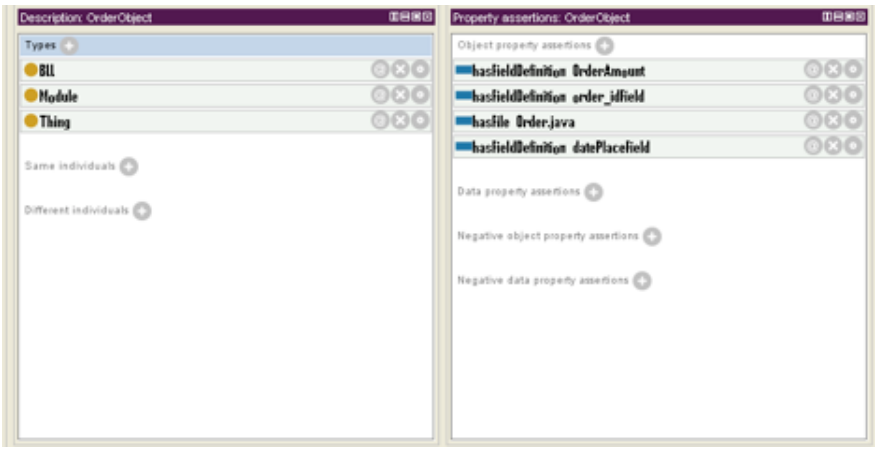

Рисунок 2.1 — Представление класса Order в OWL. Визуализация Protege.

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

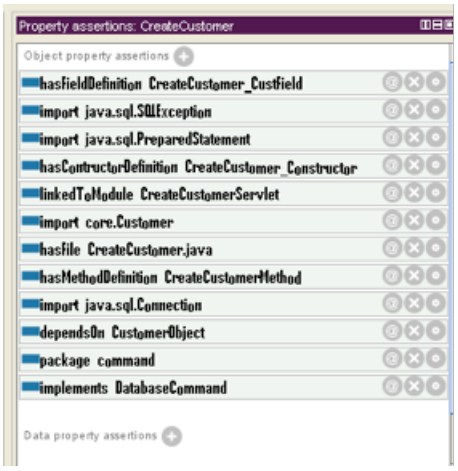

Рисунок 2.2 — Представление класса CreateCustiner в OWL. Визуализация Protege.

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 Action объект из результатов работы парсера, основываясь на Деревьях принятия решений [72] 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 поддерживается 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”; модуль StanfordParser вычлиняет из запроса связи типа dobj (связь объекта и действия); модуль ActionGenerator создает на основе связей действие; модуль ActionApplier, используя действие изменяет модель приложения; модуль ApplicationGenerator применяет изменения к приложению. В результате у класса Customer (представляет клиента магазина в приложении) 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 подобную операцию — добавить метод — сделать не может. Поиск решения также требовал специального обучения: система не могла путем перебора информации в своей базе знаний найти решение. В системе также не было возможности обучения.

После анализа ошибок была предпринята попытка найти более универсальное решение. Результатом стало построение Menta 0.3, описанной ниже.

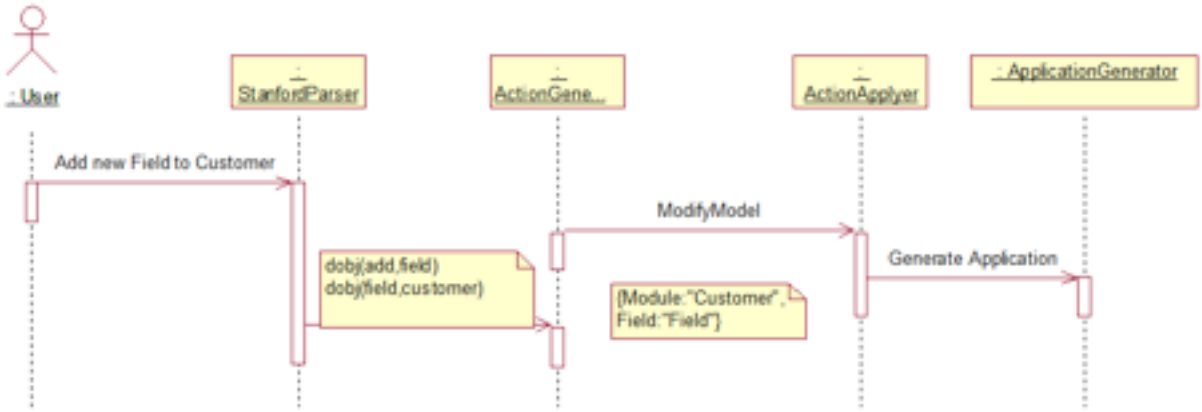

Рисунок 2.3 — UML диаграмма последовательности для основного потока в модели Menta 0.1

2.2 Модель Menta 0.3 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 [73], [74]. Кроме того, есть примеры их использования и в системах поддержки принятия решений, однако эффективность таких систем не подтверждена [75]. 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 решение проверяла его при помощи логического движка NARS [76] на соответствие входным критериям приемки. С точки зрения генетических алгоритмов это — функция отбора особей из поколения [77].

2.2.1 Основные компоненты модели

Модель состоит из компонентов, представленных в Таблице 2.3.

Таблица 2.3 — Компонетны модели Menta 0.3

Компонент	Описание
MentaController	Веб-служба [78], который 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 ecj [79] . 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 максимально лучше 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.

Таблица 2.3 – продолжение

Компонент	Описание
ReasonerAdapter	Транслирует How-To в термины NARS. NARS – non-axiomatic reasoning system [76](система логических суждений, разработанная профессором Пеем Вонгом). Принцип действия NARS – это всевозможная комбинация фактов. Каждый факт имеет свои частоту и вероятность. Их сочетанием получается композиция данных фактов.
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 в файл, что было неудобно в случае, если приложение работает параллельно над несколькими запросами. В системе Menta 0.3 стал использоваться специальный сервер баз данных, речь о котором пойдет далее.

2.2.2 База знаний на основе графов

При реализации KBServer был создан промежуточный слой DAO (Data Access Object), данный подход широко используется в проектировании про-

граммного обеспечения [80]. 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, хранения объектов из базы данных.

Configuration. Этот класс хранит параметры настройки базы данных, такие 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 были проанализированы 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**
Единицы измерения	**мс.**	**мс.**	**мс.**
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 лучшее результаты, был выбран HGDB, так как HGDB 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 кон-

трольных примерах: Acceptance Criteria; Обучение; Поиск и применение решения; Отсутсвие 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” [64], в которой предложил свой взгляд на систему мышления и памяти человека. В основу теории легла парадигма триплета Критик – Селектор – Образ мышления, k-line (линия, которая связывает приобретенные знаний, например, огонь — горячо) для сопоставления знаний. На рисунке 2.4 представлена схематичное изображение триплета Критик – Селектор – Образ мышления.

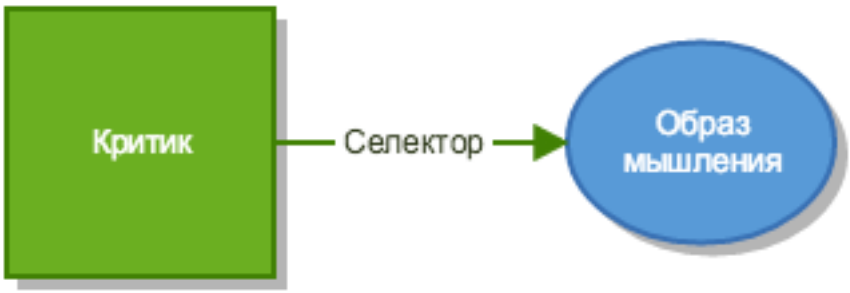

Рисунок 2.4 — Критик – Селектор – Образ мышления

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 могут активироваться несколько критиков. Например, человек решает сложную задачу, идет активация множество критиков: выполнить расчет, уточнить технические детали. Кроме того, параллельно может активироваться критик переработки, сообщающий о необходимости отдыха.

Селектор занимается выбором определенных ресурсов, которыми также являются Образ мышления.

Образ мышления — это способ решения проблемы. Образ мышления может быть сложным и, например, активировать другие критики. Например, размышляя над проблемой, специалист понимает, что нужно произвести полный перебор, и тут он решает поискать готовое решение: а может кто-то уже сделал такой перебор и можно будет его использовать. Здесь ”поиск готового решения”является критиком внутри образа мышления ”поиск решения”.

На рисунке 2.5 представлена расширенная модель работы триплета Критик – Селектор – Образ мышления. Критик активирует Селектор, который активирует Образ мышления (синий круг). 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.

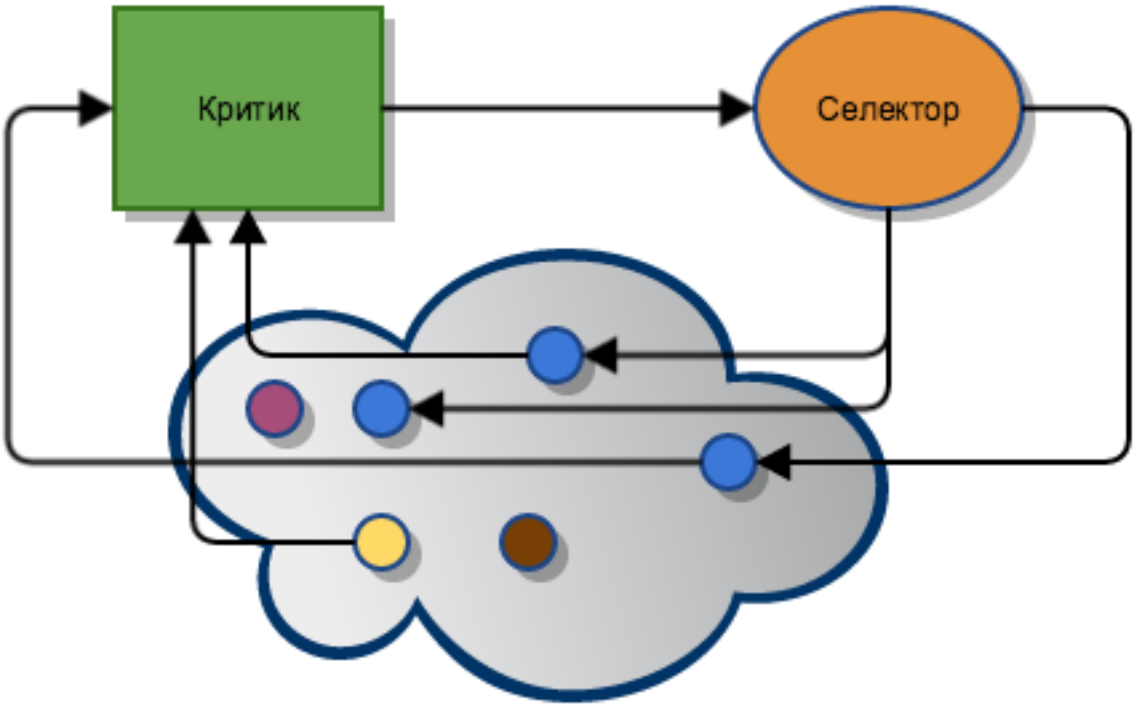

Рисунок 2.5 — Критик-Селектор-Путь мышления в разрезе ресурсов

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. Но наш взгляд, все это компенсируется возможностью изобретения образов мышления. Например, чтобы быть быстрыми, мы изобрели различные механизмы (самолеты, машины и др.). Чтобы быть сильными, мы изобрели оружие.

Все изобретения и являются результатом взаимодействие человека с окружающим миром. Именно данное взаимодействие заставляет людей изобретать что-то новое, создавать шедевры литературы и летать в космос. По ходу своего развития человек проходит от инстинктивного одергивания руки до создания Теории всего [81]. И в этом ему помогает возможность гибкого мышления: изобретение различных подходов к решению проблемы. Далее мы рассмотрим концепцию уровней мышления, следуя Марвину Мински.

1. Инстинктивный уровень
2. Уровень обученных реакций
3. Уровень рассуждений
4. Рефлексивный уровень
5. Саморефлексивный уровень
6. Самосознательный уровень

В Таблице 2.5 представлено описание уровней мышления.

Деление на данные уровне носит условный характер. Например уровень 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.

На рисунке 2.6 представлено схематичное изображение уровней мышления. 1-3 уровни составляют личность человека. 2-5 представляют ЭГО человека (Человеческое Я) - осознание человека в общении с окружающими. 3-6 представляют собой сверх ЭГО человека (сверх Я) - его моральные установки.

Таблица 2.5 — Описание уровней мышления Марвина Мински

Уровень	**Описание**
Инстинктивный уровень	На данном уровне происходят инстинктивные реакции (врожденные). Например, боязнь обжечься. Не прыгать под машину. Общую формулу для этого уровня можно выразить как ”Если ..., то сделать так”.
Уровень обученных реакций	На данном уровне происходит мышление обученных реакций, то есть тех реакций, которыми человек обучается в течение жизни. Например, переходить дорогу на зеленых свет. Общую формулу для этого уровня можно выразить как ”Если ..., то сделать так”.
Уровень рассуждений	На данном уровне происходит мышление с использованием рассуждений. Если я сделаю так, то будет ... Например, если перебежать дорогу на зеленый свет, то можно успеть вовремя. Здесь сравниваются последствия нескольких решений и выбирается оптимальное. Общую формулу для этого уровня можно выразить как ”Если ..., то сделать так, тогда будет так”.
Рефлексивный уровень	На данном уровне происходит рассуждение с учетом анализа прошлых событий. Например, прошлый раз я побежал на моргающий зеленый и чуть не попал под машину.
Саморефлексивный уровень	На данном уровне происходит оценка себя. Строится определенная модель 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 легкомысленный.
Самосознательный уровень	На данный момент характерен только для человека. На данном уровне идет оценка поступков человека с точки зрения высших идеалов и внешних оценок. Например, а что подумают мои друзья? А как бы поступил мой герой?

Рисунок 2.6 — Иллюстрация концепции Уровней мышления

Концепция k-line

Концепция K-line была первый раз упомянута Марвином Мински в 1987 году в журнале Cognitive Science. В книге ”The Society of Mind” [82] Марвин Мински раскрывает концепцию K-line. Полностью концепция описана позже в книге ”The Emotion Machine” [64]. K-line представляет собой связь между двумя событиями, объединяющими их в знание. Например, объединение Пути мышления, найденного решения и активированной проблемы. Данная линия объединяет то как мы думали, решение. На Рисунке 2.7 показана K-line, которая объединяет пу-

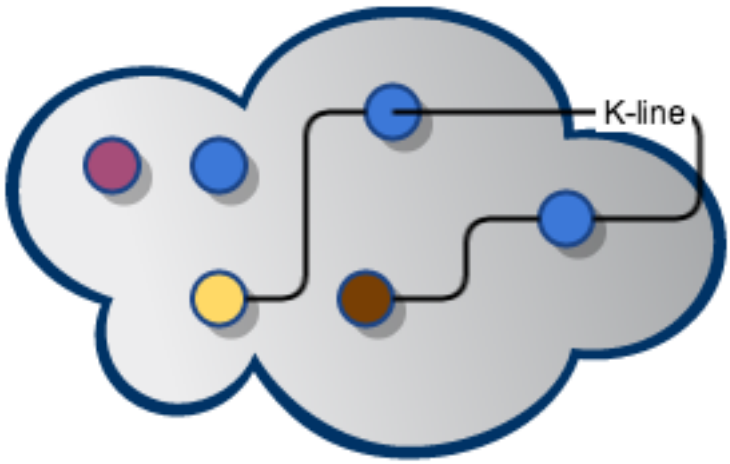

Рисунок 2.7 — Иллюстрация концепции K-line

ти мышления, решение и другие Критики. Данная концепция позволяет ”запоминать”удачные решения.

2.4 Выводы

Модель Menta 0.1 имеет следующие недостатки: отсутствие устойчивости к ошибкам входной информации: грамматическим и содержательным. 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 [72], он в рамках данной модели не рассматривался. Модель Menta 0.3 имеет следующие недостатки: отсутсвие обучения; отсутсвие обработки естественного языка; HyperGraphDB оказалась непригодный 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. Например при 10 фактов количество сочетаний будет равно 45 на первом уровне, далее будут сравнивать результаты этих сочетаний; апробации оказалось, что критерии приемки практически описывают необходимое решение, что являлось недопустимым. Данный подход был описан в статье [**?**].

Для программной экспертной системы очень важно обладать способностью мыслить и рассуждать. Например, очень важно для системы уметь действовать по аналогии. Так как множество запросов типичны и отличаются лишь параметрами. Например, пожалуйста, установить Office, Antivirus и т.д.

Также для экспертной системы важно уметь абстрагировать специализированные рецепты решения. К примеру, система научилась решать инцидент ”Please install Firefox”. Абстрагировав данный инцидент до степени ”Please install browser”система сможет теми же способами попробовать решить новый инцидент.

После рассмотрения нескольких моделей была выбрана модель мышления Марвина Мински, так как данная модель наиболее точно ложится на целевую область решения инцидентов в области IT. 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 система и программная реализация. Архитектура была создана с учетом принципов проектирования Enterprise [83] систем.

3.1 Архитектура системы

Данный раздел описывает основные режимы функционирования системы и концепцию построения системы. Архитектура системы представляет собой модульную систему для того, чтобы компоненты можно было удобно заменять [84]. Например, подключать различные обработчики естественного языка. Основные компоненты системы описаны в таблице 3.1.

Таблица 3.1 — Основные компоненты системы Thinking-Understanding

Компонент	Описание
TU Webservice	Основной компонент взаимодействия со внешними система, 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 спе-

циалист технической поддержки (TSS) (в общем случае это Пользователь (User)). Специалист технической поддержки может выполнять следующие действия: обучать систему; предоставить 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 формализован системой, но также решение данного запроса. Система формализует запрос, формализует 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 в посредством запроса CorrectSystemSolutions

Второй вариант использования это основной поток. Главными действующим лицом системы является заказчик (Customer)(в общем случае это базовый класс пользователь (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 сталкивается с проблемой, то она задаст вопрос пользователю.

