

Казанский (Приволжский) Федеральный Университет

На правах рукописи

УДК 004.8

Тощев Александр Сергеевич

**Разработка эффективного подхода обработки
производственных задач прикладного характера в области
обслуживания программного обеспечения информационной
инфраструктуры предприятия**

Специальность 05.13.01 —

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

Диссертация на соискание учёной степени

Кандидат технических наук

Научный руководитель:

доктор физико-математических наук, профессор

Елизаров А.М.

Казань — 2015

Оглавление

	Стр.
ВВЕДЕНИЕ	5
 Глава 1. ПОСТАНОВКА ЗАДАЧИ ПОЛУЧЕНИЯ, АНАЛИЗА И ОБРАБОТКИ ЭКСПЕРТНОЙ ИНФОРМАЦИИ	 12
1.1 Возникновение области	12
1.2 Прогноз развития области	13
1.3 Методологии, используемые в области IT аутсорсинга: ITIL и ITSM	14
1.4 Техничко-экономическое обоснование	15
1.5 Постановка задачи	16
 Глава 2. АНАЛИЗ ТЕКУЩИХ РЕШЕНИЙ ПОЛУЧЕНИЯ, АНАЛИЗА И ОБРАБОТКИ ЭКСПЕРТНОЙ ИНФОРМАЦИИ В ОБЛАСТИ ОБСЛУЖИВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ИНФОРМАЦИОННОЙ ИНФРАСТРУКТУРЫ	 17
2.1 Обзор решений	17
2.2 Требования к системе	19
2.3 Методы и комплексы обработки естественного языка	21
2.3.1 Обработка Эталонных Текстов	21
2.3.2 Обработка текстов с ошибками	24
2.3.3 Сравнение средств обработки русского и английского языка	26
2.4 Вывод по главе	27
 Глава 3. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ СИСТЕМЫ	 29
3.1 Модели мышления	29
3.2 Модель мышления на базе нейронных сетей	29
3.3 Модель с использованием Деревьев Принятия Решений (Menta 0.1)	30
3.3.1 База знаний на основе OWL	30
3.3.2 Основные компоненты модели	31
3.3.3 Выводы по модели	33

3.4	Модель с использованием Генетических алгоритмов на базе модели мышления Питера Норвига (Menta 0.2-0.3)	33
3.4.1	Особенности решения	34
3.4.2	Выводы по модели	36
3.5	Модель мышления Марвина Мински (TU)	37
3.5.1	Крити-Селектор-Путь мышления	37
3.5.2	Уровни мышления	40
3.5.3	K-line	43
3.6	Выводы по главе	44
Глава 4.	РЕЗУЛЬТАТЫ РАБОТЫ	45
4.1	Архитектура системы	45
4.1.1	Компоненты системы	49
4.1.2	Компонент Webservice	52
4.1.3	Компонент CoreService.ThinkingLifeCycle	54
4.1.4	Компонент CoreService.Selector	66
4.1.5	Компонент CoreService.Critics	74
4.1.6	Компонент CoreService.WayToThink	78
4.1.7	Компонент CoreService.PreliminaryAnnotator	82
4.1.8	Компонент CoreService.KnowledgeBaseAnnotator	83
4.1.9	Компонент DataService	84
4.1.10	Компонент ClientAgent	85
4.1.11	Компонент Reasoner	86
4.2	Модель данных системы - TU Knowledge	87
4.3	Прототип	90
4.3.1	UML диаграмма действий приложения	91
4.3.2	Технологии прототипа	91
4.4	Апробация прототипа	93
4.4.1	Экспериментальные данные	93
4.4.2	Верификация	95
4.5	Выводы по главе	96
	ЗАКЛЮЧЕНИЕ	97
	Список литературы	99

Список рисунков	103
Список таблиц	105
Приложение А. Интерфейсная модель	107
Приложение Б. Описание модуля Action	109
Приложение В. Описание модуля Цели	110
Приложение Г. Рецепты решений	112
Приложение Д. Экспериментальные данные	115

ВВЕДЕНИЕ

В настоящее время набрали большую популярность системы удаленной поддержки информационной инфраструктуры предприятия. Явление это стало называться «Аутсорсинг» (от англ. "out source" вне источника). Ввиду развития рынка компаниям становится невыгодно держать свой штат службы поддержки, и они отдают ее сторонней компании.

Из-за возросшей популярности данного бизнеса и появления большого количества игроков на рынке возникла большая конкуренция, что потребовало увеличения эффективности и сокращения издержек, что в свою очередь привело к необходимости системного анализа области и выработке решения сложившихся проблем. В контексте этой проблемы рассматривается модель области и модель системы, которая увеличивает эффективность работы путем частичной (в некоторых случаях полной) автоматизации обработки инцидентов, начиная с разбора на естественном языке и заканчивая применением найденного решения.

Главным требованием к подобной системе является замена части функций, которые сейчас обеспечивают специалисты:

1. Обработка запросов на естественном языке
2. Возможность обучения
3. Общение с человеческим специалистом
4. Проведение логических рассуждений: аналогия, дедукция, индукция
5. Умение абстрагировать решение одной проблемы и, экстраполируя его, применить для других решений
6. Способность поддерживать диалог с пользователем

На данный момент компании ведут разработку подобных систем. Примером является набирающая популярность IBM Watson. Подобный класс систем также называется вопросно-ответными системами.

В данной работе была произведена попытка создания подобной системы на основе исследования целевой области (удаленная поддержка информационной структуры предприятия) и построения ее модели. Акцент был сделан на создании интеллектуальной системы для решения широкого круга проблем.

Целью данной работы является комплексное исследование области удаленной поддержки информационной структуры предприятия, создание ее модели, выработка списка проблем, оценка подходов к их решению, создание архитектуры и реализация базового прототипа программного комплекса, обеспечивающего разбор и понимание входного запроса пользователя и поиск решения данной проблемы.

Для достижения поставленной цели необходимо было решить следующие задачи:

1. Провести теоретико-множественный и теоретико-информационный анализ сложных систем в области поддержки информационной инфраструктуры
2. Рассчитать технико-экономическое обоснование возможности автоматизации целевой области
3. Создать модель целевой области
4. Исследовать модели мышления и выбрать наиболее подходящую
5. На основе выбранной модели мышления разработать модель проблемно-ориентированной системы управления, принятия решений и оптимизации технических объектов в области обслуживания информационной структуры предприятия
6. Создать архитектуру приложения на основе модели
7. Реализовать прототип на основе архитектуры
8. Провести апробацию прототипа на тестовых данных

Основные положения, выносимые на защиту:

1. Теоретико-множественный и теоретико-информационный анализ сложных систем в области поддержки информационной инфраструктуры
2. Модель проблемно-ориентированной системы управления, принятия решений и оптимизации технических объектов в области обслуживания ИТ, ее технико-экономическое обоснование
3. Прототип программной реализации модели проблемно-ориентированной системы управления, принятия решений и оптимизации технических объектов в области обслуживания ИТ
4. Апробация системы на контрольных примерах и ее результаты

Научная новизна:

1. Была создана модель проблемно-ориентированной системы управления, принятия решений в области обслуживания информационной структуры предприятия на основе модели мышления
2. Доказана применимость модели для других областей
3. Была представлена новая модель данных для модели мышления и оригинальный способ ее хранения, эффективный по сравнению с другими базами данных
4. Было выполнено оригинальное исследование моделей мышления в области обслуживания информационной структуры предприятия
5. На основе модели была создана архитектура системы и ее прототип

Научная и практическая значимость Система, разрабатываемая в рамках данной работы носит значимый практический характер. Идея работы зародилась из производственных проблем в IT отрасли, с которыми автор сталкивался каждый день. Только глубокое понимание области помогло выбрать правильное решение. Более подробное описание представлено в Главе 1. **Степень достоверности** полученных результатов обеспечивается выполнением тестов на контрольных примерах. Отчеты находятся в соответствии с полученными другими авторами, экспертными системами и специалистами.

В работе были проведены исследования согласно паспорту специальности 05.13.01, сопоставление приведено в Таблице 1.

Таблица 1 — Сопоставление направлений исследования специальности 05.13.01 и исследований, проведенных в работе

Направление исследования	Результат работы
Разработка критериев и моделей описания и оценки эффективности решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации	В рамках работы была разработана модель системы принятия решения и обработки информации в области решения запросов пользователя на естественном языке.
Продолжение следует	

Таблица 1 – продолжение

Направление исследования	Результат работы
Разработка проблемно-ориентированных систем управления, принятия решений и оптимизации технических объектов	По модели, разработанной в предыдущем пункте был разработан прототип системы принятия решения Thinking-Understanding, который был испытан на модельных данных.
Методы получения, анализа и обработки экспертной информации	В рамках системы TU был разработан метод обработки экспертной информации - обучение при помощи модели мышления TU, основанной на принципах модели б-ти Марвина Мински.
Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений и обработки информации	В рамках разработки системы TU были созданы специальные алгоритмы для анализа запросов пользователя и принятия решений.
Теоретико-множественный и теоретико-информационный анализ сложных систем	В рамках работы был проведен комплексный анализ области поддержки программного обеспечения, с помощью которого была построена система данной области и выделены участки для оптимизации принятия решений.
Методы и алгоритмы интеллектуальной поддержки при принятии управленческих решений в технических системах	Система, разработанная в рамках данной работы включает в себя инновационные методы и алгоритмы поддержки принятия решений, использующих в своей основе модель мышления на базе модели мышления Человека, описанной в книге Марвина Мински.
Продолжение следует	

Таблица 1 – продолжение

Направление исследования	Результат работы
Визуализация, трансформация и анализ информации на основе компьютерных методов обработки информации	В Главе 1 представлена наглядная визуализация данных по системному анализу области удаленной поддержки инфраструктуры.

Апробация работы. Основные результаты работы докладывались на:

- Конференция Лобачевского - 2011
- WCIT-2012
- AINL-2013
- RCDL-2014
- AMSTA-2015

Апробация работы проводилась на выгрузка инцидентов из систем регистрации ОАО "АйСиЭл КПО ВС". Система показала хорошие результаты обработки данной информации. **Личный вклад.** Автор принимал активное участие в исследовании целевой области, разработке архитектуры приложения, реализации прототипа, проработки теории, тестировании прототипа.

Публикации. Основные результаты по теме диссертации изложены в 8 печатных изданиях [1], [2], [3], [4], [5], [6], [7], [8], 1 из которых изданы в журналах Scopus [7], 1 в журнале Web of Science [8], 1 в журнале РИНЦ [4], 4 в тезисах докладов [1], [2], [3], [4].

Объем и структура работы. Диссертация состоит из введения, четырех глав, заключения и пяти приложений. Полный объем диссертации составляет 122 страницы с 45 рисунками и 27 таблицами. Список литературы содержит 33 наименования.

Таблица 2 — Словарь терминов

Термин	Значение
База Знаний	База данных приложения, представленная в виде онтологии знаний
WayToThink	Путь мышления. Основан на определении Марвина Мински [9]. Класс объектов, которые модифицируют данные
Critic	Критик. Основан на определении Марвина Мински [9]. Класс объектов, которые выступают триггерами при наступлении определенного события
ThinkingLifeCycle	TLC. Основан на определении Марвина Мински [9]. 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".
TU	Сокращение от ThinkingUnderstanding.

Таблица 3 — Принятые аннотации для изложения

Аннотация	Описание
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.

Глава 1. ПОСТАНОВКА ЗАДАЧИ ПОЛУЧЕНИЯ, АНАЛИЗА И ОБРАБОТКИ ЭКСПЕРТНОЙ ИНФОРМАЦИИ

1.1 Возникновение области

В настоящее время в области ИТ набрало большую популярность системы удаленной поддержки информационной инфраструктуры, так называемый «Аутсорсинг». Ввиду развития рынка компаниям становится невыгодно держать свой штат службы поддержки, и они отдают свою инфраструктуру сторонней компании. Большинство проблем, которые решает удаленная служба поддержки носят весьма тривиальный характер.

- Установить приложение
- Переустановить приложение
- Решить проблему с доступом к тому или иному ресурсу

Данные проблемы решают специалисты технической поддержки. Обычно техническая поддержка делится на несколько линий.

Таблица 1.1 — Описание работы специалистов различных уровней поддержки

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

Каждая линия поддержки представлена специалистами. В среднем команда, обслуживающая одного заказчика насчитывает 60 человек. Процентное соотношение специалистов разных линий поддержки отображено на Диаграмме 1.1



Рисунок 1.1 — Диаграмма состава команд

Работа специалиста 1 линии поддержки состоит из множества рутинных и простых задач. На Диаграмме 1.2 показано соотношение разных типов проблем, встречающихся во время работы поддержки

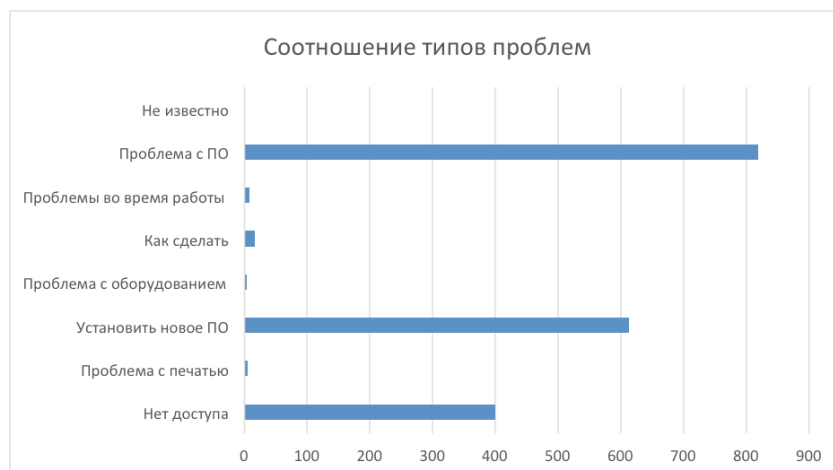


Рисунок 1.2 — Диаграмма соотношений типов проблем

Решение части задач может быть автоматизировано, а специалисты получают дополнительное время на решение более интересных задач. Проблема заключается в автоматизации решения рутинных задач в области удаленной поддержки инфраструктуры.

1.2 Прогноз развития области

Основной тенденцией в развитии области удаленной поддержки инфраструктуры является попытки удешевить и улучшить стоимость предоставления

Таблица 1.2 — Категории инцидентов в области удаленной поддержки инфраструктуры

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

услуг.

Компании, работающие на этом рынке вкладывают большие деньги в автоматизацию. Кроме того современное развитие науки и техники, а точнее вычислительных мощностей позволяет автоматизацию даже самых наукоемких процессов.

Дальнейшим развитием области является замена человеческих специалистов на автоматические системы. Многие ведущие компании ведут разработки в этом направлении. Например, компания HP. Данная компания имеет свои системы по регистрации подобных инцидентов и сейчас ведется работа над автоматизацией системы.

1.3 Методологии, используемые в области IT аутсорсинга: ITIL и ITSM

В области IT аутсорсинга есть несколько готовых стандартов ведения работ. Одним из таких стандартов является библиотека ITIL. Данный стандарт описывает лучшие практики организации работ в области IT аутсорсинга. Используемый

в библиотеки подход соответствует стандартам ISO 9000 (ГОСТ Р ИСО 9000) . Наличие стандартов в области диктует унифицированность постановки проблем, а также унифицированность алгоритмов решения. Такие предпосылки говорят о возможности частично или в некоторых случаях полной автоматизации решения проблем.

1.4 Техничко-экономическое обоснование

По данным аналитики портала SuperJob [10] средняя зарплата системного администратора с опытом работы в среднем по Казани составляет 30000-35000р за 2014 года. Из расчета на 1 час с учетом 21 рабочих дней в месяце - 179-208 рублей в час. Расход компании на человека складывается по следующий формуле [11] :

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

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 в час. Таким образом за 8 часовой рабочий день от 2280 до 2512. Стоимость аренды выделенного сервера (Xeon X3, 1.7 GHz, 8GB RAM, 256GB SSD) стоит 8 900 руб./мес [12]. Таким образом за 1 час стоит 53 рубля с учетом 8 часового рабочего дня. Но сервер может работать 24 часа в сутки за исключением простоев на обслуживание, которые составляют не более 5 %. Итого сервер работает 478,8 часов в месяц. С этой точки зрения сервер будет стоить 18,5 рублей в час. Один сервер в своем быстроедействие может заменить несколько специалистов при решении задач. Чтобы решение было экономически эффективным необходимо, чтобы оно сокращало расходы как минимум на 30%. Грубый подсчет на основе стоимости часа и пропорции показывает, что работа специалиста это 6% работы сервера (без учета работы сервера как несколько специалистов и примерно одинаковой скорости решения инцидентов). Таким образом уровень решения инцидентов системой в 50% выполнит требования по прибыли примерно 1027 %.

1.5 Постановка задачи

Задачами данного исследования являются:

- Изучение возможности автоматизации области удаленной поддержки инфраструктуры путем анализа области
- Выработка критериев и сравнительный анализ существующих решений в области
- Создание модели проблемно-ориентированной системы принятия решений для решения задачи автоматизации поддержки удаленной инфраструктуры
- Создание проблемно-ориентированной системы принятия решений для автоматизации поддержки удаленной инфраструктуры
- Создание методических рекомендаций для проведения верификации, проведение верификации и представление результатов
- Подсчет статистических результатов работы комплекса

HP OpenView [13] является комплексным программным решением по мониторингу ИТ инфраструктуры предприятия. Система имеет множество модулей. Данная система охватывает широкий спектр возможностей:

- Система не поддерживает:

-

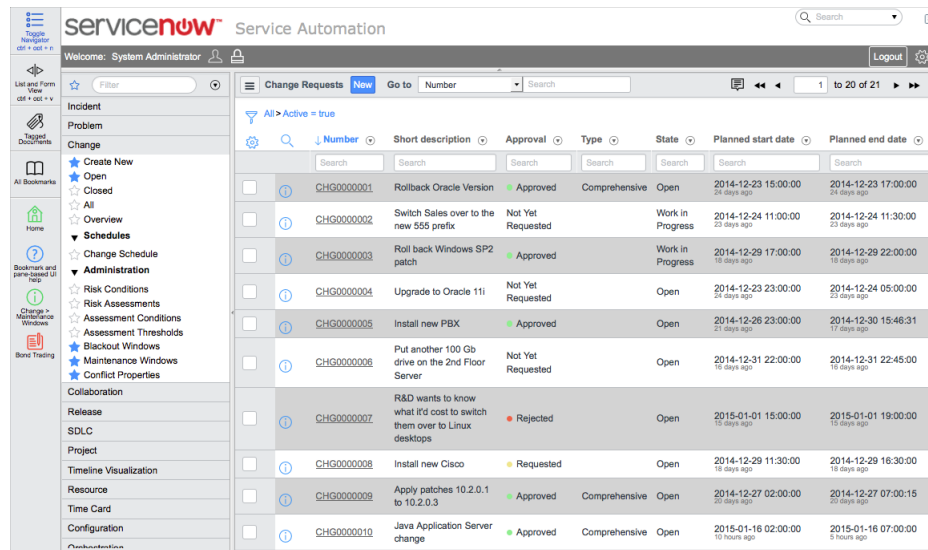
Рисунок 2.1 — HP OpenView

ServiceNOW Средства автоматизации сервиса. Предоставляет следующие возможности:

- Регистрация инцидентов
- Создание цепи обработки инцидента

Система не поддерживает:

- Понимание и формализация запросов
- Автоматическое исправление проблемы на основе формализации запроса



The screenshot shows the ServiceNOW Service Automation interface. The left sidebar contains navigation links for Incident, Problem, Change, and various administrative functions. The main area displays a table of change requests with columns for Number, Short description, Approval, Type, State, Planned start date, and Planned end date.

