МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
Одесский национальный политехнический университет

На правах рукописи
УДК 004.932.72'1

Маковецкий Александр Сергеевич

МОДЕЛИ И ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ ПОСТРОЕНИЯ БАЗ ЗНАНИЙ ДЛЯ ДИАГНОСТИКИ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

05.13.06 – Информационные технологии

Диссертация
на соискание научной степени кандидата технических наук

Научный руководитель –
кандидат физико-математических наук,
доцент Тишин Петр Метталинович

Одесса – 2015

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность. Распределенная информационная система (РИС) - это сложная система, в среде которой функционирует ряд информационных сервисов. Отличительной особенностью таких систем является высокая трудоемкость обеспечения необходимого уровня их функциональности, производительности и надежности.

Диагностические службы являются одним из важнейших компонентов систем обслуживания РИС. Диагностические услуги используются для обнаружения, идентификации и локализации неисправностей, а также для поиска возможных причин их возникновения. Высокая сложность современных РИС приводит к существенному увеличению времени, которое тратит эксперт на ее диагностирование.

Наиболее результативным способом сокращения временных затрат на выполнение процедур диагностирования современных РИС, без потери достоверности, является разработка автоматизированных систем, которые могли бы накапливать и сохранять знания высококвалифицированных экспертов и использовать их при решении задач диагностики. Такие системы строятся с использованием баз знаний, которые позволяют хранить и обрабатывать знания многих экспертов о возможных причинах неисправностей при решении задач диагностики РИС.

В современных системах диагностирования РИС, основанных на знаниях, присутствуют две актуальные задачи, которые требуют своего решения. Первая - связана с тем, что часть параметров функционирования РИС не могут быть заданы в виде конкретных значений, а задаются в виде нечетких данных. Вторая - большое разнообразие и постоянное изменение аппаратных и программных компонентов РИС приводит к существенным трудностям при поддержке актуальности ее базы знаний. Анализ современных моделей и технологий построения баз знаний РИС показал, что, с одной стороны, они не позволяют

работать с нечетко заданными параметрами, с другой стороны, для внесения изменений требуют привлечения и эксперта по базам знаний, и эксперта по РИС.

Таким образом, разработка новых моделей и информационной технологии создания баз знаний для автоматизированных систем диагностирования РИС, которые позволяют оперативно учитывать в базе все изменения, происходящие в РИС и позволяют проводить диагностирование на основе нечетких диагностических данных, является актуальной.

Связь работы с научными программами, планами и темами. Работа выполнена в соответствии с приоритетными направлениями научно-исследовательских работ Одесского национального политехнического университета (ОНПУ), в соответствии с координационными планами Министерства образования и науки Украины, в частности, в рамках научных исследований по госбюджетным научно-исследовательскими работами 700-145 "Модели, методы и инструментальные средства поддержки принятия решений по повышению эффективности гидродинамических процессов в действующем энергетическом оборудовании ".

Роль автора в указанных проектах, в которых он был непосредственным исполнителем, заключается в разработке моделей баз знаний для диагностирования распределенных информационных систем.

Цель и задачи исследования. Целью исследования является разработка моделей и информационной технологии построения баз знаний автоматизированных систем диагностирования РИС, использование которых позволяет сократить время на выполнение диагностики и время, затрачиваемое на поддержку актуальности базы знаний.

Для достижения поставленной цели в диссертационной работе решены следующие **основные научно-технические задачи**:

1. Проведен анализ существующих систем диагностирования РИС, основанных на знаниях, и методов и моделей их построения. Выявлены их ограничения и недостатки и возможные пути преодоления.

2. Разработана структура базы знаний для систем диагностирования РИС.

3. Усовершенствована обобщенная модель диагностики РИС, которая, с применением аппарата нечетких множеств, позволяет использовать для диагностирования РИС экспертные знания о нечетко заданных параметрах.

4. Разработаны онтологические модели расширенного описания элементов и зависимостей в РИС, а также онтологическая модель описания диагностических тестов в формате OWL с использованием дескриптивной логики.

5. Разработана онтологическая модель описания знаний о штатных и нештатных ситуациях на основе многосортного языка прикладной логики.

6. Развит метод диагностирования РИС, который может быть применен в рамках разработанной модели базы знаний диагностики РИС, и может проводить ранжирование диагнозов по степени достоверности.

7. Разработана информационная технология построения баз знаний для систем диагностирования РИС, которая позволяет сократить временные затраты на выполнение диагностики и поддержку актуальности базы знаний РИС.

Объектом исследования является процесс диагностирования РИС.

Предметом исследования является информационная технология создания баз знаний систем диагностирования РИС.

Методы исследования. Методы математического моделирования систем и основные положения теории нечетких множеств для описания обобщенной модели диагностики РИС; основные положения дескриптивной логики, теории графов и аппарата онтологического вывода при построении зависимостей между сервисами РИС, множествами диагностических параметров и их источниками; основы прикладной многосортной логики для описания модели онтологии штатных и нештатных ситуаций.

Научная новизна исследования заключается в следующем:

- усовершенствована обобщенная модель диагностики РИС, которая, в отличие от существующих, благодаря применению аппарата нечетких множеств и лингвистических переменных, позволяет сократить затраты времени на поддержку актуальности базы знаний диагностических ситуаций в РИС;

- впервые разработаны онтологические модели расширенного описания элементов и зависимостей в РИС, а также онтологическая модель описания диагностических тестов, в формате OWL с использованием дескриптивной логики, которые позволяют сократить время, затрачиваемое экспертами-пользователями в поддержку актуальности базы знаний структурных и функциональных зависимостей между элементами РИС в условиях высокой динамики развития РИС;

- впервые разработана онтологическая модель описания знаний о штатных и нештатных ситуациях на основе многосортного языка прикладной логики, что позволяет сократить затраты времени эксперта-пользователя на поддержку актуальности базы знаний диагностических ситуаций в РИС и их причин;

- получил дальнейшее развитие метод диагностирования РИС для использования в рамках разработанной модели базы знаний диагностики РИС, который, в отличие от существующих, за счет использования знаний о нечеткие значения диагностических параметров, позволяет ранжировать диагнозы по степени вероятности, что позволяет диагносту-пользователю сократить время на диагностирование неисправности в РИС.

Практическое значение полученных результатов:

- Впервые разработана информационная технология построения баз знаний для диагностирования нечетких ситуаций в РИС, основанных на экспертных знаниях.

- На основе разработанной информационной технологии были созданы экспериментальная база знаний и диагностическая система РИС. Их

применение в распределенной информационной системе обеспечило сокращение времени на диагностирование каждой из неисправностей, описанных в базе знаний, в 5-10 раз, в зависимости от неисправности, без потери достоверности, при этом коэффициент простоя системы уменьшился на 7% по сравнению с коэффициентом простоя без использования базы знаний. Кроме того, сократилось время на поддержку актуальности базы знаний в целом в 2,5 раза.

- Результаты работы были внедрены в учебный процесс Одесского национального политехнического университета, кафедры компьютерных интеллектуальных систем и сетей в курсы дисциплин: «Исследования в области искусственного интеллекта», «Компьютерные сети», «Анализ и проектирование компьютерных систем», и задействованы в научно-промышленном предприятии "Дискрет" в качестве системы диагностирования информационных сетей (акт о внедрении от 27.05.15).

Личный вклад соискателя.

Основные положения и результаты диссертационной работы, выносимые на защиту, были получены автором лично. В работах, написанных в соавторстве, вклад диссертанта заключается в следующем: разработана формализованная онтологическая модель описания диагностических тестов с использованием дескриптивной логики [1, 10]; разработана онтологическая модель для описания знаний о штатных и нештатных ситуациях при диагностике РИС на основе многосортного языка прикладной логики [2, 8-9]; разработана методика расчета сходства между текущей ситуацией и ситуациями, которые описаны в базе знаний [3]; разработан новый подход с применением нечетких множеств для выявления неисправностей в распределенных информационных системах [4, 11]; разработана онтологическая модель описания закономерностей при диагностике распределенных информационных систем [5-7, 12].

Апробация результатов диссертации. Научные положения, выносимые на защиту, выводы и рекомендации диссертации принадлежат автору. Основная часть полученных в диссертации результатов докладывались автором лично на международных научно-технических конференциях. Материалы диссертации прошли апробацию на пяти научных конференциях и семинарах различного уровня, а именно:

X Всеукраинская НТК студентов и аспирантов "Информационные системы и технологии" - ОГАХ, Одесса, 2010 г.;

XIII и XIV Международные НТК «Современные информационные и электронные технологии», Одесса, Украина, 2012 - 2013 гг;

III Украинско-немецкая конференция «Информатика. Культура. Техника», Одесса, 2015 г.;

XXIII Международная научно-практическая конференция «Информационные технологии: наука, техника, технология, образование, здоровье», Харьков, 2015

Публикации. По теме диссертации опубликованы 12 работ, в том числе 6 статей в научных журналах из перечня профессиональных изданий Украины, 5 из которых внесены в международные наукометрические базы (BASE, Index Copernicus, DOAJ и др.), А также 6 тезисов докладов в трудах международных и всеукраинских конференций.

Структура диссертации. Диссертация состоит из введения, четырёх разделов, выводов, списка использованных источников из 112 наименований и 3 приложений. Полный объем диссертации составляет 137 страниц; работа содержит 18 рисунков и 5 таблиц.

РАЗДЕЛ 1
АНАЛИЗ СУЩЕСТВУЮЩИХ СИСТЕМ ДИАГОНОСТИРОВАНИЯ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ, МОДЕЛЕЙ И МЕТОДОВ ИХ ПОСТРОЕНИЯ

1.1. Системы диагностирования РИС, основанные на знаниях, и подходы, применяемые для их создания.

Согласно исследованиям Андреаса Ханемана (Andreas Hanemann) [13], существующие методы и модели создания систем, базирующихся на знаниях, для диагностирования РИС можно условно разделить на несколько классов: те, которые основаны на моделях (MBR), те, которые основаны на правилах (RBR) и те, которые основаны на прецедентах (CBR).

1.1.1 Системы основанные на моделях.

Рассуждения основанные на моделях (MBR) [14-16] представляет собой систему основанную на моделировании каждой из ее компонентов. Модель может представлять собой либо физическую сущность или логическую сущность (например, LAN, WAN, домен, сервисы, бизнес-процессы). Модель содержащая все физические сущности называется функциональной моделью, а модель содержащая все логические сущности называется логической моделью. Описание каждой модели содержит три категории информации: атрибуты, отношения с другими моделями, поведение. Решение задач диагностики является результатом сотрудничества между виртуальными автономными моделями. Так как все компоненты сети представлены с их поведением в имитационной модели, то можно выполнять моделирование, чтобы предсказать, как вся сеть будет вести себя [17]. В работе используется оригинальная MBR из [16], которая показывает, использование MBR в качестве способа рассуждений

на основе модели системы. Примером системы MBR является NETeXPERT2 [18]. Для телекоммуникаций система была внедрена в [19], еще один пример системы был представлен в [20, 21].

Yemanja [22] представляет собой MBR, которая направлена на корреляцию низкоуровневых сетевых событий. Здесь используются модели поведения для субъектов и правил корреляции. Слоистая структура, которая используется в данной работе, позволяет отслеживать, как выявленные причины и симптомы распространяются от нижних слоев к более высоким уровням. Агенты используются для сбора информации с помощью SNMP и базы данных конфигурации.

В работе [23] используются диаграммы зависимостей для определения неисправности. Кроме этого, модели поведения используются при моделирования производительности компонентов, участвующих в сервисных операциях. Идея состоит в том, чтобы не использовать фиксированные пороговые значения для отчетности о нарушениях ресурсами порогов, а рассчитывать их динамически из условий SLA. В ходе мониторинга SLA, также запоминаются значения производительностей ресурсов, которые подразделяются на "хорошие" или "плохие". "Плохие" модели состояния для компонентов привязаны к проблемной ситуации, в то время как хорошие модели состояния универсальны. Отношение "плохой" модели состояния к динамическим порогам определяет серьезность проблемы с ресурсами. Подход использует одну первопричину для определения неисправности. Похожие подходы, относящиеся к MBR описаны в работах [24-26].

Графы переходов между состояниями [27] являются одним из типов моделей MBR. Он использует состояния и переходы между состояниями, а также токены, которыми можно разметить состояния. Действиям соответствует переход между состояниями. Графы, которые построены из этих элементов описывают процедуру идентификации неисправности. Этот подход используется в Open Service’s Nerve Center [28].

Таким образом можно сделать вывод, что подход на основе MBR, позволяет моделировать сложные отношения, которые встречаются в сценариях управления обслуживания и поэтому используется для диагностики неисправностей. Применение этого подхода требует моделировать каждый сервис, как логический объект, что включает в себя детальное моделирование взаимодействия этого сервиса с другими сервисами и инфраструктурой [15]. Эффективность моделей во многом определяется знаниями, которые заложены в MBR и может приводить к дополнительным сложностям. Поэтому методика часто используется в сочетании с другими методами.

1.1.2 Системы основанные на правилах.

Рассуждения на основе правил (RBR) [14, 17] использует набор правил для диагностики событий. Условия используют полученные события и информацию о системе, в то время как заключение содержит либо действия, либо инициирует дополнительные факты, которые позволяют выбирать следующее правило. На рис. 1.1 показана основная структура системы, основанной на правилах.

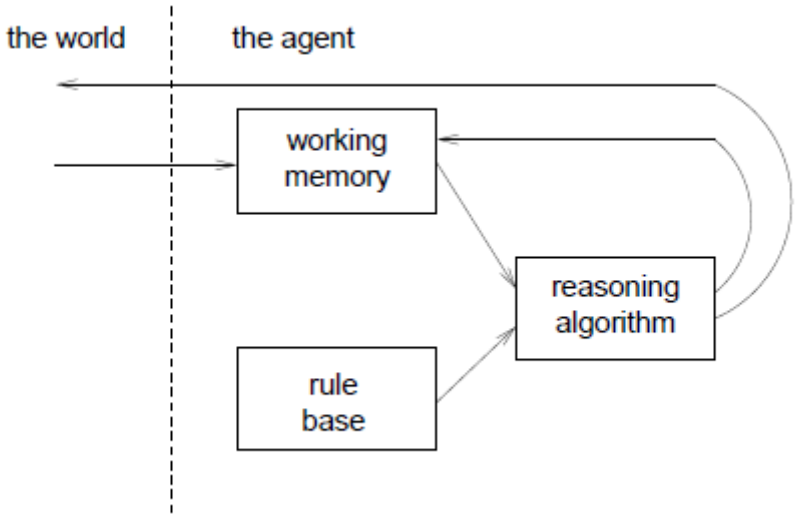


Рисунок 1.1 – Основная структура RBR [14]

События, полученные от внешнего мира, сохраняется в рабочей памяти. Правила из базы правил применяются в соответствии информацией в рабочей памяти, что приводит к обновлениям рабочей памяти, а также может запускать действия во внешний мир. Коммерческие системы, такие как IBM Tivoli Enterprise Console [29], HP OpenView Event Correlation Services [30] (на основе инструмента с открытым исходным кодом Simple Event Correlator [31, 32), Micromuse Netcool Impact [33] и т.д., основаны на правилах, так что этот подход можно рассматривать как доминирующий в промышленности. В дополнение к использованию для проводных сетей, он также применяется для беспроводных сетей GSM [34]. Подход, который обращается к сервисно-ориентированный архитектуре можно найти в [35]. В данной работе аномалии выявляются в транзакциях с использованием динамических порогов. События получаются с помощью SNMP ловушек или агентов.

Важным вопросом для применения системы на основе правил является генерация и поддержание базы правил. Чтобы избежать ручного ввода знаний в правилах, были разработаны автоматизированные методы. В [36] дается подход для генерации правил из базы данных которая предназначена для сотовых сетей и в состоянии иметь дело с зашумленными данными. Статья содержит хороший обзор соответствующей работы. Дальнейшие подходы для автоматического определения правил можно найти в [37, 38].

RBR может быть связано с политиками управления [39], в том смысле, что некоторая политика представляет особый вид правила. Этот подход к управлению QoS был предложен в [40], причём он позволяет применять RBR к сервисным архитектурам. Здесь предлагается система мониторинга QoS , а правила используются для выполнения определенных действий при отклонения QoS.

Рассуждения на основе правил имеет ряд преимуществ, которые привели к успеху этого метода в сетевой области. В общем, подход позволяет компактно представить общие знания о домене [41]. Атомарные фрагменты информации,

состоящие из правил, могут быть добавлены (удалены), не затрагивая другие фрагменты. Модули, состоящие из правил могут быть проверены по отдельности. Объяснение результата диагностики, в системах основанных на правилах, легко может быть осуществлено путем отслеживания выполненных правил в обратном порядке.

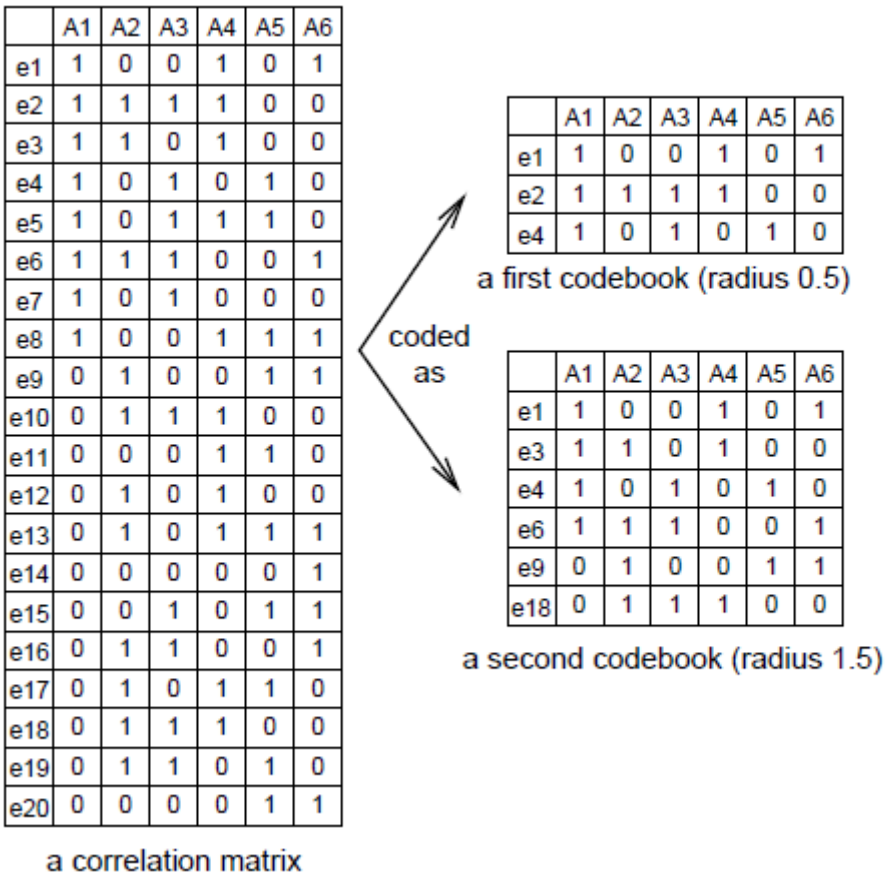
Однако в работах [15, 22] системы RBR классифицируются как относительно негибкие. Частые изменения в моделируемой среде приводят к обновлению правил. Эти изменения должны быть выполнены специалистами так, как ни один метод автоматизации не имеет в настоящему времени достаточной проработки. В свою очередь это имеет тот недостаток, что эксперты могут быть недоступны, чтобы обеспечить свои знания соответствующим образом. В некоторых системах информация о топологии сети, которая необходима для диагностики не используется явно, но кодируется в правилах. Это делает непрозрачным использование правил для обновления и изменения топологии. Кроме того, для систем RBR, является проблемой возникновение неизвестной ей ситуации, потому что существующие правила не совпадают с этой ситуацией. Выход систем RBR также трудно предсказать, вследствие непредвиденных взаимодействий правил в большом наборе правил и потенциально конфликтующих правил [14].

Таким образом, системы RBR подходят для сервис-ориентированной диагностики, если возможно найти масштабируемое решение для генерации и поддержания правил.

Подход на основе кодовой книги [42, 43] использует матрицу, содержащую отношения симптомов и неисправностей. Эта матрица называется кодовой книгой. Подход начинается с использования графа зависимостей с двумя видами узлов для моделирования. Первым видом узлов являются неисправности, которые должны быть обнаружены, в то время как вторым видом узлов являются наблюдаемые события (симптомы). Зависимости между узлами обозначены как ориентированные рёбра. Можно задать веса для ребер,

например вероятности того, что одна ошибка или событие, вызывает другую ошибку или событие. Другой возможный вес может указывать зависимость от времени. Есть несколько возможностей снижения размерности исходного графа.

После окончательного ввода графа, он превращается в матрицу зависимостей, где столбцы содержать неисправности, а строки содержат события (рис. 1.2).



	A1	A2	A3	A4	A5	A6
e1	1	0	0	1	0	1
e2	1	1	1	1	0	0
e3	1	1	0	1	0	0
e4	1	0	1	0	1	0
e5	1	0	1	1	1	0
e6	1	1	1	0	0	1
e7	1	0	1	0	0	0
e8	1	0	0	1	1	1
e9	0	1	0	0	1	1
e10	0	1	1	1	0	0
e11	0	0	0	1	1	0
e12	0	1	0	1	0	0
e13	0	1	0	1	1	1
e14	0	0	0	0	0	1
e15	0	0	1	0	1	1
e16	0	1	1	0	0	1
e17	0	1	0	1	1	0
e18	0	1	1	1	0	0
e19	0	1	1	0	1	0
e20	0	0	0	0	1	1

	A1	A2	A3	A4	A5	A6
e1	1	0	0	1	0	1
e2	1	1	1	1	0	0
e4	1	0	1	0	1	0

	A1	A2	A3	A4	A5	A6
e1	1	0	0	1	0	1
e3	1	1	0	1	0	0
e4	1	0	1	0	1	0
e6	1	1	1	0	0	1
e9	0	1	0	0	1	1
e18	0	1	1	1	0	0

Рисунок 1.2 – Корреляционная матрица и две производных кодовые книги [43]

Если существует зависимость в графе, вес соответствующего ребра помещают в ячейку матрицы. Впоследствии, можно сделать упрощение, за счет удаления событий, которые не помогают различать неисправности. Существует компромисс между минимизацией матрицы и устойчивостью результатов. Если матрица сводится к минимуму, насколько это возможно, некоторые неисправности могут быть выделены только одним событием. Если это событие

не может быть надежно обнаружено, то система не может различать две неисправности. Мера, как много ошибок в наблюдении событий компенсируется системой определяется расстоянием Хэмминга. С теоретической точки зрения расчет минимального размера кодовой книги является NP-трудной задачей, но эвристические методы существуют и хорошо работают [44].

В [45] возможности для преобразования представления из баз знаний на основе правил в кодовые книги и наоборот были показаны. Они применялись для оптимизации. Преобразование может быть полезным поскольку правила представляют более удобный способ принятия решений, в то время как матрицы зависимостей являются более понятными. Развитие техники кодовой книги привязано к коммерческому инструменту Smarts5 InCharge [46].