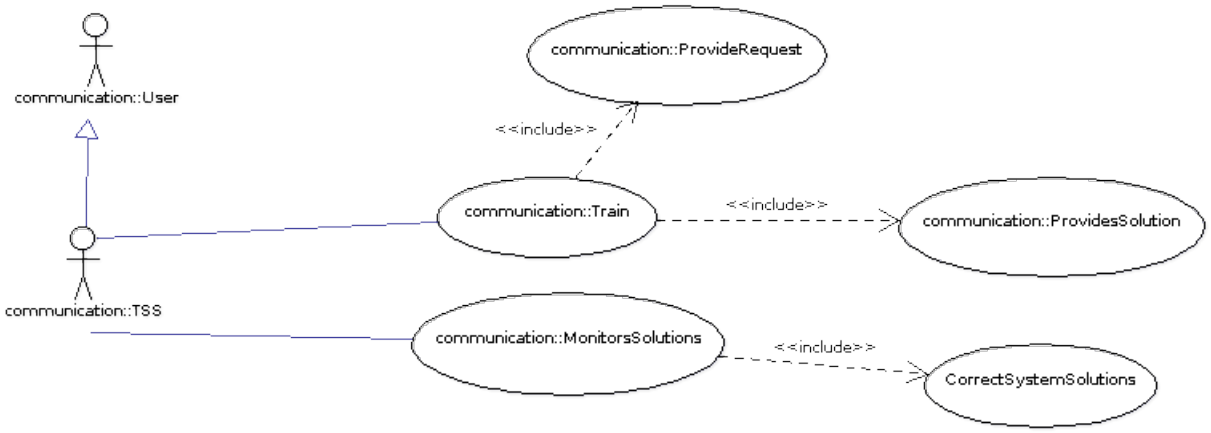

Рисунок 3.1 — Вариант использования. Обучение.

Таблица 3.3 — Описание ветвей в варианте использования ”Основной режим”

Ветвь	**Описание**
ProvideRequest	Заказчик вводит запрос в систему на естественном языке. Это может 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, то система запрашивает пользователя детали
communication:ProvideConfirmationResponse	В случае, когда система нашла решение, она запрашивает пользователя подтверждение о том, 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 заключительных главах будет приведен подробный алгоритм взаимодействия всех компонентов системы.

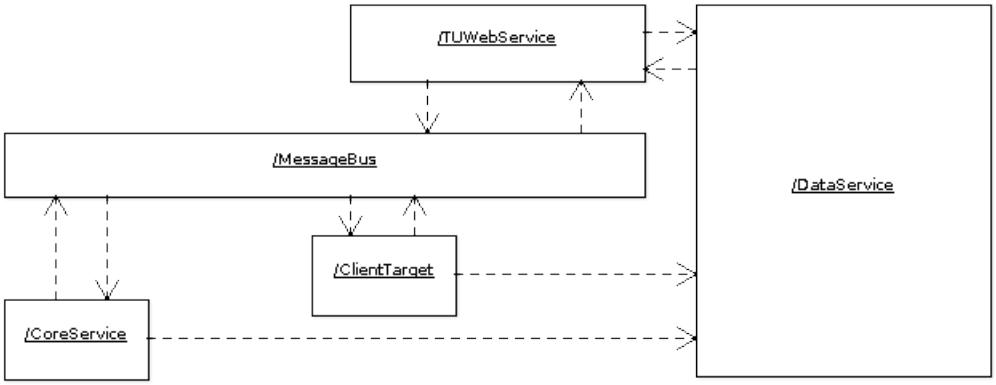

Рисунок 3.2 — Диграмма взаимодействия компонентов

Основной точкой взаимодействия (с позиции пользователя) с системой является компонент WebService 3.1.2. Данный компонент построен на стандарте WS SOAP для универсального использования в различных внешних системах [85] (иными словами легким предоставлением API). Взаимодействие происходит по следующей схеме.

1. WebService получает запрос пользователя. Сохраняет запрос в Базу Знаний (Базу данных) **??**.
2. WebService отправляет сообщение типа Request с информацией о запросе в компонент MessageBus (шина);
3. Один из экземпляров CoreService компонента обрабатывает запрос;
4. Компонент CoreService обрабатывает запрос и сохраняет результаты в Базу Знаний, затем он отправляет в MessageBus сообщение

RequestCompleted и сообщение ActionsToExecute с действиями, которые необходимо исполнить;

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 посылает их указанным компонентам. TUWebService компонент взаимодействия с ”внешней средой который предоставляет список функций и типов, посредствам вызова которых можем обработать запрос в системе. DataService — отвечает за хранения данных, предоставляет базу заний для приложения. CoreService — ядро приложения, управляет основным жизненным циклом системы. ClientTarget — клиентский компонент для выполнения команд на машинах клиента. Сюда относятся как и удаленное исполнение посредством WMI и т.п. так и непосредственная установка клиента на машины пользователей. На рисунке 3.3 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 главах данного раздела.

В этой главе были представлены компоненты и их декомпозиция в разрезе системы. На рисунках видна крупноблочная структура системы, а также детальная в разрезе общих блоков. Кроме того описано взаимодействие с системой с точки зрения конечного пользователя, а также взаимодействия со стороны других систем. Дано описание крупноблочных компонентов, а также детальное описание одного из основных компонентов.

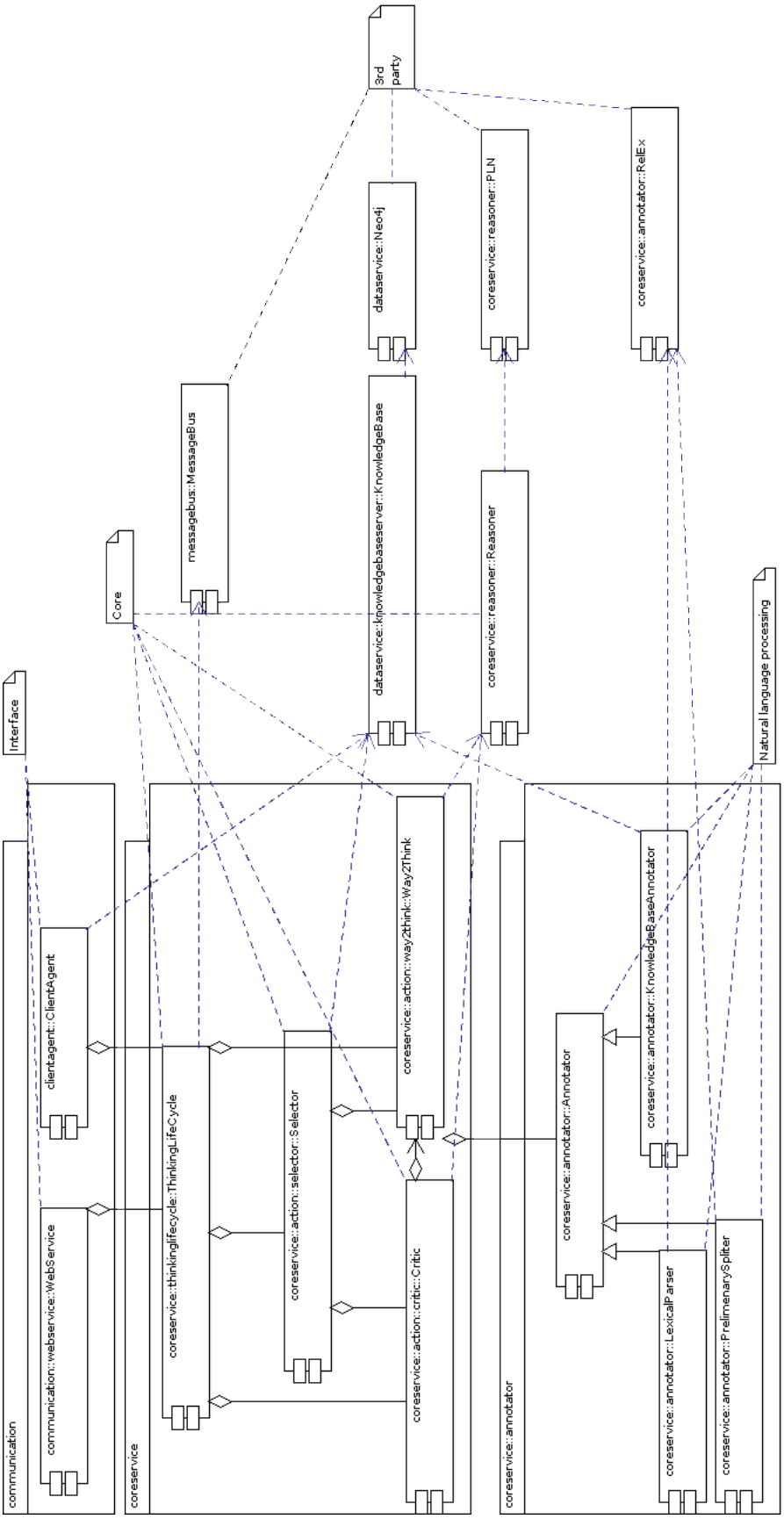

Рисунок 3.3 — Детальная диаграмма компонентов системы

3.1.2 Компонент WebService

Данный компонент обрабатывает запросы пользователей, а также внешних систем. Запрос пользователя представляется посредством объекта Request, который содержит информацию о пользователя, 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-

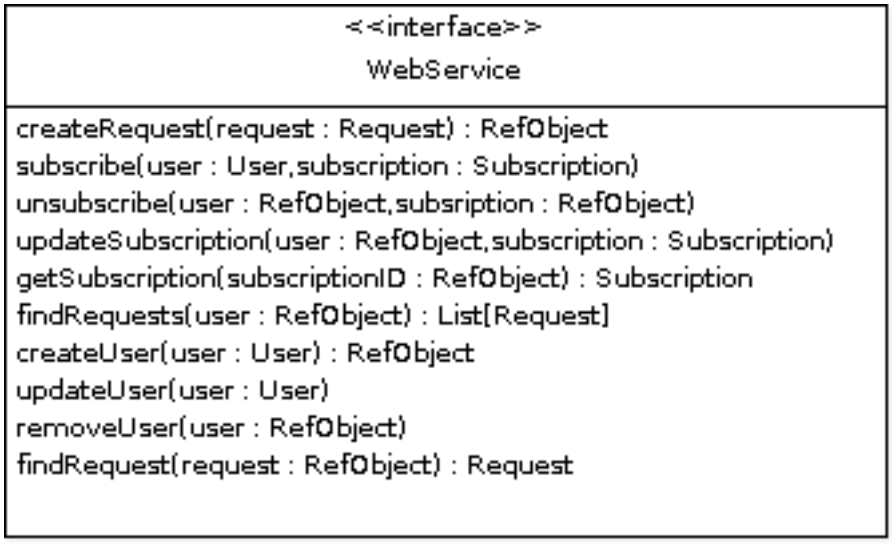

Рисунок 3.4 — Интерфейс компонента WebService

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 Базу Знаний и начинает его обработку;
3. Когда изменяется статус запрос 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 взаимодействия с системой и инкапсулировать от пользователя основную логику системы. Методы были созданы так, чтобы внешним системам не нужно было реализовывать модель данных системы, а использовать базовые облегченные объекты для приема и переда-

чи информации. Данный подход подробно описан в книге Мартина Фаулера ”Архитектура приложений” [86] и имеет название Data Transfer Object. В данном компоненте также широко исользуется подход Фасад, который также описан в книге. Архитектура компонента была проверена на предмет наличие антипаттернов проектирования, описанных в книге Вильяма Брауна ”Антипаттерны проектирования” [87].

В данной главе был описан компонент взаимодействия с пользователям и внешними системами. Описана архитектура компонента, основные методы, а также приведен пример основного потока.

Таблица 3.4 — Описание методов компонента WebService

Метод	**Описание**
createRequest(request:Request): [RefObject]	Создает запрос от пользователя. В качестве параметра в метод передается SubscriptionID, по которому идет проверка запроса.
subscribe(user:User, subscription:Subscription)	Создает подписку для 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), осуществляет обмен данных между компонентами. Компонент построен на фреймворке Akka Concurrency, который позволяет разрабатывать приложения, работающие параллельно [88]. 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 генерированных по шаблону инцидентов. Объект, который используется для обработки использует паттерн Akka [88]. 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 приводится в следующих главах.

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

На уровне Reflective работает Критик контроля времени, которые 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 в 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 (компонента) ThinkingLifeCycle

Метод	**Описание**
onMessage(message : Message)	Данный метод вызывается при получении сообщения от шины. После этого происходит обработка запроса, вычисляется список действий, которые нужно выполнить. После этого запускается исполнение этих действий. На рисунке 3.6 представлена диаграмма действий для 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. На рисунке 3.8 представлена диаграмма действий для 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 (пути мышления, непосредственно обработчики данных, классы, которые производят изменения данных) На рисунке 3.9 представлена диаграмма действий для этого метода.
	Продолжение следует

Таблица 3.5 – продолжение

Метод	Описание
processWay2Think(inputContext: Context, outputContext: Context): TransFrame	Данный метод запускает обработку WayToThink **??**. Данный метод 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 для этого метода.
processCritic(context: Context): List[SelectorRequestRulePair]	Данный метод запускает обработку Critic **??**. На рисунке 3.11 представлена диаграмма действий для этого метода.
init(): Boolean	Данный метод инициализирует экземпляр класса ThinkingLifeCycle. Во время инициализации происходит Базы Знаний **??**, подключения к Шине данных. На рисунке 3.12 представлена диаграмма действий для этого метода.
start(): Boolean	Данный метод является оберткой для поддержки Akka Concurrency. Он вызывает метод init.
stop(): Boolean	Данный метод является оберткой для поддержки Akka Concurrency. Он останавливает работу экземпляра класса: останавливается сессия к шине данных, останавливается подключение к Базе Знаний.
registerProcess(process : Process,level : Level) : Process	Данный метод регистрирует процесс в пуле. В качестве параметра принимается Level (уровень приоритета процесса).
stop(processLevel : Level) : List[Process]	Данный метод регистрирует останавливает процесс. В качестве параметра принимается ссылка на процесс. На рисунке 3.13 представлена диаграмма действий для этого метода.