Number	Short description	Approval	Type	State	Planned start date	Planned end date
CHG00000001	Rollback Oracle Version	Approved	Comprehensive	Open	2014-12-23 15:00:00	2014-12-23 17:00:00
CHG00000002	Switch Sales over to the new 555 prefix	Not Yet Requested		Work in Progress	2014-12-24 11:00:00	2014-12-24 11:30:00
CHG00000003	Roll back Windows SP2 patch	Approved		Work in Progress	2014-12-29 17:00:00	2014-12-29 22:00:00
CHG00000004	Upgrade to Oracle 11i	Not Yet Requested		Open	2014-12-23 23:00:00	2014-12-24 05:00:00
CHG00000005	Install new PBX	Approved		Open	2014-12-26 23:00:00	2014-12-30 15:46:31
CHG00000006	Put another 100 Gb drive on the 2nd Floor Server	Not Yet Requested		Open	2014-12-31 22:00:00	2014-12-31 22:45:00
CHG00000007	R&D wants to know what it'd cost to switch them over to Linux desktops	Rejected		Open	2015-01-01 15:00:00	2015-01-01 19:00:00
CHG00000008	Install new Cisco	Requested		Open	2014-12-29 11:30:00	2014-12-29 16:30:00
CHG00000009	Apply patches 10.2.0.1 to 10.2.0.3	Approved	Comprehensive	Open	2014-12-27 02:00:00	2014-12-27 07:00:15
CHG00000010	Java Application Server change	Approved	Comprehensive	Open	2015-01-16 02:00:00	2015-01-16 07:00:00

Рисунок 2.2 — Service NOW

IBM Watson 2.3 Вопрос-ответная система поддерживает:

- Понимания и формализацию запросов
- Поиск решений

Система не поддерживает:

- Автоматическое исправление проблемы на основе формализации запроса

Прочие системы Кроме того существуют дополнительные способы автоматизации

- Обработка инцидентов посредством регулярных выражений. В таком решении нет гибкости, так как обработка идет путем поиска ключевых слов вне контекста
- Обработка инцидентов при помощи скриптов. Автоматизирует лишь рутинные операции

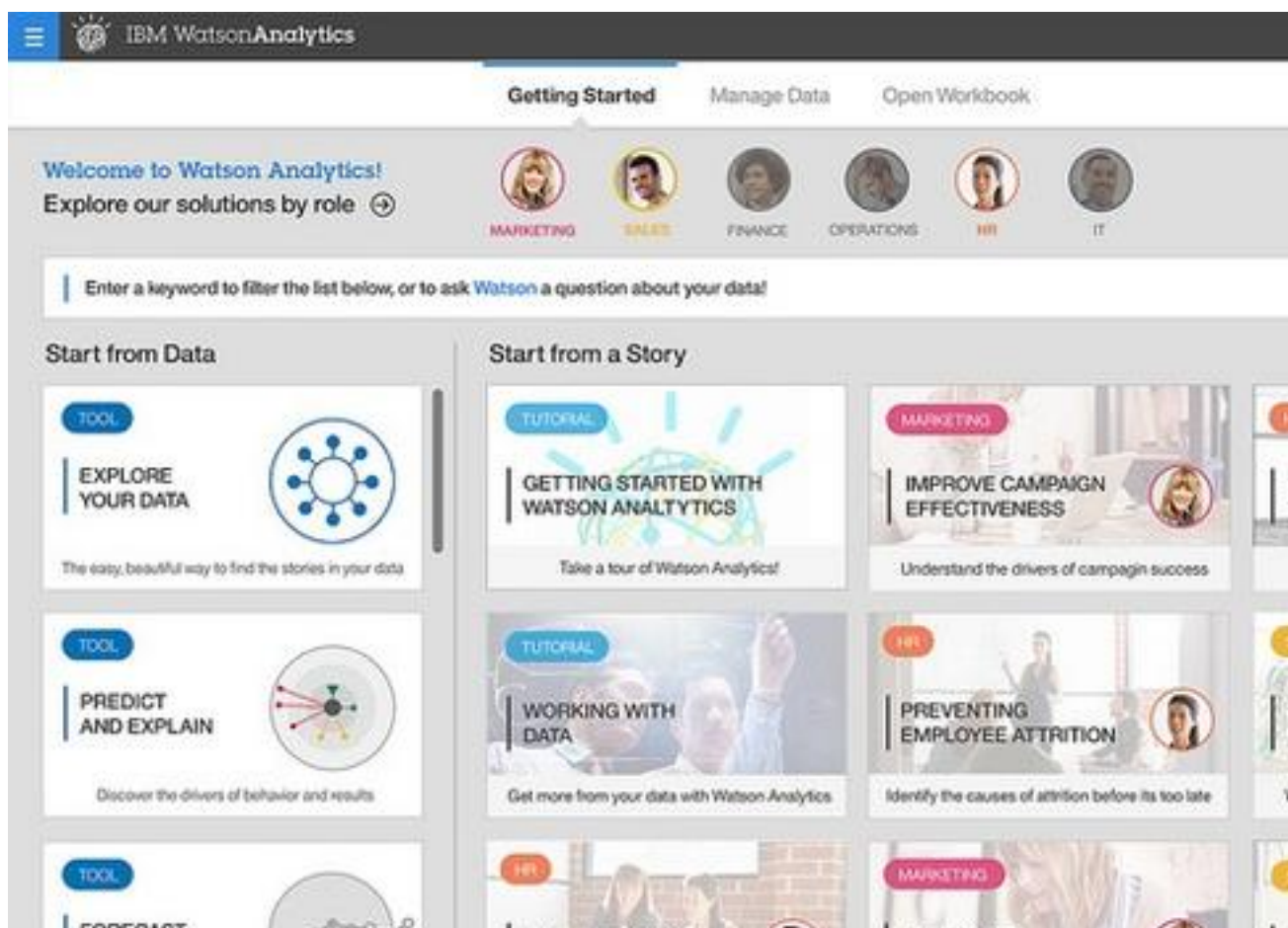


Рисунок 2.3 — Пример работы системы Watson

2.2 Требования к системе

Основными требованиями к системе являются следующие:

- Мониторинг
- Регистрация инцидентов
- Управление системами
- Создание цепи обработки (Workflow) инцидента
- Понимания и формализацию запросов на естественном языке
- Поиск решений
- Применение решений
- Обучение решению инцидента
- Умение проводить логические рассуждения: генерализацию, специализацию, синонимичный поиск

Требования к системе формировались исходя из возможностей специалистов поддержки, а также анализа проблем, которыми они занимаются. Большин-

ство инцидентов тривиальные и типичные, но все они разные. Для человека проблема "Please insall Firefox"и "Please install Chrome"идентичные, но с точки зрения формализации - нет. Общее в них можно найти взглянув на генерализацию различающейся части. Firefox и Chrome являются пакетами программного обеспечения.

2.3 Методы и комплексы обработки естественного языка

2.3.1 Обработка Эталонных Текстов

В данном разделе проводится обзор обработчиков естественного языка. За основу были взяты инциденты из выгрузки систем поддержки ОАО "ICL КПО-ВС".

Ввиду специфики области 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 инциденты:

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.*

Во время анализа были использованы следующие обработчики естественного языка:

1. Open NLP [14]
2. RelEx [15]
3. StanfordParser [16]

Результат работы вычислялся при помощи метрик, представленных в Таблице 2.1.

Результаты приведены на сводной диаграмме Рисунок 2.4

Из диаграммы видно, что наилучшие результаты показывает обработчик RelEx [15]. После анализа необработанных инцидентов было выявлено несколько проблем у всех обработчиков:

Таблица 2.1 — Таблица метрик

Метрика	Описание	Формула
Аккуратность	Понимание текста обработчиком	$Ac = \frac{1 - x}{y}$ <p>где x - количество нераспознанных слов, y количество распознанных</p>
Успешно обработанные	Успешно обработанные инциденты	$P = \frac{x}{100}$ <p>где x успешно обработанные</p>
Не успешно обработанные	Неуспешно обработанные инциденты	$N = \frac{y}{100}$ <p>где y неуспешные инциденты</p>
Результативность	Общая результативность обработчика	$R = \frac{P}{N}$
Общий бал	Общая оценка обработчика	$T = Ac + R$

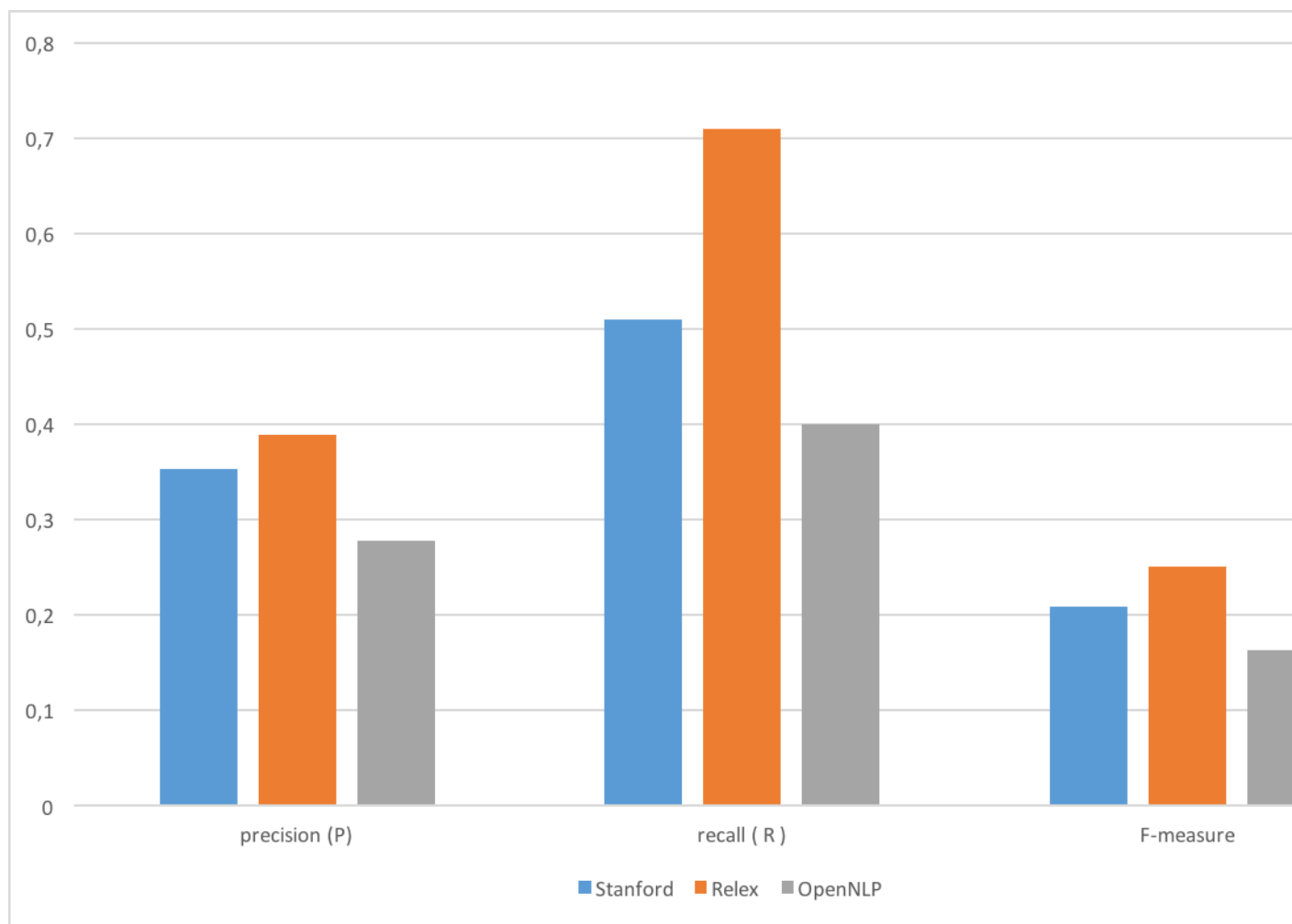


Рисунок 2.4 — Результаты обработки текстов

1. Невозможности корректировки простых грамматических ошибок, связанных с пропущенными пробелами или неверным форматированием. Ошибки первого типа.
2. Ошибки неверной интерпретации слов в предложении. Например, слово *please* интерпретировалось как глагол, хотя является по смыслу «формой вежливости». Ошибки второго типа.

2.3.2 Обработка текстов с ошибками

По результатам прошлого раздела было решено выбрать в качестве обработчика естественного языка RelEx, но были выявлены некоторые проблемы. Было принято решение исправить данные проблемы при помощи предварительной обработки текста. Предварительная обработка текста была разбита на несколько фаз:

1. Комплексная корректировка ошибок
2. Обработка при помощи внутренней базы знаний

Для того, чтобы избавиться от орфографических, синтаксических ошибок используется составной корректор. Данный компонент имеет модульную структуру и применяет корректировку последовательно.

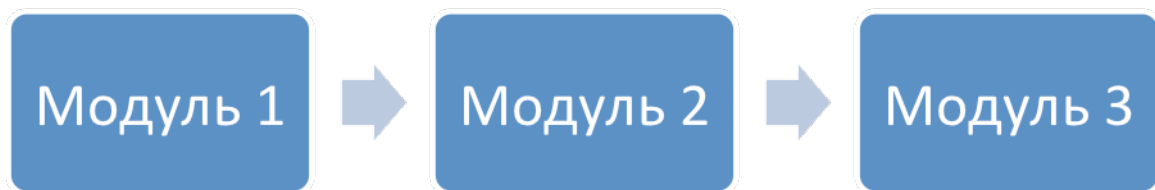


Рисунок 2.5 — Архитектура предварительной обработки текста

Для данного компонента были написаны модули корректировки:

- Google API
- After The Deadline

Таким образом удалось исправить большинство ошибок, связанных с синтаксисом, грамматикой, орфографией. Также удалось исправить ошибки неверного написания: лишних пробелов, пропущенных запятых, пропущенных точек. По-прежнему остается проблема обработки неверной интерпретации слов в

тексте.

Для корректировка ошибок второго типа был использована инъекция в работу парсера RelEx. Ввиду OpenSource природы проекта, модульности был подменен модуль извлечения и обработки слов в предложения. Стандартный процесс обработки был разбит на «предобработку» и «обработку». Стадия «обработки» включала в себя алгоритм работы такой же как был до этого в модули, на стадии «предобработки» управление передается модулю основному приложению, который проверяет данное слово на предмет его вхождения во внутреннюю Базу Знаний и если таковое имеется, то приложение передает соответствующие корректировки в модуль

2.3.3 Сравнение средств обработки русского и английского языка

Средства обработки естественного языка принято относить в большому классу средств NLP – Natural Language Processing. 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 обработки Link Grammar [17]. Link Grammar поддерживает множество языков: английский, русский, турецкий, немецкий и т.д. RelEx использует вывод LG и преобразует его в формат связей.