Подход на основе кодовой книги имеет то преимущество, что он использует долгосрочный опыт работы с графами и кодированием. Этот опыт используется, чтобы минимизировать граф зависимостей и выбрать оптимальный группу событий по времени обработки и устойчивости к шуму. Подход может в некоторых ситуациях иметь дело с неизвестным комбинациями событий. Они могут быть отображены на известные комбинации с помощью расстояния Хэмминга. Тем не менее, эти оптимизации привязаны к двоичному кодированию из зависимостей.

Для применения этого подхода в РИС, ремонтопригодность корреляционной матрицы является критической проблемой, похожей на поддержание правил в RBR. Частые изменения в реализации сервиса потребует частого обновления графа зависимостей, которое может занять довольно много времени. Кроме того, это не очевидно, как кодировать сложные отношения в простом графе зависимостей. Еще один недостаток в том, что общее корреляционное окно должно быть применено для матрицы [22]. Этого недостаточно для сервис-ориентированного подхода, потому что сервисная

информация, как правило, действительна дольше, чем события на уровне ресурсов.

1.1.3 Системы основанные на прецедентах.

Рассуждения на основе прецедентов (CBR) — это подход, который основан на изучении прошлого опыта. Общую информацию о CBR и его первых применениях можно найти в работах [47, 48]. Понятия связанные с управлением сетью, приведены в работах [49, 50, 51]. Подход использует отчеты о обнаруженных симптомах, которые были формализованы и внесены в базу данных прецедентов вместе с соответствующим решением. Решение, в котором используется текущий симптом, нацелено на повторное использование его в подобных ситуациях в будущем.

Примерами систем CBR являютсr SpectroRx [52], FIXIT [53], Critter [49], и ACS [54].

При применении в РИС, в результате изменений в реализации сервисов, может выявляться ошибочность решений полученных в предыдущих случаях. Есть два способа, чтобы справиться с этой ситуацией. Первый способ - искать в базе данных, такие случаи и как-то изменять или удалять их. Второй способ - адаптация предыдущих случаев.

В заключение заметим, что CBR может выразить специальные знания о прецедентах [41]. Система CBR является модульной в том смысле, что единичные прецеденты могут быть удалены из базы данных прецедентов, не затрагивая всю систему. Во время работы прецеденты могут быть легко добавлены, но следует отметить, что нет, как правило, ни одного прецедента, приведенного заранее.

Согласно [47], процедура вывода в CBR может потребовать меньше усилий, чем в RBR. Тем не менее, при таком сравнении, нужно быть

осторожным, поскольку например, дополнительными усилиями на шаге адаптации CBR не следует пренебрегать [55].

Есть также трудности при применении данного подхода [14]. Поля, которые используются, чтобы найти аналогичный случай и их значения должны быть определены соответствующим образом. Метод не в состоянии выразить общие знания о прецедентах, и может привести к проблемам адаптивности [41].

1.1.4 Гибридный подход.

В результате рассмотрения методов диагностики РИС, известных на данный момент, в работе [13] была рассмотрена гибридная архитектура базы знаний. Рассмотрение такого подхода обусловлено тем, что каждый из методов, представленных ранее в литературе [16] и данном разделе, имеет некоторые недостатки при его применении в РИС. Объединяя RBR и CBR методы, можно использовать преимущества обоих подходов, избегая их ограничений. Гибридная архитектура, предложенная в работе [13], изображена на рис. 1.3.

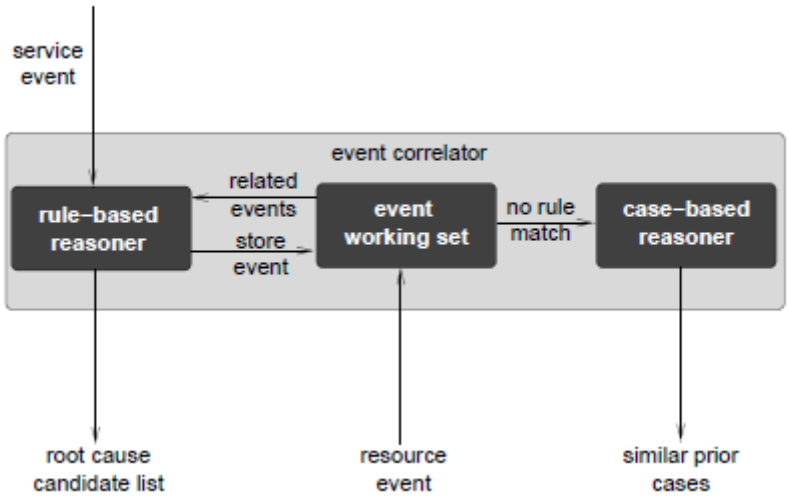


Рисунок 1.3 – Гибридная архитектура базы знаний.

Наличие эффективных алгоритмов корреляции и выразительность представления знаний в правилах привели к выбору модуля RBR. Этот модуль получает события сервиса и, в первую очередь, пытается автоматически соотнести их для того, чтобы получить список ресурсов кандидатов. Однако подход на основе набора правил не предполагает возможность обучения, что может осуществляться с помощью Service MIB и CBR модуля, соответственно.

Провайдеру необходимо иметь такую конфигурацию управляющего решения на сервисном уровне, чтобы при любом прецеденте, любые изменения в конфигурации могли выполняться безопасно.

В связи со сложностью управления услугами, бывают ситуации, когда Service MIB и, следовательно, производные правила, не правильно отражают текущую ситуацию. Как следствие, RBR решатель не в состоянии сопоставить события так, как требуется, и события остаются некоррелированными или неправильно взаимосвязанными. В таких ситуациях модуль CBR применяется для содействия в поисках причины прецедента. В этом случае создаётся база данных предыдущих прецедентов. Закономерности описывающие текущую ситуацию в связи с предыдущей должны быть определены для того, чтобы применять аналогичные решения, которые отражают способность к обучению базы знаний CBR.

Набор рабочих событий представляет собой временное хранилище для событий, которые окончательно не были сопоставлены в RBR. Доступ к набору осуществляется при корреляции сервисов для получения связи между событиями. Событие содержится в рабочем наборе до тех пор, пока оно либо не скореллируется с каким-либо другим событием, либо не завершится временное окно заданное для успешной корреляции. В последнем случае событие обрабатывается в CBR движке или понижает порядок своей важности.

Установка времени действия рассмотрения события в RBR компоненте является важным вопросом для корреляции событий. На уровне ресурсов обычно общее соотношение окон используются для всех событий, так что

событие, которое существует больше определенного порога, будут удалены. Тем не менее, этот простой метод не рекомендуется для корреляции событий сервисов, поскольку события сервисов, как правило, имеют различные временные окна, в отличии от событий ресурсов.

Еще один вопрос, который не рассматривается на уровне ресурсов, является разрыв между временем, когда о событии будет сообщено и временем, возникновения события. Это время, как правило, значительное, так как пользователь должен решить, сообщать ли симптомы провайдеру. Это означает, что 15-минутная задержка не является чем-то необычным. В отличие от этого событие ресурсов может быть передано для корреляции событий ресурсов с задержкой порядка нескольких секунд или даже меньше. Рекомендуется использовать время возникновения события, как основу для корреляции, так как при этом правильный порядок событий может быть построен.

В рассматриваемой архитектуре, описание рассматриваемых рабочих процессов, в дальнейшем превращается в псевдокод алгоритма корреляции. В связи со сложностью, этот алгоритм разрабатывается, как правило, в виде нескольких этапов, отсевающих, один за другим, предположения сформулированные ранее. Наконец, данный алгоритм объясняет, как конструируются события сервисов и ресурсов, и какими правилами и прецедентами они создаются и управляются.

Условия работы основного алгоритма корреляции можно сформулировать в виде [13]:

- все ресурсы контролируются провайдером и, следовательно, основные прецеденты могут находится только в ресурсах;

- нет работ по техническому обслуживанию, влияющих на сервисы и ресурсы;

- вся информация об состоянии сервисов и ресурсов известна через существующие события;

- все события, связанные с одним штампом времени и сообщения о них поступают практически в тот же момент времени;

- есть только отдельные зависимости;

- существует только одно событие для сервиса или ресурса. Как следствие, ситуация, когда сервис работает для одного пользователя и не работает для другого, не моделируется;

- есть только бинарные состояния сервисов и ресурсов, без ухудшения качества;

- события не меняются при корреляции;

- зависимости не изменяются в процессе корреляции;

- существуют тесты, которые обнаруживают корректно ли сервис или ресурс работает. В результате этих тестов генерируются события с указанием состояния сервисов или ресурсов;

- зависимости или события моделируются корректным образом по отношению к сервисам и их функциональным возможностям.

Не смотря на все бесспорные преимущества такого подхода, у его реализаций, существующих на данный момент, имеют два существенных недостатка. Во-первых, параметры функционирования РИС задаются только в виде конкретных (явных) значений. Не предоставляется возможность использовать при решении нечёткие данные, которые описывают качественные параметры элементов системы. Во-вторых, результатом работы таких систем является либо одиночное предполагаемое решение, либо их множество, но без ранжирования по степени вероятности неисправности, что требует от оператора системы затрачивать дополнительное время на ручной анализ полученных результатов перед принятием окончательного решения.

Для преодоления этих недостатков, в работе предлагается разработать новые и усовершенствовать существующие модели и методы построения баз знаний с гибридной архитектурой для систем диагностирования РИС которые, с одной стороны, позволят учитывать при диагностировании нечёткие данные, с

другой стороны, проводить ранжирование предполагаемых неисправностей на выходе. Для решения этой задачи предлагается применение аппарата нечётких множеств и ведение в описание элементов РИС лингвистических переменных.

1.2. Информационные модели описания элементов РИС

При диагностировании неисправностей в РИС приходится иметь дело с разнообразной информацией, которая включает в себя детальное описание сервисов, ресурсов, клиентов, соглашений об уровне обслуживания (SLA), возможных неисправностей и др. Особое внимание должно уделяется моделированию зависимостей, которое крайне важно при отслеживании неисправностей от уровня сервисов до уровня ресурсов. Информационные модели для систем диагностирования РИС должны быть способны моделировать различные виды информации, необходимые для управления неисправностями сервисов с учётом специфики функционирования РИС. Существующие информационные модели, которые используются для управления распределёнными вычислительными системами во многом решают эти задачи, однако имеют ряд существенных ограничений. Обзор работ в этой области, включающей Web сервисы и исследовательские подходы, приведен в [56].

1.2.1 Internet Information Model

Internet Information Model разработана Internet Engineering Task Force (IETF) [57] и предоставляет огромный набор MIB переменных, организованных в дерево, которые используются для управления устройствами и их взаимодействия с помощью SNMP. Например, Event MIB [58], которая основана на RMON MIB, полезна для мониторинга ресурсов и соответствующего определения событий. Кроме того, большой набор конкретных поставщиков

MIB переменных существует в так называемых корпоративных MIB. Несмотря на усилия по интеграции этой информации со сервисами [59], модель явно ориентирована на управление только ресурсами.

1.2.2 Common Information Model

Common Information Model (CIM, [60]) разработана Distributed Management Task Force (DMTF), который является преемником Desktop Management Task Force. Первоначальная цель этой инициативы по стандартизации было детальное моделирование компьютерной системы, затем была распространена на проблемы сети. Цель CIM в том, чтобы полностью заменить SNMP в связи с его ограничениями. CIM содержит диаграммы классов, которые включают большое количество атрибутов и методов, и специфицированы в MOF–файлах (формат XML). Концепция WBEM данной инициативы предусматривает доступ к информации CIM, используя модуль CIMOM, для которого существует множество реализаций [61]. CIM в основном имеет дело с сетями и системами управления. Стандартизация информации о сервисах ограничена определениями атрибутов службы непосредственно связанных с атрибутами устройства.

В упрощённом виде CIM можно представить как способ, который позволяет нескольким участникам обмениваться информацией о распределенной вычислительной системе, которая необходима для управления её элементами. Упрощение состоит в том, что CIM не только определяет представление управляемых элементов и управляющей информации, но и предоставляет возможность управлять ими и контролировать и их работу. Преимущество использования CIM заключается в том, что управляющее программное обеспечение, созданное с использованием CIM, может работать со множеством реализаций этого стандарта без потери данных или сложных перекодировок.

CIM разработанный и опубликованный Distributed Management Task Force [62]. Связанный с ним стандарт Web-Based Enterprise Management (также

разработанный DMTF) [63], определяет реализацию CIM, включая протокол определения и доступа.

Ядром CIM является CIM-метамодель (далее CIM-метасхема). CIM-метасхема представляет собой базис, на котором описываются CIM-схемы и CIM-модели. Она описывает мета-элементы, имеющие атрибуты и связи при помощи понятий языка UML (Unified Modeling Language): UML-класс, UML-обобщение, UML-свойство и UML-ассоциация, и может быть выражена в виде UML-диаграммы. CIM-метасхема изображена на рисунке 1.4.

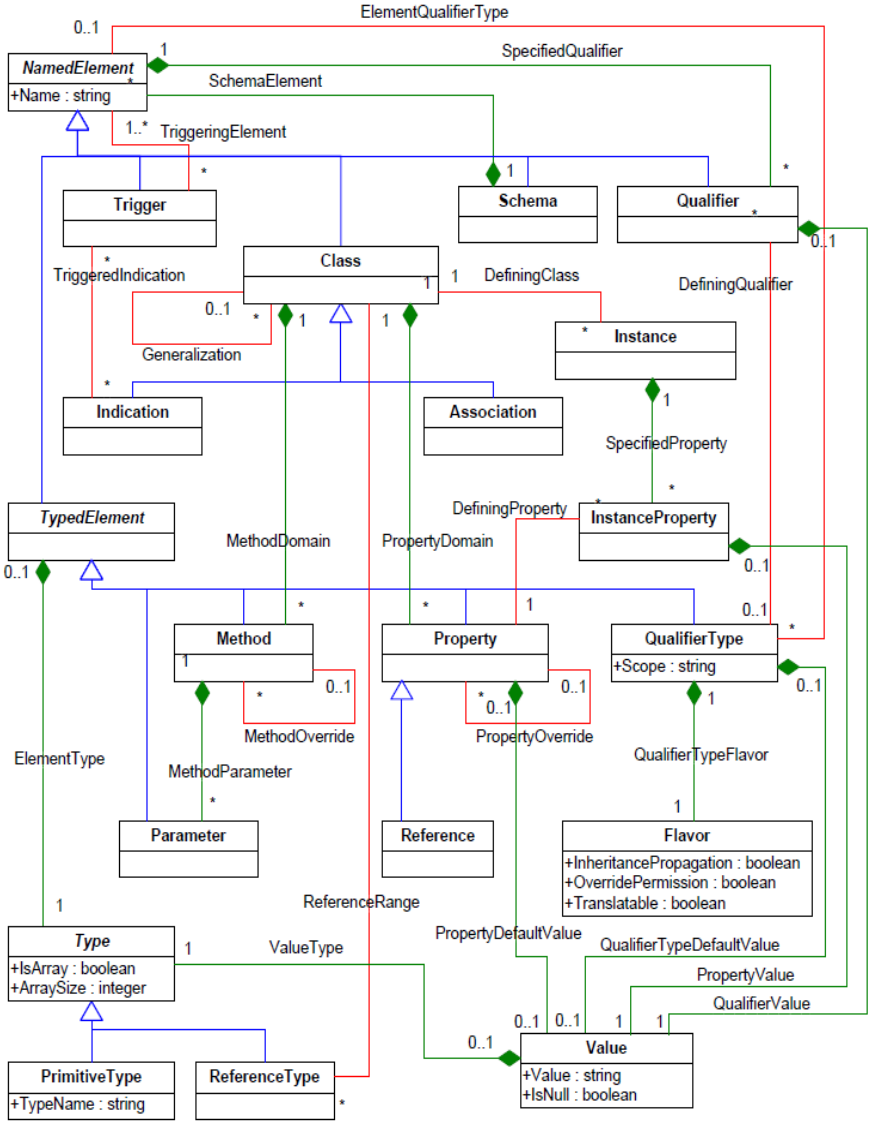


Рисунок 1.4 – CIM-метасхема

Для обеспечения информационного моделирования CIM разделена на базовую модели (Core Model), которая содержит основные классы и несколько расширений называемых Workgroup Models. Подводя итог, можно отметить, что CIM предоставляет более 1000 классов и атрибутов полезных для управления сетями и системами. Таким образом, семейство CIM-моделей является наиболее удачной информационной моделью для описания элементов РИС, благодаря гибкости выразительных средств и богатому набору готовых моделей.

1.2.3 Common Diagnostic Model

Для моделирования диагностических процессов в базе знаний отдельный интерес вызывает Common Diagnostic Model (CDM) - архитектура и методология для выражения системной диагностической аппаратуры через CIM-стандартные интерфейсы. Стандартизация этих интерфейсов означает, что клиенты, провайдеры и описания диагностических тестов увеличивают степень мобильности и, в большинстве случаев, нужно только один раз их описать чтоб обеспечить многоразовое использование в разных средах и на разных платформах.

CDM позволяет приложениям Diagnostic Client:

- выявлять, конфигурировать и выполнять диагностические тесты;
- контролировать выполнение тестов;
- рассматривать и управлять результатами тестов.

Схема CDM-модели представлена на рисунке 1.5. Центральным элементом CDM-модели является класс диагностического теста (Diagnostic Test), который ассоциирован с классами, описывающими элементы системы, которые диагностируются (ManagedElement), конкретные задачи, входящие в состав теста (ConcreteJob), компьютерные системы, на которых расположены диагностические системы (ComputerSystem), параметры диагностических тестов

(DiagnosticSettingData), журналы отчетов о выполнении диагностических тестов, и другие.

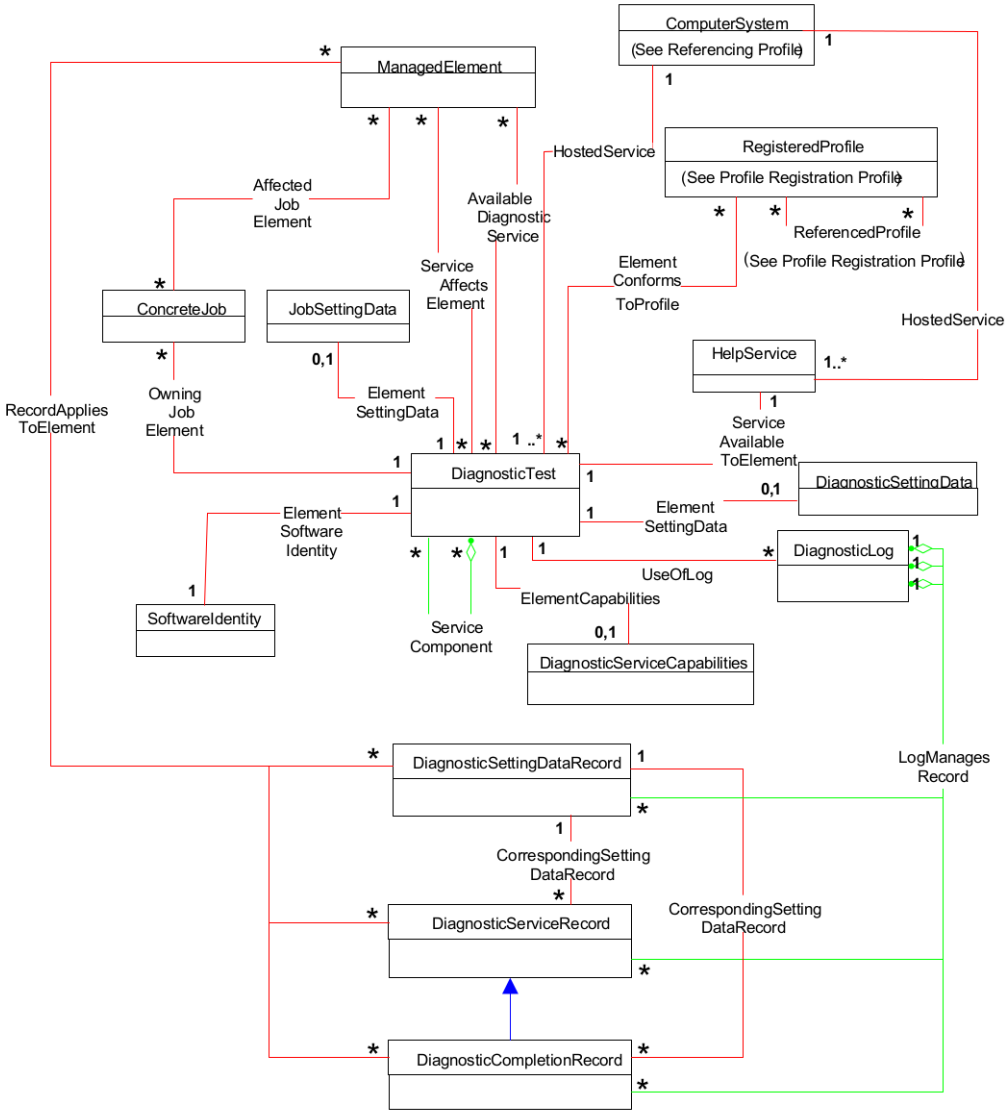


Рисунок 1.5 – CDM - модель.

Таким образом, CDM позволяет вводить в базу знаний системы диагностирования описания диагностических тестов, субъекты и объекты их выполнения, составные части и результаты в стандартизированном виде в терминах CIM-модели.

1.2.4 Сервисная модель MNM

CIM модель является широко распространенной информационной моделью, которая, в частности, полезна для моделирования ресурсов, однако класс CIM сервиса не может быть использован здесь поскольку в нём сервис понимается как процесс, выполняющийся на одном хосте. Поэтому, в работе [13], для моделирования сервисов предлагается модель MNM.

Базовая сервисная модель MNM содержит основные классы, которые необходимы для моделирования диагностики сервиса. Центральным элементом базовой модели является класс сервиса (Service), который ассоциирован с классами, описывающими функциональность (ServiceFunctionality) и параметры качества сервисов (QoSParameter). Эти два класса имеют важное значение для сервиса, так как при описании сервиса необходимо описать функциональность, которую он предоставляет, а параметры качества сервиса характеризует особенности сервиса.

Класс ресурс (Resources) определяется вместе с классом QoR. Термин «качество ресурса (QoR)» принят от термина параметра "качество устройства" и описывает особенности ресурса, которые могут иметь значение для параметра QoS. Таковы, например, параметры «загрузка процессора» и «потребление памяти устройства» при ссылке на них.

Класс данной модели, разработан для нужд интеллектуальных компонентов и, следовательно, сосредоточен на различных видах зависимостей, которые могут возникать при функционировании сервиса. Он также включает в себя атрибуты для настройки сущностей, которые необходимы для моделирования компонентов сервиса.

Цель информационного моделирования – обеспечить различную глубину моделирования для параметров качества сервисов (QoS), зависимостей и точек доступа к сервисам (SAP). Параметры QoS могут быть определены для сервиса в целом (например, доступность сервиса) или специально для функций сервиса

(например, временные условия для выполнения транзакции). Таким образом, зависимости могут быть смоделированы для сервиса в целом или для его функций. Моделирование дает возможность иметь различные SAP-ы, которые могут быть привязаны к сервису (если вся функциональность сервиса может быть доступна через этот SAP).

На рис. 1.6 представлена базовая сервисная модель MNM в виде UML-диаграммы классов, которая содержит основные классы, необходимые для моделирования диагностики сервиса.