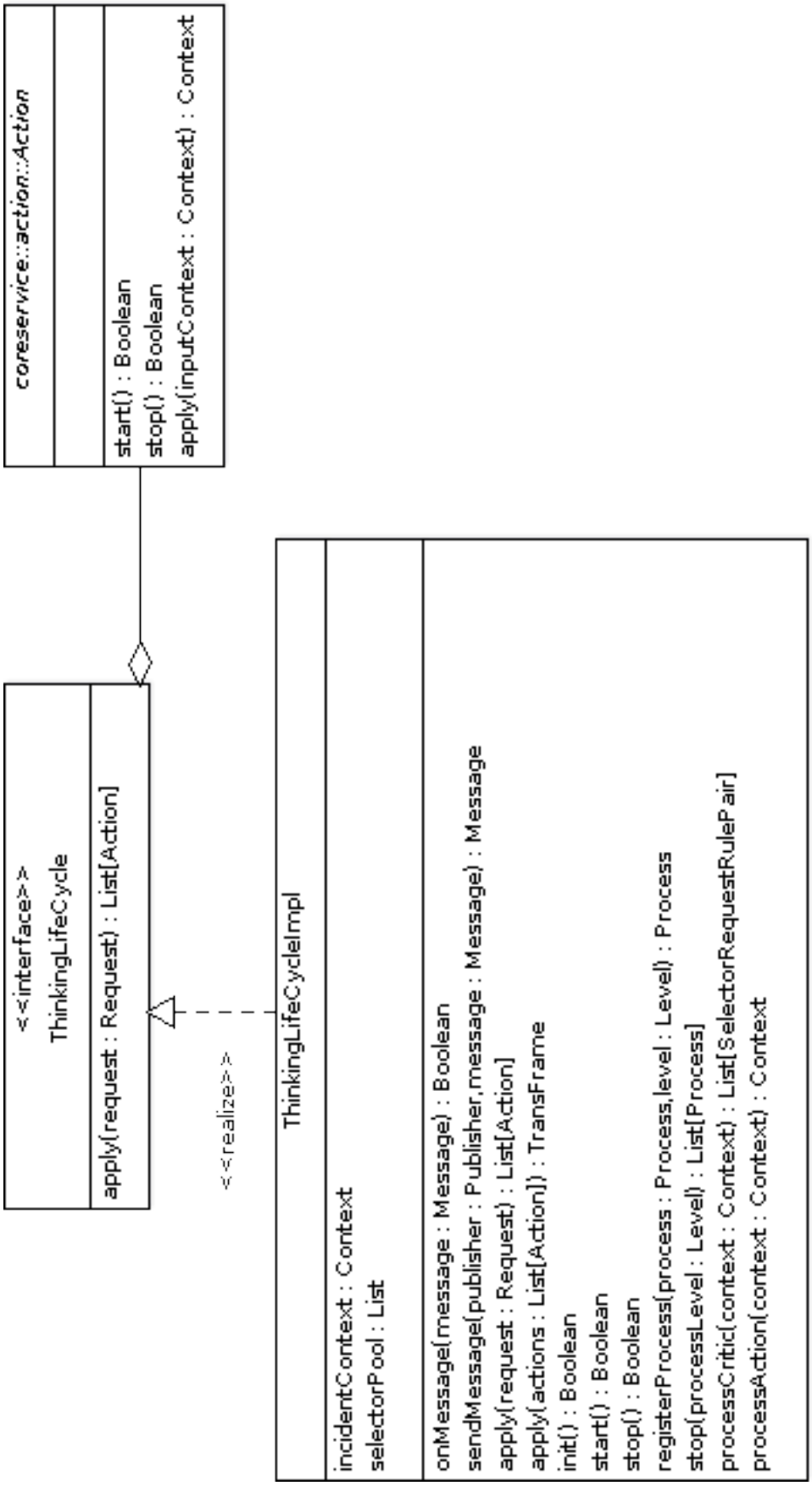

Рисунок 3.5 — Диаграмма классов ThinkingLifeCycle

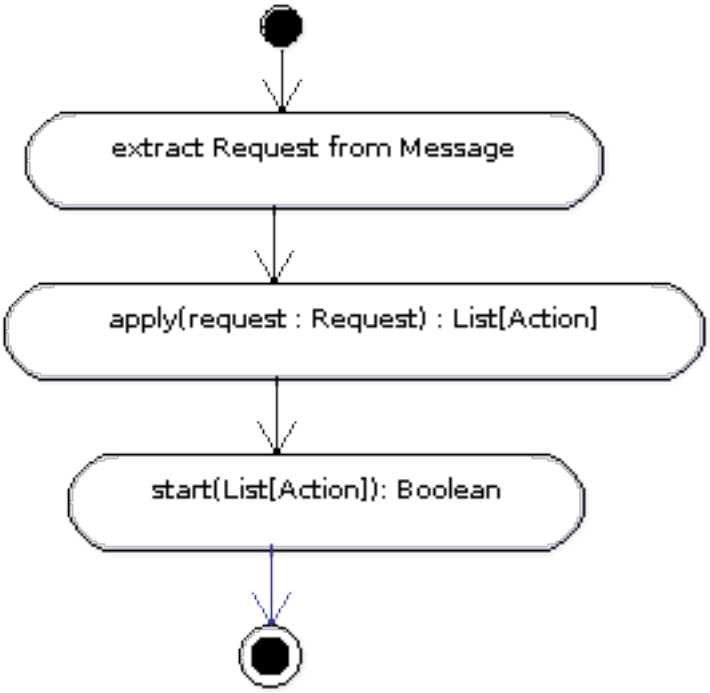

Рисунок 3.6 — Диаграмма действий метода onMessage компонента ThinkingLifeCycle

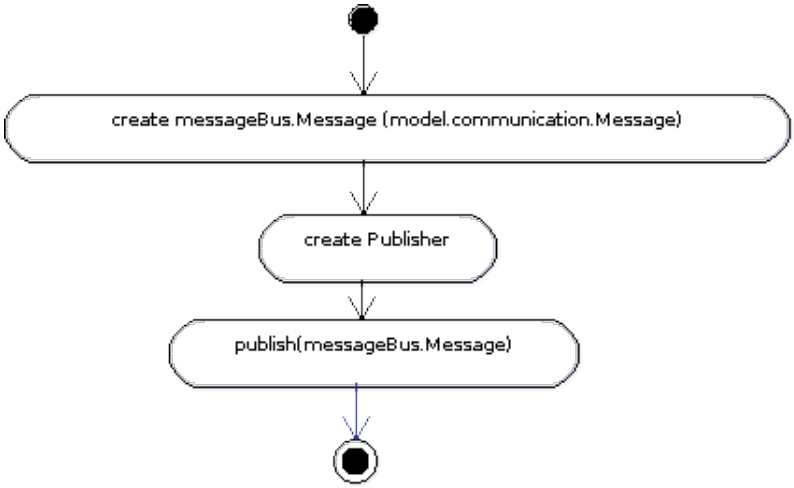

Рисунок 3.7 — Диаграмма действий метода sendMessage компонента ThinkingLifeCycle

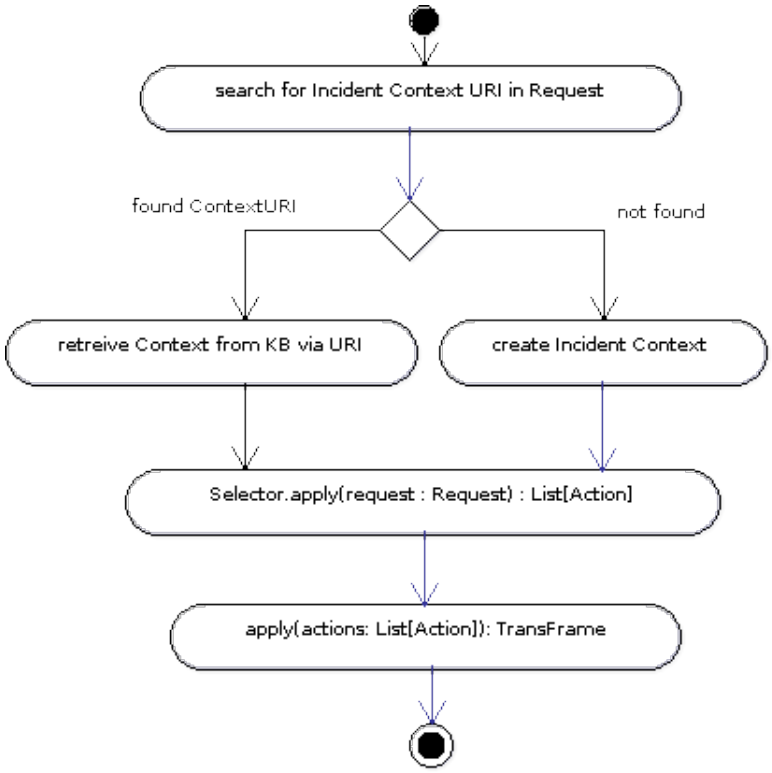

Рисунок 3.8 — Диаграмма действий метода apply компонента ThinkingLifeCycle

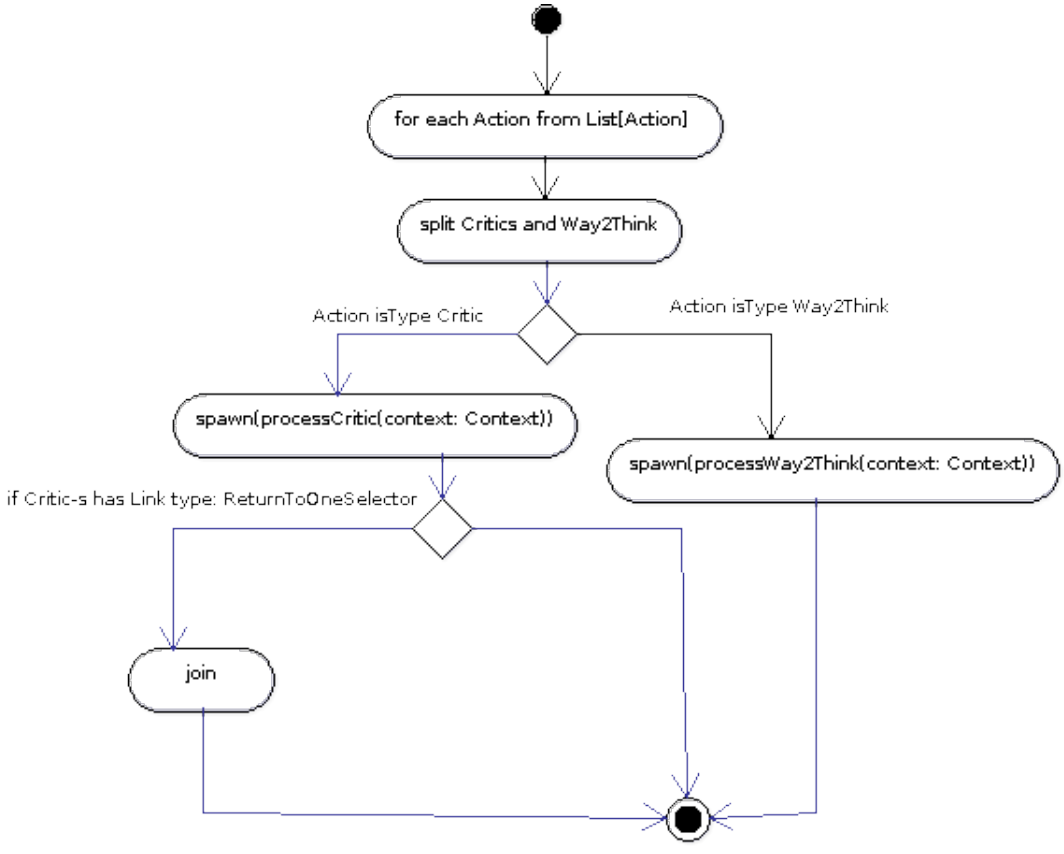

Рисунок 3.9 — Диаграмма действий метода apply компонента ThinkingLifeCycle

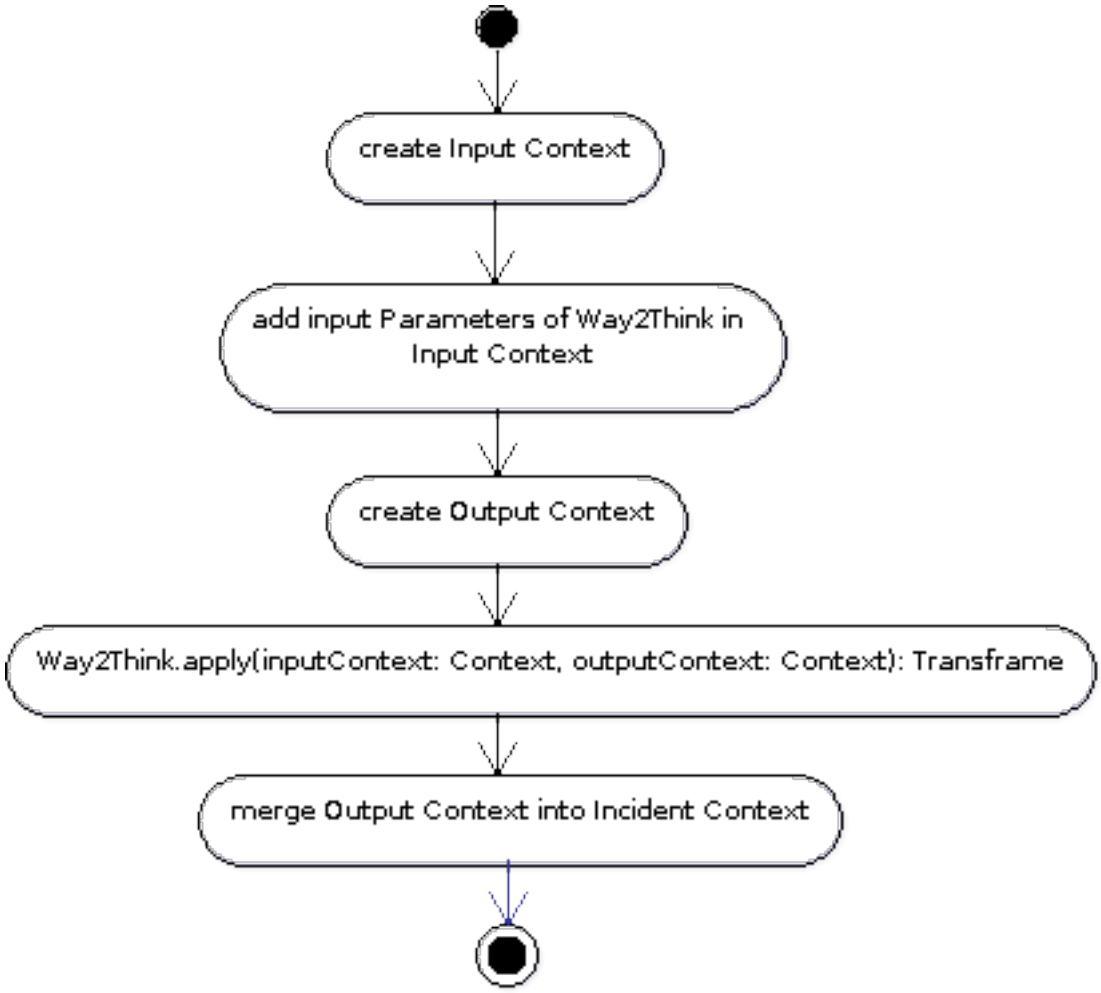

Рисунок 3.10 — Диаграмма действий метода processWay2Think компонента ThinkingLifeCycle

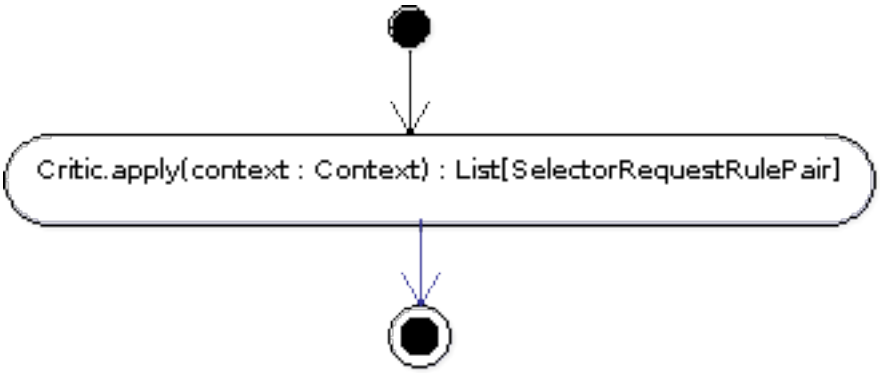

Рисунок 3.11 — Диаграмма действий метода processCritic компонента ThinkingLifeCycle

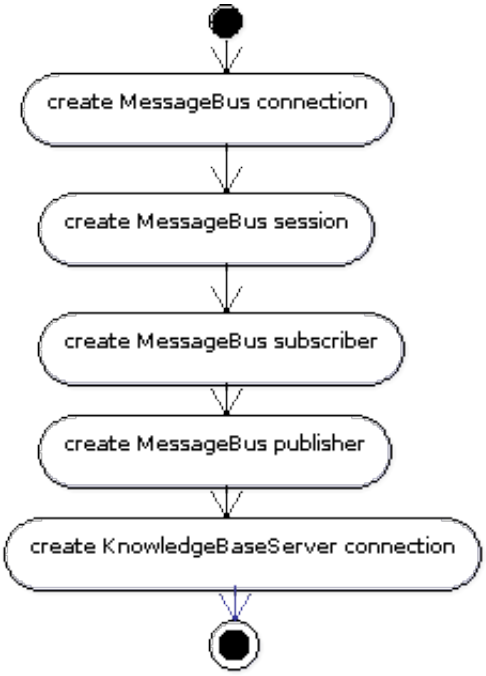

Рисунок 3.12 — Диаграмма действий метода init компонента ThinkingLifeCycle

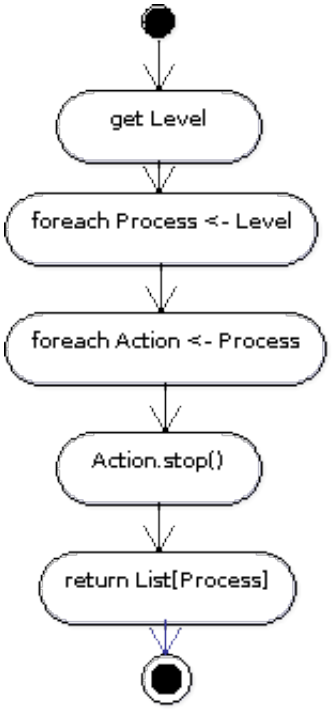

Рисунок 3.13 — Диаграмма действий метода stop компонента ThinkingLifeCycle

Описание работы компонента. Чтобы понять работу компонента далее рассмотрим описание основного потока работы с примерами.

Запуск и остановка. Когда приложение запускается, оно инициализирует ThinkingLifeCycle, 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 [89]. Далее рассмотрим подробнее взаимодействие с остальными компонентами системы.

Интеграция компонента с остальной системой

1. Критик возвращает список Селекторов (SelectorRequestRule)
    (a) ThinkingLifeCycle запускает обработку компонента Selector
    (b) Selector возвращает список Action (см. приложение Б) из базы знаний
    (c) ThinkingLifecycle параллельно запускает возвращенные Action
        i. Если Action это Critic
        ii. ThinkingLifeCycle 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 ThinkingLifeCycle **ждет** результаты и отправляет список SelectorRequestRule, которые были возвращены в качестве результата работы Critic, новому Selector. Иными словами Critic может вернуть новый Selector. В данном случае нам нужно провести операцию Join для всех потоков [90]. В иных же случаях все Action запускаются в параллельных потоках.

i. Если Action это WayToThink
ii. ThinkingLifeCycle создает InputContext (входной контекст приложения) и копирует туда все данные из Context (контекста) возвращенный Selector
iii. TLC (см. таблицу **??**) запускает WayToThink
iv. TLC сохраняет параметры в OutputContext
v. TLC сохраняет итоговый результат работы и возвращает его

В данной главе было приведено описание основного цикла приложения с примерами работы. В следующих главах идет подробное описание работы каждого компонента на более низком уровне. В конце раздела приведено развернутое описание стандартного цикла приложения, а также диаграмма расположения компонентов в разрезе уровней мышления.

3.1.4 Компонент CoreService.Selector

Selector (Селектор) 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.

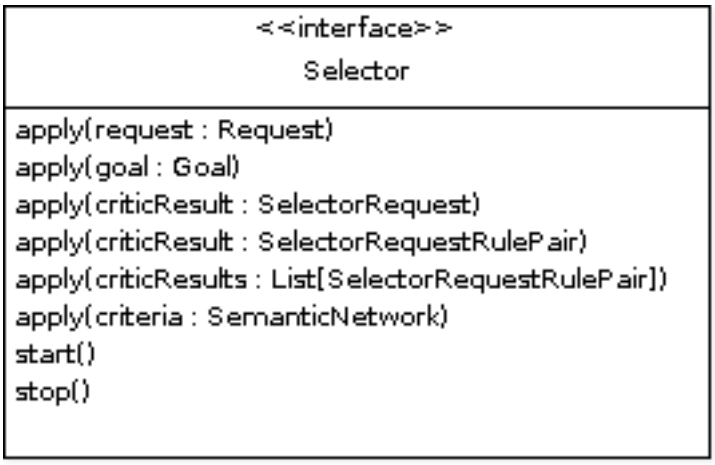

Рисунок 3.14 — Интерфейс компонента Selector

На рисунке 3.14 показан интерфейс компонента. В Таблице 3.6 приведено описание методов компонента, показанных на интерфейсе.