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)

```

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)

На их основе можно формализовать приложение на естественном языке. Остальные парсеры пока не поддерживают русский язык. Существуют открытые проекты, но они еще недостаточно развиты.

2.4 Вывод по главе

В данной главе были рассмотрены существующие на данный момент системы в области обработки экспертной информации в области обслуживания программного обеспечения и информационной архитектуры. Все рассмотренные системы не соответствуют полному комплексу необходимых требований. В Таблице 2.2 приведены сводные данные по системам. В главе также выработаны критерии сравнения обработчиков естественного языка и выполнен основной анализ об-

работчиков естественного языка. Ввиду развитости и доступности было решено использовать OpenCog RelEx.

Таблица 2.2 — 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
Итоговые очки	4	3	5

Глава 3. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ СИСТЕМЫ

В данной главе рассматриваются модели, которые были рассмотрены при выборе модели системы. Важно отметить, что работа над системой велась с 2011 года. Было выпущено 3 рабочих версии прототипа системы, реализующие различные модели мышления.

3.1 Модели мышления

В работе было рассмотрено несколько моделей мышления (в скобках кодовое имя прототипа):

- Модель мышления Марвина Мински [9] (TU)
- Модель мышления на базе нейронных сетей
- Модель с использованием Деревьев Принятия Решений (Menta 0.1)
- Модель с использованием Генетических алгоритмов на базе модели мышления Питера Норвига [18] (Menta 0.3)

3.2 Модель мышления на базе нейронных сетей

Модель на базе нейронных сетей (поддерживающая обучение) была отброшена на предварительной стадии оценки, так как имеет большие требования производительности [19] несовместимые с Технико Экономическим Обоснованием 1.4.

3.3 Модель с использованием Деревьев Принятия Решений (Menta 0.1)

Данная модель являлась одной из первых, которая была опробована. Модель была основана на деревьях принятия решений [20]. В построение модели данной системы использовались следующие компоненты:

- Обработка запросов на естественном языке
- Поиск решения
- Применение решения
- База знаний

Система была ориентирована на выполнение простых команд, например, добавить поле на форму. Основная функция модели представлена следующим потоком:

1. Получение и формализация запроса
2. Поиск решения при помощи Деревьев Принятия Решений
3. Изменение модели приложения в формате OWL [21]
4. Генерация и компиляция приложения

3.3.1 База знаний на основе OWL

В качестве базы знаний использовался owl-файл. С помощью редактора Protege [22] в базу вводились данные о приложении в виде семантической сети. Для тестирования было создано приложение, которая выполняла функциональность по добавлению заказов в базу. На Рисунке 3.1 представлен класс Order в формате OWL. Слева отображаются супер классы, к которым он привязан. Например, BLL - класс относится к бизнес логике приложения, Module - отдельный модуль в рамках системы. Справа представлены свойства класса, их описание представлено в Таблице 3.1. С помощью предикатов определяется поведение свойства: создать файл, создать новое поле. В Таблице 3.2 представлено описание иерархии предикатов. На Рисунке 3.2 представлен класс CreateCustomer в OWL. Сюда входят описание всех необходимых свойств для генерации файла на языке Java.

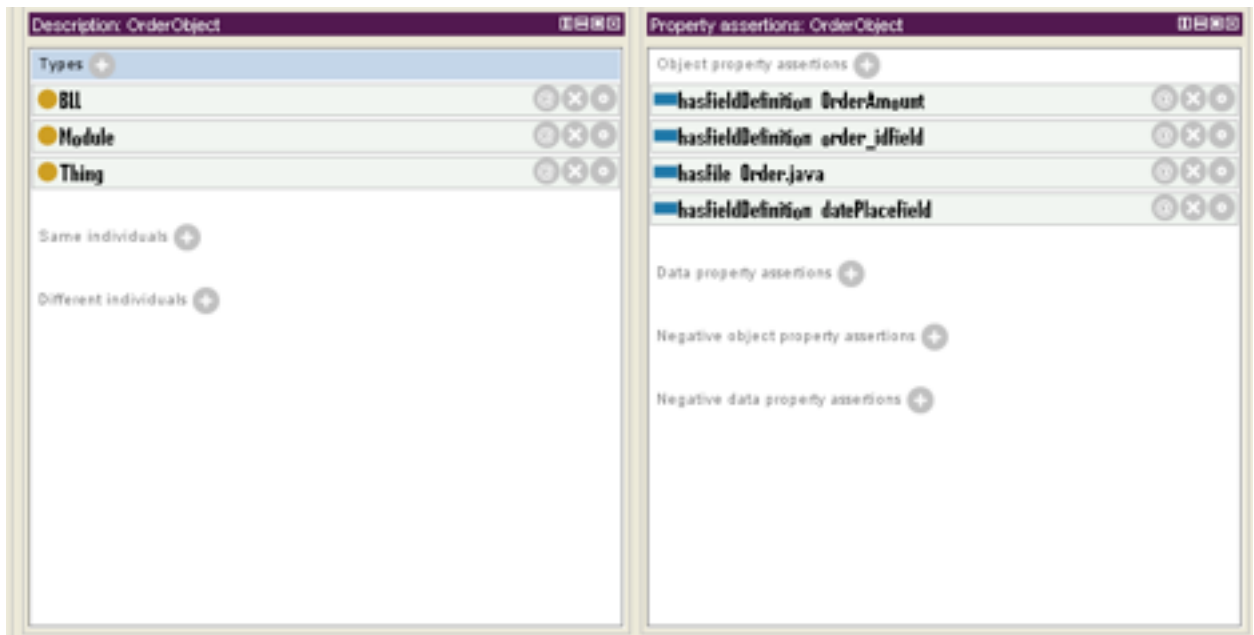


Рисунок 3.1 — Представление класса Order в OWL. Визуализация Protege.

Таблица 3.1 — 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

3.3.2 Основные компоненты модели

Основными компонентами модели является:

1. Request parser (Stanford parser)
2. Генерация Action (Action Generator)
3. Исполнение Action (Action Applier)
4. 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 и базе данных. Основной задачей данного модуля является генерация имени модуля, действия и поля. Модуль *Action Applier* ищет объект в модели по данным от Action Generator и производит действие, кроме того, используя предикат *dependOn*, он производит модификацию всех зависимых классов.

Таблица 3.2 — Описание иерархии предикатов

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



Рисунок 3.2 — Представление класса CreateCustiner в OWL. Визуализация Protege.

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 3.3 представлена диаграмма последовательности для основного рабочего потока.

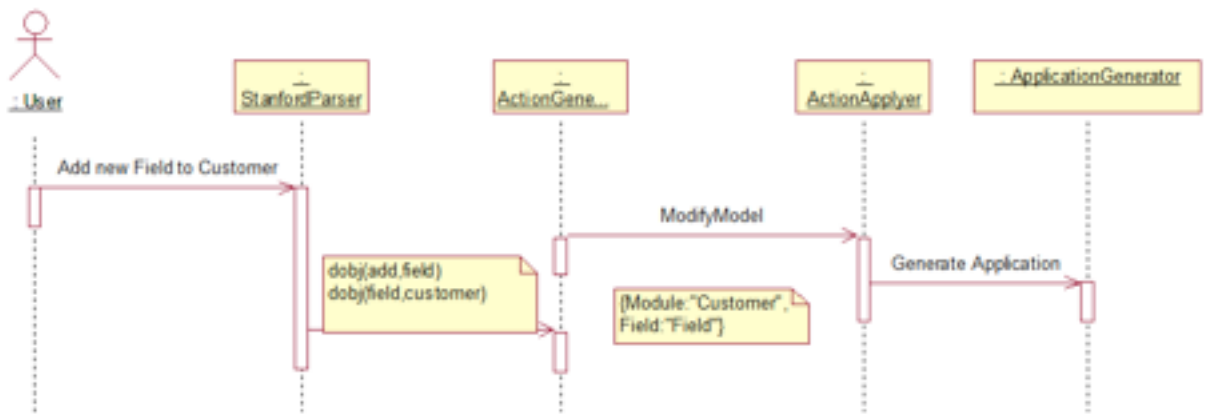


Рисунок 3.3 — Диаграмма последовательности для основного потока в модели Menta 0.1

3.3.3 Выводы по модели

Основными проблемами данной модели являлось следующее:

1. 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
2. Система поиска решения работала только в рамках модели одной программы
3. Отсутствовала функция обучения

3.4 Модель с использованием Генетических алгоритмов на базе модели мышления Питера Норвига (Menta 0.2-0.3)

После работы над ошибками была предпринята попытка сделать поиск решения более универсальным. В данной модели был добавлен модуль логики для оценки решения и модуль генетических алгоритмов для генерации решения. В рамках данной модели были сформированы основные компоненты будущей модели:

- Критерии Приемки (Acceptance Criteria)
- How-To для хранения решений проблем
- Формат данных OWL
- Использование логических вычислений для проверки решения

Система содержала внутри себя модель приложения. При помощи генетического алгоритма модель строила из частей новую систему и проверяла ее при помощи логического движка [23] соответствии входным критерия приема.

3.4.1 Особенности решения

Модель состоит из компонентов, представленных в Таблице 3.3.

Таблица 3.3 — Компоненты модели Menta 0.3

Компонент	Описание
MentaController	WebService, который предоставляет интерфейс для общения с пользователем и другими системами
Продолжение следует	

Таблица 3.3 – продолжение

Компонент	Описание
SolutionGenerator	<p>Модуль отвечает за генерацию решения. На вход он получает Acceptance Criteria. Основой является генетический алгоритм. Для него был выбран framework esj [24]. 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.</p>
SolutionChecker	<p>Проверка решения. Принимает на вход выбранные How-To, AcceptanceCriteria. Комбинирует их и передает ReasonerAdapter.</p>
ReasonerAdapter	<p>Транслирует How-To в термины NARS. NARS – non-axiomatic reasoning system [23]. Система логических суждений, разработанная профессором Pei Wang. Принцип ее действия это всевозможная комбинация фактов. Каждый факт имеет свой frequency, confidence пару. И по сочетанием получается композиция данных фактов.</p>
Продолжение следует	

Таблица 3.3 – продолжение

Компонент	Описание
Translator	<p>Транслирует объекты Базы Знаний – знания – в отчеты. Отчеты бывают следующих типов:</p> <ul style="list-style-type: none"> – Solution Report – UML Report – Patch <p>В данной версии используется первый тип отчета. Этот отчет содержит описание решения, сгенерированного системой на выбранном языке программирования.</p>
Applicator	<p>Данный модуль применяет решение к модели приложения, содержащейся в базе знаний. Также данный модель включает FileApplicator, который генерирует решение в виде файлов на выбранном языке программирования.</p>
KBServer	<p>База знаний приложения. Используется сервер non-SQL БД HypergraphDB.</p>

3.4.2 Выводы по модели

Основными недостатками подхода оказалось:

- Отсутствие обучения
- Отсутствие 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.

Кроме того после апробации оказалось, что критерии приемки практически описывают необходимое решение, что являлось недопустимым. Данный подход был описан в статье [25].

3.5 Модель мышления Марвина Мински (TU)

Модель построенная с применением модели мышления Марвина Мински содержала в себе основные концепции предыдущих моделей и показала свою состоятельность на контрольных примерах.

- Acceptance Criteria
- Обучение
- Поиск и применение решения
- Отсутствие обработки естественного языка

Данная модель является более абстрактной и представляется собой верхнеуровневую архитектуру обработки запроса (мышления), где компонентами являются лучшие части предыдущих систем.

3.5.1 Крити-Селектор-Путь мышления

В 2006 году Марвин Мински опубликовал свою книгу "The emotion machine" [9], в которой предложил свой взгляд на систему мышления и памяти человека. В основу теории легла парадигма триплета Критик-Селектор-Путь мышления, k-line для сопоставления знаний. На рисунке 3.4 представлена схематичное изображение Критика-Селектора-Пути мышления

Критик представляет собой определенный триггер: внешние обстоятельства, события или иное воздействие. Например, включился свет и зрачки сузились. Обожглись и одернули руку. Критик активируется только когда для этого достаточно обстоятельств. Одновременно могут активироваться несколько критиков. Например, человек решает сложную задачу. Идет активация мно-

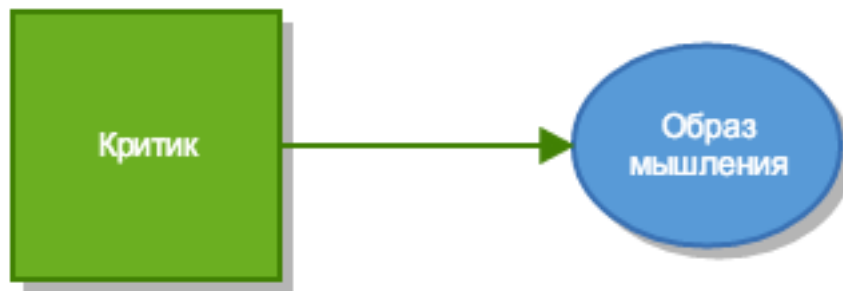


Рисунок 3.4 — Критик-Селектор-Путь мышления

жество критиков: считать, технические детали, кроме того параллельно может активироваться критик переработки, сообщающей о необходимости отдыха.

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

Путь мышления это способ решения проблемы. Путь мышления также может активировать следующий критик.

На рисунке 3.5 представления расширенная модель работы триплета Критик-Селектор-Путь мышления. Критик активирует селектор, который активирует путь мышления (синий круг). Путь мышления в свою очередь может активировать критик или же совершить определенные действия. Например, зажегся зеленый свет светофора, значит можно переходить дорогу.

Если активировалось много критиков, значит проблему нужно уточнить, так как степень неопределенности слишком высока. Если проблема очень похожа, то можно судить по аналогии.