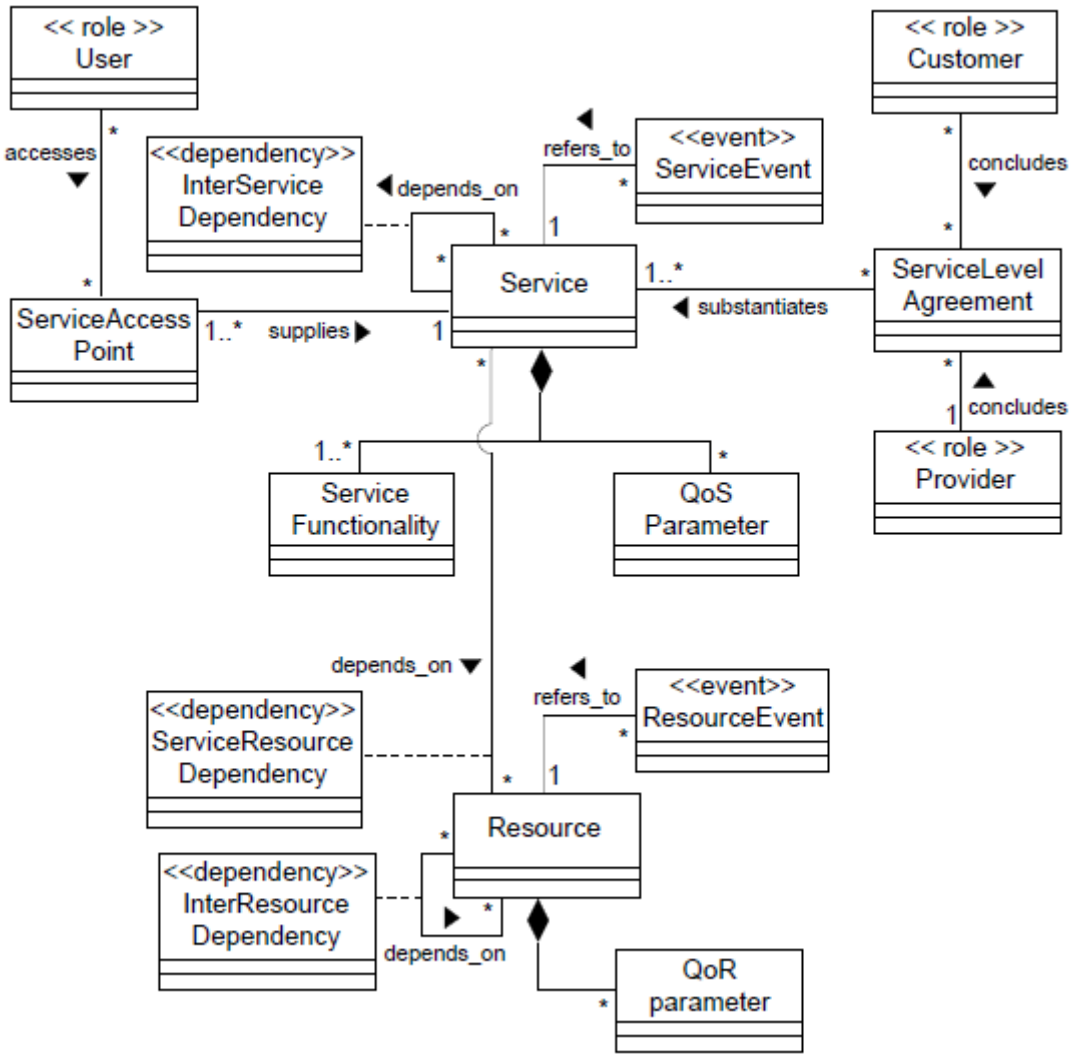


Рисунок 1.7 – Базовая сервисная модель MNM

Применение сервисной модели MNM при построении баз знаний систем диагностики РИС позволяет описывать РИС не только на уровне ресурсов, а и на более высоком уровне — сервисов что, в свою очередь, существенно расширяет множество доступных для диагностирования параметров и зависимостей между ними.

1.3. Моделирование зависимостей между сервисами и ресурсами.

Для диагностирования неисправностей сервисов, помимо описаний самих сервисов, необходимы средства моделирования зависимостей, которые описывают сложные взаимодействия на сервисном уровне и на уровне ресурсов. Это необходимо, для автоматического определения связи признака неисправности на сервисном уровне с ресурсами, которые используются для реализации этого сервиса.

Несмотря на важность зависимостей для диагностики неисправностей, зависимости и их особенности часто описываются лишь поверхностно. В дополнение к модели для телекоммуникаций [64], ядро CIM модели является одним из немногих стандартов позволяющим явно определять зависимости. Ядро CIM модели содержит общий класс зависимостей от которого другие классы наследуются. Однако, зависимости почти не содержат атрибутов и не привязаны к сервисам.

Термины antecedent – обозначающий объект, от которого зависят другие объекты и dependent – обозначающий объект, который зависит от других объектов которые используются в CIM могут применяться для описания связей сервисов и ресурсов [13]. Графы зависимостей, которые строятся из CIM зависимостей используются в [65] для определения неисправностей. Расширение CIM для диагностики неисправностей и анализа влияния на телекоммуникационные услуги, рассматривается в [66]. Графы зависимостей представляют общую концепцию организации зависимостей для диагностики

неисправностей. В [67, 68] общий подход для применения такого графа дается. Согласно [69, 70] множество взаимных зависимостей на сервисном уровне, как правило, показатель их плохого проектирования. В данной работе модели зависимостей делятся на функциональные, структурные, и оперативные. Для моделирования зависимостей используется RDF. Есть только примеры потенциальных параметров зависимостей, такие как сила (strength) (вероятность того, что компонент выйдет из строя, если связанный с ним компонент вышел из строя).

В [71] при описании зависимостей для сервисов, которые предоставляются интернет-провайдерами сервисов, было введено несколько видов зависимостей:

- зависимость выполнения (execution dependency),
- ссылочная зависимость (link dependency),
- межсервисная зависимость (inter-service dependency)
- организационная зависимость (organization dependency).

Методология обнаружения этих зависимостей представлена в [72].

Атрибут силы для зависимостей определяется в [73] и имеет значения «сильный», «средний», «слабый» или «отсутствует».

В целом, моделирование зависимостей для сервисов, не может рассматриваться как удовлетворительное, поскольку оно не направлено на удовлетворение потребностей сервис-ориентированного подхода. Предложенная в работах [74, 75] структура, нацеленая на решение этой задачи, изображена на рис. 1.7, в виде иерархии зависимостей с использованием композиционного паттерна.

Нахождение зависимостей на практике, как правило, является сложной задачей. В [76] описаны несколько методов для обнаружения неисправностей. Для зависимостей, которые представляют собой самую важную часть этих моделей, запрос информации из существующих источников информации обычно ограничивается зависимой технологией. Например, зависимости,

связанные с DNS не могут быть сохранены непосредственно, но могут быть определены из файлов, таких как "resolve.conf". Для IP-сетей физической топологии, MIB [77-79] может быть использован в качестве источника информации.

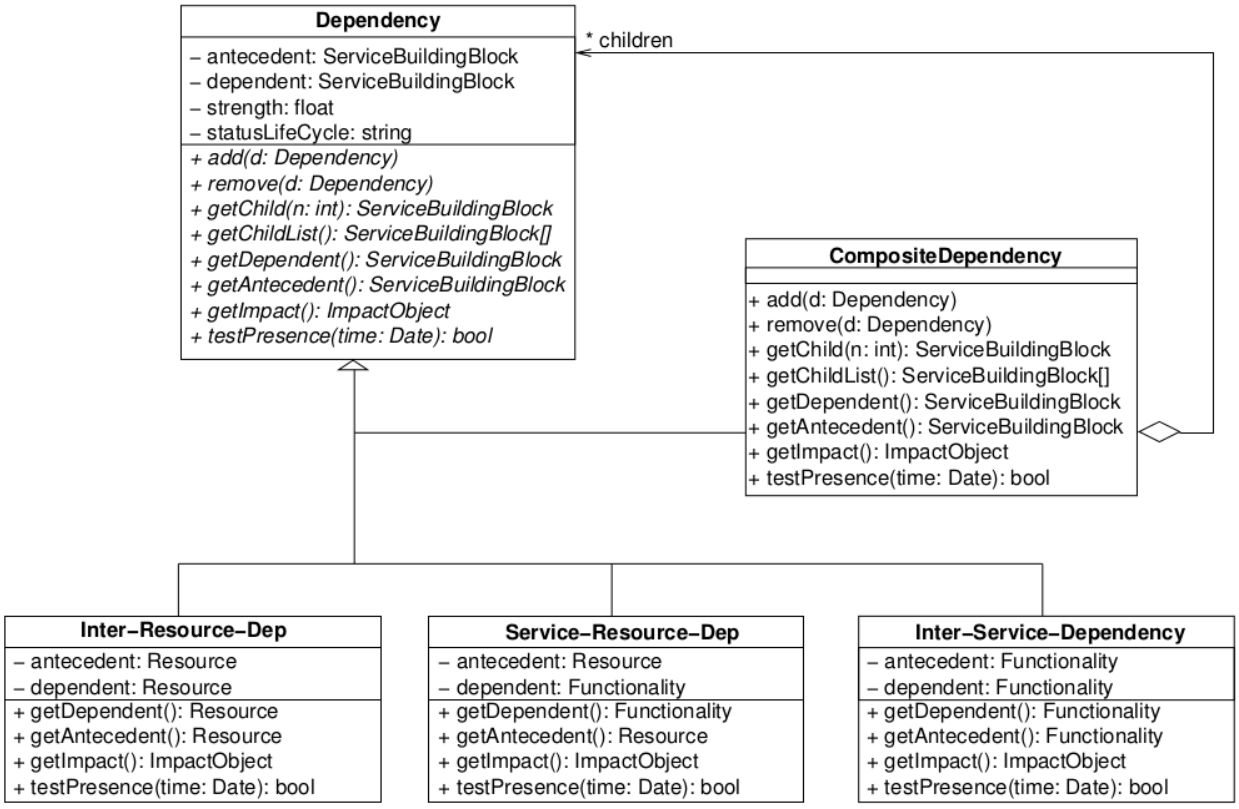


Рисунок 1.7 – Иерархия зависимостей с использованием композиционного паттерна

Когда методы для запроса существующей информации не возможно осуществить, должны быть определены методы позволяющие установить факты зависимостей. Application Response Measurement (APM) [80] может быть использован, чтобы обеспечивать дополнительную информацию, при решении этой задачи. Набор библиотек для решения этого вопроса приведен в [81]. В соответствии с подходом, который используется для определения зависимостей [82, 73], критические узлы идентифицируются и мониторятся. Последствия

неисправностей в узлах контролируются, а изменения в поведении системы используется для определения силы зависимостей.

Пассивный метод нахождения зависимостей путем анализа взаимодействий дается в работах [83, 84]. Он использует данные, доступные на узлах. Кроме того изучение данных о трассировке сообщений было использовано в [85]. При этом, число взаимодействий используется в качестве индикатора силы. Дополнительные публикации показывают влияние неточности моделирования на точность определения неисправности [86]. Эти неточности возникают из-за отсутствия зависимостей или ложных зависимостей.

В работах [87, 88] предложено использовать подход, основанный на нейронной сети. Для каждой пары ресурсов в сети контролируется активность с использованием таких показателей, как загрузка процессора, использование полосы пропускания или их комбинаций. Кривые деятельности вводятся для нейронной сети, которая решает, существует ли связь между узлами.

Существуют также подходы [89-91], основанные на интеллектуальном анализе данных из файлов журнала событий с целью выявления этих зависимостей.

В работах [92-97] представлены другие изученные информационные модели, которые не вошли в этот раздел.

Проведённый анализ показал важность более глубокого описания зависимостей меду элементами РИС чем простая её констатация. Создание гибких моделей описания зависимостей позволит расширить возможности для автоматического, автоматизированного и «ручного» описания моделей РИС для решения задач диагностики.

1.4. Подходы и языки описания знаний

1.4.1 Онтологический подход

Несмотря на все неоспоримые преимущества моделей семейства CIM, следует отметить, что все они представлены в виде UML- моделей и не могут напрямую быть использованы в механизмах рассуждения. CIM-модели изначально ориентированы на базы данных, а не на базы знаний. При этом, большинство существующих подходов к описанию знаний существенно усложняют процесс поддержания актуальности базы знаний поскольку требуют от эксперта-диагноста оперировать сложными и непривычными понятиями из области управления знаниями. Таким образом, необходимо было выбрать подход который, одной стороны, позволял бы описывать диагностические аспекты РИС в приемлемом для хранения и проведения логического вывода виде, с другой стороны, сохранял гибкость и простоту использования при автоматизированном или «ручном» заполнении и поддержании в актуальном состоянии базы, которая присуща CIM-моделям.

Анализ существующих подходов описания знаний показал [98], что только онтологический подход позволяет описывать знания на различных уровнях абстракции, что существенно упрощает процесс модификации знаний в базах знаний, построенных на их терминологической базе. Тем самым позволяет отказаться от привлечения эксперта из баз знаний при поддержке ее актуальности в условиях высокой динамики развития РИС. В связи с этим, было принято решение применить онтологический подход при разработке модулей модели базы знаний.

Онтологические модели составляют наиболее общие концептуальные понятия моделируемой области, которые полностью абстрагированы от конкретных моделей представления знаний и практической реализации. Применение онтологических моделей при формализации базовых категорий

предметной области имеет ряд преимуществ, среди которых основными являются универсальность, применимость на различных уровнях детализации и др.

Таким образом, онтологическая модель представляет собой систему логических смысловых соотношений, с которыми согласны эксперты, а вся система представляет явное описание предметной области. Такая модель предметной области является основой многоуровневой информационной модели.

Такой подход позволяет сократить не только время на разработку базы знаний, за счёт использования готовых смысловых шаблонов на разных уровнях абстракции, но и время на поддержание актуальности рабочей базы знаний, за счёт возможности у эксперта-диагноста оперировать привычными, при этом формализованными понятиями.

1.4.2 Web Ontology Language

Для формализованного описания онтологий существует множество языков, как ориентированных на использование человеком, так и ориентированные на программную обработку. Для описания онтологий для систем диагоностирования РИС, нужен язык обладающий с одной стороны поддержкой современных эффективных решателей, с другой стороны простотой в использовании при разработке и расширении терминологической базы. Именно благодаря такому сочетанию свойств, наиболее популярным на данный момент, является OWL.

Web Ontology Language (OWL) – язык описания онтологий, ориентированный на обработку, в первую очередь, приложениями. OWL может использоваться, чтобы явно представлять значения терминов и отношения между этими терминами в словарях[99]. OWL имеет больше средств для выражения значения и семантики, чем XML, RDF, и RDF-S, и, таким образом,

OWL идет дальше этих языков в способности представить поддающийся машинной обработке онтолгии.

OWL обеспечивает три различных по выразительности диалекта, спроектированных для использования отдельными группами разработчиков и пользователей:

– OWL Lite предназначен для тех пользователей, которые нуждаются, прежде всего, в классификационной иерархии и простых ограничениях. Например, притом что он поддерживает ограничения кардинальности, допускаются значения кардинальности только 0 или 1. Разработчикам проще обеспечить в своих продуктах поддержку OWL Lite, чем более выразительные диалекты, в частности, OWL Lite позволяет быструю миграцию тезаурусов и других таксономий. OWL Lite также имеет более низкую формальную сложность, чем OWL DL.

– OWL DL предназначен для тех пользователей, которые хотят максимальной выразительности при сохранении полноты вычислений, и разрешаемости. OWL DL включает все языковые конструкции OWL, но они могут использоваться только согласно определенным ограничениям (например, в то время как класс может быть подклассом многих классов, класс не может быть представителем другого класса). OWL DL так назван из-за его соответствия дескриптивной логике, дисциплине, в которой разработаны логики, формирующие формальную основу OWL.

– OWL Full предназначается для пользователей, которые хотят максимальную выразительность и синтаксическую свободу RDF без гарантий вычисления. Например, в OWL Full класс может рассматриваться одновременно как собрание индивидов и как один индивид в своем собственном значении. OWL Full позволяет такие онтологии, которые расширяют состав предопределенного (RDF или OWL) словаря.

Каждый из этих диалектов - расширение его более простого предшественника, и в том, что касается выразительных возможностей, и в том, что касается возможностей производимых заключений, но не наоборот.

1.4.3 Многосортный язык прикладной логики

Для описания онтологий, которые предполагают изменения значений параметров во времени, язык OWL не применим, поскольку он основывается на дескриптивной логике, которая является монотонной. Для преодоления этих ограничений, может быть использован многосортный язык прикладной логики [100-103].

Необходимость этого языка обусловлена следующими обстоятельствами. Язык классической математической логики - язык исчисления предикатов - был разработан, для достижения двух во многом противоречащих друг другу целей: во-первых, как средство описания состояний порождающего процесса в исчислении предикатов, во-вторых, как средство формализации идей.

Язык описания состояний порождающего процесса (множеств формул) в исчислении предикатов должен иметь такой семантический базис (совокупность символов языка, семантика которых не зависит от интерпретации символов сигнатуры), чтобы исчисление предикатов, обладающее свойством полноты, содержало конечное множество правил вывода. Семантический базис классического языка исчисления предикатов образуют пропозициональные связки и кванторы. Кроме того, в семантический базис языка неявно входят множества (области значений переменных), декартово произведение множеств (области определения функций и предикатов являются декартовыми произведениями), а также функциональные отображения (интерпретациями функциональных и предикатных символов являются элементы множества функциональных отображений). Правила вывода исчисления предикатов обычно соответствуют семантике пропозициональных связок и кванторов.

При использовании языка исчисления предикатов как средства формализации идей каждую логическую теорию можно рассматривать как способ интенсионального задания множества функций интерпретации, имеющих одну и ту же конечную область определения - сигнатуру языка.

В этом случае логический язык должен содержать такой семантический базис, чтобы с помощью конечного множества предложений можно было задать возможно более точную аппроксимацию любого подразумеваемого множества функций интерпретации. Понятно, что чем шире такой семантический базис, тем в большем числе случаев может быть достигнута эта цель.

В реальных приложениях при задании декларативных моделей (систем алгебраических, дифференциальных и других уравнений, а также систем неравенств и задач оптимизации) этот семантический базис включает, как минимум, арифметику. Однако, известно, что исчисление предикатов, обладающее свойством полноты и основанное на таком языке, не может содержать конечного множества правил вывода.

Таким образом, сосуществуют как бы несколько логических языков: одни (для описания формул) с ограниченным семантическим базисом, а другие (как средство формализации идей) с расширенным семантическим базисом. Языки первой группы являются средством представления состояний порождающего процесса исчисления предикатов, изучаемого в рамках математической логики, а остальные - средством представления декларативных моделей, изучаемых в чистой и прикладной математике.

Определение языка содержит только описание ядра. При расширении семантического базиса для конкретных приложений допустимы два класса элементов: элементы первого класса невозможно или нежелательно определять средствами ядра языка и уже построенных расширений, напротив, элементы второго класса могут быть естественно определены средствами ядра языка и уже построенных расширений.

Элементы первого класса описываются в стандартном и специализированных расширениях языка в такой же форме, которая принята и при описании ядра языка. Стандартное расширение языка определяет элементы семантического базиса, относительно которых можно предположить, что они будут полезны практически во всех приложениях.

Специализированное расширение языка определяет элементы семантического базиса, которые необходимы для сравнительно узкого класса приложений. Поскольку одни и же специализированные расширения могут быть полезны в различных приложениях, такие расширения имеют названия. Каждый конкретный язык прикладной логики включает ядро, а также обычно стандартное расширение и некоторые специализированные расширения. Таким образом, каждый конкретный язык прикладной логики характеризуется некоторой совокупностью названий расширений, а не сигнатурой. Сигнатура же вводится в каждой конкретной логической теории, заданной на таком языке. При этом предложения этой теории могут сопоставлять элементам сигнатуры (именам) их значения (интерпретацию) или сорта, либо ограничивать возможные функции интерпретации этих имен в зависимости от интерпретации других имен. Каждая теория в свою очередь имеет имя, параметрами которого являются имена расширений языка, на котором эта теория записана. Элементами теории, кроме предложений, могут быть и другие теории, представленные их именами.

Ядро языка прикладной логики.

Синтаксис термов. Термом является:

1. имя n;

2. переменная v;

3. N и L;

4. $t_1 \rightarrow t_2$, где t_1 и t_2 - термы;

5. $(\times t_1, t_2, ..., t_k)$, где $t_1,...,t_k$ - термы;

6. $t(t_1,...,t_k)$, где t , $t_1,...,t_k$ - термы;

7. j(t), где t терм.

Синтаксис формул. Формулами являются:

1. $t(t_1,...,t_k)$, где t , $t_1,...,t_k$ - термы

2. $\neg f_1$, $f_1 \& f_2$, $f_1 \vee f_2$, $f_1 \Rightarrow f_2$, $f_1 \Leftrightarrow f_2$, где f_1 и f_2 формулы.

Если переменная не является связанной в терме или формуле, то она считается свободной в этом терме или формуле. Если переменная связана в терме или формуле, то она связана и в предложении, куда входит этот терм или формула. В ядре языка нет связанных переменных.

Синтаксис предложений. Предложение состоит из префикса и тела. Префикс есть последовательность описаний переменных

$$(v_1:t_1)\ (v_2:t_2)...(v_m:t_m)$$

(ограниченных кванторов всеобщности), где $(v_i:t_i)$ - описание переменной, v_i - переменная, t_i - терм для всех i=1,...,m.

Тело предложения зависит от типа этого предложения. Типами предложений являются: описание значения имени, описание сорта имени, ограничение на интерпретацию имен. Любая свободная переменная, входящая в тело предложения, должна быть описана в его префиксе. Если переменная является связанной в теле предложения, то она не может входить в префикс этого предложения.

Тело описания значения имени имеет вид: $t_1 \equiv t_2$, где t_1, t_2 - термы.

Тело описания сорта имени имеет вид: $\chi(t_1)=t_2$, где t_1, t_2 - термы.

Тело ограничения на интерпретацию имен является формулой.

Семантика термов и формул определяет значения термов и формул, а также условия, при которых эти значения существуют. При этом предполагается, что на множестве имен задана функция α, значение которой для каждого имени есть интерпретация этого имени.

Значения термов и формул будут определены относительно функции интерпретации α и произвольной допустимой подстановки θ значений вместо всех свободных переменных терма или формулы.

Выводы по первому разделу

1. Проведен анализ существующих систем диагностирования РИС, основанных на знаниях, а так же подходов, которые применяются для их построения. По итогам был сделан вывод, что существующие подходы обладают двумя существенными недостатками. Во-первых, они не позволяют использовать нечёткие данные, которые описывают качественные параметры элементов системы, во-вторых, не позволяют проводить ранжирование множества предполагаемых диагнозов по степени их вероятности.

2. Проведён анализ информационных моделей описания элементов РИС и зависимостей между ними. Результаты показали, что динамичность развития, присущие РИС, требует уделять особое внимания способности базы знаний к лёгкости поддержания её в актуальном состоянии, и вводит дополнительные ограничения при выборе подходов и информационных моделей для описания элементов РИС и зависимостей между ними. Существующие информационные модели такие как CDM и MNM из семейства CIM, MIB и другие, хоть и обладают широкими возможностями в описании диагностических аспектов РИС, тем не менее ориентированны на создание баз данных, а не баз знаний и не предназначены для применения в механизмах логического вывода.