Таблица 3.6 — Описание методов класса (компонента) Selector

Метод	Описание
apply(request : Request) : Action	Данный метод на основе запроса пользователя получает из Базы знаний необходимые Critic 3.1.5. На рисунке 3.15 представлена диаграмма действий для этого метода.

Таблица 3.6 – продолжение

Метод	Описание
apply(goal: Goal) : Action	Данный метод на основе цели системы получает из Базы знаний необходимые Critic 3.1.5. На рисунке 3.16 представлена диаграмма действий для 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 для этого метода.

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

Действия при классификации входящего запроса

1. TLC 3.1.3 запускает входящие Critic 3.1.5 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 Reflective уровне мышления сработал Critic, который посчитал, что данное решение некорректно или же пользователь признал его таким

На рисунке 3.18 представлена диаграмма действий классификации инцидента. TLC 3.1.3 получает цель классифицировать инцидент, затем Selector по этой цели возвращает необходимые Critic. Затем TLC запускает обработку Critic в разных потоках (параллельно). В данном случае рассматривается три Critic: DirectInstruction — прямые инструкции, данный Critic возвращает WayToThink Simulate 3.1.6, 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 пытается преобразовать запрос к DirectInstruction запросу (прямым инструкциям);

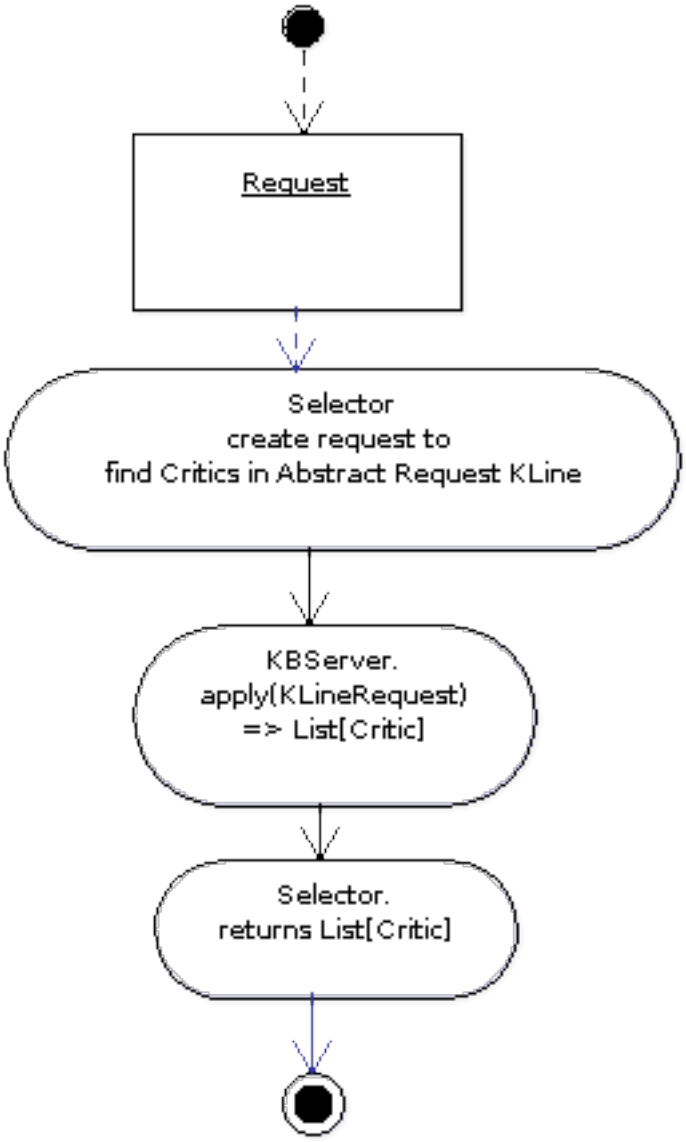

Рисунок 3.15 — Диаграмма действий метода Selector.apply(request : Request) компонента Selector

ProblemWithoutDesiredState - проблема без ожидаемого результата, данный Critic возвращает Simulate, Reformulate, InferDesiredState, который пытается преобразовать проблему к ProblemWithDesiredState.

После работы компонента TLC 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 данного ком-

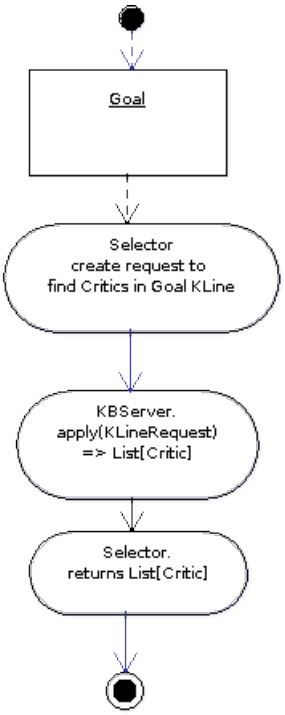

Рисунок 3.16 — Диаграмма действий метода Selector.apply(goal: Goal) компонента Selector

понента является то, что список ресурсов, который он возвращает можно задавать динамически, он также может формироваться во время работы системы.

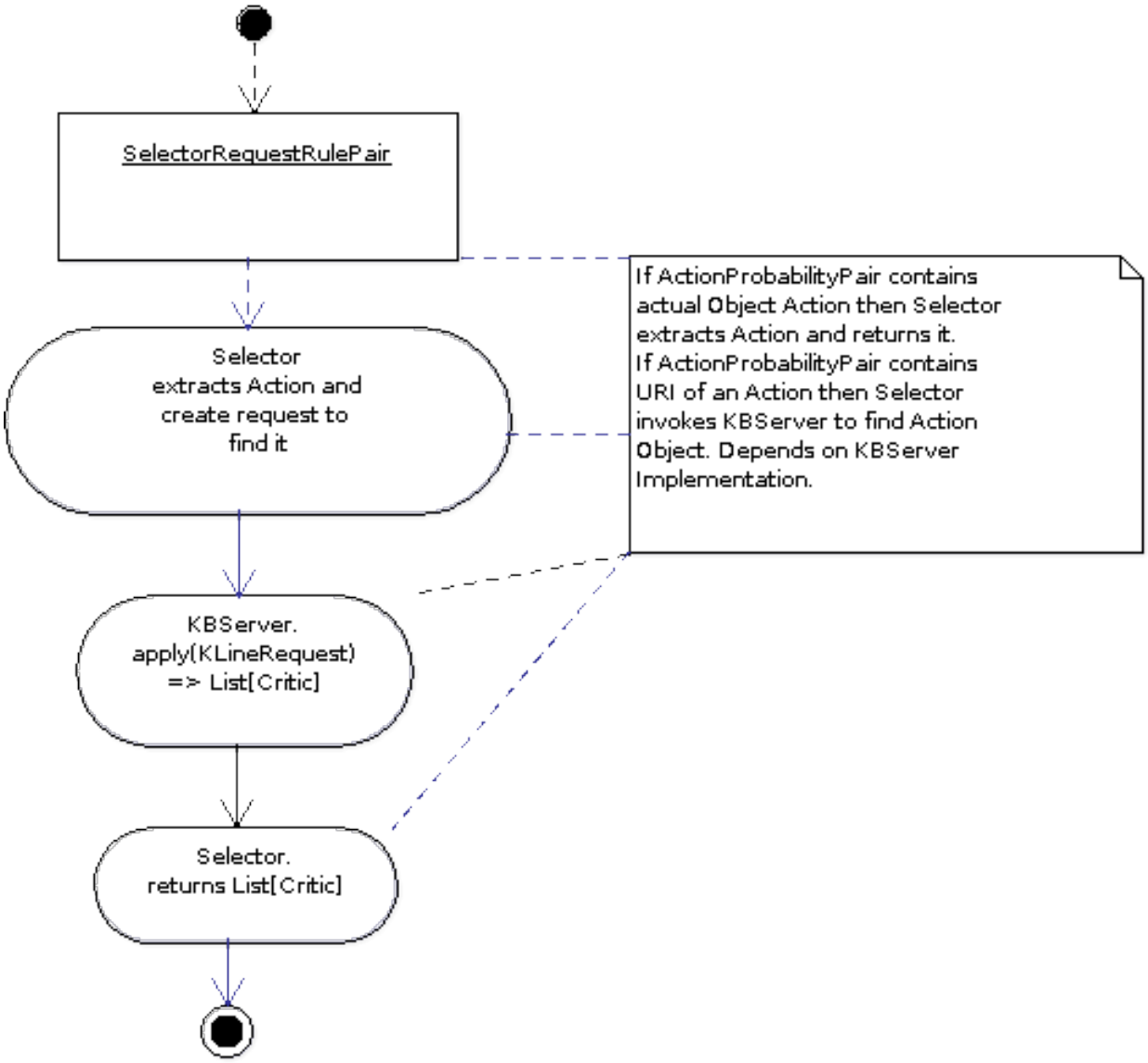

Рисунок 3.17 — Диаграмма действий метода Selector.apply(criticResult : ActionProbabilityRule) компонента Selector

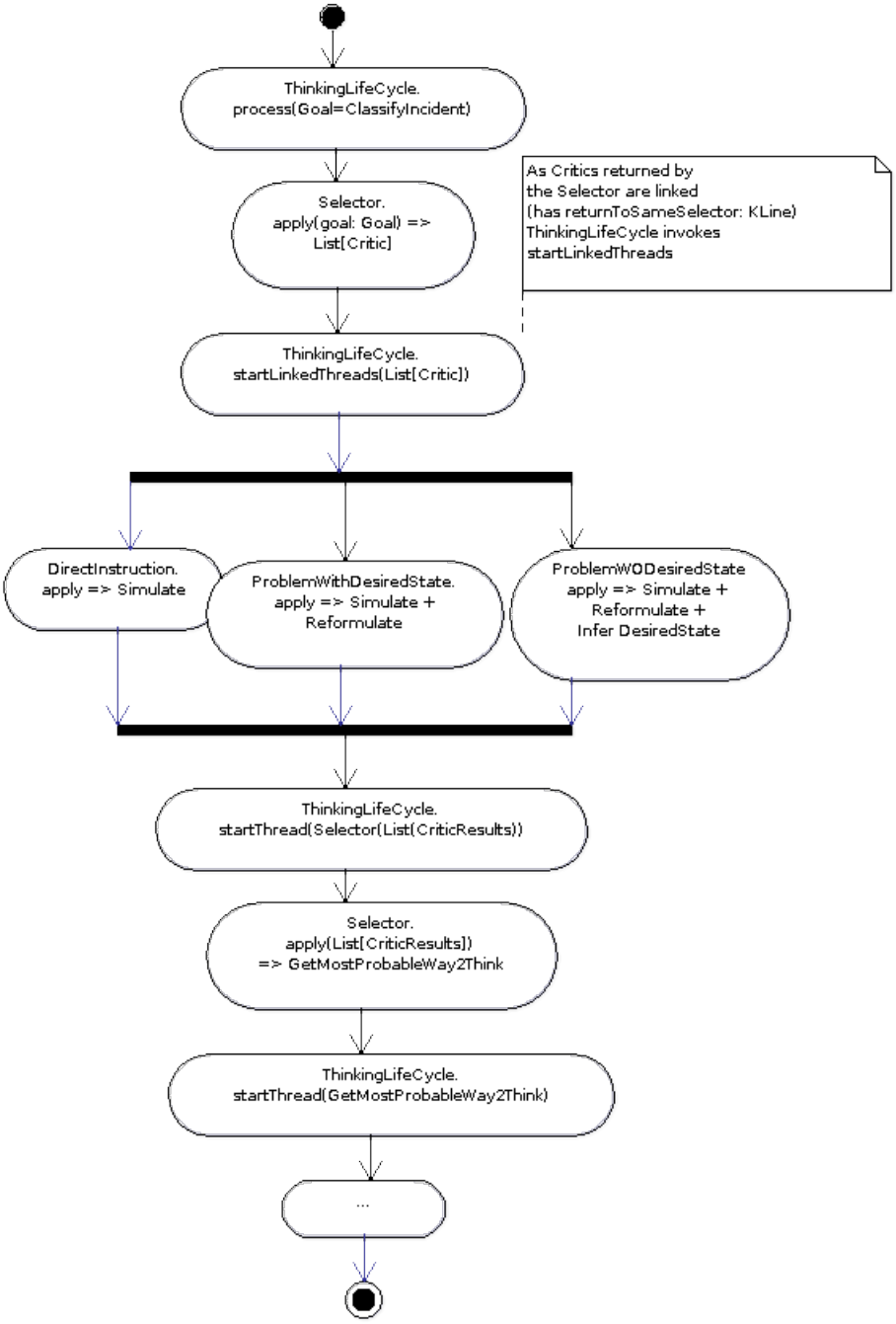

Рисунок 3.18 — Диаграмма действий классификации инцидента

3.1.5 Компонент CoreService.Critics

Critic является основным компонентом для анализа в триплете Критик – Селектор – Образ мышления. Critic используется для классификации входной информации, рефлексии, само-анализа и служит определенным вероятностным переключатетелем. Например, компоненты контроля времени, контроля эмоционального состояния системы — это тоже Critic.

Входной критерий. TLC 3.1.3 запускает Critic согласно Goal (Цель) (см. Приложение B) 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 - 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. То есть 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 B), который запускает необходимый WayToThink.
Control	контролирующий Critic, который ждет определенного события (срабатывает на определенное событие). Например, заканчивается отведенное на решение время.

Таблица 3.7 – продолжение

Critic	Описание
Analyser	анализатор, обрабатывает и выявляет тип инцидента. Например, прямые инструкции, проблема с желаемым состоянием, выбор наиболее вероятного действия.

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.
include():List[Critic]	Данный метод возвращает список 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 3.1.4 с определенной веротяностью, после чего TLC 3.1.3 их активирует.

Как уже писалось ранее Critic действуют на разных уровнях мышления. Далее приводится 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 у нас будет запрос вида: install antivirus. DesiredState найдет здесь действие — install, но не найдет желаемого состояния, то есть его вероятность будет 60%. DirectInstruction будет искать действие и объект, которые присутствуют в запросе, то есть его вероятность будет 100%, его TLC 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 данной главе 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.

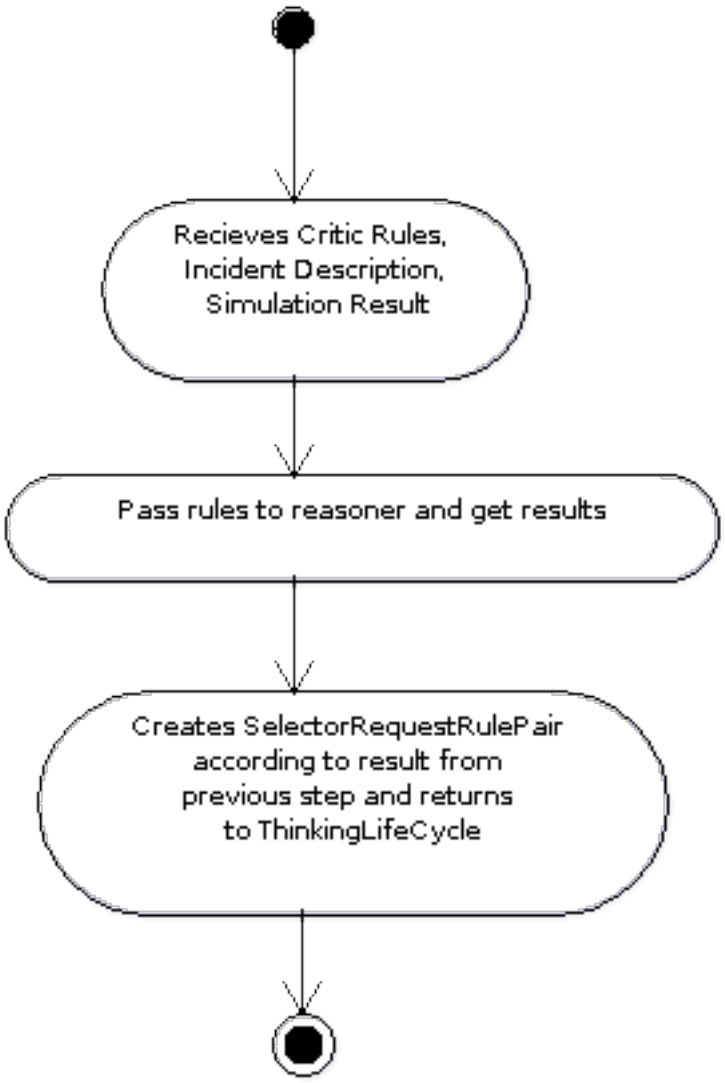

Рисунок 3.19 — Диаграмма действий компонента Critic

3.1.6 Компонент CoreService.WayToThink

WayToThink является основным операционным компонентом триплета Критик – Селектор – Образ мышления. 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 из компонента ThinkingLufeCycle 3.1.3. Входными данными является InputContext, который содержит параметры WayToThink.

Выходной критерий. WayToThink завершил работу. На выходе возвращается измененные в ходе работы данные. На Рисунке 3.20 представлен интерфейс компонента.