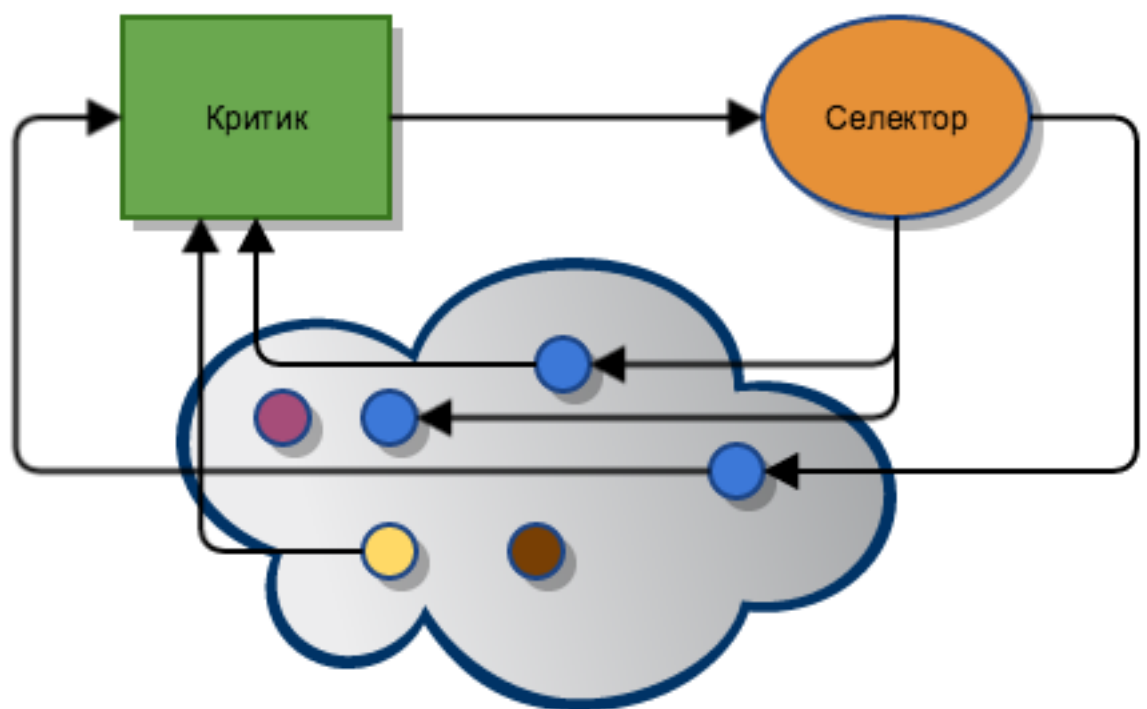


Рисунок 3.5 — Критик-Селектор-Путь мышления в разрезе ресурсов

3.5.2 Уровни мышления

Концепция уровней мышления представляет собой модель степени ментальной активности человека. Никто из людей не может похвастаться скоростью гепарда, гибкостью кошки, силой медведя. Но наш вид все это компенсирует возможностью изобретения путей мышления. Например, чтобы быть быстрыми мы изобрели самолеты, машины. Чтобы быть сильными, мы изобрели оружие. Что же делает это возможным? Безусловно результатом всего является взаимодействие человека с окружающим миром. Именно данное взаимодействие заставляет людей изобретать что-то новое, создавать шедевры литературы и летать в космос. Но как же мы всего этого добиваемся начиная от инстинктивного одергивания руки до создания Теории всего [26]. Далее мы рассмотрим концепцию уровней мышления.

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

В Таблице 3.4 представлено описание уровней мышления.

Деление на данные уровни носит условный характер. Например уровень 5 и 6 можно объединить. Но по словам Марвина Мински принцип бритвы Оккама успешно применяется в физики, но в психологии он не должен применяться так же легко.

На рисунке 3.6 представлена 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 3.4 — 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 данной 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	а данной уровне происходит мышление с использованием рассуждений. Если я сделаю так, то будет ... Например, если перебежать дорогу на зеленый свет, то можно успеть вовремя. На данном уровне сравниваются последствия нескольких решений и выбирается оптимальное. Общую формулу для этого уровня можно выразить как "Если ..., то сделать так, тогда будет так".
Рефлексивный уровень	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	Самозонательный 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 3.6 — Иллюстрация концепции Уровней мышления

3.5.3 K-line

Концепция K-line была первый раз упомянута Марвином Мински в 1987 году в журнале Cognitive Science. В книге "The Society of Mind" [27] Марвин Мински раскрывает концепцию K-line. Полностью концепция описана позже в книге "The Emotion Machine" [9]. 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 3.7 показана K-line, которая объединяет пу-

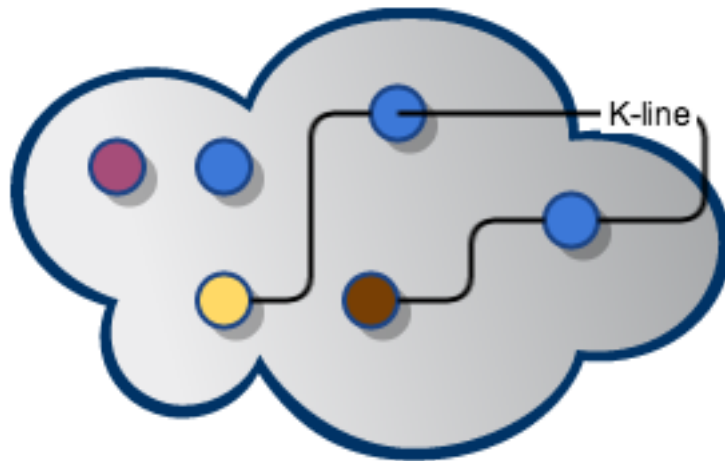


Рисунок 3.7 — Иллюстрация концепции K-line

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

3.6 Выводы по главе

Для программной экспертной системы очень важно иметь способность мышления и рассуждения. Например, очень важно для системы уметь действовать по аналогии. Так как множество запросов типичны и отличаются частностями. Например, пожалуйста, установить Office, Antivirus и т.д.

Также для экспертной системы важно уметь абстрагировать специализированные рецепты решения. Например, 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 модели мышления Мински была построена модель системы, которая поддерживает основные функции: обучение, понимание инцидента, поиск решения, применение решения. Более подробно с результатами апробации моделей можно ознакомиться на сайте <http://tu-project.com>

Глава 4. РЕЗУЛЬТАТЫ РАБОТЫ

В качестве результатов работы была произведен теоретико-множественный и теоретико-информационный анализ сложных систем в области поддержки информационной инфраструктуры (было представлено в предыдущих главах). Была разработана модель системы, по которой была разработана проблемно-ориентированная систем управления, принятия решений и оптимизации технических объектов в области обслуживания IT. В данной главе представлена модель разработанной системы - ThinkingUnderstanding (TU), архитектура, реализация и результаты испытаний. Данная глава включает общее описание компонентов и их интеграции, затем каждый компонент рассмотрен в отдельности. В конце главы рассмотрен пример работы системы и приведены результаты испытаний. Описание компонентов классов приводится на английском языке, так как того требует нотация UML [28].

4.1 Архитектура системы

Архитектура системы представляет собой модульную систему. Такой подход был выбран для того, чтобы компоненты можно было удобно заменять. Основными компоненты системы описаны в Таблице 4.1.

Таблица 4.1 — Основные компоненты системы ThinkingUnderstanding

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.
Продолжение следует	

Таблица 4.1 – продолжение

Компонент	Описание
ClientAgent	Компонент выполнения скриптов на целевой машине.
MessageBus	Шина данных для системы.

Система может работать в 2-х режимах: режим обучения и режим запроса. Вариант использования для режима обучения представлен на Рисунке 4.1. Главными действующими лицами является специалист технической поддержки (TSS), в общем случае это Пользователь (User). Данный вариант использования имеет несколько ветвей, которые представлены в Таблице 4.2.

Таблица 4.2 — 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 кейс. Главными действующими лицами системы является заказчик (Customer), в общем случае это базовый

класс Пользователь (User). Он также имеет несколько ветвей, представленных в Таблице 4.3

Таблица 4.3 — Описание ветвей в Варианте использования
”Основной режим обучения”

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:ProvideClarification	Response , если система не может формализовать запрос, либо нашлось множество решений, то система запрашивает пользователя детали
communication:ProvideConfirmation	Response , когда система нашла решение, она запрашивает пользователя подтверждение о том, что искомое решение решило его проблему

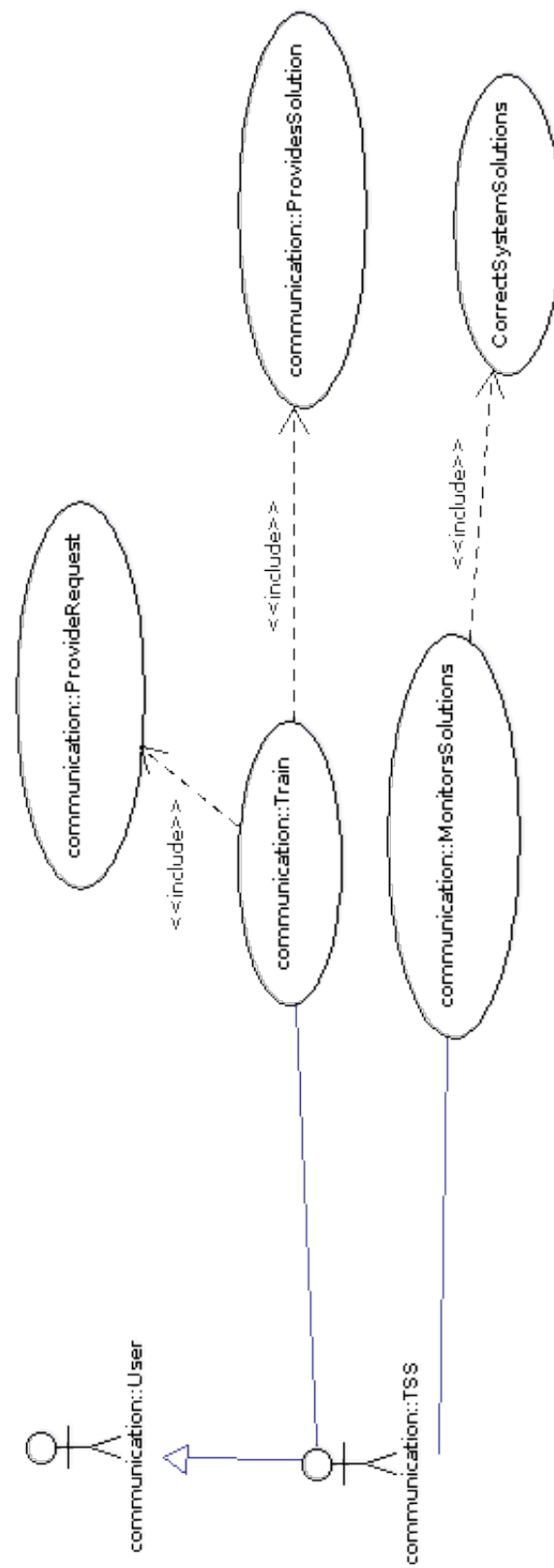


Рисунок 4.1 — Вариант использования. Обучение.

4.1.1 Компоненты системы

На Рисунке 4.2 представлена диаграмма компонентов системы. Взаимодействие компонентов системы показано на рисунке 4.3. Пользователь взаимодействует с системой посредством компонента WebService 4.1.2. Взаимодействие происходит по следующей схеме.

1. WebService получает запрос пользователя. Сохраняет запрос в Базу Знаний (Базу данных) 2.
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

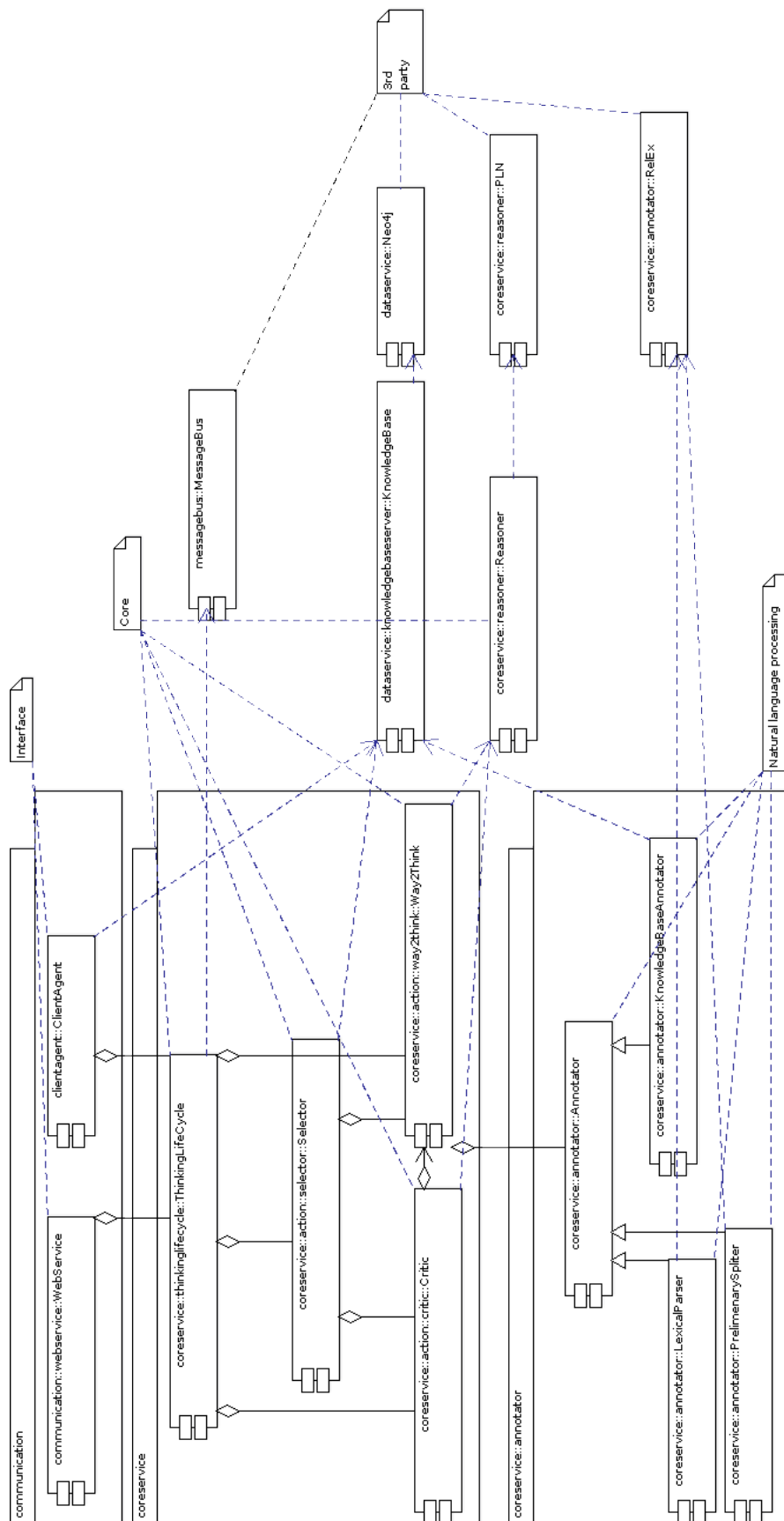


Рисунок 4.2 — Диграма компонентов системы

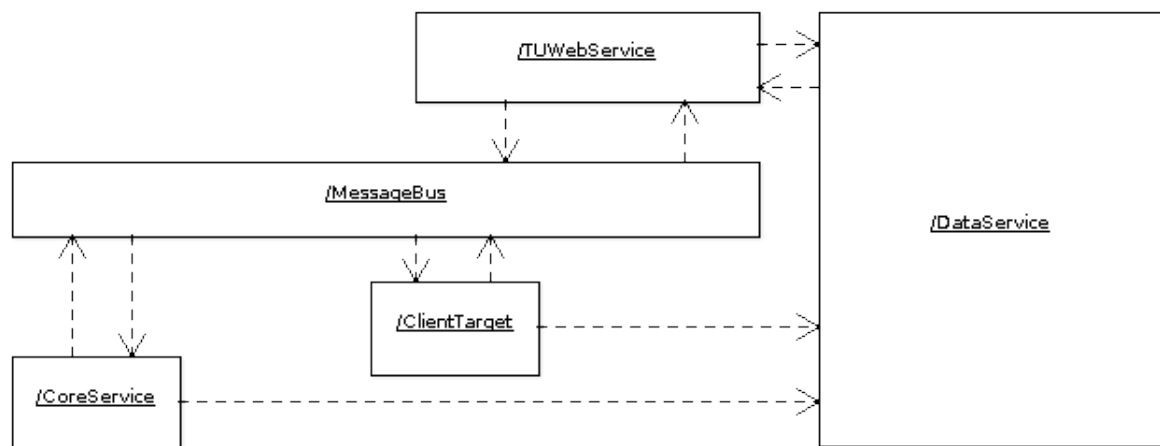


Рисунок 4.3 — Диграма взаимодействия компонентов

4.1.2 Компонент WebService

Данный компонент обрабатывает запросы пользователей. Запрос пользователя представляется объектом Request, который содержит информацию о пользователе, а также ссылку на метод, который будет вызван, когда запрос будет обработан. Вся работа происходит в компоненте CoreService. На Рисунке 4.4 представлен интерфейс компонента. В Таблице 4.4 представлено описание методов.

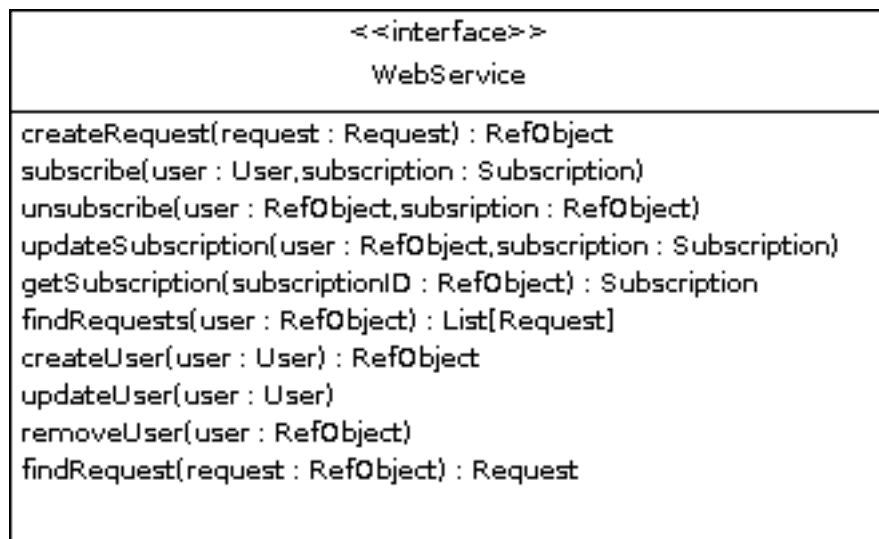


Рисунок 4.4 — Интерфейс компонента WebService

Подробное описание классов представлено в А. Основной алгоритм работы компонента:

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 на этот запрос

Таблица 4.4 — Описание методов компонента WebService

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

4.1.3 Компонент CoreService.ThinkingLifeCycle

Это основной компонент системы, ответственный непосредственно за выполнение запросов. Данный компонент управляет потоками, событиями приложения. Он запускает исполнение Критиков (Critic), Селекторов (Selector), Путь мышления (WayToThink), осуществляет обмен данных между компонентами. Компонент построен на фреймворке Akka Concurrency, который позволяет разрабатывать приложения, которые могут работать параллельно [29].

В данном компоненте реализовано шесть уровней мышления.

1. Instinctive - Инстинктивный уровень
2. Learned - Уровень обученных реакций
3. Deliberative - Уровень рассуждений
4. Reflective - Рефлексивный уровень
5. Self-Reflective Thinking - Саморефлексивный уровень
6. Self-Conscious Reflection - Самосознательный уровень

На уровне Instinctive идет обработка сгенерированных по шаблону инцидентов. Объект, который используется для обработки использует паттерн Akka [29]. На рисунке 4.5 представлена диаграмма классов компонента. В таблице 4.5 представлено описание методов компонента.

Таблица 4.5 — 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 4.6 представлена диаграмма действий для этого метода.
Продолжение следует	

Таблица 4.5 – продолжение

Метод	Описание
apply(request : Request) : List[Action]	<p>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 4.8 представлена диаграмма действий для этого метода.</p>
apply(actions : List[Action]) : TransFrame	<p>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 4.9 представлена диаграмма действий для этого метода.</p>
processWay2Think(inputContext: Context, outputContext: Context): TransFrame	<p>Данный метод запускает обработку WayToThink 2. Данный метод создает входной контекст (InputContext), заполняет его параметрами, создает выходной контекст OutputContext. Затем он запускает обработку данных во входном контексте. На рисунке 4.10 представлена диаграмма действий для этого метода.</p>
Продолжение следует	

Таблица 4.5 – продолжение

Метод	Описание
processCritic(context: Context): List[SelectorRequestRulePair] Critic 2.	Данный метод запускает обработку Critic 2. На рисунке 4.11 представлена диаграмма действий для этого метода.
init(): Boolean	Данный метод инициализирует экземпляр класса ThinkingLifeCycle. Во время инициализации происходит Базы Знаний 2, подключения к Шине данных. На рисунке 4.12 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 4.13 представлена диаграмма действий для этого метода.

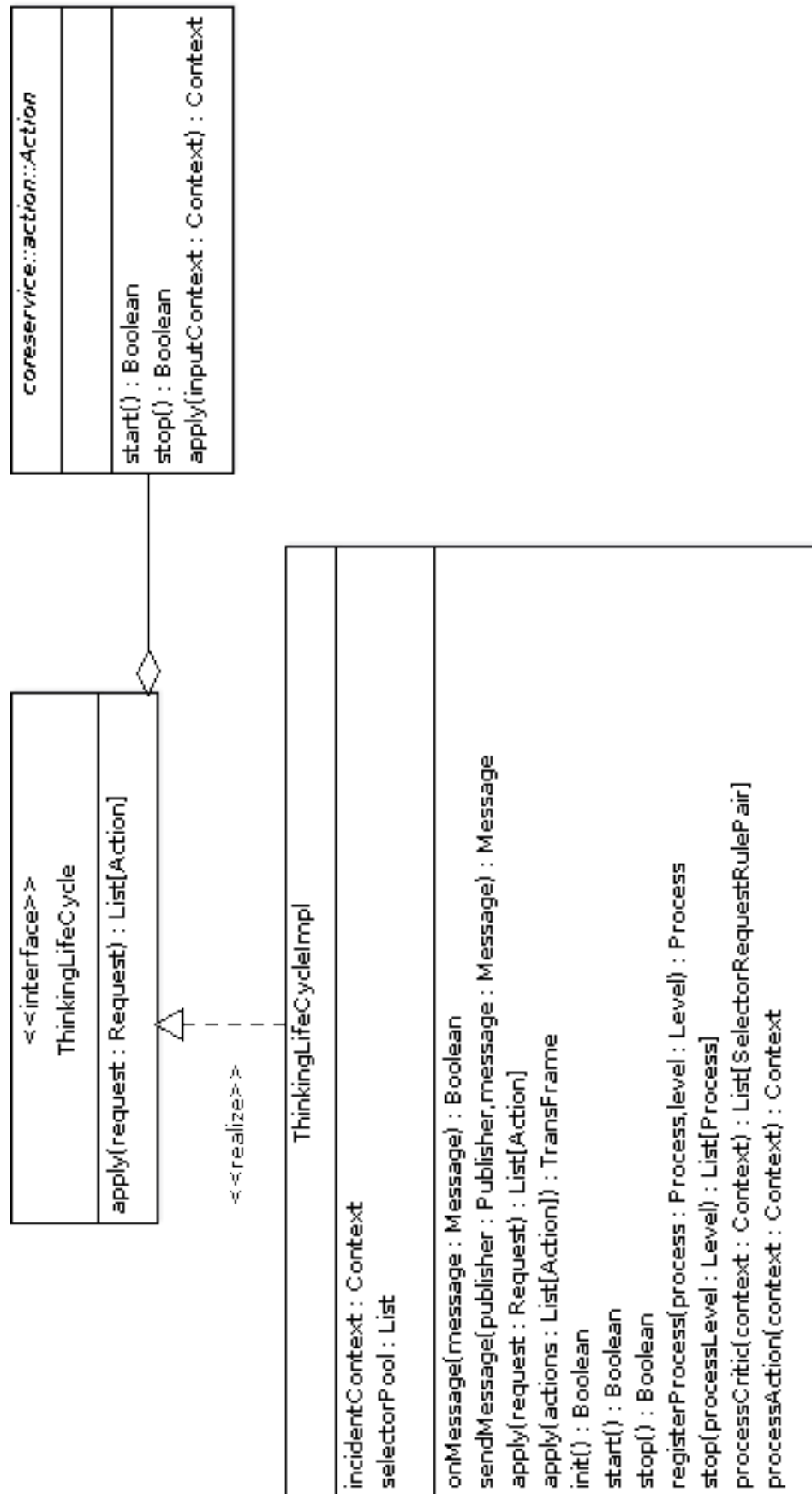


Рисунок 4.5 — Диаграмма классов ThinkingLifeCycle

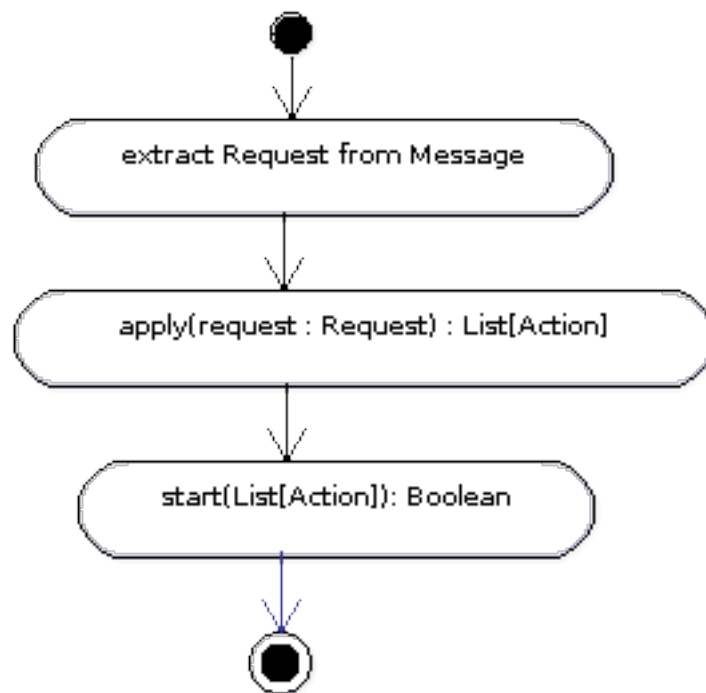


Рисунок 4.6 — Диаграмма действий метода `onMessage` компонента `ThinkingLifeCycle`