3. Анализ подходов и языков описания знаний показал, что большинство из них являются чересчур сложными для использования не экспертами по базам знаний, что делает невозможным их применение при диагностировании РИС, поскольку прикладная база знаний в этой предметной области постоянно требует адаптации к постоянно меняющимся условиям. Только онтологический подход позволяет строить базы знаний в терминах доступные диагностам, при этом позволяет строить логические выводы на основе своих описаний.

РАЗДЕЛ 2
РАЗРАБОТКА СТРУКТУРЫ БАЗЫ ЗНАНИЙ И ОБОБЩЕННОЙ МОДЕЛИ ДИАГНОСТИКИ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

2.1 Разработка структуры базы знаний системы диагностирования РИС

Опираясь на проведённый анализ в первом разделе, были сформулированы основные требования к архитектуре баз знаний систем диагностирования РИС.

В первую очередь структура должна быть модульной и иерархичной, для обеспечения гибкости и простоты при построении и поддержании актуальности базы знаний. Модульность базы знаний позволяет проводить независимое управление знаниями о конкретных аспектах предметной области, проводить обобщение и адаптацию элементов знаний при построении и обновлении прикладных баз знаний. Модульный иерархический подход позволяет оперативно расширять модель описания знаний дополнительными модулями на архитектурном уровне.

Кроме того, как показал анализ приведенный в пункте 1.4.1, в основу разработки модулей высокого уровня, которые описывают базовые концептуальные понятия, должен закладываться онтологический подход. Он позволит описать базовые термины и понятия предметной области в формализованном и пригодном для автоматизированных рассуждений виде, не нарушая гибкости всей архитектуры базы знаний.

Структура базы знаний, так же должна учитывать специфику гибридного подхода к рассуждениям, описанного в пункте 1.1.4. База знаний должна учитывать возможность проведения рассуждений как на основе RBR, так и на основе CBR рассуждений.

Структура модулей должна обеспечивать возможность описания нечётких диагностических данных на всех уровнях модели, для полноценного их вовлечения в процесс обнаружения, локализации и классификации неисправности.

И в завершение, модули базы знаний должны позволять использовать стандартные существующие информационные модели для описания диагностических аспектов РИС, описанных в подразделе 1.2.

Базовая структура базы знаний строилась в виде иерархии модулей, описывающих различные аспекты диагностики РИС на разных уровнях абстракции. Предложенная структура базы знаний предусматривает 4 модуля, при этом на верхнем уровне используется модуль обобщенной модели диагностики РИС, а на среднем и нижнем уровнях - модули, представленные онтологическими моделями по принципу степени абстракции их описания.

Разработанная структура базы знаний представлена на рисунке 2.1.

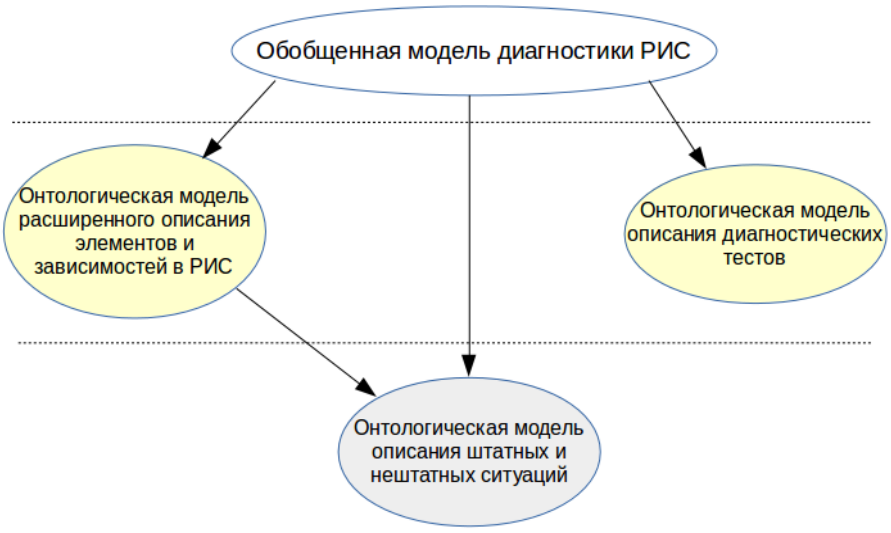


Рисунок 2.1 – Структура базы знаний для диагностики РИС

На верхнем уровне абстракции диагностические аспекты РИС описывает обобщенная модель диагностики РИС. Она вводит основные понятия для описания диагностики РИС. Эти понятия используются для построения

моделей низшего уровня. Применение аппарата нечетких множеств в этой модели позволяет использовать для диагностирования системы экспертные знания о нечетко заданных параметрах и уменьшить время на описание возможных диагностических ситуаций.

На втором уровне абстракции располагаются онтологические модели расширенного описания элементов и зависимостей в РИС и онтологическая модель описания диагностических тестов. Они расширяют обобщенную модель диагностики РИС, вводя дополнительные термины и зависимости. Онтологическая модель расширенного описания элементов и зависимостей в РИС предоставляет более выразительные средства для описания зависимостей между сервисами РИС и множествами диагностических параметров и уменьшает, таким образом, время, которое тратят эксперты на поддержку актуальности базы знаний в условиях высокой динамики развития РИС. Введенные понятия используются в онтологических моделях более низкого уровня абстракции.

Онтологическая модель описания диагностических тестов вводит основные понятия для описания диагностических тестов как источников диагностических данных в случае, если первоначальный набор диагностических данных недостаточен для диагноза.

На третьем уровне абстракции располагается онтологическая модель описания знаний о штатных и нештатных ситуациях. Данная модель вводит базовые термины, используемые для описания знаний предметной области, а также ограничений на их значения. Она позволяет описывать знания о штатных ситуациях, которые будут использоваться для определения нормального состояния функционирования сервисов, а также знания о вариантах нештатных ситуаций и их возможных причинах для определения списка наиболее вероятных причин неисправности, если фиксируется нештатная ситуация.

Такая структура базы знаний позволяет создавать гибкие и простые в создании и обслуживании базы знаний, которые способны учитывать не только

количественные, но и качественные параметры элементов РИС, в виде нечётких данных, при применении их в системах диагностирования РИС. Базовая структура может расширяться за счёт добавления новых модулей, основанных на понятиях и терминах уже существующих модулей.

2.2 Усовершенствование обобщенной модели диагностики РИС

На самом верхнем, первом, уровне абстракции, структура базы знаний предполагает обобщенную модель диагностики, которая задаёт основные понятия диагностики, используемые на более низких уровнях. Существующие модели [104-105] представления диагностических аспектов РИС не предполагают хранение и использования нечетких данных о параметрах РИС и, тем самым, ограничивают возможности системы диагностики. Усовершенствование существующих обобщенных моделей диагностики позволит, с одной стороны, расширить набор знаний, которые могут использоваться при решении задач диагностики, с другой стороны, позволят проводить ранжирование предполагаемых диагнозов по степени их вероятности.

Поскольку, основными структурными элементами современных РИС являются сервисы, то неисправное состояние сети передачи данных ведет к невозможности выполнять некоторыми сервисами заданные функции. Под сервисом в данной работе понимается взаимодействующие посредством сети функциональные компоненты, составные части которых могут выполняться в отдельных аппаратно-программных окружениях.

В усовершенствованной модели совокупность сервисов подаётся в виде множества

$$S = \{s_l\}_{l=1}^{L}, \qquad (2.1)$$

где L – общее количество сервисов.

В соответствии с работой [101] вводится определение источника диагностической информации. Источником диагностической информации (ИДИ) называется компонент РИС, предоставляющий необходимую для определения состояния сервиса информацию в виде диагностических параметров.

Предполагается, что в качестве источников диагностической информации будут рассматривать только те компоненты, которые предоставляют доступ к значениям диагностических параметров посредством протокола SNMP.

Совокупность источников диагностической информации подаётся в виде множества

$$A = \{a_i\}_{i=1}^{I}, \tag{2.2}$$

где I – общее количество источников диагностической информации в рассматриваемой сети. Каждый источник, поддерживает одну или несколько MIB и может реализовать описанные в ней SNMP-объекты управления в виде переменных, которые содержат текущие значения соответствующих параметров. Совокупность диагностических параметров некоторого источника диагностической информации $a \in A$ в задаётся виде

$$P(a) = \{p_j(a)\}_{j=1}^{J(a)}, \tag{2.3}$$

где $J(a)$ - общее количество диагностических параметров источника диагностической информации $a \in A$.

В соответствии с рассматриваемой диагностической моделью в процессе функционирования РИС из множества $P(a)$ формируется множества диагностических параметров $\{B(s_l)\}_{l=1}^{L}$, значения которых в дальнейшем

необходимо учитывать в процессе идентификации состояния некоторой сервиса $s_l, l = 1,...,L$. Совокупность диагностических параметров $B(s)$ ($s \in S$) задаётся в виде

$$B(s) = \{b_m(s)\}_{m=1}^{M(s)}, \qquad (2.4)$$

где $M(s)$ - общее количество диагностических параметров сервиса $s \in S$.

Если для всех сервисов известны множества $B(s_l)$, то для любого диагностического параметра b можно определить множество сервисов $S(b)$, в которых параметр b используется в процессе определения ее состояния.

Однако, учитывая, что тип переменной, которая реализует SNMP-объект управления, в большинстве случаев является целочисленным на интервале [1, 4294967295] то идентифицировать такое количество состояний для сохранения и проведения последующего анализа является трудоемкой процедурой и нецелесообразно. На практике все множество значений параметра на конечное число подмножеств. Каждое подмножество включает, значения которые параметр b принимает при одном и том же состоянии сервиса $s_l \in S(b)$.

Все множество значений $D(b)$ параметра b разбивается на конечное число подмножеств

$$D(b) = \bigcup_{k=1}^{K(b)} D_k(b), \qquad (2.5)$$

где $K(b)$ – общее количество подмножеств для параметра b. Причем каждое подмножество $D_k(b)$ включает значения параметра b, которые он принимает при одном и том же состоянии сервиса $s_l \in S(b)$.

Для использования лингвистического описания значений параметра b необходимо построить полные ортогональные семантические пространства

(ПОСП), которые будут служить областями лингвистических значений каждого из параметров.

Для построения ПОСП некоторого параметра b в соответствии с формулой (2.5) определим множества нечетких значений $\overline{D(b)} = \{\bar{b}^k\}_{k=1..K(b)}$, где $K(b)$ - количество нечетких значений, принимаемых параметром, в виде нечетких чисел с трапецеидальной функцией принадлежности μ^k, которая положительно определена на некотором интервале $(p_b^k(b), p_e^k(b))$, зависящим от $D_k(b)$.

Для того чтобы построенные множества $\overline{D(b)}$ являлись ПОСП, необходимо, чтобы они удовлетворяли следующим аксиомам [106]:

Аксиома 1 – нормальность: каждая функция принадлежности μ^k нечетких значений $\bar{b}^k$ достигает единицы на некотором ненулевом отрезке значений своего базового множества $D(b)$.

Аксиома 2 – функция μ^k не убывает слева от $p_b^k(b)$ и не возрастает справа от $p_e^k(b)$.

Аксиома 3 – функции μ^k не могут иметь более двух точек разрыва первого рода.

Аксиома 4 – полнота: для любого значения b из базового множества $D(b)$ найдется нечеткое значение $\bar{b}^k$ с ненулевым значением функции принадлежности $\mu^k(b)$ в данной точке.

Аксиома 5 – ортогональность: сумма всех значений функций принадлежности μ^k в некоторой точке базового множества $x \in D(b)$ должна быть равна единице, т.е.

$$\sum_{k=1}^{K(b)} \mu^k(x) = 1.$$

Каждое нечеткое число $\bar{b}^k \in \overline{D(b)}$ определено через функцию принадлежности следующего вида:

$$\bar{b}^k \Rightarrow \mu^k(x) = \begin{cases} 0, & x \le p_b^k, x \ge p_e^k \\ \dfrac{x - p_b^k}{p_{b_1}^k - p_b^k}, & p_b^k < x < p_{b_1}^k \\ 1, & p_{b_1}^k \le x \le p_{e_1}^k \\ \dfrac{x - p_e^k}{p_{e_1}^k - p_e^k}, & p_{e_1}^k < x < p_e^k \end{cases}, \quad (2.6)$$
$$k = 1..K(b)$$

где x – некоторое четкое значение параметра, p_b^k, p_e^k – начальное и конечное значения соответственно интервала значений базового множества $D_k(b)$, на котором функция принадлежности *k*-го нечеткого значения параметра положительно определена, $p_{b_1}^k, p_{e_1}^k$ – начальное и конечное значения соответственно интервала значений базового множества $D_k(b)$, на котором функция принадлежности *k*-го нечеткого значения параметра b равна единице.

Можно показать справедливость следующего утверждения:

Утверждение. Если множество нечетких значений $\overline{D(b)} = \{\bar{b}^k\}_{k=1..K(b)}$ удовлетворяет соотношениям (6), то оно является ПОСП.

Для формализованного использования параметров системы, определим для них лингвистические переменные (ЛП):

$$p_m^l = < n_m^l, \{\bar{b}_m^k(s_l)\}_{k=1,\dots,K(b_m(s_l))}, D(b_m(s_l)) >, \quad (2.7)$$

где $n_m^l, \{\bar{b}_m^k(s_l)\}_{k=1,\dots,K(b_m(s_l))}, D(b_m(s_l))$ – соответственно имя, терм-множество и базовое множество *m*-го параметра *l*-ого сервиса.

Предположим теперь, что для некоторого диагностического параметра сервиса $s_l \in S(b)$, можно задать разбиение

$$D(b_m) = D_l^0(b_m) \cup D_l^1(b_m) \quad (2.8)$$

такое, что значения из $D_l^0(b)$ соответствуют работоспособному состоянию сервиса $s_l \in S(b)$, а значения из $D_l^1(b)$ – неработоспособному состоянию сервиса. Данное разбиение позволяет ввести следующее определение характеристической лингвистической переменной сервиса $s_l \in S(b)$.

Определение. Характеристической лингвистической переменной сервиса $s_l \in S(b_m)$ называется такая переменная p_m^l из множества (2.7), что:

1. базовое множество *m*-го параметра *l*-ой сервиса удовлетворяет (2.8)
2. p_m^l представима в виде

$$p_m^l = p_m^l(N) \cup p_m^l(S)$$

где $p_m^l(N)$ лингвистическая переменная, терм-множество которой состоит из термов соответствующих работоспособному состоянию сервиса, а $p_m^l(S)$ лингвистическая переменная терм-множество которой состоит из термов соответствующих не работоспособному состоянию сервиса.

На множестве всех возможных ситуаций для сервиса $s_l \in S(b)$ в РИС определены класс штатных и класс нештатных ситуаций.

Определение. Нечеткой ситуацией называется нечеткое множество второго уровня [107]:

$$\Theta = \left\{ \mu(p_m^l) \Big/ p_m^l \right\}$$

$$\mu(p_m^l) = \left\{ \mu(\overline{b_m(s_l)}^k) \Big/ \overline{b_m(s_l)}^k \right\}$$

где $\mu(\overline{b_m(s_l)}^k)$ – значение функции принадлежности признака к определенному терму $\overline{b_m(s_l)}^k$ для конкретного значения диагностического параметра. Пример графического представления функций принадлежности представлен на рис. 2.2.

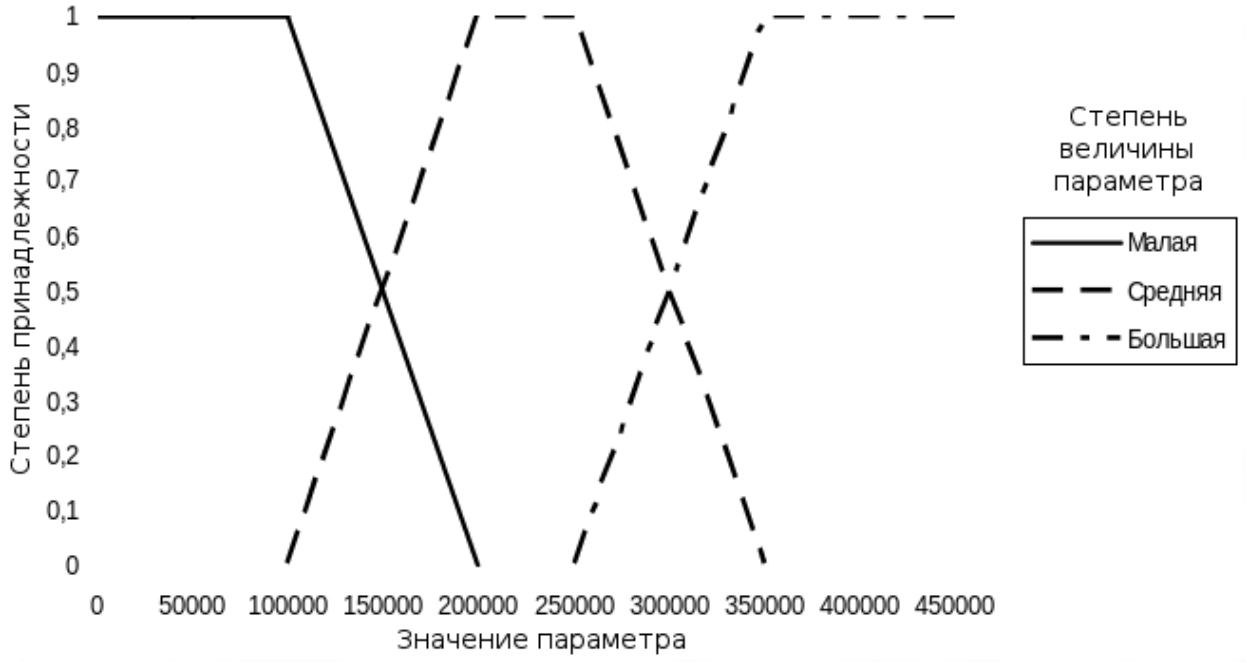


Рисунок 2.2 – Пример графического представления функций принадлежности

Оценить степень схожести между текущей нечеткой ситуацией $\Theta(t)$ и теми, что наблюдались ранее $\Theta(t_0)$, можно с помощью степени нечеткого равенства или степени нечеткой общности, которые описаны в [106]. Расчет степени нечеткого равенства применяется в том случае, когда в обеих сравниваемых нечетких ситуациях $\Theta(t)$ и $\Theta(t_0)$ все наборы наблюдаемых лингвистических переменных совпадают. Если в ситуации $\Theta(t)$ произошли некоторые события, которые в ситуации $\Theta(t_0)$ не происходили, либо в ситуации $\Theta(t)$ не учитываются или не происходят те события, которые наблюдались в ситуации $\Theta(t_0)$, то следует рассчитывать степень нечеткой общности, которая является более общим показателем, чем степень нечеткого равенства ситуаций.

Через $k(\Theta(t),\Theta(t_0))$ обозначена степень схожести рассматриваемых ситуаций $\Theta(t)$ и $\Theta(t_0)$, которая будет равна нечеткой степени равенства или нечеткой степени общности этих ситуаций, в зависимости от того, совпадают наборы наблюдаемых лингвистических переменных или нет соответственно. Коэффициент $k(\Theta(t),\Theta(t_0))$ принимает значения из отрезка [0,1], причем, чем больше значение коэффициента, тем ситуации более схожи.

Определение. Штатной ситуацией для сервиса $s_l \in S(b)$ и набора характеристических лингвистических переменных p_m^l на заданном временном интервале T_p называется ситуация, когда термы лингвистических переменных p_m^l описывающих значения диагностических параметров $b_m(s_l)$ на заданном временном интервале T_p принимают значения из терм-множеств $p_m^l(N)$.

Определение. Нештатной ситуацией для сервиса $s_l \in S(b)$ и набора характеристических лингвистических переменных p_m^l на заданном временном интервале T_p называется ситуация, когда существует лингвистическая переменной p_m^l описывающая значения диагностического параметра $b_m(s_l)$ на заданном временном интервале T_p которая принимает значения из терм-множества $p_m^l(S)$.

Таким образом, используя введенные определения и степень схожести нечетких ситуаций $k(\Theta(t),\Theta(t_0))$ получено множество вариантов нечетких ситуаций для сервиса $s_l \in S(b)$, которое можно описать соотношениями:

$$\begin{aligned}
&\tilde{C}^l = \tilde{C}^l_{шт} \cup \tilde{C}^{1l}_{не},\\
&\tilde{C}^l_{шт} = \left\{\Theta^l_{шт,s}\right\}_{s=1}^{N_{шт}(l)}\\
&\tilde{C}^l_{не} = \left\{\Theta^l_{не,s}\right\}_{s=1}^{N_{не}(l)}.
\end{aligned}$$

где $N_{шт}(l)$ – количество вариантов нечетких ситуаций, описывающих штатный режим функционирования l-ой сервиса, а $N_{не}(l)$ – количество вариантов нечетких ситуаций, описывающих нештатный режим функционирования l-ой сервиса.

На множестве всех нештатных ситуаций введено понятие приоритета причины значений параметра. Приоритет зависит от модальности причинно-следственной связи действующей в данной нештатной ситуации и от того, к какому классу оно принадлежит. Та причинно-следственная связь, которая определяет значения диагностического параметра на этом интервале его развития, имеет максимальный приоритет в множестве возможных причин.

Усовершенствование обобщенной модели диагностики РИС позволило упростить процесс модификации знаний о диагностических ситуациях за счет использования нечеткого описания значений диагностических параметров. Так, например, сравнительный анализ затрат времени экспертов-пользователей различной квалификации на поддержку актуальности базы знаний диагностических ситуаций на основе разработанной модели и на основе модели, базирующейся на четко заданных параметрах, показало уменьшение расходов, в среднем, на 57% (см. пункт 4.2). Такого результата удалось достичь за счёт использования обобщённых описаний штатных и нештатных ситуаций на удобном для экспертов-пользователей языке и использованию лингвистических переменных. Использование лингвистических переменных позволило избавится от громоздких и сложных правил на основе чётких параметров.

Выводы по второму разделу

1. Опираясь на проведённый анализе, были сформулированы основные требования к архитектуре баз знаний систем диагностирования РИС. В качестве основных требований были выделены модульность и иерархичность

архитектуры для обеспечения гибкости и простоты разработки и модификации базы знаний. С той же целью в качестве основного подхода к описанию знаний в модулях базы знаний выбран онтологический подход. При построении модулей базы знаний решено ориентироваться на применение в рассуждениях, основанных на гибридном RBR-CBR подходе, а так же использование информационных моделей, описанных в подразделе 1.2. Кроме того, решено расширить описание диагностических аспектов РИС нечёткими данными.

2. На основе сформулированных требований разработана структура базы знаний системы диагностирования РИС. Базовая структура предполагает разработку четырех модулей на трёх уровнях абстракции, где модули более низкого уровня строятся она основе понятий и терминов модулей более высокого уровня. При этом, базовая структура может расширятся за счёт добавления новых модулей, основанных на понятиях и терминах уже существующих модулей.

3. С целью сокращения времени затрачиваемого на поддержание актуальности базы знаний, была усовершенствована обобщенная модель диагнотики РИС, которая позволяет использовать при описании диагностических аспектов РИС нечёткие данные, за счёт введения лингвистических переменных. Проведённый сравнительный анализ временных затрат экспертов-диагоностов разного уровня квалификации при поддержании актуальности базы знаний системы диагностирования РИС показал сокращение затраченного времени в среднем на 57% при использовании усовершенствованной обобщенной модели.