В общем виде компонент описывает последовательность действий. В системе

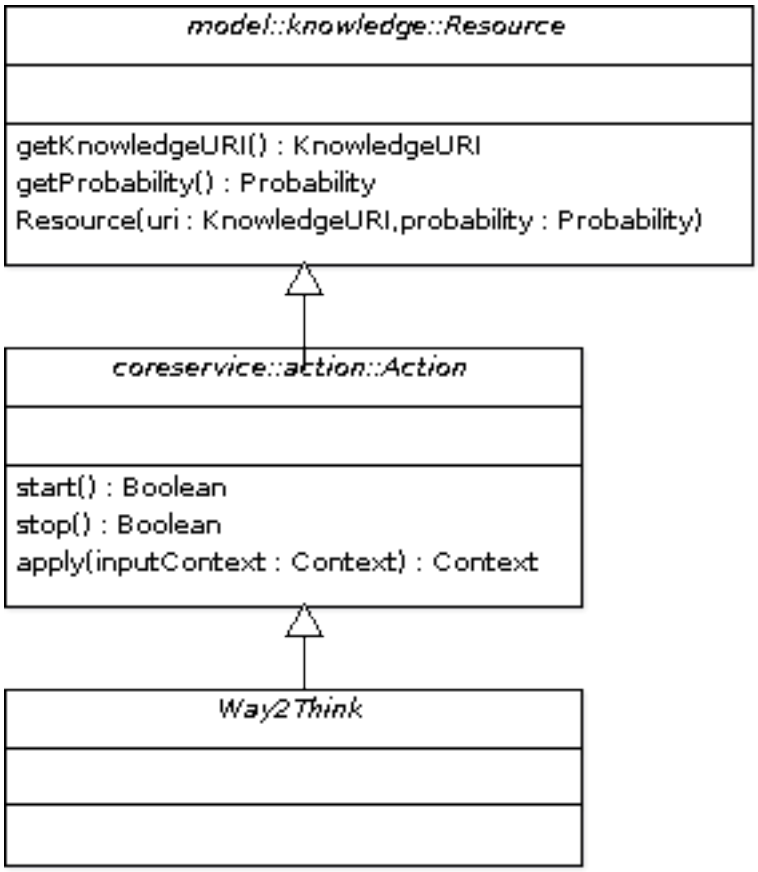

Рисунок 3.20 — Интерфейс компонента WayToThink

используется два больших класса WaytToThink простой и составной (сложный).

Простые WayToThink являются встроенными в систему, остальные являются комбинацией компонентов: Critic 3.1.5, Selector 3.1.4, WayToThink 3.1.6. В Таблице 3.9 приведено описание встроенных в систему WayToThink. 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.
Установить общий статус системы	Данный WayToThink устанавливает состояние системы в глобальном контексте.
Установить цель системы	Данный WayToThink устанавливает цель запроса в текущем контексте 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 пользотеля.
Смоделировать (Simulate)	Данный WayToThink ищет связь между текущим контекстом и проблемами уже сохраненными в базе.
	Продолжение следует

Таблица 3.9 – продолжение

WayToThink	Описание
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

Метод	Описание
start()	Запустить обработку информации.
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 решение проблемы (HowTo см. приложение Г), 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 данной главе 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 взять скрипт, который был написан на языке интерпретации Bash и ввести его в систе-

му, можно разбить каждый шаг и вызов на отдельную часть и сделать сложный WayToThink с ветвлениями и циклами.

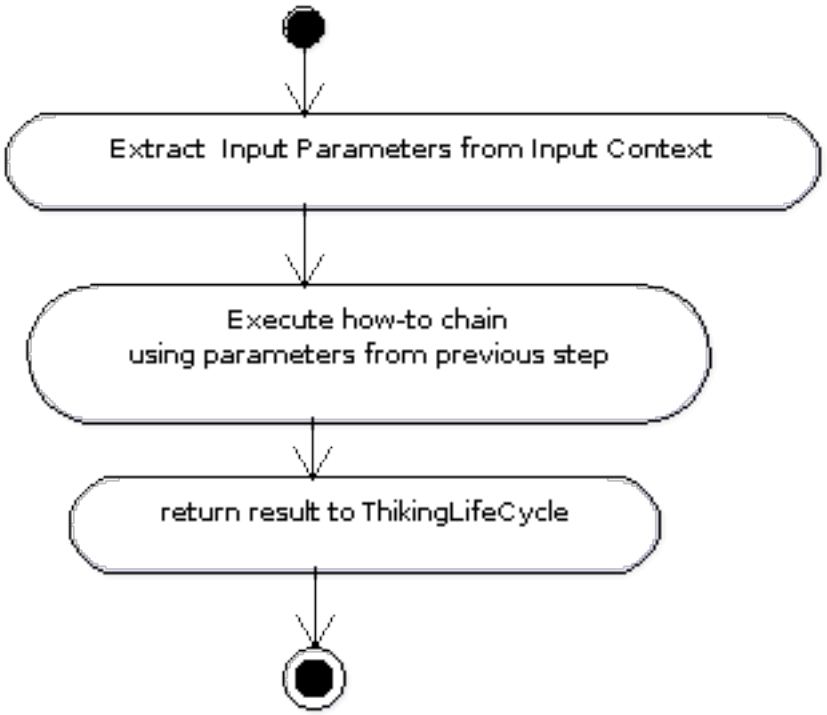

Рисунок 3.21 — Работа компонента WayToThink в режиме описания решения проблемы (HowTo)

3.1.7 Компонент CoreService.PrelimenaryAnnotator

Данный компонент проводит предварительную подготовку текста: грамматическую и орфографическую коррекцию 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.

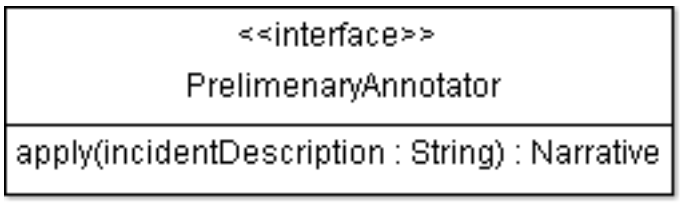

Рисунок 3.22 — Интерфейс компонента PrelimenaryAnnotator

Таблица 3.11 — Описание методов компонента PrelimenaryAnnotator

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 [91], а также Google API. В части разбиения текста на слова используется алгоритм из открытого комплекса Link Grammar [**?**]. В целом компонент содержит в себе также составную системы разбора, что отличает его от прямого использования алгоритма. Он манипулирует результатом из нескольких подсистем, для увеличения степени точности.

Одной из особенностью компонента является использования 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 анно-

тированный текст, что позволяет на 20% (согласно экспериментальным данным) увеличить точность обработки.

3.1.8 Компонент CoreService.KnowledgeBaseAnnotator

Данный компонент устанавливает связи между терминами во входной фразе и базой знаний. Данный компонент также является WayToThink 3.1.6.

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 аналогия в база знаний program-computer software.

3.1.9 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, BusiessObject - 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.

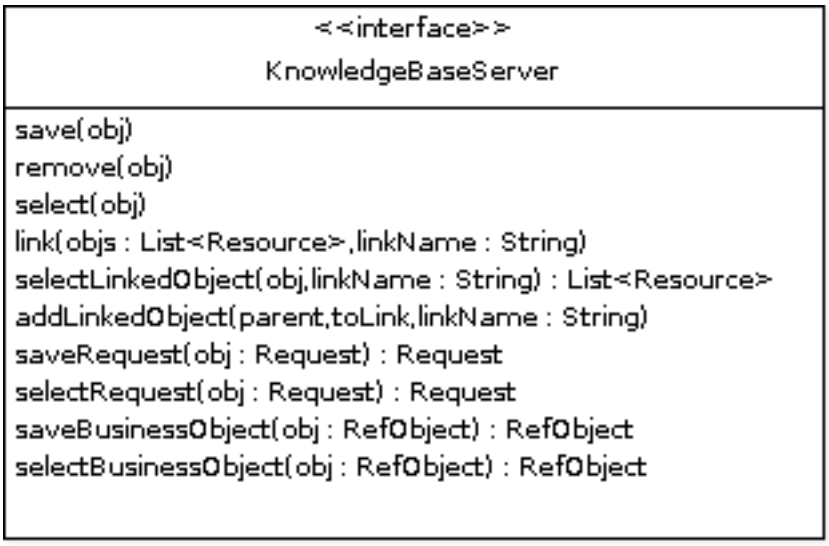

Рисунок 3.23 — Интерфейс компонента KnowledgeBaseServer

Таблица 3.12 — Описание методов компонента DataService

Метод	Описание
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.
	Продолжение следует

Таблица 3.12 – продолжение

Метод	Описание
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 полуить объект из Базы Знаний.

3.1.10 Компонент Reasoner

Данный компонент осуществляет логические вычисления для системы, например, для обработки правил в компоненте Critic 3.1.5. На рисунке 3.24 представлен интерфейс компонента. В Таблице 3.13 приведено описание методов компонента.

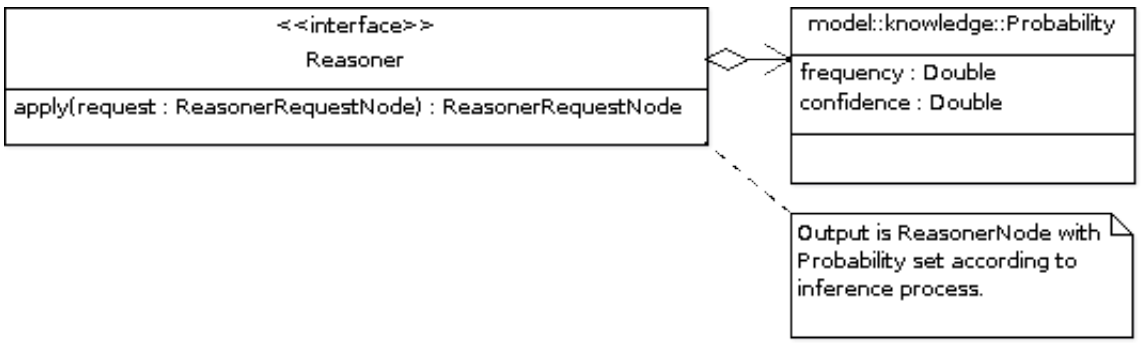

Рисунок 3.24 — Интерфейс компонента Reasoner

Таблица 3.13 — Описание методов компонента Reasoner

Метод	Описание
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 [92] и NARS [76]. Основным применением данного компонента является 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.

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, родившейся для структурирования информации Web [70] 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.

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 работу 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 3.1.5.
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.
	Продолжение следует

Таблица 3.14 – продолжение

Класс	Описание
Frame	Коллекция объектов Knowledge, с возможностью проставление специального тега для семантической группировки.
TransFrame	Коллекция Frame, содержащая два состояния одного фрейма: до и после.
Goal	Цель. Приложение В.
Tag	Тоже что и цель, но использующиеся для меток.
Preliminary annotation	SemanticNetwork входного запроса.
KnowledgeBase annotation	SemanticNetwork с сопостовлением концепциям Базы Знаний.
Domain model	SemanticNetwork доменной модели.
Situation model	SemanticNetwork, часть DomainModel, созданной для обработки текущего запроса. Приложение Б.
Incident	SemanticNetwork входного запроса к систему.
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.

3.2.1 Описание запросов в рамках TU Knowledge

В рамках модели данных TU Knowledge проблема имеет следующее описание:

- Область (Микронема)
- Дата обращения
- Автор
- Приоритет
- Категория

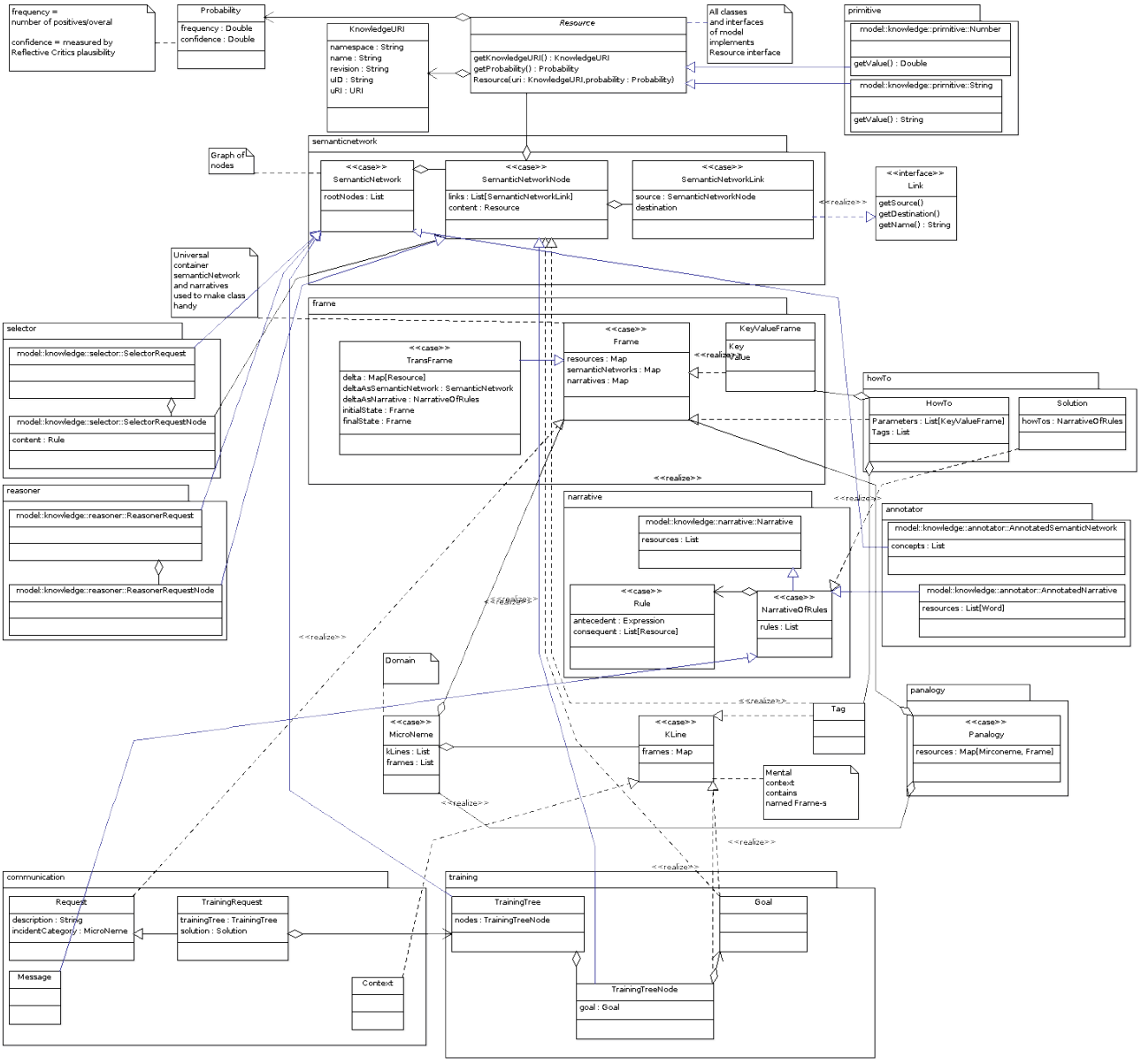

Рисунок 3.25 — Схема данных TU Knowledge в формате UML

– Теги
– Описание

Запрос на обучение включает следующие части:

– Область (Микронема)
– Дерево обучения
– Ограничения (например, время на решение)

Микронема

Данная концепция 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.

3.2.2 Дерево обучения

Дерево обучения базируется на структуре Цель-Подцель. Получение целей происходит из:

– Из подцелей

– Вручную добавленные при запросе

Во время процедуры обучения возможно, что на одно 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 обучение провалиться, то будет выбран менее вероятный.