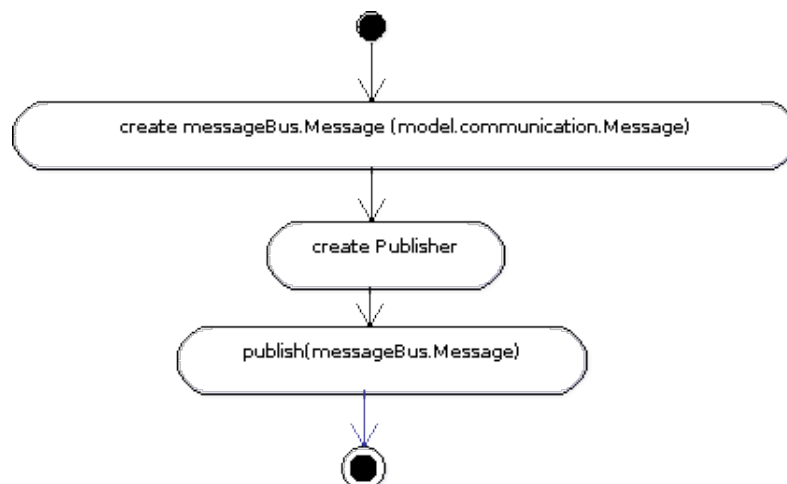


Рисунок 4.7 — Диаграмма действий метода `sendMessage` компонента `ThinkingLifeCycle`

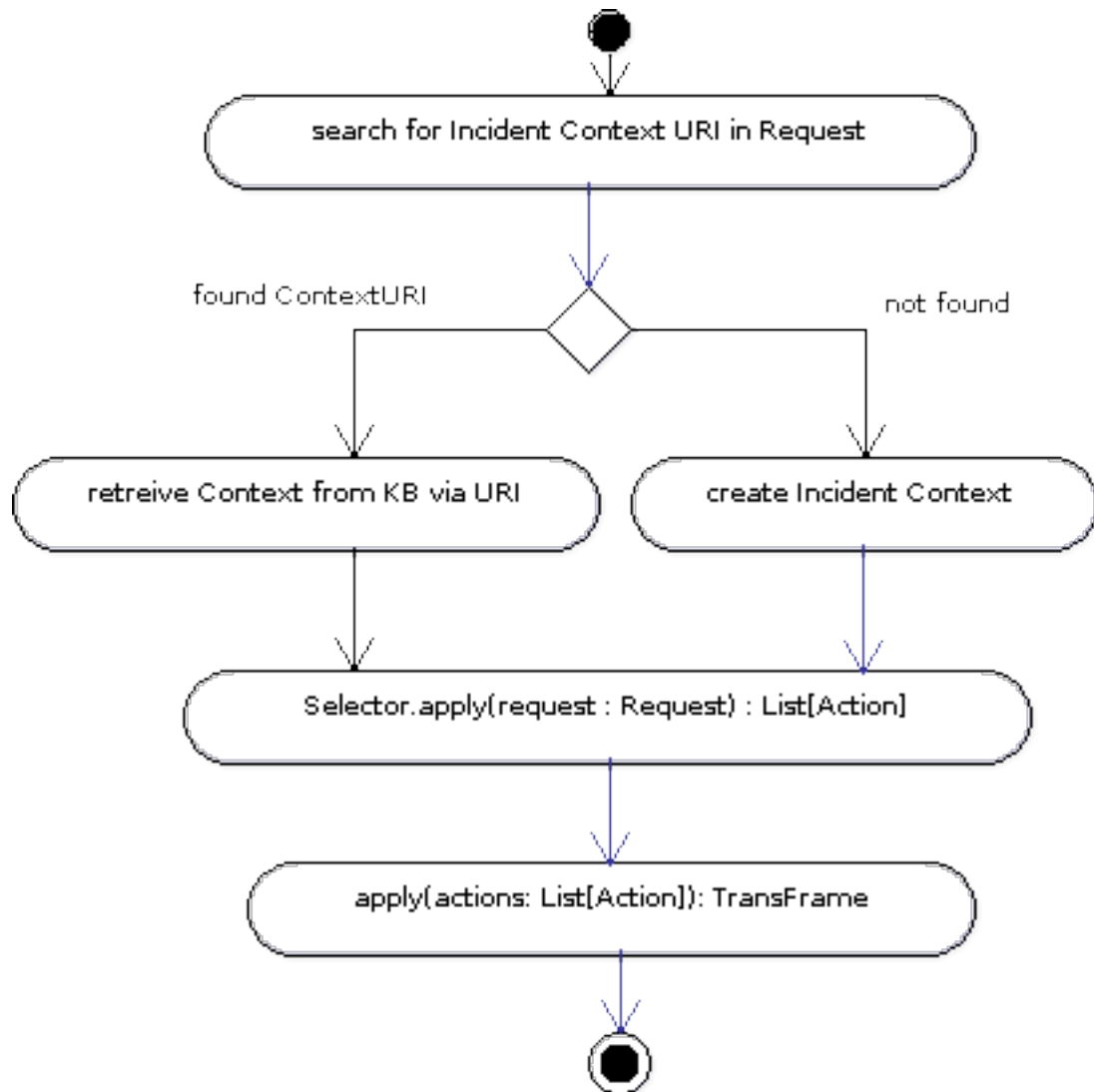


Рисунок 4.8 — Диаграмма действий метода `apply` компонента `ThinkingLifeCycle`

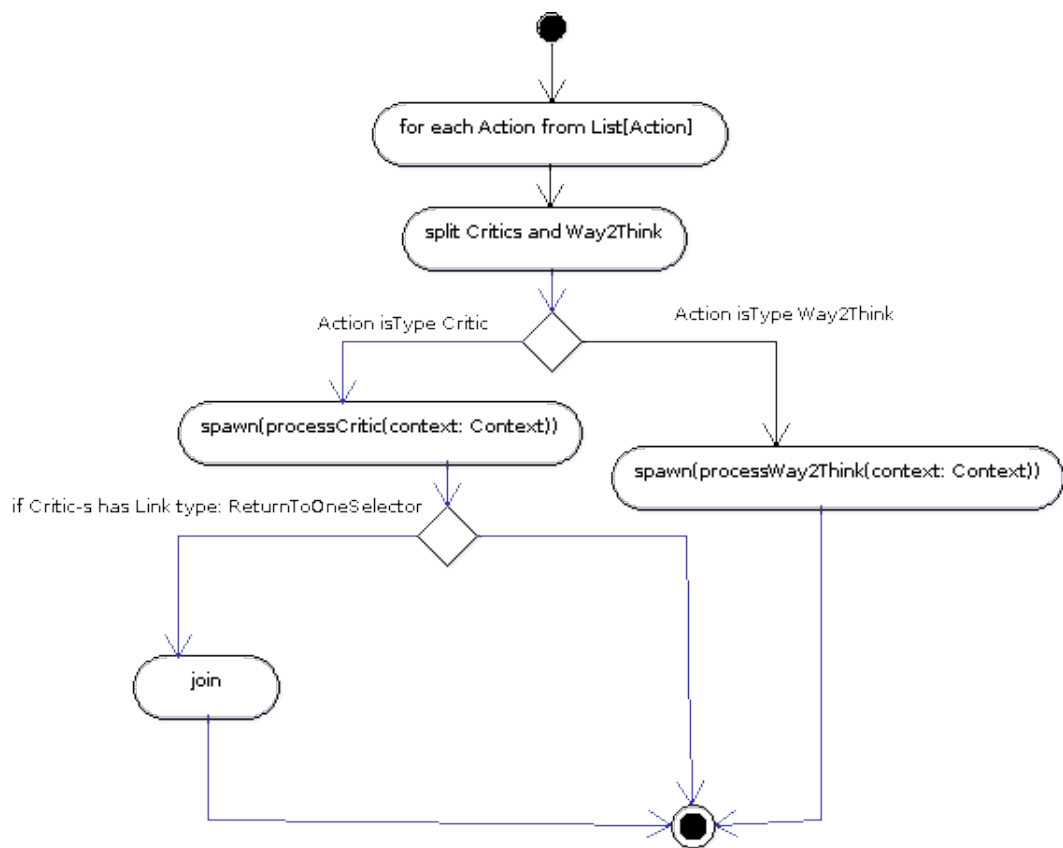


Рисунок 4.9 — Диаграмма действий метода `apply` компонента `ThinkingLifeCycle`

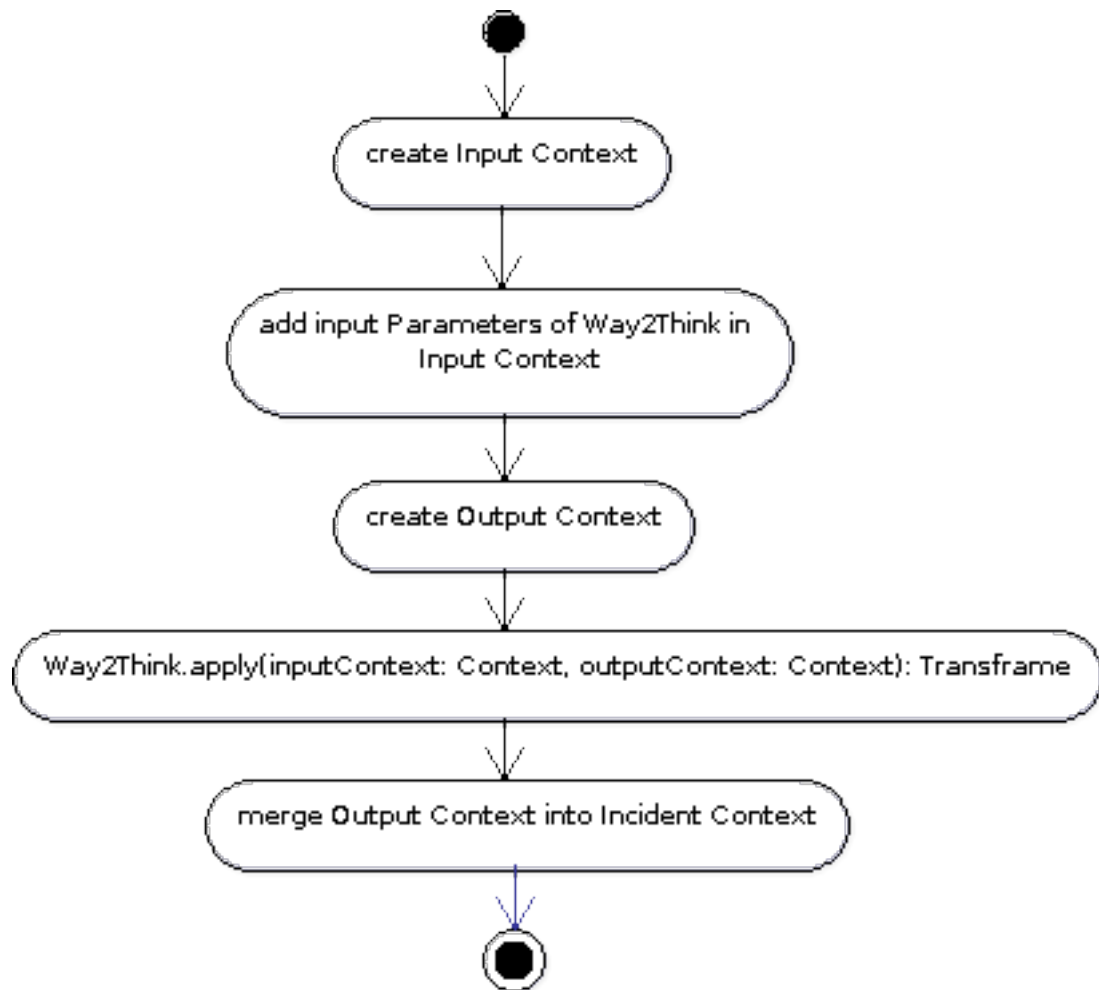


Рисунок 4.10 — Диаграмма действий метода `processWay2Think` компонента `ThinkingLifeCycle`

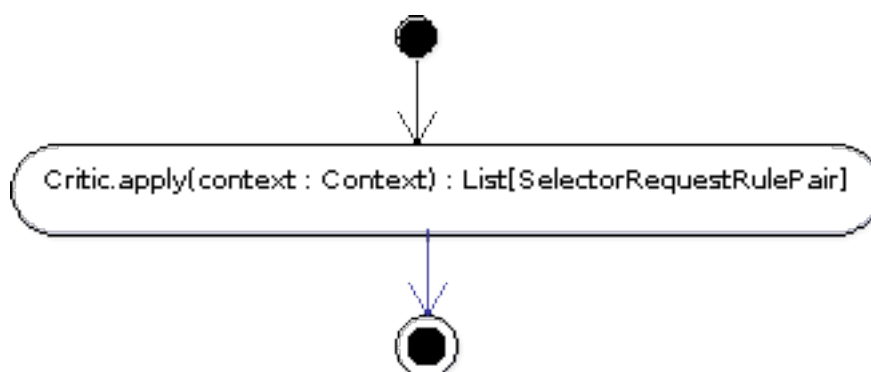


Рисунок 4.11 — Диаграмма действий метода `processCritic` компонента `ThinkingLifeCycle`

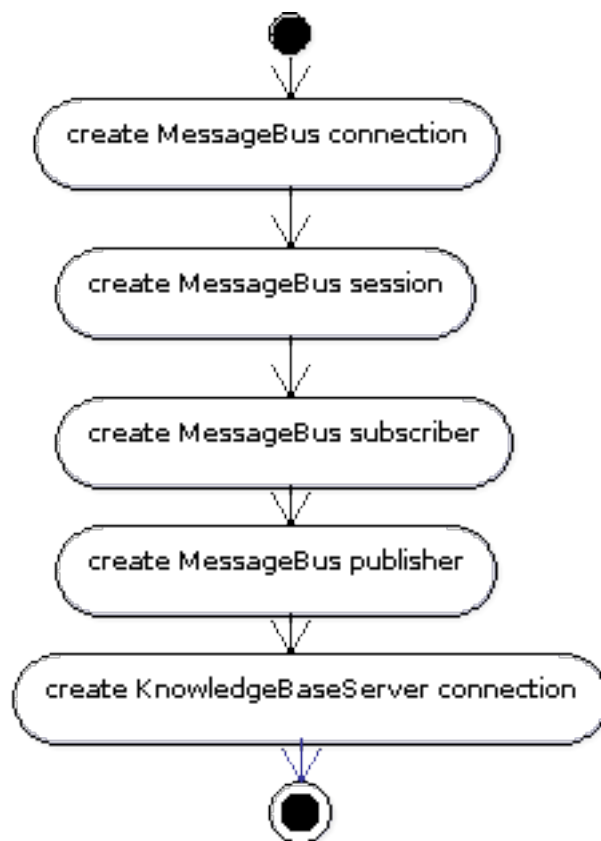


Рисунок 4.12 — Диаграмма действий метода `init` компонента `ThinkingLifeCycle`

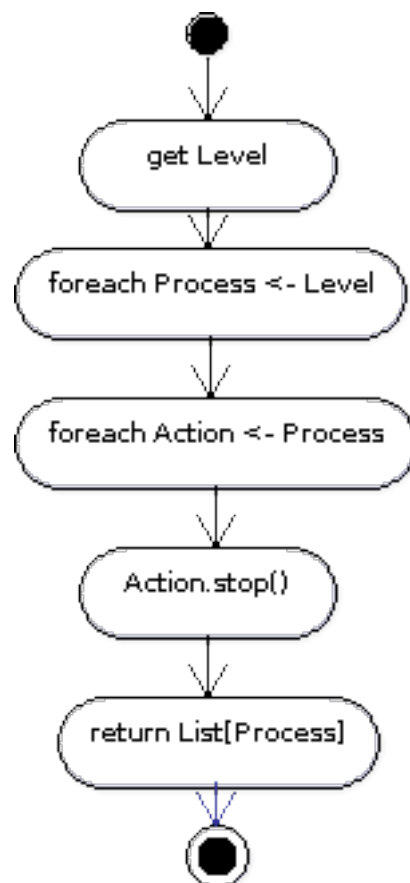


Рисунок 4.13 — Диаграмма действий метода `stop` компонента `ThinkingLifeCycle`

Описание работы компонента

Запуск и остановка

1. Когда приложение стартует оно инициализирует ThinkingLifeCycle, который активирует набор критиков, базируясь на текущей цели системы. Например, цель-классифицировать инцидент, активируется набор критиков: разобрать, проверить, найти категорию.
2. Когда приложение останавливается - оно останавливает все объекты класса и подклассов Actions (Critics, WayToThink), Selectors и ThinkingLifeCycle.

Коммуникация происходит посредством сообщений, отправленных через MessageBus (Шину Данных) [2](#) JMS [\[30\]](#). *Взаимодействие компонента с другими компонентами*

1. Критик возвращает список Селекторов (SelectorRequestRule)
 - (a) ThinkingLifeCycle запускает обработку компонента Selector
 - (b) Selector возвращает список Action **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
 - iii. Если Action это Critic с ссылками ReturnToSameSelector ThinkingLifeCycle ждет результаты и отправляет список SelectorRequestRule, возвращенные Critic новому Selector. Иными словами Critic может вернуть новый Selector. В данном случае нам нужно провести операцию Join для всех потоков [\[31\]](#). 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 2 запускает WayToThink
- iv. TLC сохраняет параметры в OutputContext
- v. TLC сохраняет итоговый результат работы и возвращает его

4.1.4 Компонент CoreService.Selector

Selector (Селектор) это компонент, который ответственен за получение списка действий из Базы знаний, согласно входным параметрам. **Входной критерий.** TLC запускает Selector с параметрами. **Выходной критерий.** Selector получает список Action: WayToThink или Critic.

На Рисунке 4.14 показан интерфейс компонента. В Таблице 4.6 приведено

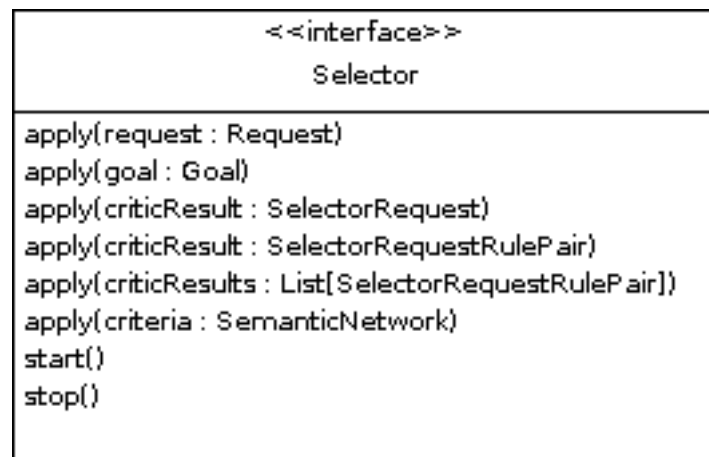


Рисунок 4.14 — Интерфейс компонента Selector

описание методов компонента.

Таблица 4.6 — Описание методов класса (компонента) Selector

Метод	Описание
<code>apply(request : Request) : Action</code>	Данный метод на основе запроса пользователя получает из Базы знаний необходимые Critic 4.1.5. На рисунке 4.15 представлена диаграмма действий для этого метода.
Продолжение следует	

Таблица 4.6 – продолжение

Метод	Описание
apply(goal: Goal) : Action	Данный метод на основе цели системы получает из Базы знаний необходимые Critic 4.1.5. На рисунке 4.16 представлена диаграмма действий для этого метода.
apply(criticResult : ActionProbabilityRule) : Action	Данный метод на основе работы Critic получает из Базы знаний необходимые Action ??. На рисунке 4.17 представлена диаграмма действий для этого метода.

Описание работы компонента

Действия при классификации инцидента

1. TLC 4.1.3 запускает входящие Critic 4.1.5 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 Refelective уровне мышления сработал Critic, который посчитал, что данное решение некорректно или же пользователь признал его таким

На Рисунке 4.18 представлена диаграмма действий выбора наиболее вероятного WayToThink. На Рисунке 4.19 представлена диаграмма действий классификации инцидента. TLC 4.1.3 получает цель Классифицировать инцидент, затем Selector по этой цели возвращает необходимые Critic. Затем TLC запускает обработку Critic в разных потоках (параллельно). В данном случае рассматривает 3 Critic.

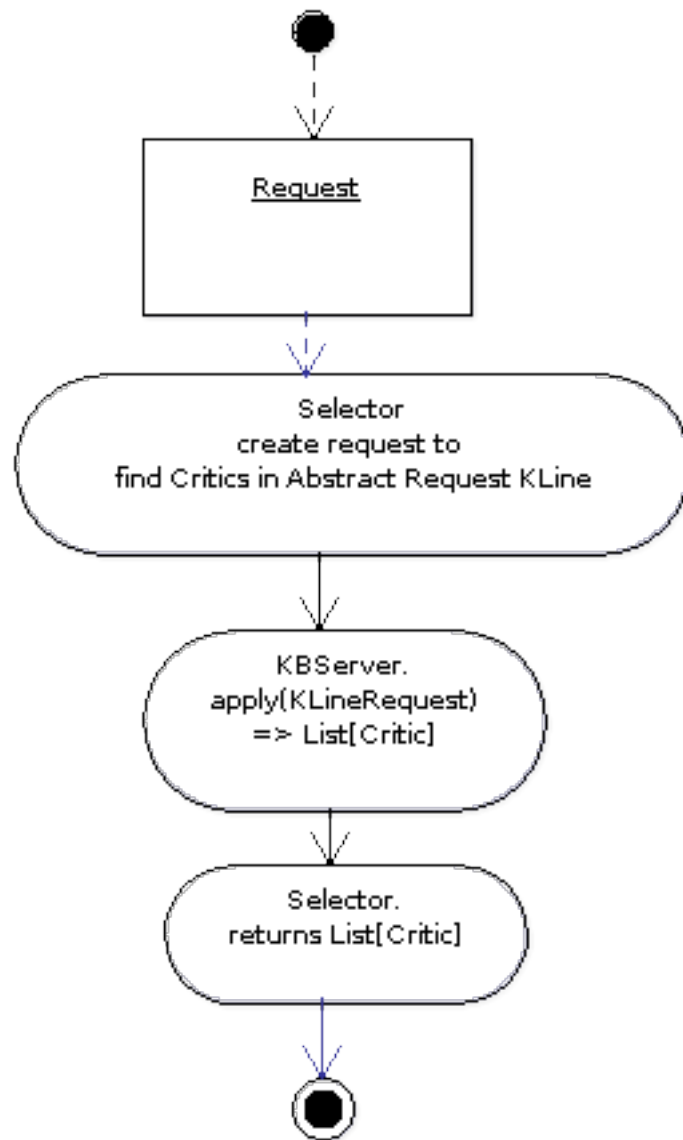


Рисунок 4.15 — Диаграмма действий метода `Selector.apply(request : Request)` компонента `Selector`

- `DirectInstruction` - прямые инструкции, данный `Critic` возвращает `WayToThink Simulate` 4.1.6, который ищет связь между концепциями в запросе и концепциями в Базе Знаний.
- `ProblemWithDesiredState` - проблема с ожидаемым результатом, данный `Critic` возвращает `Simulate+Reformulate WayToThink`, которые ищут сопоставление концепциями в Базе Знаний и пытается преобразовать запрос к `DirectInstruction` запросу (прямым инструкциям).

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

Затем TLC собирает результаты выполнения всех Critic и запускает их по новой, пока не будет достигнута изначальная цель.

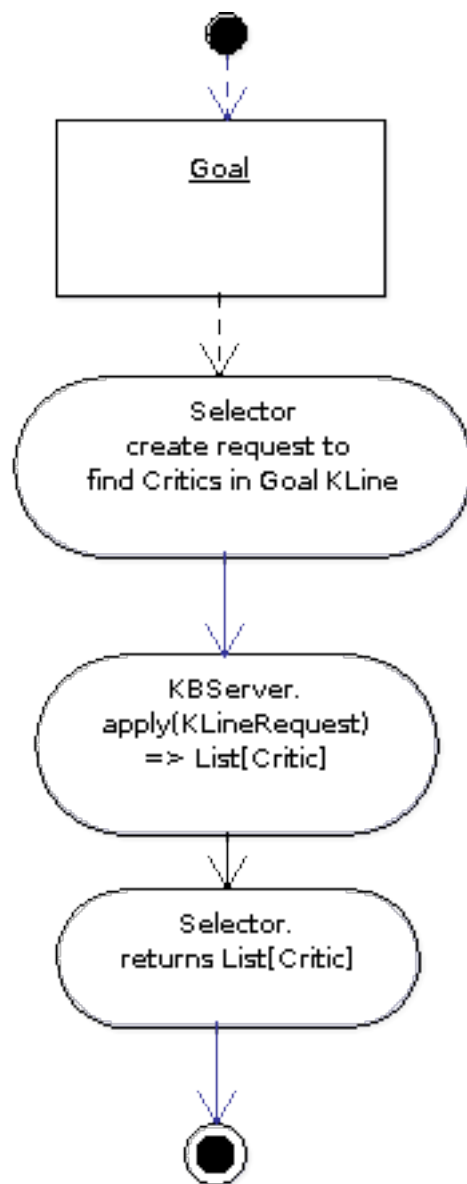


Рисунок 4.16 — Диаграмма действий метода `Selector.apply(goal: Goal)` компонента `Selector`

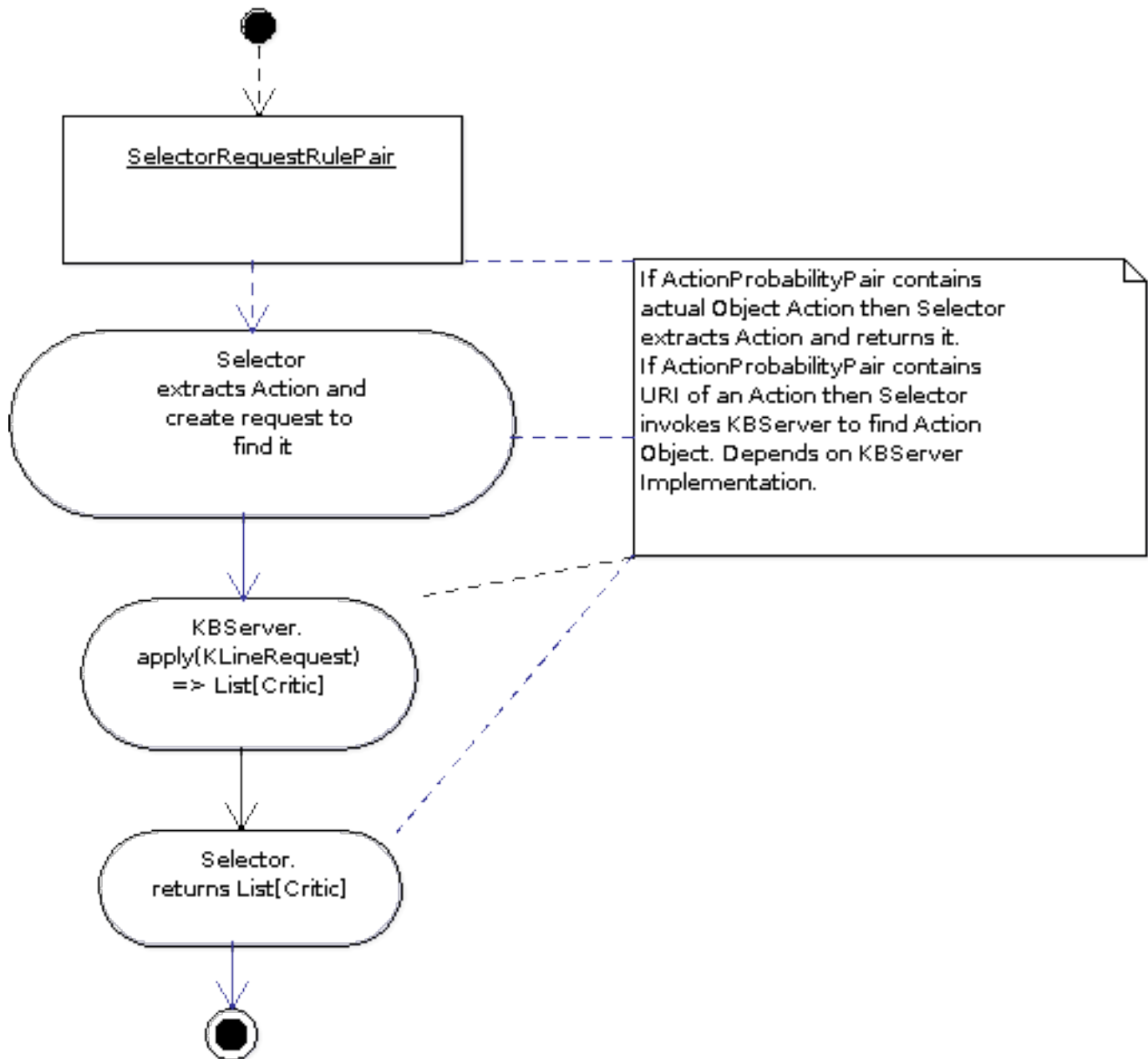


Рисунок 4.17 — Диаграмма действий метода `Selector.apply(criticResult : ActionProbabilityRule)` компонента `Selector`

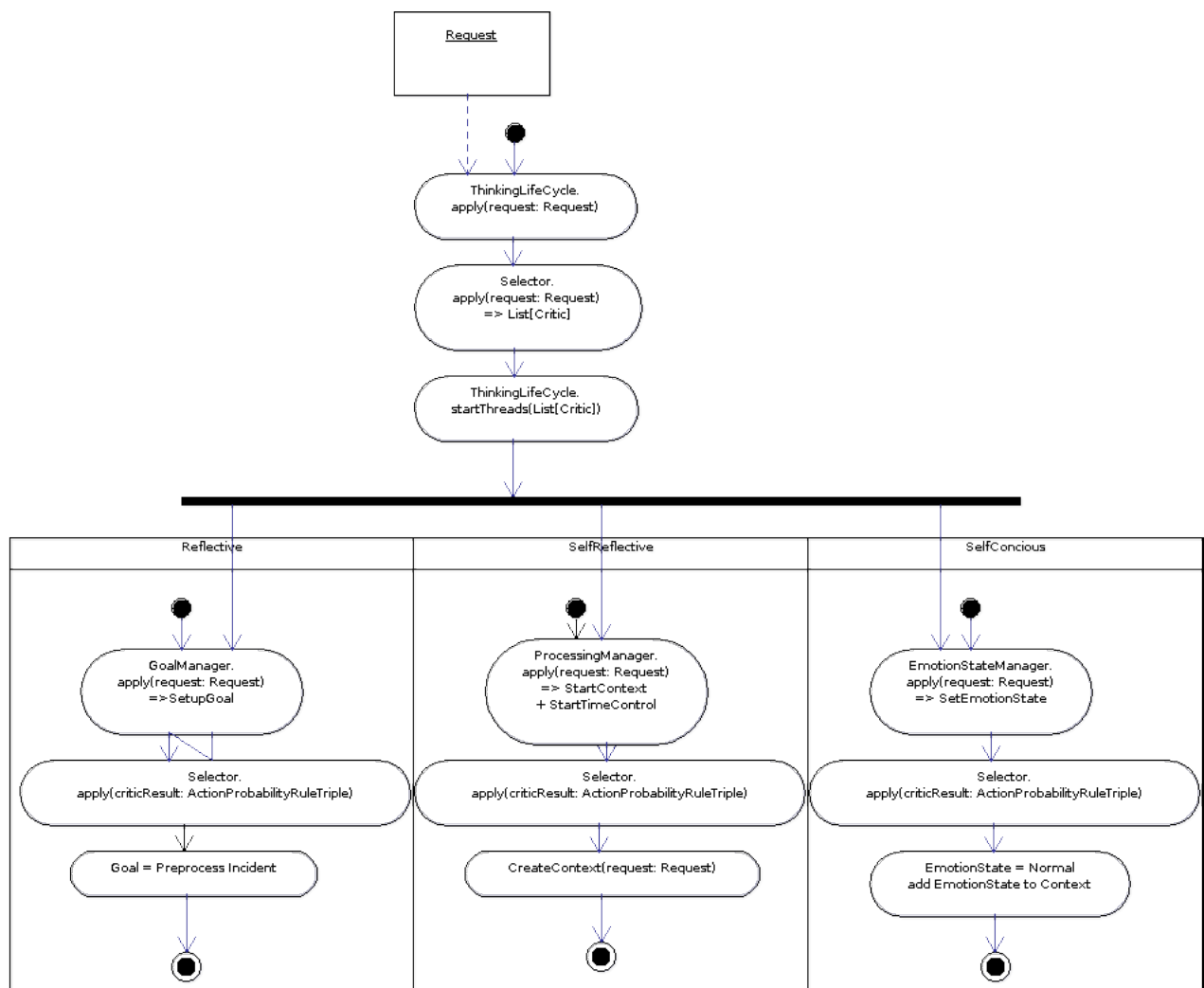


Рисунок 4.18 — Диаграмма действий выбора WayToThink

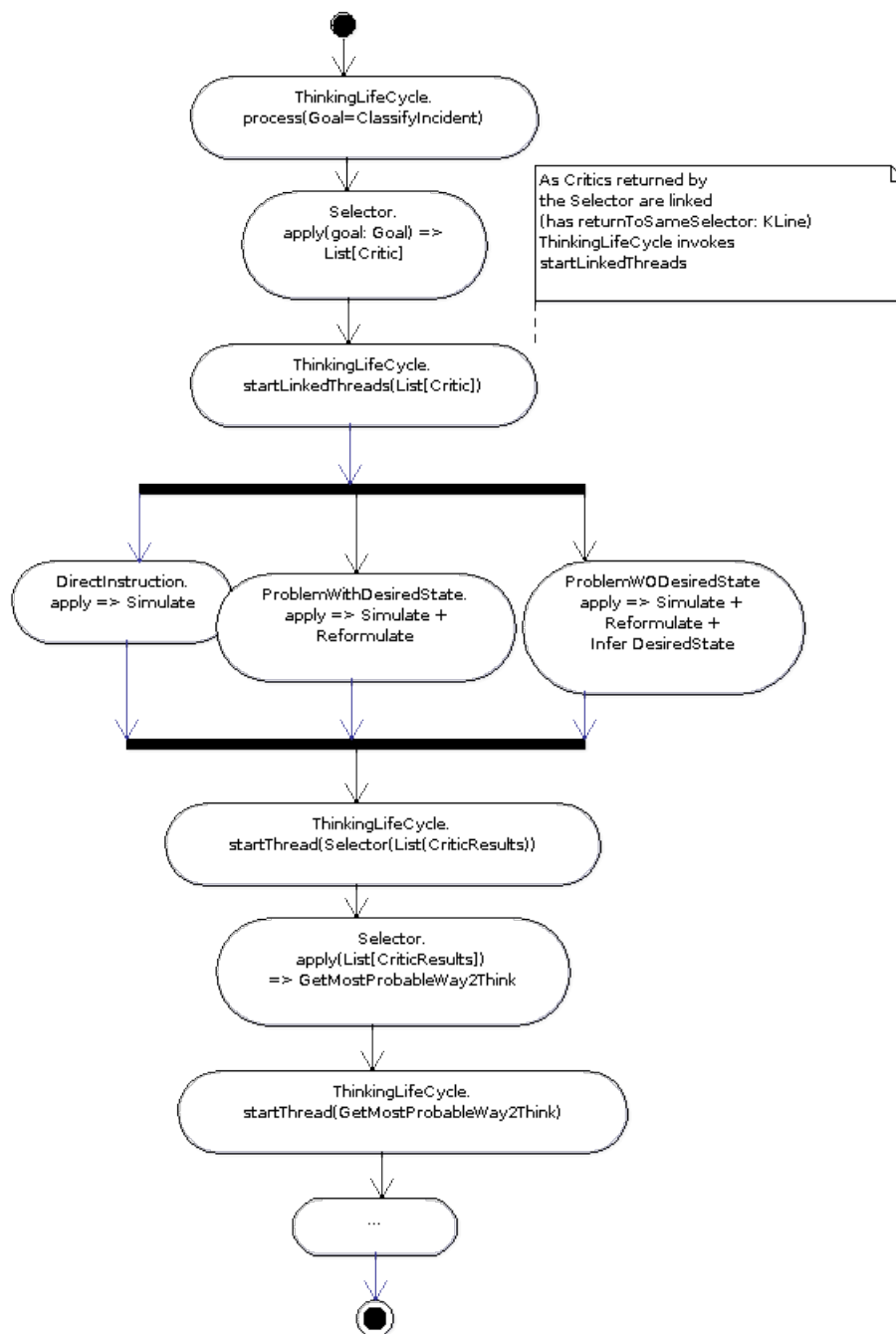


Рисунок 4.19 — Диаграмма действий классификации инцидента

4.1.5 Компонент CoreService.Critics

Critic является основным компонентом для анализа в триплете Critic->Selector->WayToThink. Critic используется для классификации входной информации, рефлексии, само-анализа и т.д. и служит определенным вероятностным триггером.

Входной критерий

TLC 4.1.3 запускает Critic согласно Goal В (Цель) или Request.

Выходной критерий

Critic генерирует SelectorRequest 4.1.4. На входе Critic принимает:

- Загруженные из базы правила для работы Critic (CriticRules)
- DomainModel:SemanticNetwork - доменная модель, представляющая собой семантическую сеть
- Описание инцидента, представляющая собой семантическую сеть

На выходе Critic предоставляет следующую информацию:

- SelectorRequest 4.1.4 - запрос на выбор Selector из базы знаний
- CriticRules - правило, которое сработало для активации. Данное правило является логическим предикатом

На Рисунку 4.20 представлена диаграмма действий Critic. В Таблице 4.7 приведено описание основных классов Critic. В Таблице 4.8 представлено описание методов компонента Critic.

Таблица 4.7 — Описание основных классов Critic, используемых в системе

Critic	Описание
Manager	Простой тип критика, который работает как триггер Goal В, чтобы запустить необходимый WayToThink.
Control	контролирующий Critic, который ждет определенного события (срабатывает на определенное событие). Например, заканчивается, отведенное на решение время.
Продолжение следует	

Таблица 4.7 – продолжение

Critic	Описание
Analyser	Анализатор, обрабатывает и выявляет тип инцидента. Например, прямые инструкции, проблема с желаемым состоянием, наиболее вероятное действие.

Таблица 4.8 — Описание методов компонента Critic

Метод	Описание
exclude():List[CriticLink]	Данный метод возвращает список CriticLink, которые при срабатывании данного Critic будут игнорироваться с определенной вероятностью. После срабатывания Critic, будет посчитана суммарная вероятность активации. После чего система решит, какой Critic был в действительности активирован.
include():List[Critic]	Данный метод возвращает список CriticLink, которые при срабатывании данного Critic будут включаться с определенной вероятностью.
apply(currentSituation:SemanticNetwork, domainModel:SemanticNetwork): List[SelectorRequestRulePair]	Данный метод запускает Critic, после чего вернется список Selector 4.1.4 с определенной вероятностью, после чего TLC 4.1.3 их активирует.

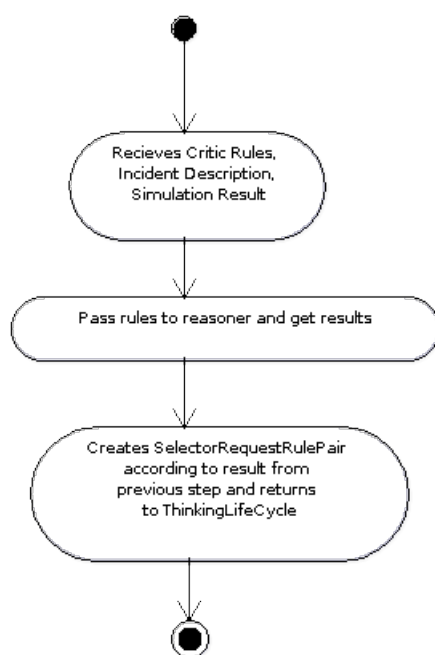


Рисунок 4.20 — Диаграмма действий компонента Critic

Основные примеры 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

4.1.6 Компонент CoreService.WayToThink

WayToThink является основным операционным компонентом триплета Critic->Selector->WayToThink. Основными задачами данного компонента являются: обновление, преобразование, сохранение данных и коммуникация с пользователем.

Входной критерий

Запуска из компонента ThinkingLufeCycle 4.1.3. Входными данными является InputContext, который содержит параметры WayToThink.

Выходной критерий

WayToThink завершил работу. На выходе возвращается измененные данные в ходе работе.

На Рисунке 4.21 представлен интерфейс компонента.

В общем виде компонент описывает последовательность действий. В системе

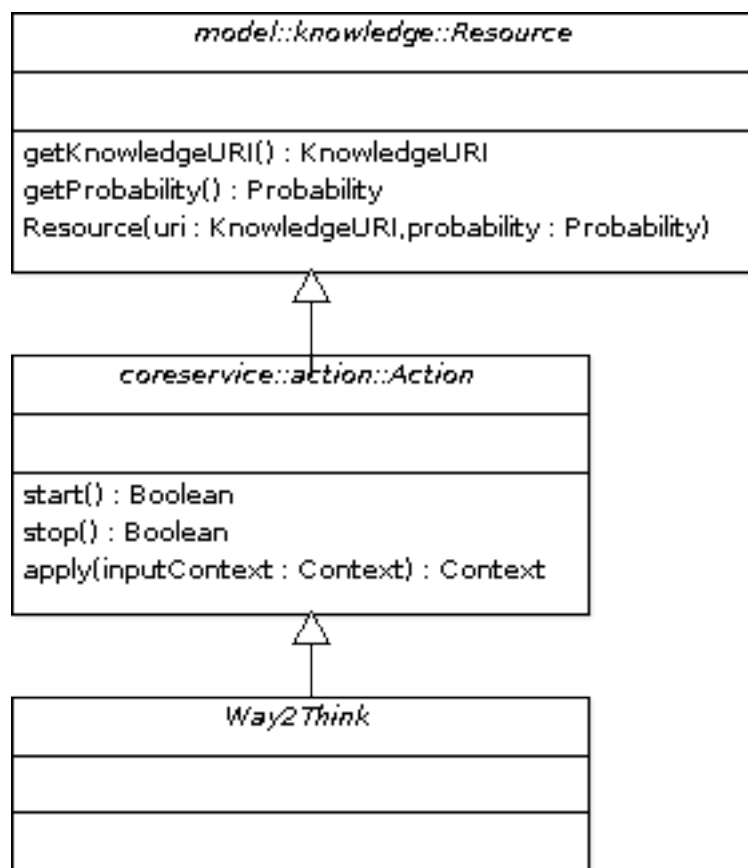


Рисунок 4.21 — Интерфейс компонента WayToThink

используется два больших класса WayToThink простой и составной (сложный). Простые WayToThink являются встроенными в систему, остальные являются комбинацией компонентов: Critic 4.1.5, Selector 4.1.4, WayToThink 4.1.6. В Таблице 4.9 приведено описание встроенных в систему WayToThink. В Таблице 4.10 представлено описание методов WayToThink.

Таблица 4.9 — 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 В.
Разделить фразу на слова и предложения	Данный WayToThink разбивает фразу на слова и возвращает список слов.
Найти связи между входной информацией и базой знаний	Данный WayToThink ищет связь между входной информацией и базой знаний.
Извлечь связи	Данный WayToThink возвращает список связей из фразы.
Сохранить наиболее вероятное решение	Данный WayToThink сохраняет наиболее вероятное решение.
Перефразировать (Reformulate)	Данный WayToThink ищет связь между текущим контекстом и известными проблемами, если есть неизвестные концепции, то он пытается их переформулировать при помощи пользователя.
Продолжение следует	

Таблица 4.9 – продолжение