РАЗДЕЛ 3
РАЗРАБОТКА ОНТОЛОГИЧЕСКИХ МОДЕЛЕЙ И УСОВЕРШЕНСТВОВАНИЕ МЕТОДА ДИАГНОСТИРОВАНИЯ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

3.1 Описание CIM-метаонтологии при помощи языка дескриптивной логики

В качестве модулей второго уровня абстракции предложенной структуры базы знаний систем диагностирования РИС, в данной работе использовались онтологические модели на основе информационных моделей семейства CIM, описанные в подразделе 1.2.

Для описания онтологических моделей на втором уровне абстракции, в результате анализа, приведенного в пункте 1.4.2, был выбран один из самых распространенных в настоящее время стандартов описания онтологий OWL, который позволяет не только четко описывать модели онтологий, но и обладает мощными средствами логического вывода. Кроме того, данный формат является привлекательным, поскольку он позволяет описывать необычные классы понятий и вводить обобщенные понятия. Из трёх предложенных диалектов был выбран OWL DL, основанный на дескриптивной логике (ДЛ), поскольку он обладает достаточными выразительными средствами, и при этом поддерживается большинством существующих программных реализаций решателей.

Прежде чем приступать к непосредственной разработке онтологических моделей, сначала необходимо описать онтологическую модель метаэлементов CIM, так называемую CIM-метаонтологию. Эта онтологическая модель описывает термины и понятия из которых строятся все CIM-модели и основывается на CIM-метамодели.

Для представления CIM-элементов при помощи ДЛ, в работе, предлагается рассмотреть их как концепты и роли, а также ввести специальные аксиомы, представленные в таблице 3.

Таблица 3.1 – Представление элементов CIM-метамодели в ДЛ.

CIM-элементы	Концепты и роли	Введённые аксиомы ДЛ
Класс C	Концепт C	
Атрибут a у C типа T	Бинарная роль	$C \sqsubseteq \langle = 1a \rangle \sqcap \exists a.T$
Ключевой атрибут a у C	Бинарная роль a	$C \sqsubseteq \langle = 1a \rangle \sqcap \exists A.D \sqcap \langle \leq 1A^{-} \rangle$
Атрибут a у C типа T	Бинарная роль a	$C \sqsubseteq \langle \geq 1a \rangle \sqcap \forall a.T$
Атрибут a в диапазоне $(n_i, \cdots, n_j)$	Бинарная роль a	$C \sqsubseteq \langle \geq n_i a \rangle \sqcap \langle \leq n_j a \rangle \sqcap \forall a.T$
Зависимость A	Бинарная роль A Роли R1 и R2	$\top \sqsubseteq \forall A.C_2 \sqcap \forall A^{-}.C_1$ $C_1 \sqsubseteq \forall A.C_2 \sqcap \langle \geq n_i A \rangle \sqcap \langle \leq n_j A \rangle$ $C_2 \sqsubseteq \forall A^{-}.C_1 \sqcap \langle \geq m_i A^{-} \rangle \sqcap \langle \leq m_j A^{-} \rangle$ $A \sqsubseteq R_1,\ R_1 \sqsubseteq A,\ A^{-} \sqsubseteq R_2,\ R_2 \sqsubseteq A^{-}$
N-арная ассоциация	Концепт A Роли Ar, R1 … Rn	$A \sqsubseteq \exists R_1.C_1 \sqcap \cdots \sqcap \exists R_n.C_n \sqcap \langle \leq 1R_1 \rangle \sqcap \cdots \sqcap \langle \leq 1R_n \rangle$ $C_i \sqsubseteq \forall R_i^{-}.A \sqcap \langle \geq n_i R_i^{-} \rangle \sqcap \langle \leq n_j R_i^{-} \rangle$ $i = 1, \cdots, n$
Бинарная ассоциация	Концепт A Роли Ar, R1 и R2	$A \sqsubseteq \exists R_1.C_1 \sqcap \exists R_2.C_2 \sqcap \langle \leq 1R_1 \rangle \sqcap \langle \leq 1R_2 \rangle$ $C_1 \sqsubseteq \forall R_1^{-}.A \sqcap \langle \geq m_i R_1^{-} \rangle \sqcap \langle \leq m_j R_1^{-} \rangle$ $C_2 \sqsubseteq \forall R_2^{-}.A \sqcap \langle \geq n_i R_2^{-} \rangle \sqcap \langle \leq n_j R_2^{-} \rangle$
Наследованные отношения (частичные и не различные)		$C_i \sqsubseteq C,\ i = 1, \cdots, n$
Наследованные отношения (частичные и различные)		$C_i \sqsubseteq C,\ i = 1, \cdots, n\ C_i \sqsubseteq \neg C_j,\ \forall i \neq j$
Наследованные отношения (общие и не различные)		$C_i \sqsubseteq C,\ i = 1, \cdots, n\ C \sqsubseteq \bigsqcup_{i=1}^{n} C_i$
Наследованные отношения (общие и различные)		$C_i \sqsubseteq C,\ i = 1, \cdots, n\ C_i \sqsubseteq \neg C_j,\ \forall i \neq j$ $C \sqsubseteq \bigsqcup_{i=1}^{n} C_i$

С помощью приведенных в таблице аксиом описываются все элементы CIM-метамодели. Так, например, элемент *CIM_Class*, согласно CIM-метамодели, должен расширять элемент *CIM_NamedElement* и быть связанным ассоциациями: *PropertyDomain* с *CIM_Property*, *MethodDomain* с *CIM_Method*, *ReferenceRange* с *CIM_ReferenceType*, *Generalization* с *CIM_Class* и *DefiningClass* с *CIM_Instance*. С

учетом ограничения множественности, на ДЛ *CIM_Class* будет описан так как показано на рисунке 3.1.

$$\begin{aligned}
&\mathrm{CIM_Class} \sqsubseteq \mathrm{CIM_NamedElement} \sqcap \\
&\neg(\mathrm{CIM_TypedElement} \sqcup \mathrm{CIM_Trigger} \sqcup \mathrm{CIM_Schema} \sqcup \mathrm{CIM_Qualifier}) \sqcap \\
&\forall \mathrm{PropertyDomain.CIM_Property} \sqcap \forall \mathrm{MethodDomain.CIM_Method} \sqcap \\
&\forall \mathrm{ReferenceRange.CIM_ReferenceType} \sqcap \forall \mathrm{Generalization}^{-}.\mathrm{CIM_Class} \sqcap \\
&\left\langle \leq 1 \mathrm{Generalization}^{-} \right\rangle \sqcap \forall \mathrm{Generalization.CIM_Class} \sqcap \forall \mathrm{DefiningClass.CIM_Instance}
\end{aligned}$$

Рисунок 3.1 – Описание *CIM_Class* на языке ДЛ

Аналогично описываются все остальные элементы CIM-метамодели: *CIM_NamedElement, CIM_TypedElement, CIM_Type, CIM_PrimitiveType, CIM_ReferenceType, CIM_Schema, CIM_Class, CIM_Property, CIM_Method, CIM_Parameter, CIM_Trigger, CIM_Indication, CIM_Association, CIM_Reference, CIM_QualifierType, CIM_Qualifier, CIM_Flavor, CIM_Instance, CIM_InstanceProperty, CIM_Value*.

Совокупность описаний всех элементов CIM-метамодели образуют CIM-метаонтологию, которая позволят описывать любую CIM-модель в виде онтологии, в соответствии с CIM-стандартом.

3.2 Разработка онтологической модели описания диагностических тестов

При разработки онтологической модели описания диагностических тестов, за основу взята CDM-модель из семейства CIM, описанная в пункте 1.2.3, поскольку она определяет все необходимые при описании диагностических тестов элементы и связи в CIM-совместимой терминологии.

Определение концептов онтологической модели описания диагностических тестов проводилось на основе понятий, описанных в CIM-метаонтологии, так же в формате OWL на языке ДЛ. Это позволило создать

онтологическую модель, основу сигнатуры которой составляют атомарные концепты роли, определяющие соответствующие элементы CDM-модели.

Описание обязательных атомарных концептов приведено в таблице 4.

Таблица Рисунок 3.2 – Обязательные концепты CDM модели.

CIM_AvailableDiagnostic Service	Ассоциативная связь диагностической услуги, которая может быть выполнена по отношению к управляемому элементу.
CIM_ConcreteJob	Класс, представляющий конкретную задачу. Используется клиентской стороной для наблюдения и контроля за выполнением диагностической услуги.
CIM_Diagnostic CompletionRecord	Запись, содержащая итоговую информацию работы услуги.
CIM_DiagnosticLog	Журнал диагностики, агрегирующий все диагностические записи (существует несколько легитимных механизмов протоколирования)
CIM_DiagnosticService Record	Используется в качестве сообщений-отчетов диагностических услуг, сообщая результаты, ошибки, предупреждения и статусы.
CIM_DiagnosticTest	Класс, который представляет диагностическую услугу, разработанный для проверки и наблюдения за поведением устройства, которое причастно к неисправности на каком-либо уровне системы.
CIM_ElementSoftware Identity	Связь диагностической услуги с информацией о версии
CIM_HostedService	Предназначен для связи конкретного диагностического теста и конкретной компьютерной системы, в пределах которой он работает, а также конкретной справочной услуги и конкретной компьютерной системы, на

	которую она распространяется.
CIM_LogManagesRecord	Связывает журнал с его записями
CIM_OwningJobElement	Связывает диагностическую услугу с конкретными задачами
CIM_RegisteredProfile	Идентифицирует Diagnostics Profile для определения клиентской стороной совместимости конкретной диагностической услуги.
CIM_ServiceAffects Element	Связь диагностической услуги с любым управляемым элементом, на который она воздействует.
CIM_ServiceAvailableTo Element	Связывает диагностическую услугу с его информацией справочной услуги.
CIM_SoftwareIdentity	Класс, использующийся для представления версии диагностической услуги.
CIM_UseOfLog	Связь журнала с управляемым элементом (устройство или диагностическая услуга), информация о котором хранится в журнале.

К необязательным концептам разработанной базы знаний относятся: *CIM_AffectedJobElement,, CIM_CorrespondingSettingDataRecord, CIM_DiagnosticCompletionRecord, CIM_DiagnosticLog, CIM_DiagnosticServiceCapabilities, CIM_DiagnosticServiceRecord, CIM_DiagnosticSettingData, CIM_DiagnosticSettingDataRecord, CIM_DiagnosticTest, CIM_ElementCapabilities, CIM_ElementSettingData, CIM_ElementSoftwareIdentity, CIM_HelpService, CIM_HostedService, CIM_JobSettingData, CIM_LogManagesRecord, CIM_OwningJobElement, CIM_RecordAppliesToElement, CIM_RegisteredProfile, CIM_ServiceAffectsElement, CIM_ServiceAvailableToElement, CIM_ServiceComponent, CIM_SoftwareIdentity, CIM_UseOfLog.*

Все перечисленные концепты обязательно являются тем или иным концептом CIM-метаонтологии, что является необходимым условием соблюдения стандарта CIM.

В качестве примера, можно привести описание класса *CIM_DiagnosticTest* (является интерпретацией концепта *CIM_Class*) и ассоциации *CIM_HostedService* (интерпретация концепта *CIM_Association*).

Согласно CDM-модели, класс *CIM_DiagnosticTest* содержит свойства: *SystemCreationClassName, SystemName, CreationClassName, Name, ElementName, Characteristics, OtherCharacteristicsDescriptions*, и метод *RunDiagnosticService()*.

Данное утверждение в ДЛ выглядит так, как показано на рис. 3.2.

$$
\begin{aligned}
&\mathrm{CIM\_DiagnosticTest} \sqsubseteq \mathrm{CIM\_Class} \sqcap \\
&\exists \mathrm{hasSystemCreationClassName.SystemCreationClassName\_DT} \sqcap \\
&\langle \leq 1 \mathrm{hasSystemCreationClassName} \rangle \sqcap \exists \mathrm{hasSystemName.SystemName\_DT} \sqcap \\
&\langle \leq 1 \mathrm{hasSystemName} \rangle \sqcap \exists \mathrm{hasCreationClassName.CreationClassName\_DT} \sqcap \\
&\langle \leq 1 \mathrm{hasCreationClassName} \rangle \sqcap \exists \mathrm{hasName.Name\_DT} \sqcap \langle \leq 1 \mathrm{hasName} \rangle \sqcap \\
&\exists \mathrm{hasElementName.ElementName\_DT} \sqcap \langle \leq 1 \mathrm{hasElementName} \rangle \sqcap \\
&\exists \mathrm{hasCharacteristics.Characteristics\_DT} \sqcap \langle \leq 1 \mathrm{hasCharacteristics} \rangle \sqcap \\
&\forall \mathrm{hasOtherCharacteristicsDescriptions.OtherCharacteristicsDescriptions\_DT} \sqcap \\
&\langle \leq 1 \mathrm{hasOtherCharacteristicsDescriptions} \rangle \sqcap \\
&\exists \mathrm{hasRunDiagnosticService.RunDiagnosticService\_DT} \sqcap \langle \leq 1 \mathrm{hasRunDiagnosticService} \rangle
\end{aligned}
$$

Рисунок 3.2 – Описание *CIM_DiagnosticTest* на языке ДЛ

Аналогично задаются остальные понятие CDM-модели.

3.3 Разработка онтологических моделей расширенного описания элементов и зависимостей в РИС

Непосредственное использование понятий, введённых в обобщенной модели диагностики РИС, при построении и обслуживании базы знаний систем диагностирования РИС является непрактичным, поскольку они являются чересчур абстрактными, для практического применения. Для расширения

понятийной базы описания диагностических аспектов РИС удобными, с практической точки зрения, терминами и сокращения, таким образом, трудозатрат экспертов-пользователей, были построены 4 онтологические модели. Выбранная структура базы знаний и применение онтологического подхода позволяют, при необходимости, расширять описания элементов и зависимостей в РИС за счёт добавления новых моделей, описанных в терминах обобщенной модели диагностики РИС.

Разработанные онтологические моделей расширенного описания элементов и зависимостей в РИС основываются на сервисной модели MNM, описанной в пункте 1.2.4. В качестве языка описания знаний так же выбран язык OWL основанный на дескриптивной логике. Поскольку онтологические моделей расширенного описания элементов и зависимостей в РИС, как и онтологическая модель описания диагностических тестов, основываются на моделях одного семейства и, в целом, решают схожие задачи, аргументы в пользу выбора OWL остаются прежними.

В рамках данной работы классы и зависимости сервисной модели MNM, описываются в виде 4 онтологических моделей, базовой, описывающей основные понятий сервисной модели, и 3 дополнительных, для расширенного описания сервиса с точки зрения пользователей и SLA, описания событий в сервисах и ресурсах, расширенного описания зависимостей между элементами РИС.

Класс данной модели, разработан для нужд интеллектуальных компонентов и, следовательно, сосредоточен на различных видах зависимостей, которые могут возникать при функционировании сервиса. Он также включает в себя атрибуты для настройки сущностей, которые необходимы для моделирования компонентов сервиса.

Разработанные онтологические моделей расширенного описания элементов и зависимостей в РИС позволяют обеспечить различную глубину моделирования для параметров качества сервисов (QoS), зависимостей и точек

доступа к сервисам (SAP). Параметры QoS могут быть определены для сервиса в целом (например, доступность сервиса) или специально для функций сервиса (например, временные условия для выполнения транзакции). Таким образом, зависимости могут быть смоделированы для сервиса в целом или для его функций. Моделирование дает возможность иметь различные SAP-ы, которые могут быть привязаны к сервису (если вся функциональность сервиса может быть доступна через этот SAP).

3.3.1 Базовая онтологическая модель.

Центральными понятиями базовой модели является классы (являются интерпретацией концепта *CIM_Class*) *Service*, который ассоциирован с классами *ServiceFunctionality* и *QoSParameter*, и *Resource*, определяемый вместе с классом *QoR*. Оба класса являются подклассами *ManagedServiceElement*, который не был явно представлен в сервисной модели MNM, однако стал одним из основных понятий онтологической модели.

На рис. 3.3 представлена базовая онтологическая модель сервисной модели MNM.

Данная онтологическая модель включает в себя такие понятия:

- класс *ManagedServiceElement*, который представляет управляемый элемент сервиса MSE и является суперклассом для классов сервисов и ресурсов;

- класс *Service*, который представляет сервис, участвующий в диагностике;

- класс *Resource*, который представляет ресурс, участвующий в диагностике;

- класс *MSEproperty*, который представляет свойства управляемого элемента сервиса MSE, и является суперклассом для классов *ServiceProperty*, *ResourceProperty* и *QualityParameter*;

Рисунок 3.3 – Базовая онтологическая модель сервисной модели MNM

- класс *ServiceProperty*, который представляет свойства управляемого элемента сервиса *MSE*, когда он является объектом класса *Service* и является суперклассом для класса *ServiceFunctionality*;

- класс *ResourceProperty*, который представляет свойства управляемого элемента сервиса MSE, когда он является объектом класса *Resource*;

- класс *QualityParameter*, который представляет свойства управляемого элемента сервиса *MSE*, и используется для задания параметров качества сервисов и ресурсов;

- класс *ServiceFunctionality*, который представляет сервисные функции объекта класса *Service*, являющегося управляемым элементом сервиса;

- класс *QoRParameter*, который представляет свойства управляемого элемента сервиса *MSE*, и используется для задания параметров качества ресурсов;

- класс *QoSParameter*, который представляет свойства управляемого элемента сервиса *MSE*, и используется для задания параметров качества ресурсов;

- класс *TestSchedule*, который представляет свойства управляемого элемента сервиса *MSE*, и используется для задания расписания проведения тестов;

- класс *StaffMember*, который представляет свойства управляемого элемента сервиса *MSE*, и используется для задания сотрудника, ответственного за управляемый элемент;

- класс *GenericEvent*, абстрактный класс, который используется для описания событий, эквивалентный абстрактному классу *GenericEvent* из онтологии *Event*;

- класс *TemporalEntity*, абстрактный класс, который используется для описания времени, осуществляется с помощью онтологии OWL-Time. Применяется стандарт W3C—OWL-Time[99];

- класс *ParameterHistory*, используется для описания хронологии значений параметров качества ресурсов и сервисов, и привязывается ко времени наблюдения за конкретным параметром;

- класс *AdminStatus*, административный статус со ссылками на историю состояния сервисов или ресурсов с административной точки зрения, которые связаны с жизненным циклом сервисов. Это означает что *MSE*, который планируется, реализуется, используется, в настоящее время изменен или отозван. Изменения фаз необходимы для определения зависимостей. Например, ресурс, который больше не используется, не может быть причиной симптомов дальше. История изменений сохраняется, чтобы иметь возможность определить ошибки в управлении изменениями, когда они используются в качестве основной причины

текущих симптомов. Она также может быть использована для отчетности о неисправности сервиса, как объяснение некоторых симптомов, которые могут быть вызваны плановым ремонтом;

- класс *OperStatus*, отличается от *AdminStatus*, поскольку история оперативного статуса отражает результаты тестов, выполняемых на *MSE* и то, как *MSE* отреагировал. В противном случае должна быть добавлена ссылка на полученное событие. История такого статуса также полезна для идентификации коренной причины отклонения, если известно когда работал *MSE* в последний раз и известны изменения, которые произошли потом. Оперативный статус, это сжатое представление *QoS* / *QoR* измерений, которые проводятся для *MSE*, в том смысле, какие симптомы существуют в описываемый момент времени;

- класс *AdminStatusEnum*, используется для описания набора состояний, в которых может находится *AdminStatus*, соответствующего элемента *MSE*;

- класс *OperStatusEnum*, используется для описания набора состояний, в которых может находится *OperStatus*, соответствующего элемента *MSE*.

Базовая онтологическая модель сервисной модели *MNM*, содержит следующие отношения между классами:

- *propertyOfMSE* задаёт отношения между классами *ManagedServiceElement* и *MSEproperty*. Класс *ManagedServiceElement* является доменом, а класс *MSEproperty* — областью значений;

- *propertyOfService* задаёт отношения между классами *Service* и множеством терминов, получающихся объединением классов *QoSParameter* и *ServiceFunctionality*. Класс *Service* является доменом, а класс, являющийся объединением калссов *QoSParameter* и *ServiceFunctionality* — областью значений;

- *propertyOfResource* задаёт отношения между классами *Resource* и классом *QoRParameter*. Класс *Resource* является доменом, а класс *QoRParameter* — областью значений;

- *subServiceFunctionality* задаёт отношения агрегации между классами *ServiceFunctionality*. Класс *ServiceFunctionality* является и доменом, и областью значений;

- *hasParameterHistory* задаёт отношения композиции между классом *QualityParameter* и классом *ParameterHistory*. Класс *QualityParameter* является доменом, а класс *ParameterHistory* — областью значений;

- *testSchedule* задаёт отношение между композицией классов *ManagedServiceElement*, *QualityParameter* и классом *TestSchedule*. Объединение классов *ManagedServiceElement* и *QualityParameter* является доменом, а класс *TestSchedule* — областью значений;

- *responsible* задаёт отношения между классом *ManagedServiceElement* и классом *StaffMember*. Класс *ManagedServiceElement* является доменом, а класс *StaffMember* — областью значений;

- *date* задаёт отношения между классом *ParameterHistory* и классом TemporalEntity. Класс ParameterHistory является доменом, а класс TemporalEntity — областью значений;

- *event* задаёт отношения между классом ParameterHistory и классом *GenericEvent*. Класс *ParameterHistory* доменом, а класс *GenericEvent* — областью значений;

- *adminStatus* задаёт отношения между классом *AdminStatus* и классом *AdminStatusEnum*. Класс *AdminStatus* является доменом, а класс *AdminStatusEnum* — областью значений;

- *adminStatusDate* задаёт отношения между классом *AdminStatus* и классом *TemporalEntity*. Класс *AdminStatus* является доменом, а класс *TemporalEntity* — областью значений;

- *adminStatusCurrent* и *adminStatusHistory* задают отношения между классом *ManagedServiceElement* и классом *AdminStatus*. Класс *ManagedServiceElement* является доменом, а класс *AdminStatus* — областью значений. При этом отношение *adminStatusCurrent* используется для определения

текущего статуса, а отношение *adminStatusHistory* для определения прошлых статусов;

- *operStatus* задаёт отношения между классом *OperStatus* и классом *OperStatusEnum*. Класс *OperStatus* является доменом, а класс *OperStatusEnum* — областью значений;

- *operStatusDate* задаёт отношения между классом *OperStatus* и классом *TemporalEntity*. Класс *OperStatus* доменом, а класс *TemporalEntity* — областью значений;

- *operStatusEvent* задаёт отношения между классом *OperStatus* и классом *GenericEvent*. Класс *OperStatus* доменом, а класс *GenericEvent* — областью значений;

- *operStatusCurrent* и *operStatusHistory* задают отношения между классом *ManagedServiceElement* и классом *OperStatus*. Класс *ManagedServiceElement* является доменом, а класс *OperStatus* — областью значений. При этом отношение *operStatusCurrent* используется для определения текущего статуса, а отношение *operStatusHistory* для определения прошлых статусов.

3.3.2 Онтологическая модель описания взаимосвязей сервиса с SLA, SAP и функциональностью сервиса.

Базовая онтологическая модель описывает только основные понятия для описания сервисов. Для расширения возможностей описания знаний о сервисе, его функционала, SLA, возможных точках доступа и пользователях, была разработана онтологическая модель описания взаимосвязей сервиса с SLA, SAP и функциональностью сервиса.