Пример

```
 1. SubGoal = Resolve incident
   2. SubGoal = ParseIncidentDescription, Way2Think = ProcessText:
       KnowingHow, SemanticNetWorkWithKLines =
{
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)

advmod(received-3, However-1)
nsubj(received-3, user-2)
ccomp(received-8, received-3)
amod(version-5, wrong-4)
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)
advmod(of—12, instead—11)
prep(Tehcnical—10, of—12)
nn(Economical—14, Business—13)
pobj(of—12, Economical—14)
}
   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
}
     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)
}
     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 от 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.
2. GoalManger устанавливает цель системы HelpUser
3. Активируется набор Critic, привязанный к данной цели
4. PreliminaryAnnorator разбирает фразу
5. KnowledgeBaseAnnotator создает семантическую сеть и ссылки на нее
6. Critic на Рефлексивном уровне запускает WayToThink ProblemSolving с целью: ResolveIncident
7. Critic на Рефлексивном уровне выбирает WayToThink KnowingHow
   (a) Запускаются параллельно все Critic, которые привязаны к IncidentClassification Critic, который привязан к ResolveIncident цели, в данном случае это DirectInstruction, ProblemWithDesiredState, ProblemWithoutDesiredState 3.1.3
   (b) Selector выбирает наиболее вероятный результат работы среди всех результатов компонентов. В данном случае будет результат работы Problem Description with desired state.
   (c) KnowingHow сохраняет варианты выбора Selector.
   (d) Simulation WayToThink с параметрами Создать модель текущий ситуации создает модель CurrentSituation. User, Software
   (e) Reformulation WayToThink, используя результаты предыдущего шага синтезирует артефакты, которых не хватает, чтобы получить CurrentState и DesiredState. DesiredState не указан явно. WayToThink запускает Critic размышления, чтобы найти корень проблемы. Critic размышления находит CurrentState- Wordfinder Tehcnical, DesiredState-Wordfinder Business Economical

(f) Рефлексивные Critic оценивают состояние системы - на каком шаге она находится, и если цель не достигнута, то запускают другой WayToThink, который был возвращен, например, DirectInstruction.

(g) Critic генерации решения запускает KnowingHow Г WayToThink, ExtensiveSearch.

(h) Selector выбирает наиболее вероятный путь мышления. В данном случае ExtensiveSearch, который будет находить решения, позволяющие привести систему в необходимое состояние (DesiredState). Если он не сможет, то он инициирует коммуникацию с пользователем.

8. Рефлексивный Critic проверяет состояние системы. Если Цель достигнута, то пользователю посылается ответ.
9. Само Сознательные Critic активируется на данном шаге и сохраняют информацию о затратах на решение.

3.3.1 UML диаграмма действий приложения

На рисунке 3.26 представлена UML диаграмма действий системы, согласно алгоритму, описанному выше и с привязкой к уровням мышления.

3.4 Выводы

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 были 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, а 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.

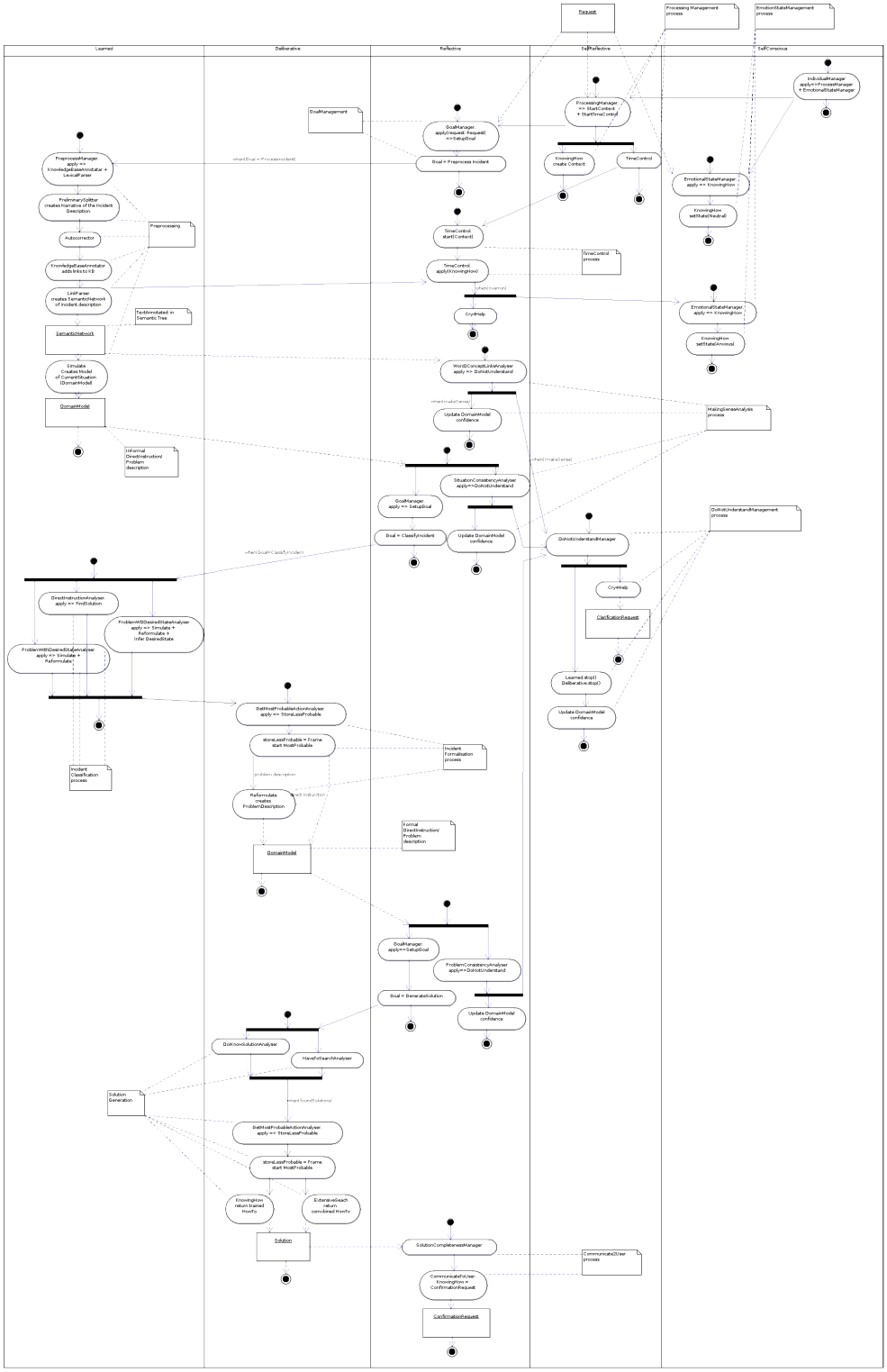

Рисунок 3.26 — Диаграмма действий LifecycleActivity

Глава 4. Экспериментальные исследования эффективности работы модели TU

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, базовая концепция для всех объектов; Action — действие. Базовая концепция 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.	Обучающий запрос. Создает связь между концепцией Tense и Concept.
Please install Firefox.	Запрос. Пользователь просит установить Firefox. Результатом должен быть найдено решение по установки Firefox.
Browser is an object.	Обучающий запрос. Создает связь между концепцией Browser и object.
Firefox is a browser.	Обучающий запрос. Создает связь между концепцией Firefox и browser.

Таблица 4.1 – продолжение

Входное предложение	Описание
Install is an action.	Обучающий запрос. Создает связь между концепцией Install и action.
User miss Internet Explorer 8.	Запрос. Проблема с желаемым состоянием (DesiredState).
User needs document portal update.	Запрос. Проблема с желаемым состоянием.
Add new alias Host name on host that alias is wanted to: hrportal.lalala.biz IP adress on host that alias is wanted to: 322.223.333.22 Wanted Alias: webadviser.lalala.net	Запрос. Сложная проблема.
Outlook Web Access (CCC) - 403 - Forbidden: Access is denied	Запрос. 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.	Запрос. Сложная проблема.

Полный список информация об экспериментальных данных представлен в приложении Д

4.2 Оценка эффективности

Для верификации экспертной системы поддержки принятия решений TU была выбрана область поддержки информационной инфраструктуры предприятия, которая была в рамках работы исследована и смоделирована в Главе 1. Для доказания 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 использовался 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%. Успешным считался инцидент, который был успешно сопоставлен концепциям в базе знаний.

Результатом успешной обработки инцидента подразумевалось 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 приведены результаты сравнения 4.2. Тесты были выполнены на машине 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.	15000	385
Please install Firefox.	9000	859
Browser is an object.	20000	400
Firefox is a browser.	5000	659
Install is an action.	8000	486
User miss Internet Explorer 8.	10000	10589
User needs document portal update.	15000	16543
Add new alias Host name on host that alias is wanted to: hrportal.lalala.biz IP adress on host that alias is wanted to: 322.223.333.22 Wanted Alias: webadviser.lalala.net	10000	18432
Outlook Web Access (CCC) - 403 - Forbidden: Access is denied	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.	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 тяжело 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**
Категория	Описание
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%

4.4 Выводы

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 сравнение со специалистом технической поддержки. Были при-

ведены 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, принятия решений в области обслуживания информационной структуры предприятия на основе модели мышления;
2. Представлены новая модель данных для модели мышления и оригинальный способ ее хранения, более эффективный по сравнению с классическими базами данных, использующими реляционный подход;
3. Выполнено оригинальное исследование моделей мышления в области обслуживания информационной структуры предприятия;
4. На основе модели, разработанной в диссертации, созданы архитектура системы и ее прототип;
5. Разработаны специальные алгоритмы для анализа запросов пользователей и принятия решений;
6. Система, разработанная в рамках данной работы, включает в себя инновационные методы и алгоритмы поддержки принятия решений, использует модель мышления, базирующуюся на модели мышления Мински;
7. Представлена наглядная 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 поддержка принятия решений, например, при постановке медицинского диагноза, чтобы отбросить ложные диагнозы.

Кроме того, в систему можно загрузить данные о взаимосвязи органов человека и болезней. Далее, к каждому заболеванию добавить симптом и способ лечения, после этого можно делать запрос с симптомами, и система выдаст список вероятных заболеваний со способами их лечения.

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

Работа велась с использованием открытых технологий, без использования проприетарного программного обеспечения. Работа была презентована автору книги Object-Oriented Software Construction [93] Бертрану Мейеру в рамках серии лекций, проведенных при содействии Университета Иннополис в Казани в 2015 году в рамках AKSES-2015 http://university.innopolis.ru/en/research/selab/events/akses и была им отмечена. Работа выполнялась при помощи компании ОАО «АйСиЭл КПО-ВС (г. Казань)», в рамках работы 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.

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 [64]. Класс объектов, которые модифицируют данные.

Critic — Основан на определении Марвина Мински [64]. Класс объектов, которые выступают триггерами при наступление определенного события.

ThinkingLifeCycle — Основан на определении Марвина Мински [64]. 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 // *Информационные технологии*. — 2006. — Т. 1311. — С. 22 –23.

2. *Коптелов А., Вишняков О.* Анализ эффективности аутсорсинга // *IT news*. — 2007. — Т. 7(80). — С. 12 –15.

3. *Коптелов А., Уштей С.* Аутсорсинг центра технической поддержки пользователей // *IT news*. — 2007. — Т. 2(75). — С. 5 –10.

4. *Коптелов А., Елманова Н.* Аутсорсинг разработки программного обеспечения // *Информационные технологии*. — 2006. — Т. 16. — С. 5 –10.

5. *Коптелов А., Караваев И.* ИТ-служба передается на аутсорсинг // *ИКС*. — 2007. — Т. 8. — С. 22 –24.

6. *Statista*. Global market size of outsourced services from 2000 to 2014 (in billion U.S. dollars). — 2015. http://www.statista.com/statistics/189788/global-outsourcing-market-size/.

7. *Hartshorne R.* Outsourcing of information and knowledge services: A supplier's view // *Business Information Review*. — 2015. — V. 32. — P. 103 –109.

8. *Зацепа С.* Рентабельность малого бизнеса и ИТ-аутсорсинг // *Управление компанией*. — 2006. — Т. 7. — С. 90 –98.

9. *Коптелов А.* Автоматизация центра поддержки пользователей // *Мобильные телекоммуникации*. — 2006. — Т. 9. — С. 103 –109.

10. *Tutubalina E.* Target-based topic model for problem phrase extraction (Conference Paper) // *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. — 2015. — V. 9022. — P. 271 –277.

11. *Bello-Orgaz G.A Jung J.J., Camacho D.A.* Social big data: Recent achievements and new challenges (Article) // *Information Fusion*. — 2015. — V. 28. — P. 45 –59.

12. *Baddoura R Venture G.* This Robot is Sociable: Close-up on the Gestures and Measured Motion of a Human Responding to a Proactive Robot (Article) // *International Journal of Social Robotics*. — 2015. — V. 7. — P. 489 –496.

13. Decision making logic for flexible assembly lines reconfiguration (Article) / G. Michalos, P. Sipsas, S. Makris, G. Chryssolouris // *Robotics and Computer-Integrated Manufacturing*. — 2016. — V. 37. — P. 233 –250.

14. Problem-oriented patient record summary: An early report on a Watson application (Conference Paper) / M. Devarakonda, D. Zhang, C.H. Tsou, M. Bornea // *2014 IEEE 16th International Conference on e-Health Networking, Applications and Services*. — 2014. — V. 1. — P. 281 –286.

15. *Wagner J.* After The Deadline. — 2012. http://www.afterthedeadline.com/.

16. *Ivanov V., Tutubalina E.* Clause-based approach to extracting problem phrases from user reviews of products // *Communications in Computer and Information Science*. — 2014. — V. 436. — P. 229 –236.

17. *Садовничей В., Савина Г.* Суперкомпьютерные технологии в науке, образовании и промышленности. — Москва: Издательство Московского университета, 2009. — 232 с.

18. *Хикс Дж.* Теория экономической истории // *Вопросы экономики*. — 2003. — Т. 8. — С. 184–188.

19. *Yang C., Chen C., Peng C.* Developing and evaluating an IT specification extraction system // *Electronic Library*. — 2006. — V. 24. — P. 832–846.

20. Intelligent telecommunication system using semantic-based information retrieval / E. Ajith Jubilson, P. Dhanavanthini, V. Victer Paul, P.and Pravinpathi et al. // *Advances in Soft Computing*. — 2015. — V. 381. — P. 137–143.

21. *Ингланд Р.* Введение в реальный ITSM / Под ред. О. Скрынник. — Москва: Лайвбук, 2010. — 131 с.

22. *Ингланд Р.* Овладевая ITIL / Под ред. О. Скрынник. — Москва: Лайвбук, 2011. — 200 с.

23. *Л. Будкова, Журавлёв Р.* Методическое руководство для подготовки к профессиональным экзаменам ISO 20000 Foundation и ISO 20000 Foundation Bridge / Под ред. О. Скрынник. — Москва: Лайвбук, 2011. — 124 с.

24. Super Job. Уровень зарплат IT специалистов. — 2014. http://www.it-analytics.ru/analytics/trends/66314.html0.11-2011.

25. Налоговый кодекс Российской Федерации. Части 1 и 2. — Москва: Эксмо, 2015. — 1344 с.

26. *Web Time*. Time Web. Стоимость аренды серверов. — 2015. http://timeweb.com/ru/services/dedicated-server/.

27. *Тощев А. С.* К новой концепции автоматизации программного обеспечения // *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, № 4. — С. 279 – 282.

28. *Toshchev A., Talanov M., Krehov A.* Thinking-Understanding approach in IT maintenance domain automation // *Global Journal on Technology: 3rd World Conference on Information Technology (WCIT-2012)*. — 2013. — V. 3. — P. 879 –894.

29. *Toshchev A., Talanov M.* Thinking model and machine understanding of English primitive texts and it's application in Infrastructure as Service domain // *Proceedings of Conference "Artificial Intelligence and Natural Language (AINL-2013)"*. — 2013. — С. 14 –19. — http://ainlconf.ru/material201303.

30. *Toshchev A., Talanov M.* Архитектура и реализация интеллектуального агента для автоматической обработки входящих заявок с помощью искусственного интеллекта и семантических сетей // *Ученые записки Института социально-гуманитарных знаний*. — 2014. — Т. 2. — С. 288 –292.

31. *Toshchev A., Talanov M.* Computational emotional thinking and virtual neurotransmitters // *International Journal of Synthetic Emotions (IJSE)*. — 2014. — V. 5. — P. 30 –35.

32. *Toshchev A., Talanov M.* Appraisal, coping and high level emotions aspects of computational emotional thinking // *International Journal of Synthetic Emotions (IJSE)*. — 2015. — V. 06. — P. 65 –72.

33. *Toshchev A.* Thinking model and machine understanding in automated user request processing // *CEUR Workshop Proceedings*. — 2014. — V. 1297. — P. 224 –226.

34. *Toshchev A., Talanov M.* Thinking lifecycle as an implementation of machine understanding in software maintenance automation domain // *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. — V. 38. — P. 301 –310.

35. *Тощев А.С.* Возможности автоматизации разрешения инцидентов для области удаленной поддержки информационной инфраструктуры предприятия // *Экономика и менеджмент систем управления*. — 2015. — Т. 4. — С. 293 –295.

36. *Тощев А.С., Таланов М.О.* Вычислительная модель эмоций в интеллектуальных информационных системах // *Электронные библиотеки*. — 2015. — Т. 18. — С. 225 –235.

37. *Тощев А.С.* Применение моделей мышления в интеллектуальных вопросно-ответных системах // *Электронные библиотеки*. — 2015. — Т. 18. — С. 216 – 224.

38. *Sokolov A., Serdobincev K.* HP OpenView System Administration Handbook: Network Node Manager, Customer Views, Service Information Portal, HP OpenView Operations / Ed. by H. Shootze. — Astrahan: Astrahan, 2004. — 688 pp.

39. *Tsvetkov A., Ponomareva O., Yurina M.* Automation of incidents' recording process in the network of the mobile radio communication of standard GSM-900/1800 (Conference Paper) // *24th International Crimean Conference Microwave and Telecommunication Technology, CriMiCo 2014; Sevastopol, Crimea; 7 September 2014 through 13 September 2014; Category numberCFP14788-CDR; Code 109221*. — 2014. — V. 1. — P. 401 –402.

40. *Padhy S. Kreutz D. Casimiro A. Pasin M.* Trustworthy and resilient monitoring system for cloud infrastructures (Conference Paper) // *Proceedings of the Workshop*

on Posters and Demos Track, PDT'11 - 12th International Middleware Conference, Middleware'11. — 2011. — V. 1. — P. 87 –95.

41. *F. Gentschen, Hegering H., Schiffers M.* IT service management across organizational boundaries (Book Chapter) // *Managing Development and Application of Digital Technologies: Research Insights in the Munich Center for Digital Technology and Management (CDTM)*. — 2006. — V. 1. — 341 pp.

42. *Catania N., Kumar P., Murray B. et al.* Web Services Management Framework. — 2003. http://devresource.hp.com/drc/specifications/wsmf/WSMF-Overview.jsp.

43. *Catania N., Kumar P., Murray B. et al.* Web Services Events (WS-Events). — 2003. http://devresource.hp.com/drc/specifications/wsmf/WS-Events.pdf.

44. Migration of the CERN IT data centre support system to servicenow (Conference Paper) / R.A. Alonso, G. Arneodo, O. Barring et al. // *Journal of Physics: Conference Series*. — 2013. — V. 513. — P. 1–80.

45. *Gross K. Hayashi S. Teige S. Quick R.* Open Science Grid (OSG) ticket synchronization: Keeping your home field advantage in a distributed environment (Conference Paper) // *Journal of Physics: Conference Series*. — 2012. — V. 396. — P. 1–80.

46. Relation extraction and scoring in DeepQA / C. Wang, A. Kalyanpur, J. Fan et al. // *IBM journal of Research and Development*. — 2012. — V. 56. — P. 1–12.

47. Watson: Beyond jeopardy! / D. Ferrucci, A. Levas, S. Bagchi et al. // *Artificial Intelligence*. — 2013. — V. 10.1016. — P. 93–105.

48. *Alterman R.* Understanding and summarization // *Artificial Intelligence Review*. — 1991. — V. 5. — P. 239–254.

49. *Chandrasekar R.* Elementary? Question answering, IBM's Watson, and the Jeopardy! challenge // *Resonance*. — 2014. — V. 19. — P. 222–241.

50. *Rajaraman V.* JohnMcCarthy - Father of artificial intelligence // *Resonance*. — 2014. — V. 19(3). — P. 198–207.

51. *Jurafsky D., Martin J.* Detecting friendly, flirtatious, awkward, and assertive speech in speed-dates // *Computer Speech and Language*. — 2013. — V. 27. — P. 89–115.

52. *Campbell M., Jr. Hoane, A.Joseph Hsu F.-H.* Deep Blue // *Artificial Intelligence*. — 2002. — V. 134. — P. 57–83.

53. *Mahdi A., Tiun S.* Utilizing WordNet and regular expressions for instance-based schema matching // *Research Journal of Applied Sciences, Engineering and Technology*. — 2014. — V. 8. — P. 460–470.

54. *Lamperti G., Zhao X.* Diagnosis of active systems by semantic patterns // *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. — 2014. — V. 8. — P. 1028–1043.

55. Automatic extraction of function-behaviour-state information from patents / G. Fantoni, R. Apreda, F. Dell'Orletta, M. Monge // *Advanced Engineering Informatics*. — 2013. — V. 27. — P. 317–334.

56. *Krasner D., Langmore I.* Flexible processing and classification for eDiscovery // *Frontiers in Artificial Intelligence and Applications*. — 2013. — V. 259. — P. 87–96.

57. *Manshadi M. Gildea-D. Allen J.* Integrating programming by example and natural language programming // *Proceedings of the 27th AAAI Conference on Artificial Intelligence*. — 2013. — P. 661–667.

58. *Foundation Apache Software*. Apache OpenNLP. — 2012. https://opennlp.apache.org/.

59. *Goetzel B.* OpenCog. — 2012. http://wiki.opencog.org/w/RelEx.

60. *Veber P., Willams D.* Introduction to information processing / Ed. by T. Zitello. — Upper Saddle River, New Jersey 07458: Prentis Hall, 2009. — 581 pp.

61. Implementing an online help desk system based on conversational agent / A. Kongthon, C. Sangkeettrakarn, S. Kongyoung, C. Haruechaiyasak // *Proceeding, MEDES '09 Proceedings of the International Conference on Management of Emergent Digital EcoSystems*. — 2009. — V. 1. — P. 450–451.

62. *Гринберг Д.* Надежный алгоритм обработки для грамматики // *Технический отчет Университета Карнеги Мелон CMU-CS-95-125*. — 1995. — Т. 9. — С. 30 –35.

63. *Russel S., Norvig P.* Artificial Intelligence. A Modern approach. — USA: Pearson, 2010. — 1152 pp.

64. *Minsky M.* The Emotion Machine. — NY, USA: Simon & Schuster, 2007. — 400 pp.

65. *Katidiotis A., Tsagkaris K.* Performance evaluation of artificial neural network-based learning schemes for cognitive radio systems // *Computers* E*lectrical Engineering*. — 2010. — V. 36. — P. 518 –535.

66. *Deng H., Runger G., Tuv E.* Bias of importance measures for multi-valued attributes and solutions // *Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN)*. — 2011. — P. 293 –300.

67. *Anaya A., Luque M., Peinado M.* A visual recommender tool in a collaborative learning experience // *Expert Systems with Applications*. — 2016. — V. 45. — P. 248–259.

68. *Sinha Y., Jain P., Kasliwal N.* Comparative study of preprocessing and classification methods in character recognition of natural scene images // *Advances in Intelligent Systems and Computing*. — 2016. — V. 45. — P. 119–129.

69. *Trujillo-Rasua R., Yero I.* K-Metric antidimension: A privacy measure for social graphs // *Information Sciences*. — 2015. — V. 328. — P. 403–417.

70. *Lesly Л.* Owl: Representing Information Using the Web Ontology Language. — 47403, Blumington, Liberty drive 1663: Tredford Publishing, 2005. — 302 pp.

71. *Noy N., McGuiness D.* Ontology Development 101: A Guide to Creating Your First Ontology // *Ontology Development 101: A Guide to Creating Your First Ontology*. — 2010. — P. 12 –35.

72. *Rokach L.* Decision forest: Twenty years of research // *Information Fusion*. — 2015. — V. 27,29. — P. 111–125.

73. Designing virtual bots for optimizing strategy-game groups / M. Bedia, L. Castillo, C. Lopez et al. // *Neurocomputing*. — 2015. — V. 172. — P. 453–458.

74. *Cheng M., Chou J., Cao M.* Nature-inspired metaheuristic multivariate adaptive regression splines for predicting refrigeration system performance // *Soft Computing*. — 2015. — P. 13.

75. *Bukharov O., Bogolyubov D.* Development of a decision support system based on neural networks and a genetic algorithm // *Expert Systems with Applications*. — 2015. — V. 42. — P. 6177–6183.

76. *Wang P.* Non-Axiomatic Logic A Model of Intelligent Reasoning. — California, USA: World Scientific Publishing Company, 2013. — 276 pp.

77. High-Speed General Purpose Genetic Algorithm Processor / S. P. Hoseini Alinodehi, S. Moshfe, M. Saber Zaeimian et al. // *IEEE Transactions on Cybernetics*. — 2015. — P. 2–3.

78. *Дергачев А. М.* Проблемы эффективного использования сетевых сервисов // *Научно-технический вестник СПбГУ ИТМО*. — 2011. — Т. 71. — С. 83–87.

79. *White D.* Software review: the ECJ toolkit // *Genetic Programming and Evolvable Machines*. — 2011. — V. 13. — P. 65 –67.

80. Layered Ensemble Architecture for Time Series Forecasting / M. M. Rahman, M. M. Islam, K. Murase, X. Yao // *IEEE Transactions on Cybernetics*. — 2015. — P. 1–5.

81. *Хокинг С.* Теория всего. — Москва: Амфора, 2009. — 160 с.

82. *Minsky M.* The Society of Mind. — NY, USA: Simon & Schuster, 2007. — 336 pp.

83. *Giachetti R.* Design of Enterprise Systems: Theory, Architecture, and Methods. — USA: CRC Press, 2010. — 448 pp.

84. An auto-tuning PID control system based on genetic algorithms to provide delay guarantees in Passive Optical Networks / T. Jiménez, N. Merayo, A. Andrés et al. // *Expert Systems with Applications*. — 2015. — V. 42. — P. 9211–9220.

85. An aggregated technique for optimization of SOAP performance in communication in Web services / K. Senagi, G. Okeyo, W. Cheruiyot, M. Kimwele // *Service Oriented Computing and Applications*. — 2015. — P. 6–7.

86. *Fowler M.* Patterns of Enterprise Application Architecture. — USA: Addison-Wesley Professional, 2010. — 448 pp.

87. *Brown W.* AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. — USA: Wiley, 2010. — 336 pp.

88. *White D.* Akka Concurrency / Ed. by K. Rolland. — Artima, 2013. — 521 pp.

89. *Robinson S.* WebSphere Application Server 7.0 Administration Guide. — PACKT publishing, 2009. — 344 pp.

90. *Goetzel B., Payrels T., Bloch D.* Java Concurrency in Practice. — Addison-Wesley Professional; 1 edition, 2006. — 384 pp.

91. *ATD*. After The Deadline. — 2012. http://www.afterthedeadline.com/.

92. *Goetzel B., I. Mathew*. Probabilistic Logic Networks: A Comprehensive Conceptual, Mathematical and Computational Framework for Uncertain Inference. — Springer: Springer, 2008. — 333 pp.

93. *Meyer B.* Object-Oriented Software Construction 2nd Edition. — Upper Sadle River, USA: Prentis Hall, 1997. — 1296 pp.

Список рисунков

Список таблиц

Приложение А

Интерфейсная модель

Интерфейсная модель содержит классы и интерфейсы для взаимодействия с пользователем.

RefObject

Представляет собой общий объект, который сохраняется в Базе Знаний. (Базовый класс для всех остальных классов и объектов)

- ObjectID- уникальный в пределах класса ключ
- Reference- уникальный в пределах всех баз знаний ключ
- Name-имя объекта

Request

Объект для хранения запроса пользователя.

- SubscriptionID - идентификатор подписки
- RequestText - запрос пользователя в виде текста
- Solution - ссылка на решение запроса
- State - статус запроса (например, Поиск Решения)
- FormalizedRequest - ссылка на формализованный запрос

Subscription

Информация о подписке пользователя на события

- Endpoints - списко UserEndpoint, которые будут использоваться для обратной связи с пользователем

UserEndpoint

- Type - тип точки связи с пользователем (например, веб-сервис)
- Address - адрес точки связи с пользователем

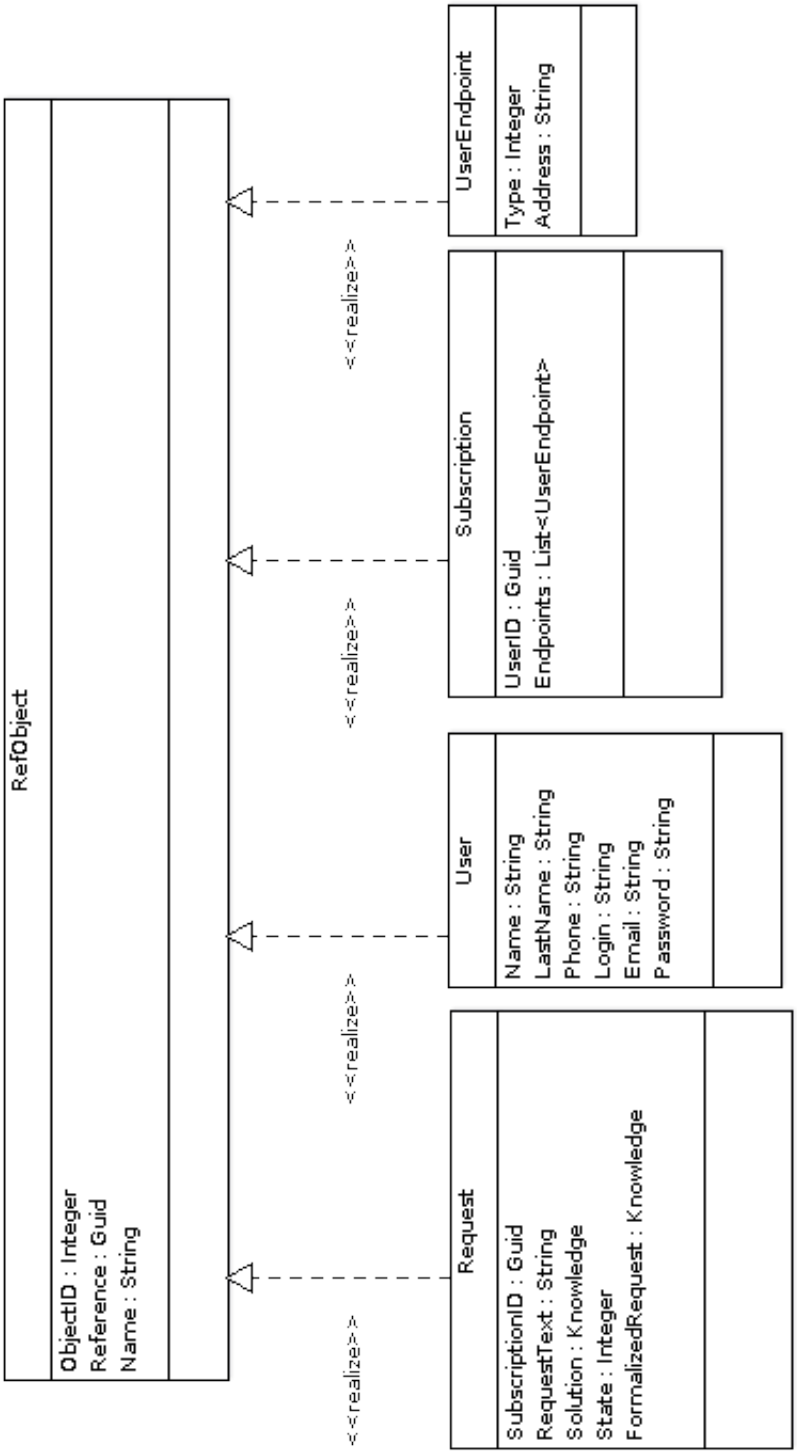

Рисунок А.1 — Диаграмма классов интерфейсной модели

Приложение Б

Описание модуля Action

Action является базовым классом для WayToThink или Critic.

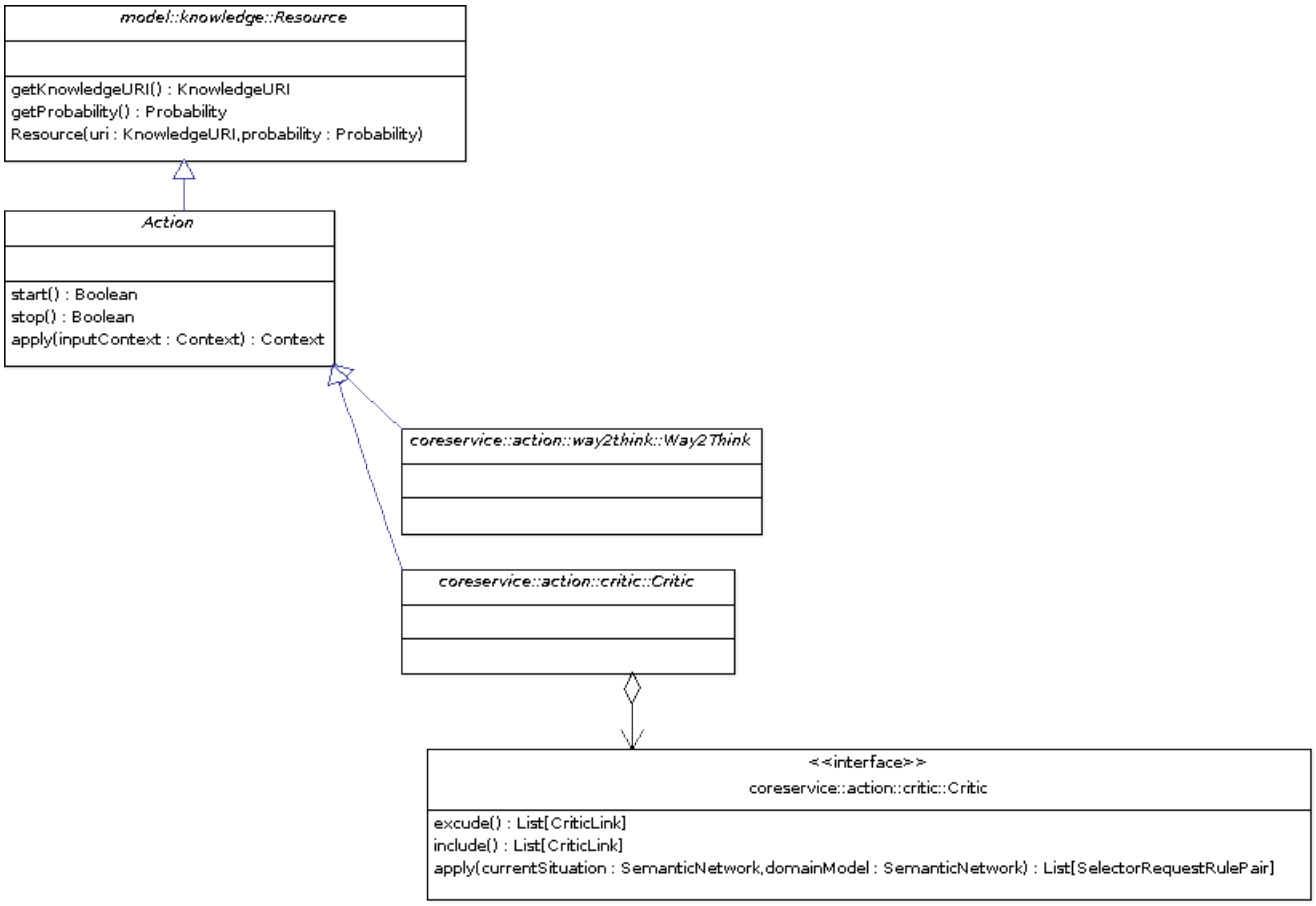

Рисунок Б.1 — Диаграмма классов Action

Приложение В

Описание модуля Цели

Goal (Цель) является набором вероятностных предикатов и последовательностью How-To необходимых для того, чтобы достичь цель. Goal и How-To тесно связаны. На Рисунке В.1 показан состав Goal. Goal состоит из:

1. Parameters - параметры, которые используются предикатами для выполнения
2. Precondition - условия, которые должны быть выполнены до выполнения проверок цели
3. Entry criteria - входной критерий, предикат, который определяет, что цель активировалась
4. Exite criteria - условия, когда цель считается выполненной
5. PostCondition - дополнительные условия для выхода
6. HowTo - набор решения. Список путей решения

Типы предикатов

В решение используется 3 типа логических предикатов: and, or, not. Представление Goal в SemanticNetwork показано на диаграмме В.2.

Иерархия целей имеет высшую цель: Помочь пользователю. Далее вниз по иерархии идут подцели: Решить инцидент, Понять тип инцидента, Найти решение инцидента и т.д.

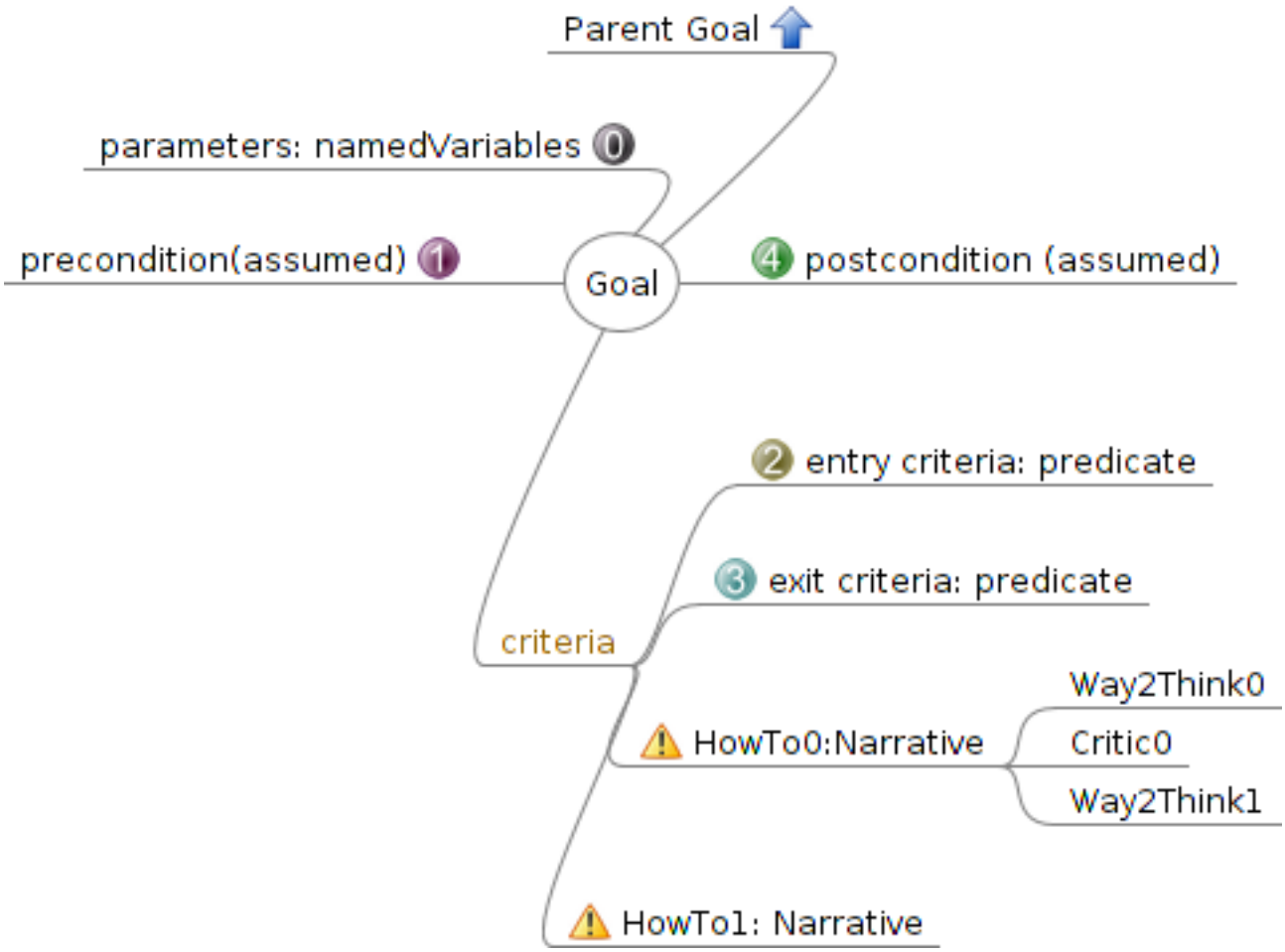

Рисунок В.1 — Диаграмма классов Goal

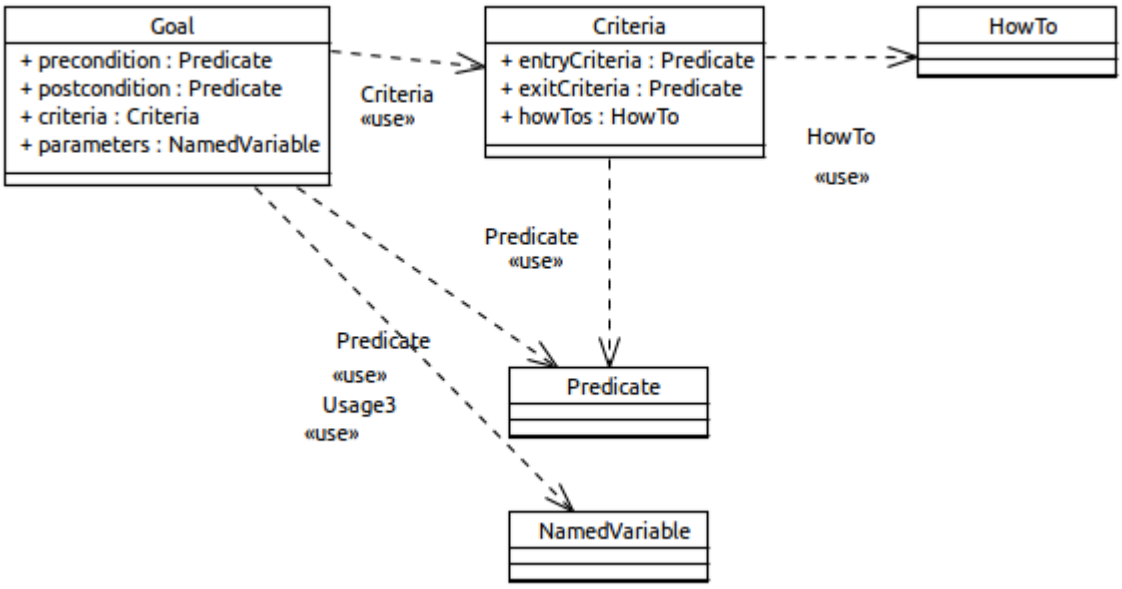

Рисунок В.2 — Диаграмма места Goal в SemanticNetwork (Семантической сети)

Приложение Г

Рецепты решений

Рецепты решений представляют собой последовательность действий выполняемых для решения проблемы, описанной в инциденте. Было разработано два типа HowTo: 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:[

    {Key:'ScriptName',
    Value:'LogonScript.bat'},
    {Key:'Description',
    Value:'Logon to server'}
  ]

  InputParameters:[
    {Key:'ServerName',
    Value:'U1'},
    {Key:'UserName',
    Value:'MyUser'}
  ]

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

  ]
}

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 Д

Экспериментальные данные

Часть экспериментальных данных (Общая длина файла примерно 10000 инцидентов).