WayToThink	Описание
Смоделировать (Simulate)	Данный WayToThink ищет связь между текущим контекстом и проблемами уже сохраненными в базе.
Найти решение	Данный WayToThink производит поиск решения, которое прикреплено к проблеме, которая была найдена при помощи моделирования и перефразирования.
Остановить работу	Данный WayToThink останавливает работу системы.

Таблица 4.10 — 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 Решение (HowTo) **Г**, то есть описывает последовательность действий, необходимых для решения проблемы.

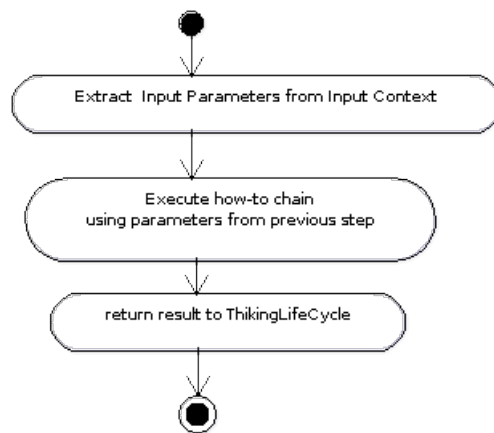


Рисунок 4.22 — Работа компонента WayToThink в режиме Рецепта решения (HowTo)

4.1.7 Компонент CoreService.PreliminaryAnnotator

Данный компонент проводит предварительную подготовку текста: грамматическую и орфографическую коррекцию текста, а также разделение на предложения. На Рисунке 4.23 представлен интерфейс компонента. Компонент также является WayToThink, так как он производит модификацию данных контекста. В Таблице 4.11 приведено описание методов класса.

<<interface>>
PreliminaryAnnotator
apply(incidentDescription : String) : Narrative

Рисунок 4.23 — Интерфейс компонента PreliminaryAnnotator

Таблица 4.11 — Описание методов компонента PreliminaryAnnotator

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

4.1.8 Компонент CoreService.KnowledgeBaseAnnotator

Данный компонент устанавливает связи между терминами во входной фразе и базой знаний. Данный компонент также является WayToThink 4.1.6.

Входными критериями является список фраз.

Выходными критериями является список ссылок на внутренние термины.

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 входящим 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.

4.1.9 Компонент DataService

Данный компонент отвечает за хранение данных в системе. База знаний построена на графах. На Рисунке 4.24 представлен интерфейс компонента. В базе знаний используется два типа объектов Object - объект базы знаний, BusinessObject - объект для Web Service (User, Request). BusinessObject является кортежем для интеграции с внешними системами. У объекта есть ID, который уникально удостоверяет его в рамках системы. В Таблице 4.12 приведено описание методов компонента.

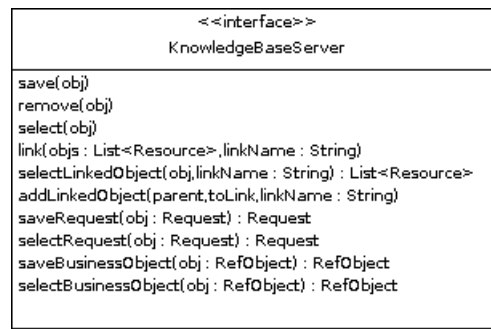


Рисунок 4.24 — Интерфейс компонента KnowledgeBaseServer

Таблица 4.12 — 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 4.12 – продолжение

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

4.1.10 Компонент ClientAgent

Данный компонент предназначен для выполнения решений на конечной машине. Данный компонент должен поддерживать обработку (см. Приложение Г). В случае проблем компонент также должен обращаться за помощью к специалисту.

4.1.11 Компонент Reasoner

Данный компонент осуществляет логические вычисления для системы, например, для обработки правил в компоненте Critic 4.1.5. На Рисунке 4.25 представлен интерфейс компонента. В Таблице 4.13 приведено описание методов компонента.

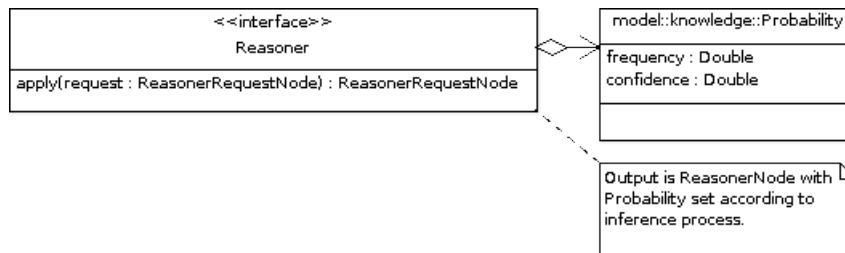


Рисунок 4.25 — Интерфейс компонента Reasoner

Таблица 4.13 — Описание методов компонента Reasoner

Метод	Описание
<code>apply(request : ReasonerRequestNode) : ReasonerRequestNode</code>	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 [32] и NARS [23].

4.2 Модель данных системы - TU Knowledge

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 [21] обрел широкое использование во многих схемах данных, так как давал возможность дополнительного расширенного описания взаимосвязи между данными. На Рисунке 4.26 представлена схема данных TU Knowledge. В Таблице 4.14 представлено описание схемы TU Knowledge.

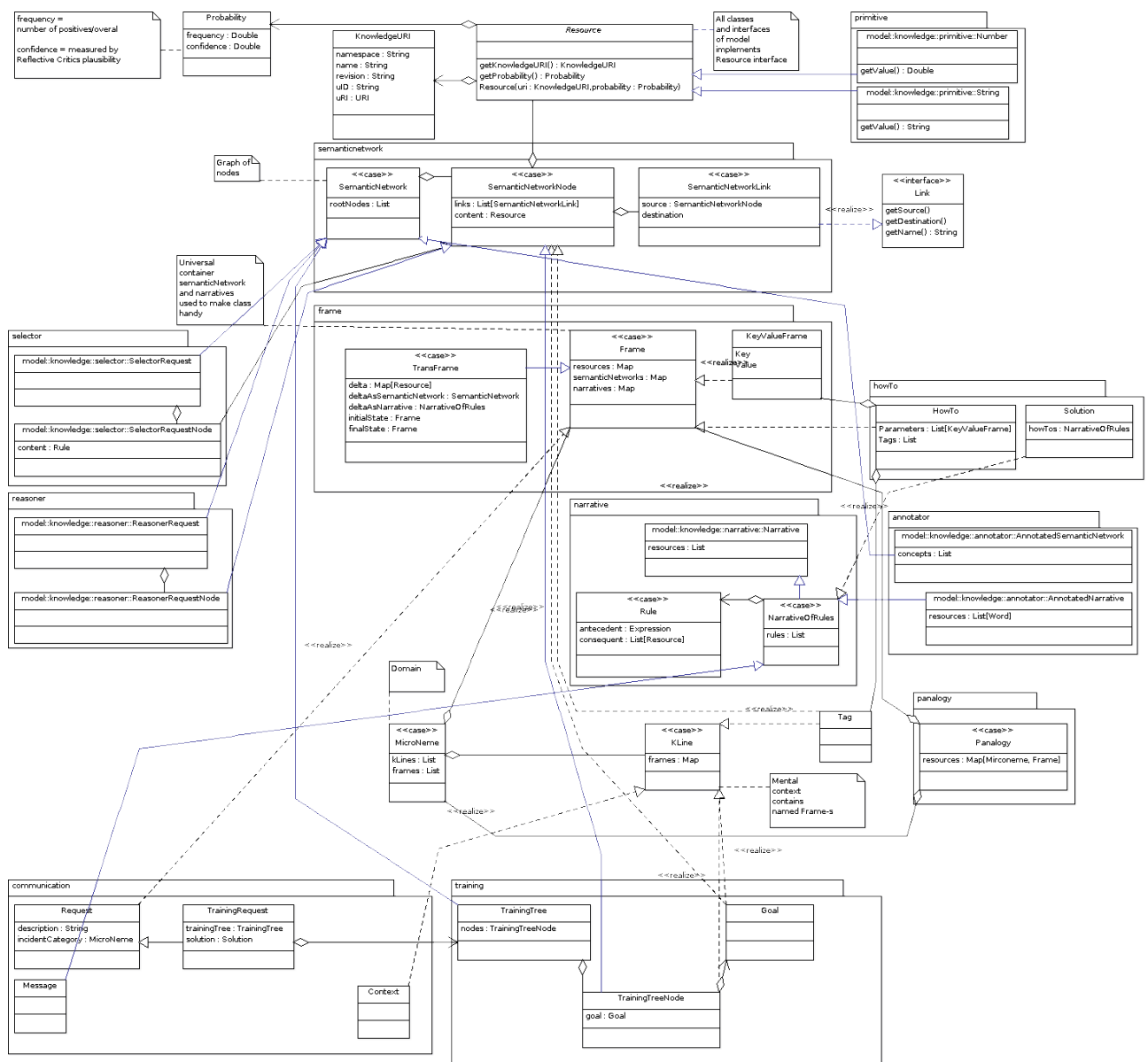


Рисунок 4.26 — Схема данных TU Knowledge в формате UML

Таблица 4.14 — Описание классов TUKnowledge

Класс	Описание
Knowledge	Базовый класс всех объектов модели. Содержит в себе URI, по которому уникально идентифицируется. Поддерживает версию. Свойствами данного объекта обладают все объекты системы. Также содержит Probability (Вероятность) и Confidence (Уверенность) поля. Например, когда в результате работы WayToThink получается Knowledge он имеет Confidence 0, так как он только что был сгенерирован, когда его проверит Critic на его состоятельность при помощи определенных в Critic правил, то он поставит ему не 0 Confidence.
Narrative	Список слов исходного запроса.
Rule	Правило. Класс описывающий правила в системы. Например, правило по которому работает Critic 4.1.5 .
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.
Frame	Коллекция объектов Knowledge, с возможностью проставление специального тега для семантической группировки.
TransFrame	Коллекция Frame, содержащая два состояния одного фрейма: до и после.
Продолжение следует	

Таблица 4.14 – продолжение

Класс	Описание
Goal	Цель. Приложение В.
Tag	Тоже что и цель, но использующиеся для меток.
Preliminary annotation	SemanticNetwork входного запроса.
KnowledgeBase annotation	SemanticNetwork с сопоставлением концепциям Базы Знаний.
Domain model	SemanticNetwork доменной модели.
Situation model	SemanticNetwork, часть DomainModel, созданной для обработки текущего запроса. Приложение В.
Incident	SemanticNetwork входного запроса к системе.
K-Line	Связь между объектами. Например, когда в систему поступает запрос она создает K-Line между Conversation, Narrative.
Conversation	SemanticNetwork, контекст инцидента.
InboundRequest	SemanticNetwork входного запроса.
Training Request	SemanticNetwork входного запроса для обучений.

4.3 Прототип

В прототипе были реализованы 4 уровня мышления. Основной рабочий поток приложения описан следующим алгоритмом:

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. PreliminaryAnnotator 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 4.1.3
 - (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 текущей 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.

4.3.1 UML диаграмма действий приложения

На Рисунке 4.27 представлена UML диаграмма действий системы.

4.3.2 Технологии прототипа

Прототип был написан на языке Scala, с применением технологии Akka - параллельного исполнения на множестве ядер. В качестве базы используется Neo4j графовая база данных.

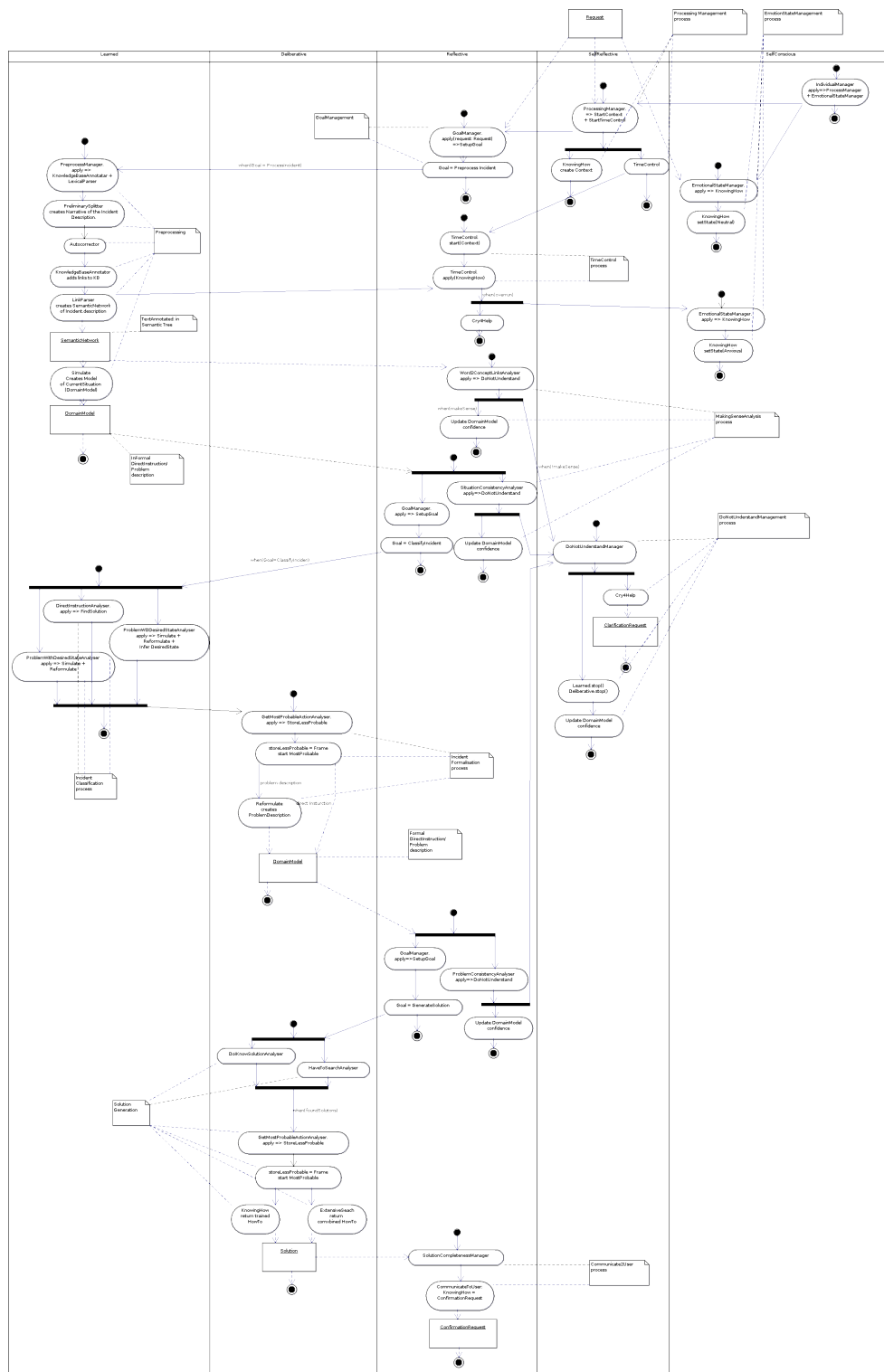


Рисунок 4.27 — Диаграмма действий LifecycleActivity

4.4 Апробация прототипа

4.4.1 Экспериментальные данные

В качестве экспериментальных данных были взяты выгрузки данных из информационных систем из главы 1.1. Для обучения на базовом уровне в систему заложено две базовые концепции:

- Object - объект. Базовая концепция для всех объектов.
- Action - действие. Базовая концепция для всех действий.

В таблице 4.15 представлен список основных тренировочных данных.

Таблица 4.15 — 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 установке 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.
Продолжение следует	

Таблица 4.15 – 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**

4.4.2 Верификация

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 доказательства жизнеспособности решения производилась верификация в 2 этапа:

- Этап 1. Разбор входящего запроса на естественном языке и вычленение концепции
- Этап 2. Обработка по разработанной архитектуре и реализации модели мышления

Для Этапа 1 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 предыдущей главе. На них запускался программный комплекс и анализировались результаты. Удалось добиться 95%. Успешным считался инцидент, который был успешно сопоставлен концепциям в Базе Знаний.

4.5 Выводы по главе

В данной главе были представлены основные результаты работы:

- Теоретико-множественный и теоретико-информационный анализ сложных систем в области поддержки информационной инфраструктуры
- Проблемно-ориентированная система управления, принятия решений и оптимизации технических объектов в области обслуживания IT
- Архитектура системы, ее реализация и испытания на модельных данных

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 4.16. Тесты были выполнены на машине Intel Core i7 1700 MHz, 8GB RAM, 256 GB SSD, FreeBSD.

Таблица 4.16 — Результаты сравнения с работой человеческого 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

ЗАКЛЮЧЕНИЕ

В работе были выполнены следующие задачи и достигнуты следующие результаты.

1. На основе анализа предметной области (поддержка информационной структуры предприятия) была выявлена потребность и возможность в автоматизации. Была построена модель предметной области. На основе модели предметной области, модели Марвина Мински была разработана модель проблемно-ориентированной системы принятия решений в области поддержки информационной структуры предприятия.
2. Испытания комплекса на модельных данных показали работоспособность модели и архитектуры.
3. Для выполнения поставленных задач был создан программный комплекс обработки, решения инцидентов и обучения на естественном языке.
4. Программный комплекс был протестирован на контрольных примерах

Представленная в данной работе модель мышления, ее архитектура и реализация является уникальной в своем роде. На момент написания это была единственная реализация модели мышления Марвина Мински.

Разработанная в рамках работы системы не является узко-специализированной. Она также подходит для других областей, где требуется поддержка принятия решений. Например, при постановке медицинского диагноза, чтобы отбросить ложные диагнозы.

Например, систему можно обучить органам человека и их взаимосвязи. Далее можно обучить каким заболеваниям подвержен тот или иной орган. Далее к каждому заболеванию добавить симптом. После этого можно делать запрос с симптомами и система выдаст список вероятных заболеваний с их вероятностью и способами их лечения.

В области диагностики проблем в машиностроении. Обучить систему узлам автомобиля, проблемам с ними связанными, признаками 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* [33] Бер-