Для описания условий использования сервиса вводится класс SLA. SLA согласовывается с одним или несколькими клиентами (*Customers*) и имеет одного поставщика (*Provider*). В SLA устанавливаются пороговые значения для параметров QoS сервиса и функционала сервиса.

На рис. 3.4 представлена онтологическая модель описания взаимосвязей сервиса с SLA, SAP и функциональностью сервиса.

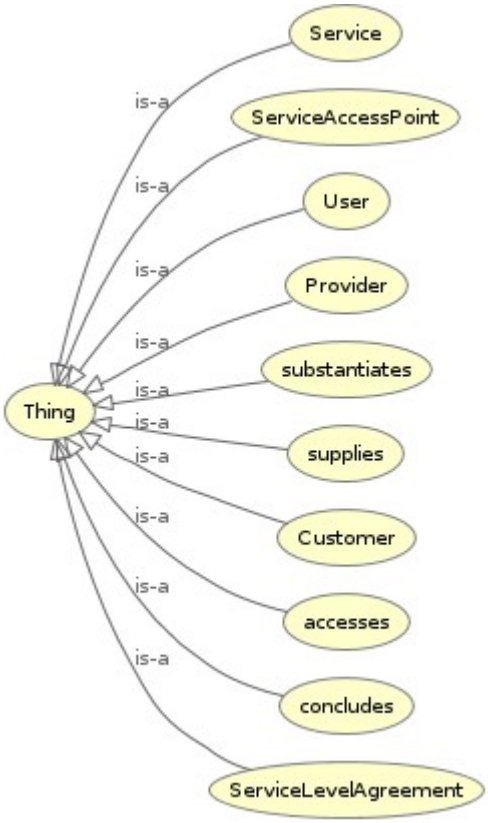


Рисунок 3.4 – Онтологическая модель описания взаимосвязей сервиса с SLA, SAP и функциональностью сервиса.

Разработанная онтологическая модель включает в себя следующие понятия:

- класс *Service* (эквивалентен понятию *Service* из базовой онтологической модели), который представляет сервис, участвующий в диагностике;

- класс *ServiceLevelAgreement*, который используется для описания условий использования сервиса. В *SLA* устанавливаются пороговые значения для параметров *QoS* сервиса и функционала сервиса;

- класс *Customer*, который используется для согласования *SLA* с одним или несколькими клиентами;

- класс *Provider*, который используется для указания поставщика услуг удовлетворяющего данные *SLA*;

- класс *ServiceAccessPoint*, используется для описания множества точек доступа, с которых пользователь может обращаться к сервису;

- класс *User*, который используется для описания пользователей, связанных с SAP;

- класс *accesses*, который используется для указания связи пользователей и *SAP*;

- класс *concludes*, который используется для описания конкретных особенностей связи *SLA* с множеством клиентов или провайдерами;

- класс *substantiates*, который используется для описания конкретных особенностей связи *SLA* и сервисом;

- класс *supplies*, который используется для описания конкретных особенностей связи *SAP* и сервисом.

Онтологическая модель описания взаимосвязей сервиса с *SLA*, *SAP* и функциональностью сервиса, содержит следующие отношения между классами:

- *hasService* задаёт отношения между множеством терминов, получающихся объединением классов *substantiates* и *supplies*, и множеством терминов задаваемых классом *Service*. Объединение классов *substantiates* и *supplies* является доменом, а класс *Service* — областью значений;

- *hasSAP* задаёт отношения между множеством терминов, получающихся объединением классов *accesses* и *supplies*, и множеством терминов задаваемых классом *ServiceAccessPoint*. Объединение классов *accesses* и *supplies* является доменом, а класс *ServiceAccessPoint* — областью значений;

- *hasSLA* задаёт отношения между множеством терминов, получающихся объединением классов *concludes* и *substantiates*, и множеством терминов задаваемых классом *ServiceLevelAgreement*. Объединение классов *concludes* и *substantiates* является доменом, а класс *ServiceLevelAgreement* — областью значений;

- *hasUser* задаёт отношения между классами *accesses* и *User*. Класс *accesses* является доменом, а класс *User* — областью значений;

- *hasCustomer* задаёт отношения между классами *concludes* и *Customer*. Класс *concludes* является доменом, а класс *Customer* — областью значений;

- *hasProvider* задаёт отношения между классами *concludes* и *Provider*. Класс *concludes* является доменом, а класс *Provider* — областью значений;

3.3.3 Онтологическая модель описания событий

При диагностировании РИС, важными признаками возникновения нештатной ситуации являются события, которые фиксируются системой мониторинга. Для описания событий, которые могут быть использованы при описании правил или прецедентов, в рамках модуля расширенного описания элементов и зависимостей в РИС, разработана онтологическая модель описания событий. Эта модель так же строится на основе CIM-метаонтологии и в соответствии с обобщенной моделью диагностики РИС.

UML-модель, представленная на рис. 3.5, изображает классы событий и их взаимосвязи при помощи элементов CIM-метамодели.

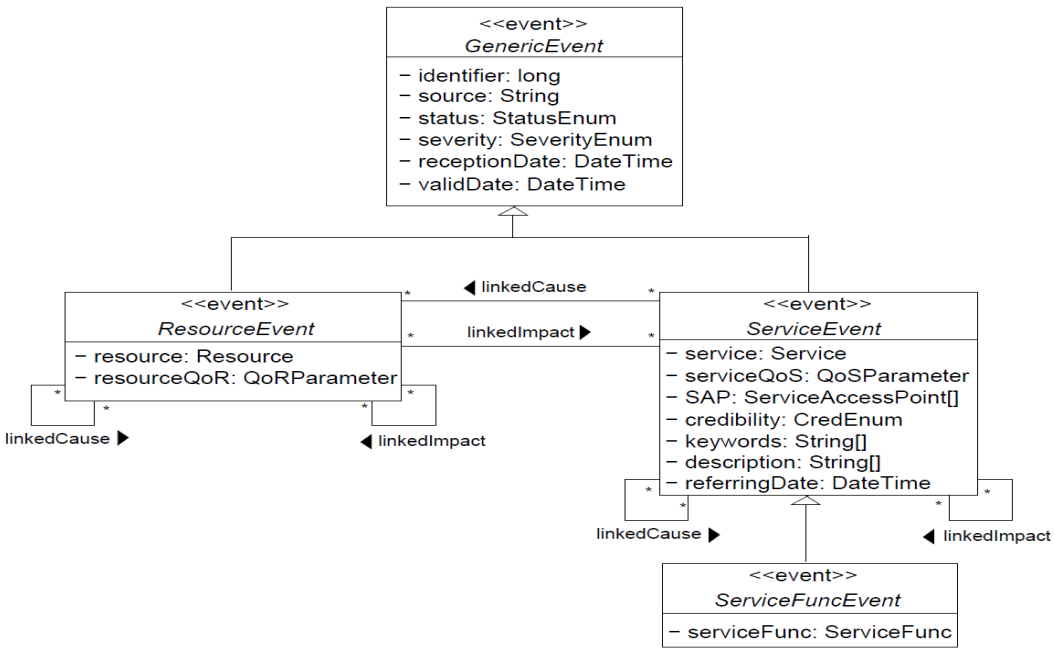


Рисунок 3.5 – CIM-модель событий

С учетом результатов работ [108-112], для описания событий введено понятие абстрактного класса *GenericEvent*, который является универсальным для событий, связанных с ресурсами и сервисами. Для уникальности ссылки на событие описываемого классом *GenericEvent*, введены атрибут-идентификатор *identifier* и атрибут *source*, который ссылается на источник события. Атрибут *status* определяет статус события. Значение атрибута *severity* определяет степень влияния, связанную с классом событий. Значение атрибута зависит от вида сервиса или ресурса, в частности, когда влияние было предварительно рассчитано, в предположении не работоспособности сервиса или ресурса. Атрибут *receptionDate* специфицирует время, когда событие было принято, как правило, с точностью до секунды. Для ресурсов предполагается, что это время имеет лишь незначительное отклонение от времени непосредственного возникновения, поэтому никаких дополнительных атрибутов времени не дано. У событий сервиса наоборот, может произойти значительная задержка. Чтобы отразить эту задержку, для сервисов вводится дополнительный атрибут *referringDate*. Продолжительность действия события задаётся атрибутом *validDate*.

Абстрактный класс для событий ресурсов *ResourceEvent* является производным от универсального класса событий. Описание должно содержать ресурсную ссылку на класс ресурса в структуре классов и фиксирует *QoR* параметр, к которому относится событие. Для причинных связей и описания воздействия применяются ассоциации *linkedCause* и *linkedImpact*.

Абстрактной класс *ServiceEvent* является производным от класса *GenericEvent* и дополняет атрибутами сервисов. Как и класс *ResourceEvent*, он содержит ассоциации *linkedCause* и *linkedImpact*.

Поскольку событие сервиса должны быть связанны с определенным сервисом и с параметром *QoS*, описание ассоциаций с данными классами так же включены. При этом атрибут *SAP* должен ссылаться на точки доступа, с которых обращались к сервису. С помощью описания событий сервиса можно

более конкретно описывать функциональности сервиса. Таким образом, класс *ServiceFunctionalityEvent* определен как подкласс *ServiceEvent* который должен иметь функциональность сервиса в качестве параметра. Подобно классу *ResourceEvent*, конкретные подклассы должны быть определены для классов *ServiceEvent* и *ServiceFunctionalityEvent*. Они описывают выполнение или нарушение порогов для параметров *QoS*.

На рис. 3.6 представлена онтологическая модель описания событий.

Рисунок 3.6 – Онтологическая модель описания событий.

Привязка онтологии ко времени для возможности наблюдать за событиями в режиме реального времени, осуществляется с помощью онтологии OWL-Time. Применяется стандарт W3C—OWL-Time[@?10].

Далее описаны понятия введённые в рамках онтологической модели описания событий:

- класс *Service*, который представляет сервис, участвующий в диагностике;

- класс *Resource*, который представляет ресурс, участвующий в диагностике;

- класс *ServiceAccessPoint*, используется для описания множества точек доступа, с которых пользователь может обращаться к сервису;

- класс *ServiceFunctionality*, который представляет сервисные функции объекта класса *Service*, являющегося управляемым элементом сервиса;

- класс *QoRParameter*, который представляет свойства управляемого элемента сервиса *MSE*, и используется для задания параметров качества ресурсов;

- класс *QoSParameter*, который представляет свойства управляемого элемента сервиса *MSE*, и используется для задания параметров качества ресурсов;

- класс *GenericEvent*, абстрактный класс, который используется для описания событий;

- класс *ServiceEvent*, который используется для описания событий сервиса;

- класс *ResourceEvent*, который используется для описания событий ресурса;

- класс *ServiceFuncEvent*, который используется для описания событий связанных с функциональностью сервиса;

- класс *EventLink*, абстрактный класс, который используется для описания взаимосвязей между событиями;

- класс *linkedCause*, который используется для описания взаимосвязи между событиями, для указания возможных причин событий;

- класс *linkedImpact*, который используется для описания взаимосвязи между событиями, для указания возможных последствий событий;

- класс *CredEnum*, который используется для описания информации о достоверности события сервиса. Он не нужен для событий, которые являются параметрами, поступающими с ресурсов, или результатом мониторинга собственного поставщика. Он важен для событий сервисов от внешних провайдеров;

- класс *SeverityEnum*, который используется для описания степени влияния, связанную с классом событий. Задаваемое значение зависит от вида сервиса или ресурса, в частности, когда влияние предварительно рассчитано, в предположении не работоспособности сервиса или ресурса;

- класс *StatusEnum*, который используется для определения статусов событий.

В онтологии введены индивидуалы, которые позволяют задавать списки значений классов *CredEnum*, *SeverityEnum* и *StatusEnum*. Таким образом, класс *CredEnum* содержит следующий список значений: NOT_REPRODUCED, REPRODUCTION_FAILED, REPRODUCTION_SUCCEEDED, CREDIBLE. При этом, NOT_REPRODUCED означает, что на данный момент не было попыток тестирования, либо потому что это не возможно для такого рода компонентов, либо возможности тестирования в настоящее время нет, или такое тестирование вовсе не реализовано. Эти ситуации могут быть также описаны различными ситуациями. REPRODUCTION_FAILED означает, что повторный тест для воспроизведения результатов не дал тот же результат. REPRODUCTION_SUCCEEDED подтверждает успешное повторение. Состояние CREDIBLE – состояние по умолчанию для события от поставщика. Можно объединить два последних состояния, так как повторение показывает, что результаты достоверны.

Класс *SeverityEnum* содержит следующий список возможных значений: SMALL, BELOW_AVERAGE, AVERAGE, ABOVE_AVERAGE, HIGH. Данный список определяет степень тяжести, которая связанна с экземпляром некоторого класса событий. В рамках работы список *SeverityEnum* представляет собой набор термов для описания лингвистической переменной, которая характеризует степень тяжести.

При этом:

- терм SMALL представляет собой нечёткое множество, которое описывает степень тяжести как маленькую;

- терм BELOW_AVERAGE представляет собой нечёткое множество, которое описывает степень тяжести как ниже средней;

- терм AVERAGE представляет собой нечёткое множество, которое описывает степень тяжести как средняя;

- терм ABOVE_AVERAGE представляет собой нечёткое множество, которое описывает степень тяжести как выше средней;

- терм HIGH представляет собой нечёткое множество, которое описывает степень тяжести как высокую.

Класс *StatusEnum* содержит следующий список: OPEN, SUSPENDED, CORRELATED, TIMEDOUT. OPEN обозначает новое событие, которое произошло и для которого начался поиск корреляции с другим событием. Статус CORRELATED означает, что событие было полностью согласовано с возможно связанным событием так, что оно не должено рассматриваться само по себе. SUSPENDED состояние вводится для обозначения, что событие ждет результатов тестирования. TIMEDOUT означает, что в текущий момент времени событие удалено из активных событий.

3.3.4 Онтологическая модель описания зависимостей между управляемыми элементами РИС

Как было указано в подразделе 1.2, для диагностирования неисправностей сервисов, помимо описаний самих сервисов, необходимы средства моделирования зависимостей, которые описывают сложные взаимодействия на сервисном уровне и на уровне ресурсов. Это необходимо, для автоматического определения связи признака неисправности на сервисном уровне с ресурсами, которые используются для реализации этого сервиса.

Для описания зависимостей между сервисами и ресурсами была разработана онтологическая модель зависимостей между управляемыми элементами, представлена на рис. 3.7.

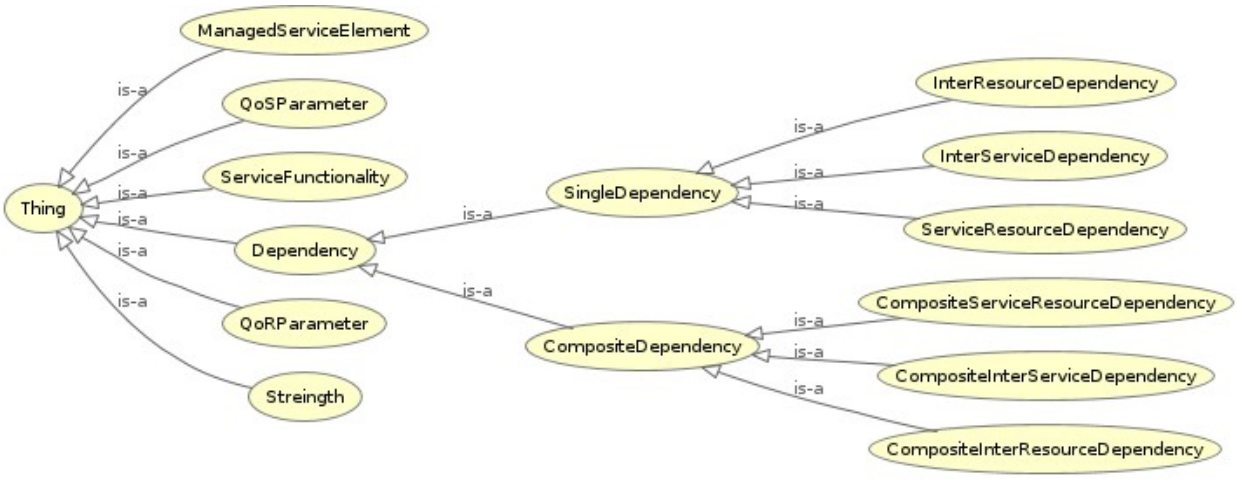


Рисунок 3.7 – Онтологическая модель зависимостей между управляемыми элементами.

Далее приведены понятия введённые в рамках онтологической модели зависимостей между управляемыми элементами:

- класс *ManagedServiceElement*, который представляет управляемый элемент сервиса *MSE* и является суперклассом для классов сервисов и ресурсов,

эквивалентный абстрактному классу *ManagedServiceElement* из онтологии *Common*;

- класс *ServiceFunctionality*, который представляет сервисные функции объекта класса *Service*, являющегося управляемым элементом сервиса, эквивалентный классу *ServiceFunctionality* из онтологии *Common*;

- класс *QoRParameter*, который представляет свойства управляемого элемента сервиса *MSE*, и используется для задания параметров качества ресурсов, эквивалентный классу *QoRParameter* из онтологии *Common*;

- класс *QoSParameter*, который представляет свойства управляемого элемента сервиса *MSE*, и используется для задания параметров качества ресурсов, эквивалентный классу *QoSParameter* из онтологии *Common*;

- класс *Dependency*, абстрактный класс, который используется для описания зависимостей между *MSE*;

- класс *SingleDependency*, абстрактный класс, который используется для описания одиночной зависимостью между двумя *MSE*. Является подклассом *Dependency*;

- класс *CompositeDependency*, абстрактный класс, который используется для описания композиции двух или более зависимостей (*Dependency*). Является подклассом *Dependency*;

- класс *InterResourceDependency*, который используется для описания зависимости между двумя ресурсами (*Resource*). Является подклассом *SingleDependency*;

- класс *InterServiceDependency*, который используется для описания зависимости между двумя сервисами (*Service*). Является подклассом *SingleDependency*;

- класс *ServiceResourceDependency*, который используется для описания зависимости между одним ресурсом (*Resource*) и одним сервисом (*Service*). Является подклассом *SingleDependency*;

- класс *CompositeInterResourceDependency*, который используется для описания композиции двух или более зависимостей (*Dependency*). Является подклассом *Dependency*;

- класс *Streingth*, который используется для описания силы зависимости.

Онтологическая модель зависимостей между управляемыми элементами содержит следующие отношения между классами:

- *dependent* задаёт отношение между классом *Dependency* и классом *ManagedServiceElement*. Класс *Dependency* является доменом, а класс *ManagedServiceElement* — областью значений;

- *strength* задаёт отношение между классом *Dependency* и классом *Streingth*. Класс *Dependency* является доменом, а класс *Streingth* — областью значений;

- *antecedent* задаёт отношение между классом *SingleDependency* и классом *ManagedServiceElement*. Класс *SingleDependency* является доменом, а класс *ManagedServiceElement* — областью значений;

- *consistsOf* задаёт отношение агрегации между составной зависимостью (*CompositeDependency*) и зависимостями из которых она состоит (*Dependency*). Класс *CompositeDependency* является доменом, а класс *Dependency* — областью значений;

- *adminStatusCurrent* и *adminStatusHistory* задают отношения между классом *Dependency* и классом *AdminStatus*. Класс *Dependency* является доменом, а класс *AdminStatus* — областью значений. При этом отношение *adminStatusCurrent* используется для определения текущего статуса, а отношение *adminStatusHistory* для определения прошлых статусов;

- *antecedentQoR* задаёт отношение между множеством терминов, получающихся объединением классов *InterResourceDependency* и *ServiceResourceDependency*, и классом *Resource*. Классы *InterResourceDependency* и *ServiceResourceDependency* является доменом, а класс *Resource* — областью значений;

- *antecedentQoS* задаёт отношение между классом *InterServiceDependency* и классом *Service*. Класс *InterServiceDependency* является доменом, а класс *Service* — областью значений;

- *antecedentFunction* задаёт отношение между классом *InterServiceDependency* и классом *ServiceFunctionality*. Класс *InterServiceDependency* является доменом, а класс *ServiceFunctionality* — областью значений;

- *dependentQoR* задаёт отношение между классом *InterResourceDependency* и классом *Resource*. Класс *InterResourceDependency* является доменом, а класс *Resource* — областью значений;

- *dependentQoS* задаёт отношение между множеством терминов, получающихся объединением классов *InterServiceDependency* и *ServiceResourceDependency*, и классом *Service*. Классы *InterServiceDependency* и *ServiceResourceDependency* является доменом, а класс *Service* — областью значений;

- *dependentFunction* задаёт отношение между множеством терминов, получающихся объединением классов *InterServiceDependency* и *ServiceResourceDependency*, и классом *ServiceFunctionality*. Классы *InterServiceDependency* и *ServiceResourceDependency* является доменом, а класс *ServiceFunctionality* — областью значений.

Разработанные онтологические модели описания элементов и зависимостей РИС и описания диагностических тестов, за счёт применения онтологического подхода, позволили упростить для экспертов-диагностов процесс поддержания базы знаний диагностики РИС в актуальном состоянии и показали сокращение времени на её обслуживание, в среднем, на 39% (см. пункт 4.2). Кроме того, онтологические модели описания элементов и зависимостей РИС вводят термины и понятия, которые могут использоваться для построения онтологической модели описания штатных и нештатных ситуаций.

3.4 Разработка онтологической модели описания штатных и нештатных ситуаций

Для формализованного описания знаний о штатных и нештатных ситуация, с использованием понятий введённых на первом и втором уровне абстракции, была разработана онтологической модели описания штатных и нештатных ситуаций. Разработанная онтологическая модель, как и предыдущие, базируется на онтологическом подходе, но строится на основе базовых терминов многосортного языка прикладной логики, используемых для описания знаний предметной области, а также ограничений на их значения. Данный подход был применен, поскольку основным недостатком подхода на основе OWL является то, что дескриптивная логика является монотонной логикой. Это не позволяет описывать в рамках OWL ситуации, при которых значения параметров изменяются во времени. Для преодоления этих ограничений использовалась прикладная логическая теория.

Описание предметной области представлено в виде четырех модулей:

- Базовые понятия и онтологические соглашения, описывающие параметры онтологической модели;
- Базовые понятия и онтологические соглашения, описывающие неизвестные онтологической модели;
- Термины знаний, описывающих штатные ситуации (ШС), и онтологические соглашения о соответствии между ними;
- Термины знаний, описывающих нештатные ситуации (НШС), и онтологические соглашения о соответствии между ними.

Как было сказано в пункте 1.4.3, каждый конкретный язык прикладной логики включает ядро, а также обычно стандартное расширение и некоторые специализированные расширения. Таким образом, каждый конкретный язык прикладной логики характеризуется некоторой совокупностью названий

расширений, а не сигнатурой. Сигнатура же вводится в каждой конкретной логической теории, заданной на таком языке.

3.4.1 Базовые понятия и онтологические соглашения расширения «Определение разбиений»

Множество имен, входящих в прикладную логическую теорию, может быть разделено на две непересекающиеся части: множество однозначно интерпретируемых имен и множество неоднозначно интерпретируемых имен. Имя является однозначно интерпретируемым, если выполнено одно из условий:

- прикладная логическая теория не определяет для имени n ни сорта, ни значения; в этом случае при любом α имеет место $\alpha(n) = n$;

- прикладная логическая теория определяет для имени n значение e, которое не зависит от интерпретаций других имен; в этом случае при любом α имеет место $\alpha(n) = e$;

- прикладная логическая теория определяет для имени n значение e, которое однозначно определяется интерпретацией других имен.