```
EUROPE DOMAIN NEW SERVER Request Form
*(M)* unable to connect remotely to other machine
Quota limit on the personal file store exceeded Europemuk176
TCP/IP Address Management Request
Quota limit exceeded
EUROPED007 caiW2kOs:w2kLVolInst C: is now Critical at 03:16:10
EUROPEM116 caiW2kOs:w2kProcInst DRWTSN32,*,* is now Critical at
   11:51:03
2011—04—29 20:16:50 EUROPEM239 LogWatcher BABBACKUP 2011—04—30
   06:05:55 EUROPEMUK212 LogWatcher NetBDBMgr File SYSTEM_LOG
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Critical a \\
CSDTS02 The NSM/TNG NT4 System Agent reports Logical Volume C: has
    reached CRITICAL utilisation at \\
CSDTS02 The NSM/TNG NT4 System Agent reports Logical Volume D: has
    reached CRITICAL utilisation at \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Warning at \\
CSAPP01 Possible hardware problem detected — Please investigate
   with HP Insight Manager \\
CSAPP02 Possible hardware problem detected — Please investigate
   with HP Insight Manager \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Warning at \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Warning at\\
EUROPEM218 CA Backup — Backup_Operation_Failed at 23:20, 30/04/11
FMSDTS02 The NSM/TNG Win2k System Agent reports Logical Volume D:
   has reached CRITICAL utilisation\\
```