трану 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/aksес> и была им отмечена. Работа выполнялась при помощи компании ОАО "АйСиЭл КПО ВС в рамках работы использовались и обрабатывались данные, собранные во время работы команд ICL над поддержкой информационной структуры предприятий-заказчиков.

Список литературы

1. *Тошчев 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. — Vol. 44. — 2 pp.
2. *Toshchev A. Talanov M. Krehov A. Khasianov A.* Thinking-Understanding approach in IT maintenance domain automation // *Global Journal on Technology, Vol 3 (2013): 3rd World Conference on Information Technology (WCIT-2012).* — 2013. — Т. 3. — Режим доступа: <http://www.world-education-center.org/index.php/P-ITCS/issue/view/96>.
3. *Toshchev A. Talanov M.* Thinking model and machine understanding of English primitive texts and it's application in Infrastructure as Service domain // *Proceedings of AINL-2013.* — 2013. — Режим доступа: <http://ainlconf.ru/material201303>.
4. *Toshchev A. Talanov M.* ARCHITECTURE AND REALIZATION OF INTELLECTUAL AGENT FOR AUTOMATIC INCIDENT PROCESSING USING THE ARTIFICIAL INTELLIGENCE AND SEMANTIC NETWORKS // *Ученые записки ИСГЗ 2078-6980.* — 2014. — Т. 2. — Режим доступа: <http://ainlconf.ru/material201303>.
5. *Toshchev A. Talanov M.* Computational emotional thinking and virtual neurotransmitters // *International Journal of Synthetic Emotions (IJSE).* — 2014. — Т. 06/2014.
6. *Toshchev A. Talanov M.* Appraisal, Coping and High Level Emotions Aspects of Computational Emotional Thinking // *International Journal of Synthetic Emotions (IJSE).* — 2015. — Т. 06/2015.
7. *Toshchev A.* Thinking model and machine understanding in automated user request processing // *CEUR Workshop Proceedings.* — 2014. — Т. 1297.
8. *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.

9. *Minsky Marvin*. The Emotion Machine. — Simon & Schuster, 2006.
10. *Job Super*. Super Job. Уровень зарплат IT специалистов. — web. — 2014. <http://www.it-analytics.ru/analytics/trends/66314.html> 10.11-2011.
11. Налоговый кодекс Российской Федерации. Части 1 и 2. — М.: Эксмо, 2015. — 1344 с.
12. *Web Time*. Time Web. Стоимость аренды серверов. — web. — 2015. <http://timeweb.com/ru/services/dedicated-server/>.
13. *Соколов А. Н., Сердобинцев К. С.* HP OpenView System Administration Handbook: Network Node Manager, Customer Views, Service Information Portal, HP OpenView Operations / Ed. by X. Шутзе. — Астрахань: Издательство, 2004. — 688 pp.
14. *Foundation Apache Software*. Apache OpenNLP. — web. — 2012. — 04. <https://opennlp.apache.org/>.
15. *Goetzel Ben*. OpenCog RelEx. — web. — 2012. — 04. <http://wiki.opencog.org/w/RelEx>.
16. *Вебер П., Вильямс Д.* Введение в обработку информации / Под ред. Т. Зителло. — Upper Saddle River, New Jersey 07458: Прентис Холл, 2009. — 581 с.
17. *Гринберг Д.* Надежный алгоритм обработки для грамматики // *Технический отчет Университета Карнеги Мелон CMU-CS-95-125*. — 1995. — 1 июля.
18. *Stuart Russell Peter Norvig*. Artificial Intelligence. A Modern approach. — Pearson, 2010.
19. *Katidiotis A. Tsagkaris K.* Performance evaluation of artificial neural network-based learning schemes for cognitive radio systems // *Computers Electrical Engineering*. — 2010. — Т. 36.

20. *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.
21. *Лейсу Л.* Owl: Representing Information Using the Web Ontology Language. — 47403, Блумингтон, Либерти драйв 1663: Трэффорд Паблишинг, 2005. — 302 с.
22. *Noy N. McGuinness D.* Ontology Development 101: A Guide to Creating Your First Ontology // *Stanford University*. — 2010.
23. *Ванг П.* Non-Axiomatic Logic A Model of Intelligent Reasoning. — США: World Scientific Publishing Company, 2013. — 276 с.
24. *White D.* Software review: the ECJ toolkit // *Genetic Programming and Evolvable Machines*. — 2012. — Т. 13.
25. *Talanov M.* Automating programming via concept mining, probabilistic reasoning over semantic knowledge base of SE domain // *Software Engineering Conference (CEE-SECR), 2010 6th Central and Eastern European*. — 2010.
26. *Хокинг С.* ТЕОРИЯ ВСЕГО. — Москва: Амфора, 2009. — 160 с.
27. *Minsky Marvin.* The Society of Mind. — Simon & Schuster, 1988.
28. *Fowler M.* UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition).
29. *Вайтм Д.* Akka Concurrency / Под ред. К. Роланд. — Артима, 2013. — 521 с.
30. *Робинсон С.* WebSphere Application Server 7.0 Administration Guide. — PACT publishing, 2009. — 344 с.
31. *Гойтз Б., Пейерлс Т., Блох Д.* Java Concurrency in Practice. — Addison-Wesley Professional; 1 edition, 2006. — 384 с.
32. *Гетзель Б. Мэтью И.* Probabilistic Logic Networks: A Comprehensive Conceptual, Mathematical and Computational Framework for Uncertain Inference. — Springer: Springer, 2008. — 333 с.

33. *Мейер Б.* Object-Oriented Software Construction 2nd Edition. — Аппер Садл Ривер, США: Прентис Холл, 1997. — 1296 с.

Список рисунков

1.1	Диаграмма состава команд	13
1.2	Диаграмма соотношений типов проблем	13
2.1	HP OpenView	17
2.2	Service NOW	18
2.3	Пример работы системы Watson	19
2.4	Результаты обработки текстов	23
2.5	Архитектура предварительной обработки текста	24
3.1	Представление класса Order в OWL. Визуализация Protege.	31
3.2	Представление класса CreateCustiner в OWL. Визуализация Protege.	32
3.3	Диаграмма последовательности для основного потока в модели Menta 0.1	33
3.4	Критик-Селектор-Путь мышления	38
3.5	Критик-Селектор-Путь мышления в разрезе ресурсов	39
3.6	Иллюстрация концепции Уровней мышления	42
3.7	Иллюстрация концепции K-line	43
4.1	Вариант использования. Обучение.	48
4.2	Диграма компонентов системы	50
4.3	Диграма взаимодействия компонентов	51
4.4	Интерфейс компонента WebService	52
4.5	Диаграмма классов ThinkingLifeCycle	57
4.6	Диаграмма действий метода onMessage компонента ThinkingLifeCycle	58
4.7	Диаграмма действий метода sendMessage компонента ThinkingLifeCycle	58
4.8	Диаграмма действий метода apply компонента ThinkingLifeCycle	59
4.9	Диаграмма действий метода apply компонента ThinkingLifeCycle	60
4.10	Диаграмма действий метода processWay2Think компонента ThinkingLifeCycle	61
4.11	Диаграмма действий метода processCritic компонента ThinkingLifeCycle	61

4.12	Диаграмма действий метода init компонента ThinkingLifeCycle . . .	62
4.13	Диаграмма действий метода stop компонента ThinkingLifeCycle . .	63
4.14	Интерфейс компонента Selector	66
4.15	Диаграмма действий метода Selector.apply(request : Request) компонента Selector	68
4.16	Диаграмма действий метода Selector.apply(goal: Goal) компонента Selector	70
4.17	Диаграмма действий метода Selector.apply(criticResult : ActionProbabilityRule) компонента Selector	71
4.18	Диаграмма действий выбора WayToThink	72
4.19	Диаграмма действий классификации инцидента	73
4.20	Диаграмма действий компонента Critic	76
4.21	Интерфейс компонента WayToThink	78
4.22	Работа компонента WayToThink в режиме Рецепта решения (HowTo)	81
4.23	Интерфейс компонента PreliminaryAnnotator	82
4.24	Интерфейс компонента KnowledgeBaseServer	84
4.25	Интерфейс компонента Reasoner	86
4.26	Схема данных TU Knowledge в формате UML	87
4.27	Диаграмма действий LifecycleActivity	92
A.1	Диаграмма классов интерфейсной модели	108
Б.1	Диаграмма классов Action	109
В.1	Диаграмма классов Goal	111
В.2	Диаграмма места Goal в SemanticNetwork (Семантической сети) . .	111

Список таблиц

1	Сопоставление направлений исследования специальности 05.13.01 и исследований, проведенных в работе	7
2	Словарь терминов	10
3	Принятые аннотации для изложения	11
1.1	Описание работы специалистов различных уровней поддержки . .	12
1.2	Категории инцидентов в области удаленной поддержки инфраструктуры	14
2.1	Таблица метрик	22
2.2	Сравнительный анализ существующих решений	28
3.1	Описание свойств класса Order в OWL	31
3.2	Описание иерархии предикатов	32
3.3	Компоненты модели Menta 0.3	34
3.4	Описание уровней мышления Марвина Мински	41
4.1	Основные компоненты системы ThinkingUnderstanding	45
4.2	Описание ветвей в Варианте использования "Режим обучения" . . .	46
4.3	Описание ветвей в Варианте использования "Основной режим обучения"	47
4.4	Описание методов компонента Webservice	53
4.5	Описание методов класса (компонента) ThinkingLifeCycle	54
4.6	Описание методов класса (компонента) Selector	66
4.7	Описание основных классов Critic, используемых в системе	74
4.8	Описание методов класса Critic	75
4.9	Описание встроенных в систему WayToThink	79
4.10	Описание методов компонента WayToThink	80
4.11	Описание методов компонента PreliminaryAnnotator	82
4.12	Описание методов компонента DataService	84
4.13	Описание методов компонента Reasoner	86
4.14	Описание классов TUKnowledge	88
4.15	Описание экспериментальных данных	93

4.16	Результаты сравнения с работой человеческого специалиста	96
------	--	----

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 - список 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

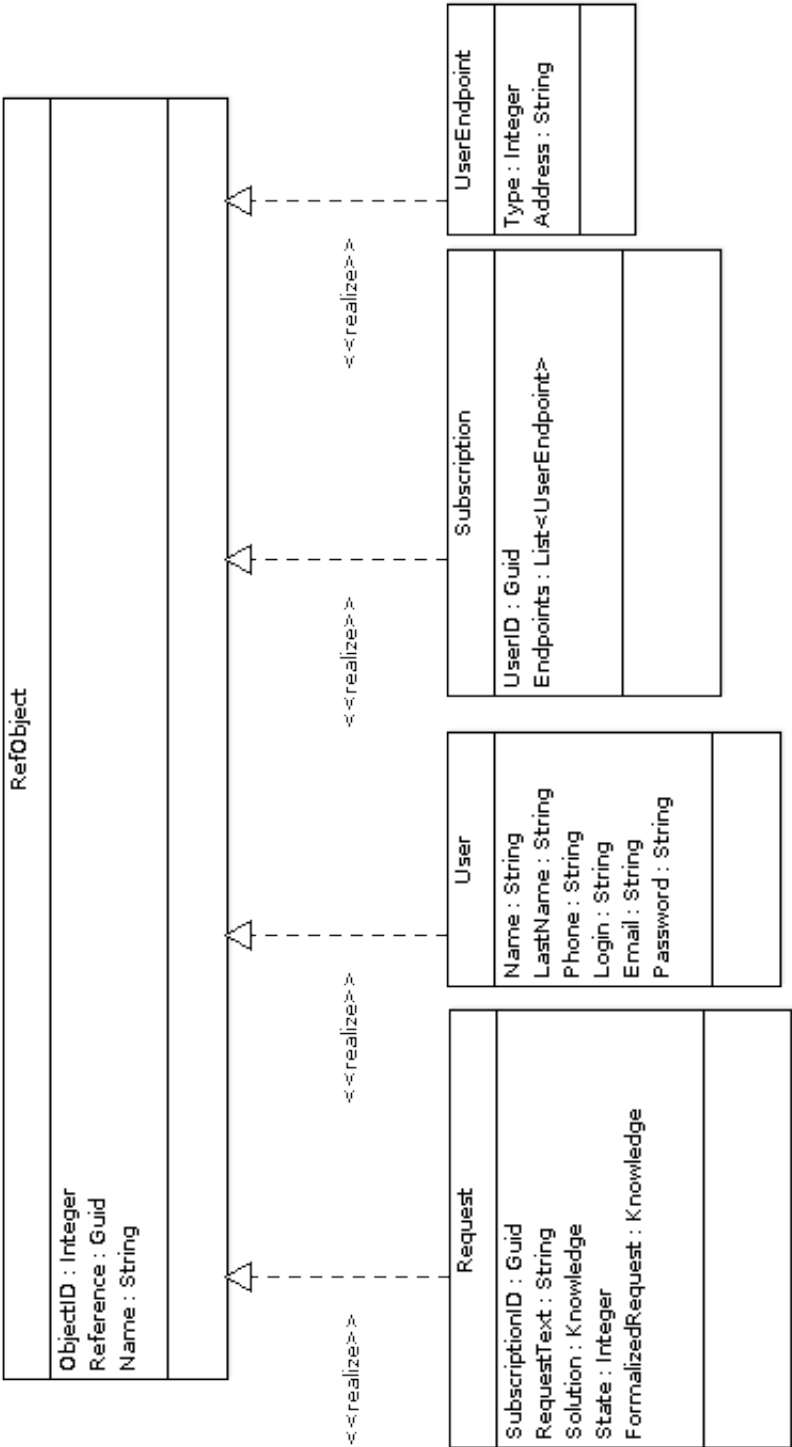


Рисунок А.1 — Диаграмма классов интерфейсной модели

Приложение Б

Описание модуля Action

Action является базовым классом для WayToThink или Critic.

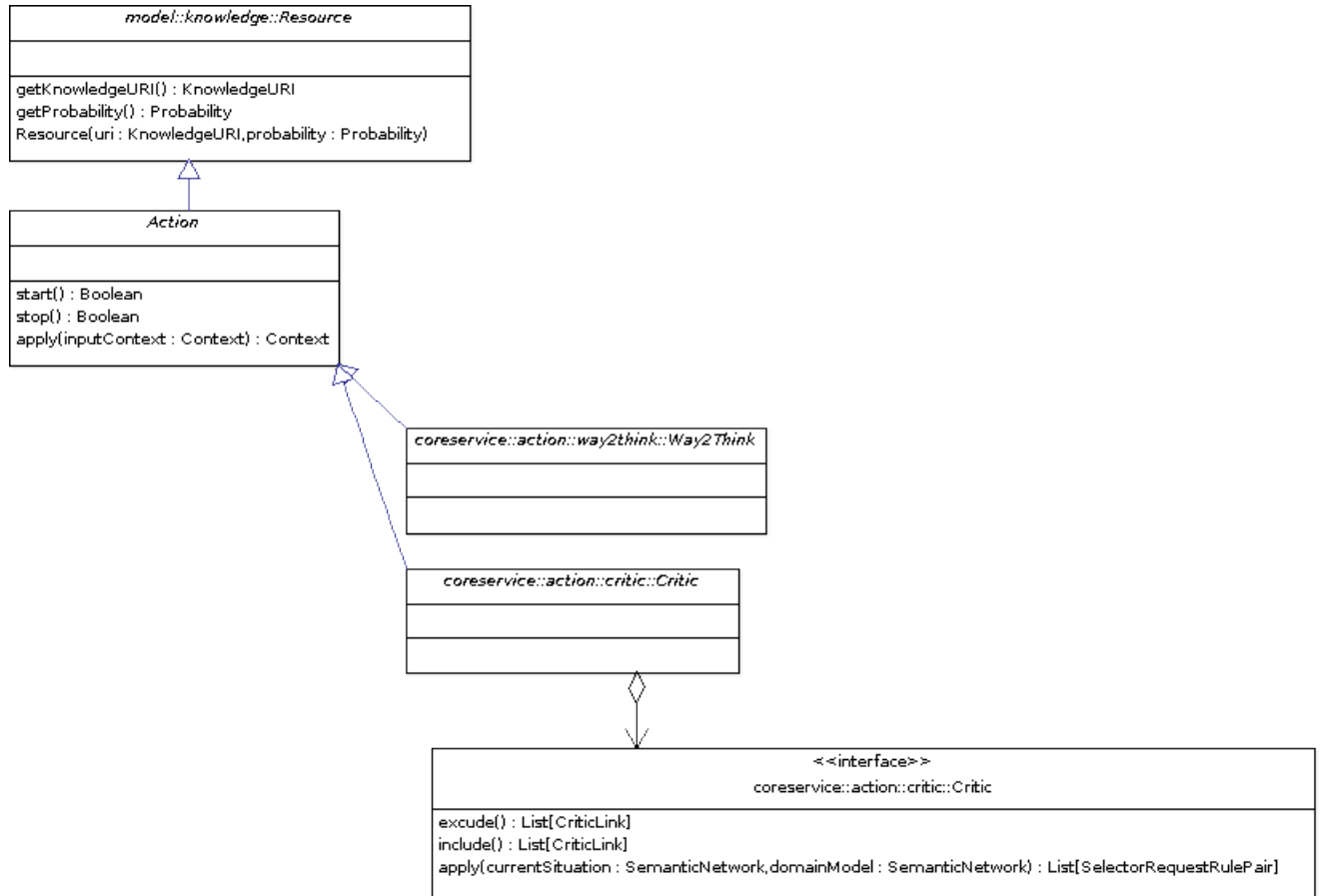


Рисунок Б.1 — Диаграмма классов Action

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.

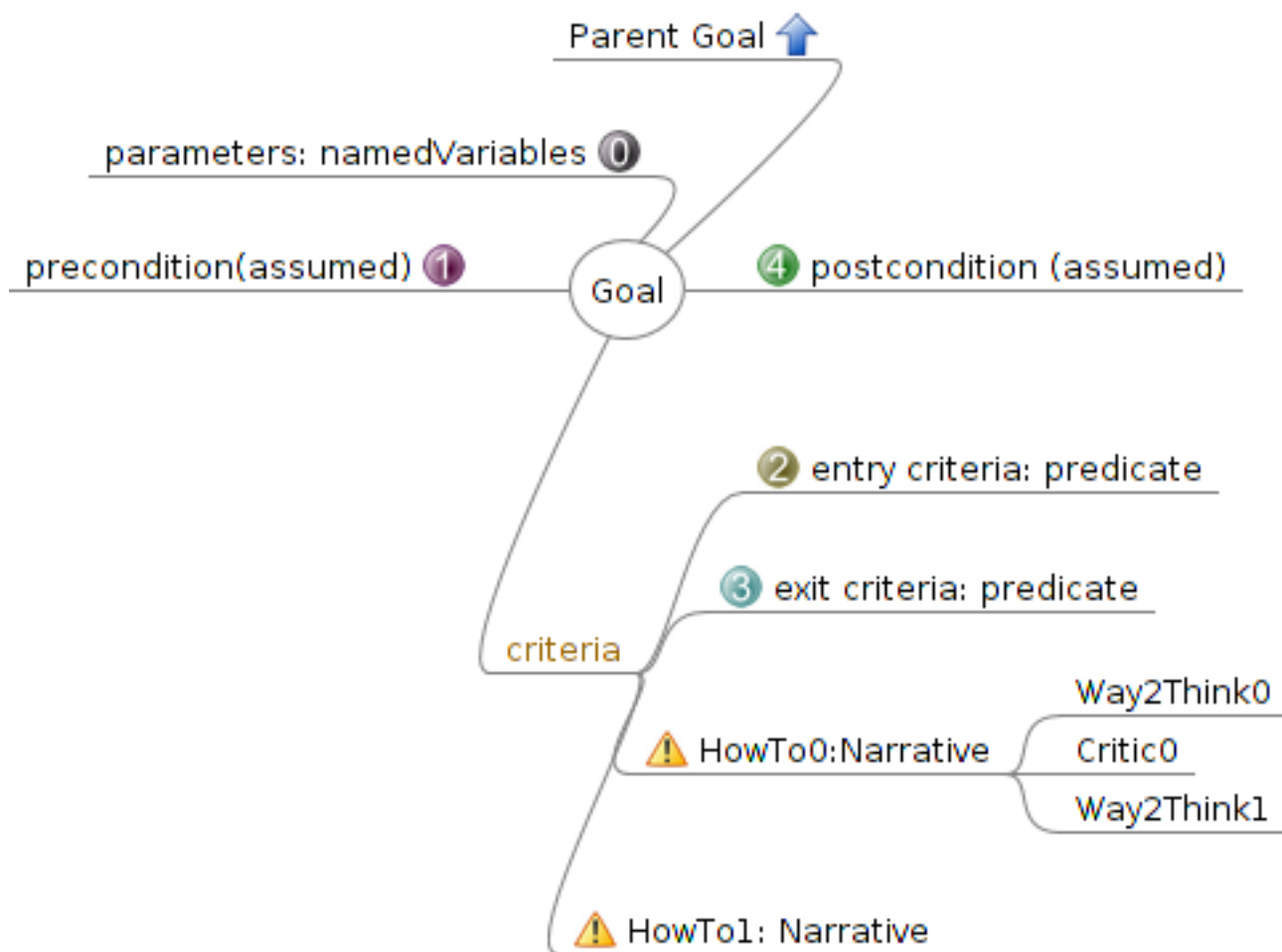


Рисунок В.1 — Диаграмма классов Goal

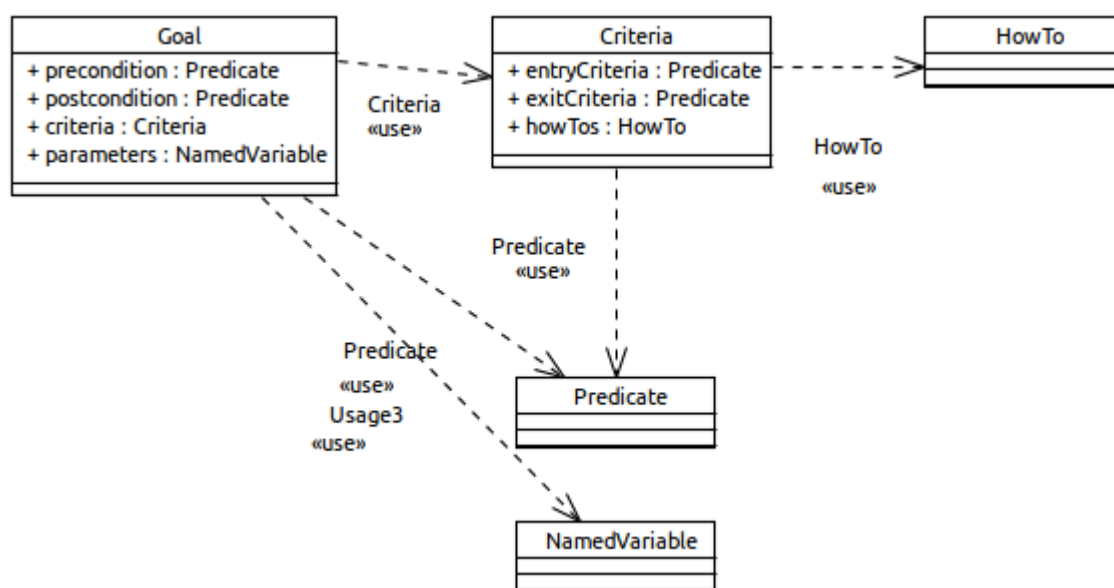


Рисунок В.2 — Диаграмма места Goal в SemanticNetwork (Семантической сети)

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 Рецептот Решения. Например, решение проблемы неработающего сегмента кластера в формате для специалиста технической поддержки.

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

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