Все остальные имена являются неоднозначно интерпретируемыми. Для каждого такого имени n прикладная логическая теория определяет сорт s, но не определяет значения. В этом случае любая функция интерпретации α удовлетворяет ограничению $\alpha(n) \in s$.

Функция интерпретации α является допустимой для прикладной логической теории, если все входящие в редукцию этой прикладной логической теории предложения имеют смысл при этой функции интерпретации. Прикладная логическая теория является семантически корректной, если существует допустимая функция интерпретации α.

Поскольку множество допустимых подстановок для каждого предложения определяется префиксом однозначно, а допустимая функция интерпретации -

неоднозначно, то семантически корректная прикладная логическая теория определяет множество допустимых функций интерпретаций.

Легко видеть, что при этих условиях множество неоднозначно интерпретируемых имен любой семантически корректной прикладной логической теории конечно для любой допустимой функции интерпретации.

Сужение допустимой функции интерпретации α на множество неоднозначно интерпретируемых имен будем называть моделью прикладной логической теории. Модель прикладной логической теории может быть задана таким множеством описаний значений имен, что после добавления этого множества к прикладной логической теории все имена полученной таким образом логической теории будут однозначно интерпретируемыми.

Термом специализированного расширения «Категории» является структура $x = (s_i \to t_i)_{i=1}^{I}$, где I - количество элементов, $(s_i)_{i=1}^{I}$ - имена, а $(t_i)_{i=1}^{I}$ - термы, значениями которых являются множества.

Значением терма x является множество структурных значений, являющееся областью определения всех возможных отображений с именами $(s_i)_{i=1}^{I}$ областями значений которых являются значения термов $(t_i)_{i=1}^{I}$ соответственно. Отображения с именами $(s_i)_{i=1}^{I}$ будем называть атрибутами, а значения этих отображений для конкретного структурного значения – значениями атрибутов этого структурного значения.

Если x есть структурное значение, принадлежащее значению терма $(s_i \to t_i)_{i=1}^{I}$ то любое s_i, которое входит в термы $(t_i)_{i=1}^{I}$, считается термом, значение которого совпадает со значением терма $s_i(x)$.

Термин *разбиения* обозначает множество всех возможных разбиений множества целых неотрицательных чисел; каждое разбиение представляет собой конечную строго возрастающую последовательность.

разбиения ≡(∪ (длина: I[0, ∞)) {(последовательность: I ⇑ (длина+1)) (&(элемент : I[1, длина]) π(элемент, последовательность) < π(элемент+1, последовательность))})

Термин *el* обозначает функцию, аргументами которой являются некоторое разбиение и целое число в диапазоне от 0 до числа элементов в этом разбиении, а результатом – элемент этого разбиения, номер которого равен второму аргументу.

el ≡ (λ(разбиение: разбиения) (элемент: I[0, length(разбиение)–1)])π(элемент+1, разбиение))

Термин *interval* обозначает функцию, аргументами которой являются некоторое разбиение и натуральное число, не превосходящее число элементов в этом разбиении, а результатом – интервал целых неотрицательных чисел между элементом этого разбиения с номером, равным второму аргументу, и элементом с предыдущим номером.

interval ≡(λ(разбиение: разбиения) (элемент: I[1, length(разбиение)–1)]) I[element(разбиение, элемент–1), element(разбиение, элемент)])

3.4.2 Базовые понятия и соглашения, описывающие параметры модели онтологии

В этом пункте описаны введенные базовые термины, используемые для описания знаний предметной области, а также ограничения на их значения (не зависящие от значений терминов для описания действительности).

(1) Термин *службы* обозначает совокупность объектов которые определяются соотношениями (2.1).

сорт службы : $\{\}N$

(2) Термин *источники* обозначает совокупность источников диагностической информации которые определяются соотношениями (2.2). В знаниях должен быть описан хотя бы один источник диагностической информации.

сорт *источники*: $\{\}\mathrm{N} \setminus \{\varnothing\}$

(3) Термин *параметры* обозначает класс понятий, соответствующих наблюдаемым диагностическим параметрам. В знаниях должен быть описан хотя бы один параметр.

сорт *параметры*: $\{\}\mathrm{N} \setminus \{\varnothing\}$

(4) Термин *параметры_ источники* обозначает функцию, которая каждому источнику диагностической информации сопоставляет множество диагностических параметров в соответствии с формулой (2.3).

сорт *параметры_ источники*: *источники* $\rightarrow \{\}$*параметры*

(5) Термин *параметры_службы* обозначает функцию, которая каждой службе сопоставляет множество диагностических параметров в соответствии с формулой (2.4).

сорт *параметры_ службы*: *службы* → {}*параметры*

(6) Термин *собст_параметры_службы* обозначает функцию, которая каждой службе сопоставляет множество собственным параметром службы

сорт *собст_параметры_службы*: *службы* → {}*параметры*

(7) Для каждой службы множество *собст_параметры_службы* принадлежит множеству *параметры_службы*.

$$(x : \textit{службы}\ )(\textit{собст\_пара}\ \ \textit{метры\_служ}\ \ \textit{бы}(x) \subset \textit{параметры\_}\ \ \textit{службы}\ (x))$$

(8) Термин *особенности* обозначает класс понятий, соответствующих особенностям объектов, которые должны учитываться при диагностике.

сорт *особенности*: {}N

(9) Термин *приоритеты* обозначает класс понятий, соответствующих множеству определяемому соответствии с формулой (*)

сорт *приоритеты*: {}N

(10) Термин *множества значений* обозначает множество всех допустимых множеств скалярных значений.

множества значений ≡ {}N \ {∅}

(11) Значения не совпадают с названиями параметров.

параметры ∩ (∪ (x: *множества значений*) x) = ∅

(12) Термин возможные значения обозначает функцию, которая параметрам сопоставляет их возможные значения.

сорт *возможные значения*: *параметры* → *множества значений*

(13) У каждого параметра не менее двух возможных значений.

(x: *параметры*) μ(возможные значения(x)) ≥ 2

(14) Термин *события* обозначает класс понятий, соответствующих событиям, которые которые должны учитываться при решении задачи.

сорт *события* = {}N

(15) Названия всех параметров и событий различны.

параметры ∩ *события* = ∅

(16) Каждый термин, входящий в класс терминов события, обозначает структурное значение с тремя атрибутами:

служба,

параметр,

источник.

Значением первого является наименование службы, второй атрибут является наименованием параметра.

(событие: *события*) сорт событие:

(*служба* → *службы*,

параметр → *параметры,*

источник → *источники*)

(17) Термин *условия* обозначает множество всех возможных условий. Это множество, состоящих из элементов – структурных значений. Каждое условие есть конечное множество структурных значений. Каждое структурное значение имеет атрибуты особенность и область значений. Значением первого является имя особенности, а второго – собственное подмножество возможных значений этой особенности. Пустое множество представляет тождественно истинное условие.

условия ≡ {} {(условие: (особенность → *особенности*,

область значений → *множества значений*))

область значений(условие) ⊂

⊂ возможные значения(особенность(условие))}

(18) Термин *условия события* обозначает функцию, которая каждому событию сопоставляет необходимые условия его существования

сорт *условия события* : *события* → *условия*

(19) Термин *причины отклонений* обозначает класс понятий, соответствующих причинам отклонений, описания которых представлены в знаниях. В знаниях должно присутствовать описание хотя бы одной причины отклонений.

сорт *причины отклонений*: {}N \ {∅}

(20) Каждый термин, входящий в класс терминов причины отклонений, обозначает структурное значение с тремя атрибутами:

число периодов развития (*чпр*),

периоды развития (*пр*),

необходимое условие (*ну*).

Значением первого является натуральное число, второй атрибут является функцией, которая номеру периода развития причины отклонений сопоставляет интервал, значение третьего атрибута – условие, необходимое для существования этого причины отклонений в ситуации (если значением атрибута является пустое множество, то условие считается истинным).

(причина: *причины отклонений*) сорт причина:

(*чпр* → I[1, ∞),

пр → (I[1, *чпр*] → *интервал*) ,

ну → *условия*)

(21) Термин *периоды динамики* это множество структурных значений с атрибутами длительность и область значений следствия. Значением первого атрибута является интервал, а второго – множество значений.

периоды динамики ≡ (

длительность → интервал,

область значений следствия → множества значений)

(22) Термин *интервал* это множество элементов структурных значений с атрибутами нижняя граница и верхняя граница. Их значениями являются натуральные числа – минимальная и максимальная длительности интервала, измеряемая целым числом, причём верхняя граница больше нижней.

интервал ≡ (нижняя граница → I[1, ∞),

верхняя граница → I[нижняя граница + 1, ∞))

4.1.3 Базовые понятия и соглашения, описывающие неизвестные модели онтологии

Действительность в диагностике рассматривается как множество ситуаций, каждая из которых соответствует диагностическому случаю. В настоящем разделе вводятся базовые термины, используемые для описания ситуаций, а также ограничения на их значения.

(23) Термин *моменты* обозначает функцию, которая каждому параметру и событию сопоставляет множество целых неотрицательных чисел – моментов времени в ситуации, когда наблюдался этот параметр. Если для некоторого параметра значение этой функции есть пустое множество, то это означает, что этот параметр не наблюдался.

сорт *моменты*: *параметры* ∪ *события* → {}I [0, ∞)

(24) Каждый термин, входящий в класс терминов *параметры* , обозначает функцию, которая сопоставляет моментам наблюдения этого параметра его значения в эти моменты.

(параметр : *параметры*) сорт *параметр* :
моменты(*параметр*) → возможные значения(*параметр*)

(25) Термин *наблюдавшиеся события* обозначает подмножество событий, наблюдавшихся в ситуации.

наблюдавшиеся события: {} *события*

(26) Каждый термин, входящий в класс терминов наблюдавшиеся события, обозначает значение, которое принимает параметр.

(событие: *наблюдавшиеся события*) сорт событие:
возможные значения(параметр(событие))

(27) Термин выполнено обозначает предикат, аргументом которого является элемент множества условия и который истинен тогда и только тогда, когда для каждой составляющей этого элемента, значение первого атрибута структурного значения в ситуации принадлежит второму атрибуту (область значений). Пустое условие тождественно истинно.

выполнено ≡ (λ (y: условия) y ≠ ∅ ⇒ (& (x: y) событие(x) ∈
∈ *наблюдавшиеся события* ⇒ j(параметр(событие(x))) ∈ область значений(x)))

(28) Если в ситуации хотя бы один раз наблюдалось некоторое событие, то для него должно быть выполнено условие.

(x: *наблюдавшиеся события*) моменты(x) $\neq \varnothing \Rightarrow$ выполнено(условия события(x))

(29) Термин диагноз обозначает множество причин отклонений. Если отклонений нет, его диагноз есть пустое множество.

сорт *диагноз*: {}*причины отклонений*

(30) Если некоторая причина отклонения входит в диагноз, то для этой причины отклонений должно быть выполнено необходимое условие.

(x: диагноз) выполнено(ну(x))

(31) Термин действительности *развитие* обозначает функцию, которая каждой причине отклонений, сопоставляет разбиение оси времени, каждый интервал которого соответствует некоторому периоду развития этой причины, и каждому наблюдавшемуся событию сопоставляет разбиение оси времени, внутри каждого интервала которого значения этого события определяются одной и той же связанной с этим интервалом причиной.

сорт *развитие*: диагноз $\cup$ {(x: *наблюдавшиеся события*)
моменты(x) $\neq \varnothing$} $\rightarrow$ разбиения

(32) Интервал, на котором наблюдается развитие некоторого параметра, покрывает все моменты наблюдения этого параметра.

(x: *параметры*) моменты(x) $\neq \varnothing \Rightarrow$ el(развитие(x), 0) $\leq$ inf(моменты(x)) && el(развитие(x),length(развитие(x))) $\geq$ sup(моменты(x))

(33) Термин *интервалы развития параметра* (*U*) это множество структурных значений, состоящих из двух атрибутов: параметр и номер интервала. Значением первого атрибута является имя параметра, второго – номер интервала его развития.

U≡ (параметр $\rightarrow$ *параметры*,

номер интервала $\rightarrow$ I[1, length(развитие(параметр)) – 1])

(34) Если причина отклонения входит в диагноз, то число периодов развития этой причины отклонений совпадает с числом периодов его развития в базе знаний, а длительность каждого периода развития лежит между нижней и верхней границами длительностями этого периода развития причины отклонений.

(x: диагноз) length(развитие(x)) = чпр(x)+1&

& (& (n: I[1, length(развитие(x)) – 1]) el(развитие(x), n) – el(развитие(x), n – 1) $\in$

$\in$ I[нижняя граница(пр(x)(n),верхняя граница(пр(x)(n)])

Описание введенных понятий оставшихся модулей вынесено в приложение А.

Разработанная онтологическая модель, как онтологические модели второго уровня абстракции структуры базы знаний, благодаря применению онтологического подхода, позволила сократить временные затраты на

поддержание актуальности базы знаний штатных и нештатных ситуаций в актуальном состоянии, в среднем, на 21% (см. пункт 4.2).

Данная онтологическая модель является заключительной моделью описания знаний в этой работе. Её завершение позволило усовершенствовать метод диагностики РИС учетом новых моделей описания знаний.

3.5 Усовершенствование метода диагностики РИС

Для применения разработанных моделей в диагностировании получил дальнейшее развитие метод диагностирования РИС. Ключевым отличием развитого метода является использование нечетких знаний для ранжирования определенных возможных диагнозов.

3.5.1 Постановка задачи диагностики

В рамках разработанного метода, задача диагностики состоит в определении всех возможных альтернативных диагнозов на основе знаний предметной области и данных наблюдений, значений особенностей (постоянные во времени). В каждый диагноз могут входить несколько причин отклонений, протекающих одновременно или последовательно. Считается, что все наблюдения протекают на определенном временном отрезке, называемом периодом наблюдения, на котором моменты времени измеряются целыми неотрицательными числами, каждое из которых есть число часов, прошедших с момента начала наблюдений (0 часов).

Такая задача относится к классу обратных задач. Ей соответствует прямая задача, состоящая в определении возможных значений наблюдаемых параметров в те или иные моменты времени на основе известного диагноза и значений особенностей.

Ниже приводится математическая постановка прямой задачи в терминах разработанных ранее онтологических моделей.

Даны знания, удовлетворяющие ограничениям целостности знаний:

- знания о наблюдениях параметрах, особенностей и их возможных значениях;

- знания о причинах отклонений и длительностях их периодов развития;

- знания о значениях параметров в штатных ситуациях;

- знания о причинно-следственных связях между причинами отклонений и наблюдениями;

Даны результаты наблюдений:

- наблюдавшиеся особенности и результаты их наблюдения (значения);

- диагноз.

Требуется найти значения параметров в определенные моменты времени и для каждого указанного выше значения - его причину - причинно-следственную связь. Связи между тем, что дано, и тем, что требуется найти, определяются моделью онтологии и системой знаний.

В терминах используемой модели онтологии математическая постановка задачи диагностики (обратной задачи) может быть сформулирована следующим образом:

Даны знания, удовлетворяющие ограничениям целостности знаний:

- знания о наблюдениях и их возможных значениях;

- знания о причинах отклонений и длительностях их периодов развития;

- знания о значениях параметров в штатных ситуациях;

- знания о причинно-следственных связях между причинами отклонений и наблюдениями;

Даны результаты наблюдений:

- наблюдавшиеся особенности и результаты их наблюдения (значения);

- наблюдавшиеся параметры, моменты их наблюдения и их значения в эти моменты.

Требуется найти диагноз.

Для решения поставленной задачи, разработан новый алгоритм, ориентированный на использование базы знаний, предложенной в подразделе 2.1.

3.5.2 Алгоритм решения задачи диагностики

Основной алгоритм решения обратной задачи.

1. Проверить гипотезу о том, что наблюдается ШС (см. функцию Проверка_ШС).

2. Если гипотеза о том, что наблюдается ШС, подтвердилась, то считать:

3.1) результатом что отклонения отсутствуют и наблюдается ШС;

3.2) иначе перебрать все причины отклонений из базы знаний, причем для каждого:

а) проверить выполнение необходимого условия для этой причины отклонений;

б) если необходимое условие выполнено, то проверить гипотезу о том, что наблюдается НШС вызванная этим отклонением (см. функцию Проверка_НШС). в случае ее подтверждения добавить это отклонение к множеству решений.

4. Завершить работу, выдав полученные результаты.

Функция *Проверка_ШС* предназначена для проверки гипотезы о том, что наблюдается ШС. Гипотеза о том, что наблюдается ШС, считается подтвержденной, если для каждого наблюдаемого параметра подтверждена гипотеза о том, что наблюдаемые значения этого параметра могут иметь место в некоторой ШС. Если хотя бы для одного наблюдаемого параметра такая гипотеза опровергается, то считается, что имеет место в НШС.

Ниже приведено описание алгоритма функции *Проверка_ШС*

гипотеза ШС = ЛОЖЬ
ДЛЯ ВСЕХ параметр ∈ наблюдавшиеся параметры ВЫПОЛНЯТЬ
гипотеза = ЛОЖЬ
гипотеза = Проверка_ШС_по_параметру(параметр)
ЕСЛИ гипотеза == ЛОЖЬ
ТО
гипотеза ШС = ЛОЖЬ
ВЫХОД ИЗ ЦИКЛА
ИНАЧЕ
гипотеза ШС = ИСТИНА
ЗАКОНЧИТЬ ТО
ЗАКОНЧИТЬ ДЛЯ ВСЕХ

Функция *Проверка_ШС_по_параметру(параметр)* предназначена для проверка гипотезы о том, что все наблюдаемые значения некоторого параметра могут иметь в ШС. Эта гипотеза считается подтвержденной, если для каждого наблюдавшегося значения параметра найден некоторый вариант ШС, протекавшей на протяжении всего периода наблюдений. Для осуществления этого поиска строится множество гипотез о развитии причинно-следственных связей (далее – МГ), куда входят варианты ШС, которые могут определять значения этого параметра. Наблюдаемые значения каждого параметра проверяются последовательно, начиная с самого раннего.

Ниже приведено описание алгоритма функции *Проверка_ШС_по_параметру (параметр)*

ДЛЯ ВСЕХ x ∈ *ШС ВЫПОЛНЯТЬ*
ЕСЛИ следствие(x) == *параметр*

ТО

ДЛЯ ВСЕХ v ∈ вариант(x) *ВЫПОЛНЯТЬ*

ЕСЛИ условия на факторы (*v*) == *ИСТИНА*

ДОБАВИТЬ В МГ(x, v, период динамики =1)

ЗАКОНЧИТЬ ДЛЯ ВСЕХ

ЗАКОНЧИТЬ ТО

ЗАКОНЧИТЬ ДЛЯ ВСЕХ

гипотеза= ЛОЖЬ

1: ДЛЯ ВСЕХ m ∈ МГ ВЫПОЛНЯТЬ

pТЕК=текущий период динамики

ЕСЛИ (поиск_ШС(pТЕК , m, параметр) == ИСТИНА)

2: ТО

запомнить, что все наблюдаемые значения рассматриваемого признака могут наблюдаться в ШС

гипотеза = ИСТИНА

2: ИНАЧЕ

запомнить, что все наблюдаемые значения рассматриваемого признака не могут наблюдаться в ШС

гипотеза= ЛОЖЬ

2*: ЗАКОНЧИТЬ ТО*

1: ЗАКОНЧИТЬ ДЛЯ ВСЕХ

Функция *поиск_ШС (p, m, параметр)* предназначена для проверка гипотезы о том, что некоторое значение параметра возможно в ШС. Для проверки этой гипотезы из МГ выбирается в качестве причины значения параметра такая причинная связь, что наблюдаемое значение принадлежит возможным значениям варианта ШС, и гипотеза о том, что следующее значение параметра возможно в данном варианте ШС не опровергнута.

Ниже приведено описание алгоритма функции *поиск_ШС (р, т, параметр)*.

гипотеза НШС = ЛОЖЬ

1: ДЛЯ ВСЕХ параметр ∈ наблюдавшиеся параметры ВЫПОЛНЯТЬ

2: ДЛЯ ВСЕХ x *∈ НШС ВЫПОЛНЯТЬ*

ЕСЛИ (НЕ (следствие(х) == параметр)) И (причина(х) == причина отклонений)

3: ТО

гипотеза НШС = ЛОЖЬ

4: ВЫХОД ИЗ ЦИКЛА

3*: ЗАКОНЧИТЬ ТО*

2: ЗАКОНЧИТЬ ДЛЯ ВСЕХ

гипотеза = Проверка_ШС_по_параметру(параметр)

ЕСЛИ гипотеза == ЛОЖЬ

2: ТО

гипотеза ШС = ЛОЖЬ

3: ВЫХОД ИЗ ЦИКЛА

2: ИНАЧЕ

гипотеза ШС = ИСТИНА

2: ЗАКОНЧИТЬ ТО

1: ЗАКОНЧИТЬ ДЛЯ ВСЕХ

ЕСЛИ (гипотеза ШС == ЛОЖЬ) И (гипотеза НШС == ЛОЖЬ)

1: ТО

1: ВЫХОД ИЗ ЗАДАЧИ

1*: ЗАКОНЧИТЬ ТО*

1: ДЛЯ ВСЕХ параметр ∈ наблюдавшиеся параметры ВЫПОЛНЯТЬ

2: ДЛЯ ВСЕХ x *∈ НШС ВЫПОЛНЯТЬ*

ЕСЛИ (следствие(x) == параметр) И (причина(x) == причина отклонений)

3: ТО

t =first (параметр)

ЕСЛИ Задача 4(v, параметр, t) == ИСТИНА

4: ТО

запомнить, что все наблюдаемые значения рассматриваемого признака могут наблюдаться в НШС

гипотеза = ИСТИНА

4: ИНАЧЕ

запомнить, что все наблюдаемые значения рассматриваемого признака не могут наблюдаться в НШС

гипотеза= ЛОЖЬ

4*: ЗАКОНЧИТЬ ТО*

3: ЗАКОНЧИТЬ ТО

2: ЗАКОНЧИТЬ ДЛЯ ВСЕХ

1: ЗАКОНЧИТЬ ДЛЯ ВСЕХ

Алгоритмы оставшихся методов подробно описаны в работе [5].

Сравнительный анализ показал сокращение времени на проведение диагностирования неисправности при применении усовершенствованного метода диагностики РИС в 5-10 раз в зависимости от неисправности и квалификации эксперта-диагноста, без потери достоверности. При этом коэффициент простоя системы в целом сократился в среднем на 7%.

Выводы по третьему разделу

1. Для разработки онтологических моделей второго уровня абстракции структуры базы знаний РИС, была разработана CIM-метаонтология на языке

OWL, для задания базовых конструктов онтологических моделей на основе CIM-моделей.

2. С целью сокращения времени затрачиваемого экспертами-диагностами на поддержание актуальности базы знаний элементов и зависимостей в РИС, а также базы знаний диагностических тестов, на основе CIM-метаонтологии на языке OWL, были разработаны: базовая онтологическая модель расширенного описания элементов и зависимостей в РИС, онтологическая модель описания взаимосвязей сервиса с SLA, SAP и функциональностью сервиса, онтологическая модель описания событий и онтологическая модель описания зависимостей между управляемыми элементами РИС. Их применение позволило сократить требуемые временные затраты на 39%.