```
FMSDTS02 The NSM/TNG Win2k System Agent reports Logical Volume D:
   has reached WARNING state at 23:4\\
EUROPEVUK140 caiW2kOs:w2kMemPhys Physical Memory is now Warning at
    00:05:25
2011—05—01 00:27:37 EUROPEMUK236 LogWatcher CA_Backup_F
   Backup_Operation_Failed File \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Warning at\\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Critical a\\
2011—05—01 00:51:37 EUROPEMUK236 LogWatcher CA_Backup_F
   Backup_Operation_Failed File \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Warning at \\
2011—05—01 01:33:37 EUROPEMUK236 LogWatcher CA_Backup_W
   Check_Device_Group File \\
EUROPEVUK232 WinA3_CPUTotal:TotalLoad CPUTotal is now Critical at
   01:50:42 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
FLETCHER The NSM/TNG Win2k System Agent reports Logical Volume C:
   has reached WARNING state at 02:4 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
EUROPEVUK232 WinA3_CPUTotal:TotalLoad CPUTotal is now Warning at
   04:56:43 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Critical a
```

```
EUROPEMUK521 WinA3_NetInst Intel[R] 82567LF-3 Gigabit Network
   Connection is now Critical at 05:40:5
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK541 caiWinA3 caiWinA3 is now DOWN at 09:46:20 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Critical a
EUROPEVUK216 caiW2kOs:w2kNetTotal Net Total is now Critical at
   11:02:05 \\
EUROPEM218 CA Backup - Backup_Operation_Failed at 11:54, 01/05/11
   \\
EUROPEVUK140 caiW2kOs:w2kMemPhys Physical Memory is now Warning at
    12:35:25 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network  \\Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\Connection is now Warning at
EUROPEMUK541 caiLogA2 caiLogA2 is now DOWN at 13:49:20 \\
EUROPEMUK541 caiWinA3 caiWinA3 is now DOWN at 13:49:31 \\
UKM145 caiW2kOs:w2kCpuInst CPU 0 is now Warning at 14:53:24 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK541 caiWinA3 caiWinA3 is now DOWN at 17:47:51 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\Connection is now Warning at
EUROPEVUK232 WinA3_CPUTotal:TotalLoad CPUTotal is now Critical at
   18:52:43 \\
EUROPEVUK039 Mib-II:IP_Interface 172.19.12.218 is now Broken at
   19:06:42 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\Connection is now Warning at
EUROPEVUK039 caiW2kOs:w2kSrvcInst CASUniversalAgent is now
   Critical at 19:24:53 \\
```

```
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\Connection is now Critical a
EUROPEVUK050A Mib-II:IP_Interface 172.19.244.7 is now Broken at
   19:52:07 \\
EDISON Mib-II:IP_Interface 172.19.244.76 is now Broken at 19:54:02
    \\
EUROPEVUK053A Mib-II:IP_Interface 172.19.244.8 is now Broken at
   19:54:59 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Warning at
CSDTS02 The NSM/TNG NT4 System Agent reports Logical Volume C: has
    reached \\ CRITICAL utilisation at
2011-05-01 22:05:36 EUROPEMUK236 LogWatcher CA_Backup_F
   Unable_To_Find_Any_Media File  \\
2011-05-01 22:05:36 EUROPEMUK236 LogWatcher CA_Backup_F
   Backup_Operation_Failed File \\
2011-05-01 22:07:36 EUROPEMUK236 LogWatcher CA_Backup_F
   Backup_Operation_Failed File \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at \\
CSAPP02 Possible hardware problem detected - Please investigate
   with HP Insight Manager \\
CSAPP01 Possible hardware problem detected - Please investigate
   with HP Insight Manager \\
EUROPEVUK232 WinA3_CPUTotal:TotalLoad CPUTotal is now Warning at
   00:32:44 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Critical a
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Critical a
EUROPEMUK541 caiWinA3 caiWinA3 is now DOWN at 01:50:22
EUROPEMUK541 caiLogA2 caiLogA2 is now DOWN at 01:50:24 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Warning at
EUROPEM116 caiW2kOs:w2kCpuInst CPU 0 is now Warning at 03:25:03
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network \\ Connection is now Warning at
```

```
EDISON Mib—II:IP_Interface 172.19.244.76 is now Broken at 19:54:02
    \\
2011—05—02 05:01:47 EUROPEMUK212 LogWatcher NetBDBMgr File
   SYSTEM_LOGapplication MatchPattern FILE \\
2011—05—02 05:13:47 EUROPEMUK212 LogWatcher NetBDBMgr File
   SYSTEM_LOGapplication MatchPattern FILE \\
EUROPEM116 caiW2kOs:w2kCpuInst CPU 0 is now Warning at 05:25:03
CSDTS02 The NSM/TNG NT4 System Agent reports Logical Volume C: has
    reached WARNING state at 05:26, \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Critical a
EUROPEMUK529 WinA3_NetInst:InBytes Intel[R] 82578DM Gigabit
   Network Connection is now Warning at 05 \\
EUROPEMUK521 WinA3_NetInst Intel[R] 82567LF—3 Gigabit Network
   Connection is now Critical at 05:40:5 \\
EUROPEMUK541 caiLogA2 caiLogA2 is now DOWN at 05:48:16 \\
EUROPEMUK541 caiWinA3 caiWinA3 is now DOWN at 05:48:52 \\
CSDTS02 The NSM/TNG NT4 System Agent reports Logical Volume C: has
    reached WARNING state at 06:08, \\
UKM145 caiW2kOs:w2kCpuInst CPU 0 is now Warning at 06:57:22 \\
UKM145 caiW2kOs:w2kCpuInst CPU 1 is now Warning at 06:57:22 \\
2011—05—02 07:01:17 UKM205 LogWatcher BABBACKUP is now Critical \\
2011—05—02 07:39:36 EUROPEMUK236 LogWatcher CA_Backup_I
   Media_Error File D:Program Files CABright \\
EUROPEVUK053A caiW2kOs:w2kProcInst DRWTSN32,*,* is now Critical at
    09:44:01 \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Critical a
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
EDISON Mib—II:IP_Interface 172.19.244.76 is now Broken at 19:54:02
    \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Warning at
LUCAS caiW2kOs:w2kCpuInst CPU 0 is now Critical at 14:27:59 \\
LUCAS caiW2kOs:w2kCpuTotal CPU Total is now Critical at 14:27:59
   \\
UKM145 caiW2kOs:w2kCpuInst CPU 0 is now Warning at 14:55:21 \\
UKM145 caiW2kOs:w2kCpuInst CPU 1 is now Warning at 14:57:21 \\
```

```
2011—05—02 15:01:17 UKM205 LogWatcher BABBACKUP is now Critical \\
2011—05—02 17:01:59 EUROPEMUK268 LogWatcher BABbackup File\\
2011—05—02 17:06:50 EUROPEMUK176 LogWatcher BABBACKUP File \\
2011—05—02 20:17:01 EUROPEM239 LogWatcher BABBACKUP File \\
 caiLogA2 caiLogA2 is now DOWN at 21:48:52 \\
CSDTS02 The NSM/TNG NT4 System Agent reports Logical Volume C: has
    reached \\ CRITICAL utilisation at
CSAPP01 Possible hardware problem detected — Please investigate
   with HP  \\ Insight Manager
CSAPP02 Possible hardware problem detected — Please investigate
   with HP \\ Insight Manager
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network \\ Connection is now Critical a
2011—05—02 23:32:02 EUROPEMUK177 LogWatcher BABBACKUP File \\
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Warning at
2011—05—03 05:01:46 EUROPEMUK212 LogWatcher NetBDBMgr File
   SYSTEM_LOG\application MatchPattern FILE
2011—05—03 05:13:46 EUROPEMUK212 LogWatcher NetBDBMgr File
   SYSTEM_LOG\application MatchPattern FILE
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Warning at
CSDTS02 The NSM/TNG NT4 System Agent reports Logical Volume C: has
    reached WARNING state at 05:30,
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Critical a
EUROPEMUK521 WinA3_NetInst Intel[R] 82567LF—3 Gigabit Network
   Connection is now Critical at 05:40:5
EUROPEMUK541 caiLogA2 caiLogA2 is now DOWN at 05:50:00
EUROPEMUK541 caiWinA3 caiWinA3 is now DOWN at 05:50:08
2011—05—03 06:23:35 EUROPEMUK236 LogWatcher CA_Backup_I
   Media_Error File D:\Program Files\CA\Bright
2011—05—03 06:31:35 EUROPEMUK236 LogWatcher CA_Backup_I
   Media_Error File D:\Program Files\CA\Bright
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Critical a
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit
   Network Connection is now Warning at
```

FLETCHER The NSM/TNG Win2k System Agent reports Logical Volume C: has reached WARNING state at 07:5

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Critical a

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Critical a

2011—05—03 09:02:02 EUROPEMUK177 LogWatcher BABBACKUP **File** C:\ **Program** Files\CA\ARCserve Backup\LOG\

D: drive on Europemde011 is **in** warning state.

EUROPE DOMAIN **NEW** SERVER Request Form Submitted via 7799 Web Site

drive on Europemde011 is **in** warning state.

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Critical a

LUCAS caiW2kOs:w2kCpuInst CPU 0 is now Critical at 14:27:59

LUCAS caiW2kOs:w2kCpuTotal CPU Total is now Critical at 14:27:59

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

EUROPE DOMAIN **NEW** SERVER

EUROPE DOMAIN **NEW** SERVER

EUROPE DOMAIN **NEW** SERVER

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF—3 Gigabit Network Connection is now Warning at

```
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at
EUROPEMUK529 WinA3_NetInst:OutPkts Intel[R] 82578DM Gigabit
   Network Connection is now Warning at 13
EUROPEMUK529 WinA3_NetInst:OutPkts Intel[R] 82578DM Gigabit
   Network Connection is now Warning at 13
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at
EUROPEVUK083 caiW2kOs:w2kCpuInst CPU 0 is now Critical at 13:25:27
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at
2011-05-03 14:26:27 EUROPEM240 LogWatcher BABHOLD File C:\Program
   Files\CA\ARCserve Backup\LOG\Back
2011-05-03 14:28:02 EUROPEMUK177 LogWatcher BABHOLD File C:\
   Program Files\CA\ARCserve Backup\LOG\Ba
2011-05-03 14:28:50 EUROPEMUK176 LogWatcher BABHOLD File C:\
   Program Files\CA\ARCserve Backup\LOG\Ba
2011-05-03 14:30:14 EUROPEMUK178 LogWatcher BABHOLD File C:\
   Program Files\CA\ARCserve Backup\LOG\Ba
2011-05-03 14:47:47 EUROPEMUK177 LogWatcher BABHOLD is now
   Critical
2011-05-03 14:56:27 EUROPEM240 LogWatcher BABHOLD File C:\Program
   Files\CA\ARCserve Backup\LOG\Back
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at
EUROPEMUK521 WinA3_NetInst:InErrors Intel[R] 82567LF-3 Gigabit
   Network Connection is now Warning at
TCP/IP Address Management Request Form
```