```

login:howto{
  Parameters:[
5    {Key:'ScriptName',
      Value:'LogonScript.bat'},
      {Key:'Description',
        Value:'Logon to server'}
  ]
10
  InputParameters:[
      {Key:'ServerName',
        Value:'U1'},
      {Key:'UserName',
15     Value:'MyUser'}
  ]

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

25 launch:howto{
  Parameters:[

      {Key:'ScriptName',
30     Value:'LaunchScript.bat'},
      {Key:'Description',
        Value:'Launch the application'}
  ]

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

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

```

$\left| \begin{array}{c} \\ \end{array} \right|$

Приложение Д

Экспериментальные данные

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

```

EUROPE DOMAIN NEW SERVER Request Form
5 * (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
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 \\
15 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 \\
20 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
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 \\
30 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
35 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
40 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

45 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 \\

50 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 \\

55 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

60 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

65 EUROPEVUK039 caiW2kOs:w2kSrvInst 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 \\

70 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** \\

75 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

80 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

85 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

2011-05-02 15:01:17 UKM205 LogWatcher BABBACKUP is now Critical \\
 2011-05-02 17:01:59 EUROPEMUK268 LogWatcher BABbackup **File**\\
 115 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
 120 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
 125 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
 130 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
 135 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

140 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.

145 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

150 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

155 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

160 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
165 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
170 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
175 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

```