3. С целью сокращения времени затрачиваемого экспертами-диагностами на поддержание актуальности базы знаний штатных и нештатных ситуаций, при помощи многосортного языка дескриптивной логики, была разработана онтологическая модель описания штатных и нештатных ситуаций. Её применение позволило сократить время на обслуживание базы знаний ситуаций на 21%.

4. Для применения знаний, описанных разработанными моделями, при диагностировании РИС, а так же сокращения времени, затрачиваемого экспертом-диагностом на обнаружение, локализацию и классификацию неисправности, был развит метод диагностирования РИС, который за счёт использования нечётких данных позволяет сократить время на диагностирования в 5-10 раз, в зависимости от неисправности и квалификации эксперта, без потери достоверности. При этом коэффициент простоя уменьшился, в среднем, на 7%.

РАЗДЕЛ 4
ЭКСПЕРЕМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ РАЗРАБОТАННОЙ ИНФОРМАЦИОННОЙ ТЕХНОЛОГИИ ПОСТРОЕНИЯ БАЗ ЗНАНИЙ ДЛЯ СИСТЕМ ДИАГНОСТИРОВАНИЯ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

4.1 Разработка информационной технологии построения баз знаний для систем диагностирования РИС

В данном разделе приводится описание разработанной информационной технологии построения баз знаний, на основе описанной в подразделе 2.1 структуре и с применением, разработанных в подразделе 2.2 и разделе 3, моделей и метода.

Разработанная информационная технология представлена на рисунке 3.

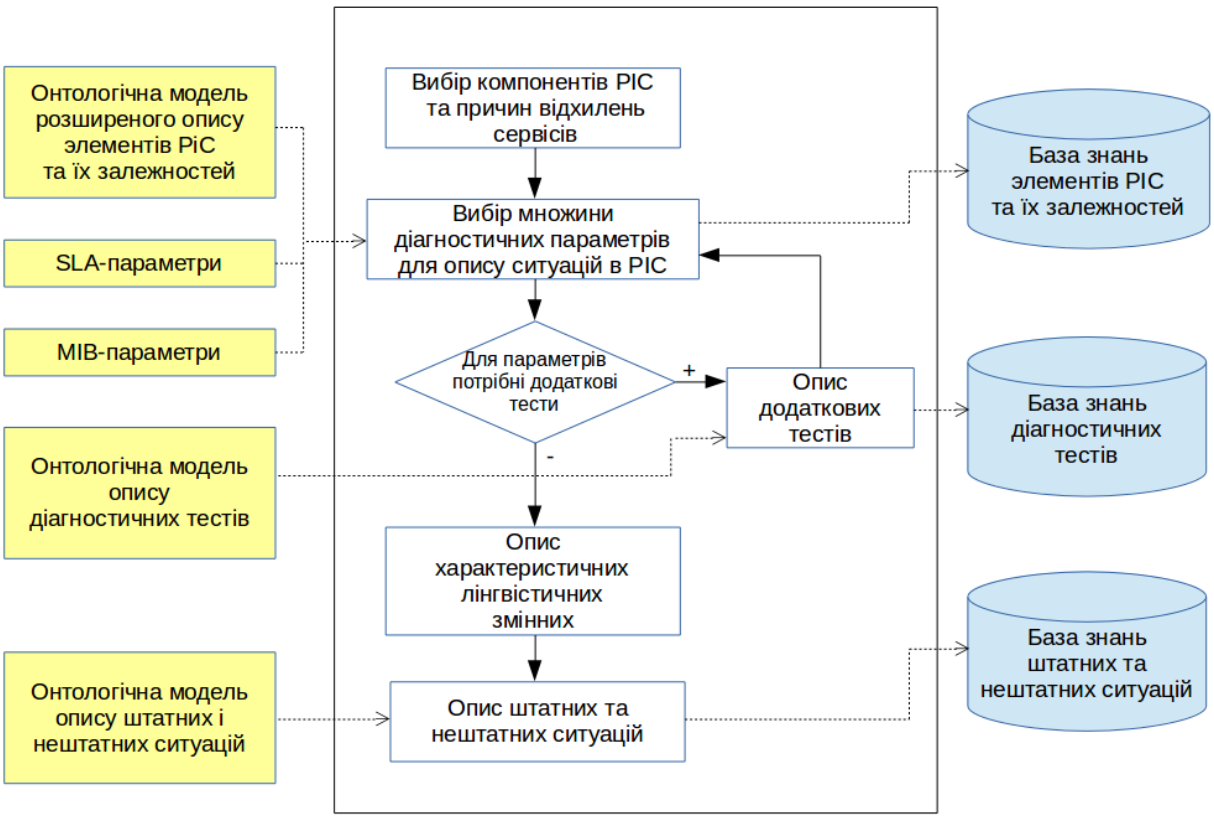


Рисунок 4.1 – Информационная технология построения баз знаний для диагностирования РИС

Информационная технология предусматривает ряд этапов построения базы знаний, которые выполняются группой экспертов в области диагностики РИС с использованием разработанных и представленных во втором и третьем разделе моделей.

На первом этапе определяются множества диагностических параметров, соответствующих сервисам, которые, в настоящее время, являются критичными для РИС.

На втором этапе определенные соответствия диагностических параметров и сервисов с использованием SLA-параметров и MIB-параметров, согласно онтологической модели расширенного описания элементов рыночной инфраструктуры, формируют базу знаний элементов РИС и их зависимостей.

Если для получения некоторых диагностических параметров необходимо выполнение дополнительных диагностических тестов, на третьем этапе проводится описание необходимых дополнительных тестов, в соответствии с онтологической моделью описания диагностических тестов. На этом этапе формируется база знаний диагностических тестов.

На четвертом этапе происходит формализация нечетких экспертных знаний. Для описанных, на втором этапе, диагностических параметров производится построение пост ортогонального семантического пространства и вводятся термы для нечеткого описания характеристических лингвистических переменных.

На заключительном пятом этапе, в соответствии с онтологической моделью описания штатных и нештатных ситуаций и используя характеристические лингвистические переменные, описанные на предыдущем этапе, описывается набор штатных ситуаций и варианты нештатных ситуаций, с фиксацией причин отклонений, которые приводят к нештатным ситуациям, для каждого сервиса, описанного на втором этапе. Данные о штатных и нештатных ситуациях формируют базу знаний о штатных и нештатных ситуациях.

4.2 Экспериментальное исследование разработанной информационной технологии

4.2.1 Описание диагностируемой системы.

В качестве диагностируемой РИС была создана локальная вычислительная сеть, состоящая из одного серверного узла и 5 клиентских. Для развёртывания диагностического программного обеспечения, через отдельный порт сервера подключена вспомогательная локальная сеть, не являющаяся частью диагностируемой системы. Вспомогательная сеть состоит из 3 узлов. На рисунке 4.2 представлена общая схема диагностируемой и вспомогательной сети.

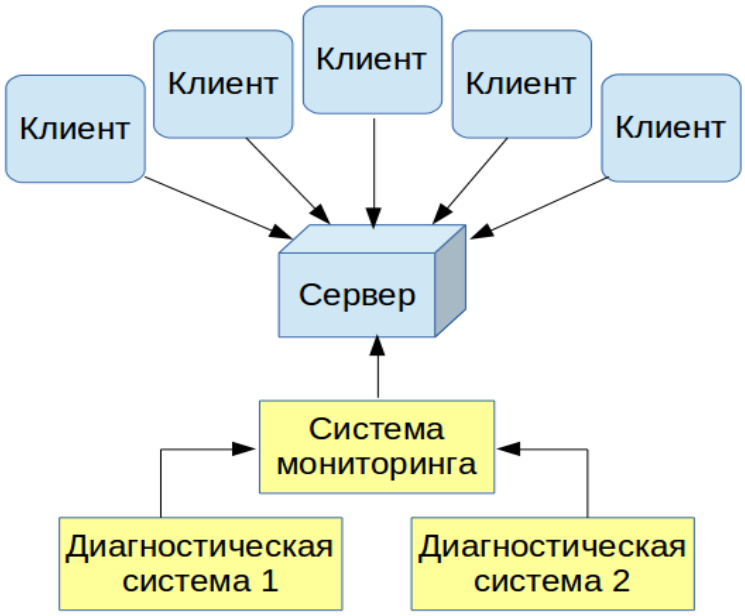


Рисунок 4.2 — Общая схема диагностируемой и вспомогательной сетей.

Клиентские узлы представляли собой персональные компьютеры, подключенные к локальной сети. Состав аппаратного и программного обеспечения изменялся согласно разработанного плана (см. пункт 4.2.3). Для

имитации активности узлов использовались специальные скрипты, отправляющие различные запросы к различным сервисам сервера.

Серверный узел представлял собой персональный компьютер повышенной мощности, который обеспечивал работу сервисов, перечисленных в таблице 4.1.

Таблица 4.1 — Сервисное программное обеспечение серверного узла.

Сервис	Программное обеспечение	Версия
СУБД	Oracle	11 xe
Файловый сервер	FileZilla	0.9.41
DNS-сервер	Windows DNS server	-
Система учёта трафика	TMeter	13.2.659

Узел системы мониторинга (СМ) представлял собой персональный компьютер без устройств ввода и вывода. Основное назначение данного узла, обеспечивать работу программного обеспечения для мониторинга параметров диагностируемой системы через сетевой интерфейс по протоколу SNMP.

Узлы диагностических систем (ДС) представляли собой персональные компьютеры, которые обеспечивали работу диагностических систем и предоставляли экспертам-диагностам интерфейс удалёного доступа к диагностируемой системе.

4.2.2 Описание экспериментальной диагностической системы и баз знаний.

Для проведения сравнительного анализа разработанных моделей и метода, в рамках испытаний, была разработана диагностическая система основанная на знаниях с использованием гибридного RBR-CBR подхода к рассуждениям. Согласно описанной в подразделе 4.1 информационной технологии, была создана экспериментальная база знаний которая, благодаря

модульности своей структуры, позволила заменять, при необходимости, отдельные модули. Для проведения сравнительных испытаний были разработаны 2 модуля описания обобщенной модели диагностирования РИС - базовая (не использующая при описании нечёткие данные) и усовершенствованная (использующая нечёткие данные), модуль расширенного описания элементов и зависимостей в РИС, модуль описания диагностических тестов и модуль описания штатных и нештатных ситуаций. Диагностическая система так же включала в себя 2 реализации метода нахождения вариантов диагнозов — базовый (основанный на базовой обобщенной модели диагностики) и усовершенствованный (основанный на моделях, предложенных в данной работе). База знаний наполнена экспертами диагностами системы.

В процессе испытаний на 2 узла ДС устанавливались разработанные диагностические системы, при этом конфигурация 1-го и 2-го узла отличались за счет выбора метода диагностирования и конфигурации базы знаний.

4.2.3 Описание этапов проведения испытаний

Сравнительные испытания проходили в 4 этапа. Каждый этап имел свою цель, характеризовался своим набором конфигураций для ДС и замеряемыми временными затратами. На первом этапе проводился сравнительный анализ временных затрат экспертов диагностов на поддержание актуальности баз знаний, построенных на основе базовой обобщенной модели диагностирования и на основе усовершенствованной. На втором этапе проводилось сравнение затрат времени на поддержание базы знаний об элементах и зависимостях в РИС, а так же диагностических тестов с использованием разработанных в работе моделях и без них. На третьем этапе сравнивалось время, затрачиваемое на поддержание актуальности базы знаний о штатных и нештатных ситуациях на основе предложенной онтологической модели и без неё. На четвёртом этапе проводился сравнительный анализ времени затрачиваемого экспертами-

диагностами, на нахождение возможных вариантов диагноза и времени затрачиваемого на поддержание актуальности базы знаний при использовании разработанной информационной технологии построения база знаний для систем диагностики РИС, и на основе базового подхода.

В таблице 4.2 приведены конфигурации и замеряемые временные показатели на каждом этапе сравнительных испытаний.

Таблица 4.2 — Сводная таблица конфигураций баз знаний и замеряемых временных характеристик по всем этапам

Этап	ДС	Обобщ. модель диагности-ки	Онтолог. модель расшир. описания	Онтолог. модель описания ш/нш сит.	Метод диагности-рования	Замеряемые временные параметры
1	1	базовая	нет	нет	базовый	поддержка
	2	усоверш.	нет	нет	базовый	
2	1	усоверш.	нет	нет	базовый	поддержка
	2	усоверш.	да	нет	базовый	
3	1	усоверш.	нет	нет	базовый	поддержка
	2	усоверш.	нет	да	базовый	
4	1	базовая	нет	нет	базовый	поддержка, диагност.
	2	усоверш.	да	да	усоверш.	

На всех этапах испытание проходило по одному принципу. Эксперты-диагносты (ЭД) на протяжении 4 часов каждый день проводили диагностирование РИС при помощи выделенной им диагностической системы и поддержание актуальности её базы знаний. В зависимости от проводимого этапа, при помощи вспомогательного программного обеспечения, отсекались временные промежутки, на выполнение тех или иных работ, с точностью до десятых минуты. Временные данные заносились в таблицу и складывались в конце дня для каждого ЭД. При оценке временных трудозатрат на поддержание

актуальности, замерялись временные промежутки, затраченные на модификацию базы знаний. При оценке затрат на проведение диагностирования, замерялись промежутки времени от получения сигнала о возникновения нештатной ситуации, до начала восстановительных работ, и от начала до завершения восстановительных работ.

На каждые 2 дня отдельными людьми готовился план модификации аппаратной и программной конфигурации диагностируемой системы, что требовало од ЭД корректировать свои базы знаний, под изменившиеся обстоятельства. Кроме того периодически создавались нештатные ситуации, требующие обнаружения, локализации и классификации.

Поскольку ЭД имели разный уровень квалификации (2 и 11 лет стажа), каждый день они менялись ДС, таким образом, на каждый двухдневный сценарий ЭД успевал поработать на обеих ДС.

4.2.4 Результаты сравнительных испытаний

Итоги первого этапа испытаний показали, что применение усовершенствованной обобщенной модели диагностики РИС, использующую при описании диагностических параметров нечеткие данные, позволяет сократить время затрачиваемое на поддержание актуальности базы знаний в среднем на 57%, по сравнению с моделью на основе исключительно чётких данных.

Второй этап испытаний показал эффективность разработанных онтологических моделей расширенного описания элементов и зависимостей в РИС и диагностических тестов в качестве инструментов упрощающих модификацию знаний о элементах и зависимостях в РИС. Так, их применение позволило сократить временные затраты на поддержку актуальности знаний на 39%.

Согласно результатам третьего этапа, применение онтологической модели описания штатных и нештатных ситуаций позволило сократить время, затрачиваемое на поддержание базы знаний штатных и нештатных ситуаций в актуальном состоянии на 21%.

Проведённые сравнительные испытания на четвёртом этапе показали преимущество усовершенствованного, в рамках работы, метода диагностики РИС, которые позволяет учитывать нечёткие диагностические данные при диагностировании неисправностей, и проводить ранжирование предполагаемых вариантов диагнозов по степени их вероятности, в частности, и всей разработанной информационной технологии построения баз знаний для диагностики РИС в целом.

Таблица 4.3 — Средние временные показатели за деньна 4-м этапе испытаний

Стаж эксперта	ДС1		ДС2	
	Время диагн.	Время вост.	Время диагн.	Время вост.
2 года	2,33 мин.	24,38 мин.	0,39 мин.	24,12 мин.
11 лет	1,07 мин.	13,12 мин.	0,21 мин.	13,15 мин.
Сумма	3,4 мин.	37,5 мин.	0,6 мин.	37,27 мин.

Согласно результатам приведённым в раблице 4.3, время на диагностирование каждой из неисправностей, описанных в базе знаний, удалось сократить в среднем в 7 раз (в 5-10 раз в отдельных случаях), в зависимости от неисправности и квалификации экспертов-пользователей, без потери достоверности. Не смотря на то что, ожидаемо время на восстановление системы практически не изменилось, коэффициент простоя удалось снизить в среднем на 7%. Экономия в 3 минуты в день может показаться несущественной однако, следует учитывать что испытания проводились сменами по 4 часа. Экономия времени за суточной смены составит 18 минут простоя системы, за месяц ежедневного обслуживания — 503-558 минуты (8,4-9,3 часа).

Так же, сравнение временных затрат показало сокращение общего времени на обслуживание базы знаний, построенной согласно разработанной технологии, на 24%.

Таким образом, в разделе четыре была описана информационная технология построения баз знаний для диагностирования РИС, которая предусматривает применение онтологических моделей, описанных в предыдущих главах. Кроме этого, в разделе также описано применение этой информационной технологии с целью сравнительной оценки временных затрат при диагностике с применением нового подхода и без него. Полученные результаты показали преимущество использования разработанной информационной технологии построения баз знаний на основе построенных онтологических моделей при диагностике РИС.

Выводы по четвёртому разделу

1. На основе разработанных моделей и метода разработана информационная технология построения баз знаний для систем диагностирования РИС.

2. Разработана экспериментальная диагностическая система, основанная на знаниях с использованием гибридного RBR-CBR подхода к рассуждениям, ориентированная на использование разработанной модульной иерархической структуры базы знаний. Разработано программно-алгоритмическое обеспечение усовершенствованного метода диагностики РИС.

3. Построена экспериментальная база знаний для диагностики РИС, на основе онтологических моделей описания диагностических аспектов РИС.

4. Сравнительная оценка временных затрат экспертов-диагностов на проведение диагностирования, по сравнению системами аналогами, показала их сокращение в 5-10 раз, и уменьшение коэффициента простоя системы в целом на 7%. Сравнительная оценка временных затрат экспертов-диагностов на

поддержание актуальности базы знаний, по сравнению системами, не использующими онтологический подход, показало их сокращение на 24%.

5. Результаты исследования были внедрены на предприятии «НПО» ДИСКРЕТ» ООО, в учебный процесс и в научно-исследовательские работы Одесского национального политехнического университета.

ВЫВОДЫ

Диссертационная работа содержит ранее не защищенные научные положения и полученные автором новые научно обоснованные результаты, которые заключаются в разработке моделей и информационной технологии построения баз знаний для автоматизированного диагностирования распределенных информационных систем с применением аппарата теории нечетких множеств и онтологического подхода. Это позволило значительно сократить временные затраты на диагностирование распределенных информационных систем без потери достоверности, и временные затраты на поддержание актуальности базы знаний.

1. Проведенный анализ существующих систем диагностирования РИС, основанных на знаниях, и методов и моделей их построения показал недостатки существующих систем диагностирования. На основании результатов анализа были сформулированы и обоснованы цели и задачи, проблемы и пути их решения.

2. Опираясь на существующие подходы и модели представления знаний о диагностических аспектах РИС, была разработана и обоснована структура базы знаний для систем диагностирования РИС.

3. Для уменьшения затрат времени на поддержку актуальности базы знаний диагностических ситуаций усовершенствована обобщенная модель диагностики РИС. Применение аппарата нечетких множеств позволило уменьшить затраты времени экспертов-пользователей различной квалификации, в среднем, на 57%.

4. Разработаны онтологические модели расширенного описания элементов и зависимостей в РИС, а также онтологическая модель описания диагностических тестов в формате OWL, которая позволила сократить время на поддержку актуальности базы знаний структурных и функциональных зависимостей между элементами рыночной инфраструктуры, в среднем, на 39%.

5. Разработана онтологическая модель описания знаний о штатных и нештатных ситуациях на основе многосортного языка прикладной логики для описания экспертных знаний о нечетких значениях параметров РИС в диагностируемых ситуациях, которая позволила уменьшить затраты времени на поддержку актуальности базы знаний диагностических ситуаций и их причин, в среднем на 21%.

6. Для использования в рамках разработанной модели базы знаний получил дальнейшее развитие метод диагностирования РИС, который, в отличие от существующих, позволяет решать задачу диагностики РИС с использованием знаний о нечетких значениях диагностических параметров, за счет чего, уменьшить затраты времени на проведение диагностирования в 5 -10 раз, без потери достоверности и уменьшить коэффициент простоя на 7%.

7. Разработана информационная технология построения баз знаний для систем диагностирования РИС, применение которой позволяет сократить временные затраты на диагностирование РИС без потери достоверности и уменьшить затраты времени диагноста-пользователя на поддержку актуальности базы знаний.

8. Экспериментальная оценка временных трудозатрат на диагностирование заданной РИС с применением базы знаний, построенной на основе разработанных моделей и информационной технологии, показало сокращение времени на диагностирование каждой из неисправностей, описанных в базе знаний, в 5-10 раз, в зависимости от неисправности и квалификации экспертов-пользователей, без потери достоверности, что привело к уменьшению коэффициента простоя системы в целом, на 7%, а также к уменьшению затрат времени на поддержку актуальности базы знаний на 24%.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Нестеренко, С. А. Разработка формализованного языка диагностики состояний на основе дескрипционной логики / С. А. Нестеренко, П. М. Тишин, А. С. Маковецкий, // Радіоелектронні і комп'ютерні системи – Харків, 2012 №6 (58) – С. 178–183.

2. Нестеренко, С. А. Разработка модели онтологии диагностики сервис-ориентированных сетевых структур на основе многосортного языка прикладной логики / С. А. Нестеренко, П. М. Тишин, А. С. Маковецкий, // Электротехнические и компьютерные системы – «Техника», 2012 №7 (83) – С. 102–108.

3. Нестеренко, С. А. Модель онтологии априорного подхода прогнозирования проблемных ситуаций в сложных вычислительных системах / С. А. Нестеренко, П. М. Тишин, А. С. Маковецкий, // Электротехнические и компьютерные системы – «Техника», 2013 №10 (86) – С. 111–119.

4. Тишин, П. М. Применение нечётких множеств для выявления неполадок в распределенных информационных системах / П. М. Тишин, А. С. Маковецкий, // Электротехнические и компьютерные системы – «Техника», 2015 №18 (94) – С. 7–11.

5. Тишин, П. М. Описание закономерностей при диагностике распределенных информационных систем, с применением многосортного языка прикладной логики / П. М. Тишин, А. С. Маковецкий, // Комп'ютерно-інтегровані технології: освіта, наука, виробництво – Луцьк, 2015 №19 – С. 156–162.

6. Тишин, П. М. Разработка модели онтологии диагностики распределенных информационных систем на основе многосортного языка прикладной логики / П. М. Тишин, А. С. Маковецкий, // Восточно-Европейский журнал передовых технологий – Харьков, 2015 №2 (74) – С. 21–26.

7. Маковецкий, А. С. MASDK как среда разработки мультиагентных систем на основе онтологии NDL / А. С. Маковецкий, П. М. Тишин // Труды X Всеукраинской научно-технической конференции студентов и аспирантов “Информационные системы и технологии” - Одесса, ОГАХ – 2010. – С. 72-73.

8. Нестеренко, С. А. Решение задач диагностики, основанных на нетривиальной онтологии / С. А. Нестеренко, П. М. Тишин, А. С. Маковецкий // Труды XIII международной научно-технической конференции «Современные информационные и электронные технологии» – Одесса, 2012. – С. 42.

9. Нестеренко, С. А. Разработка модели онтологии диагностики корпоративной сети на основе многосортного языка прикладной логики / С. А. Нестеренко, П. М. Тишин, А. С. Маковецкий // Труды XIII международной научно-технической конференции «Современные информационные и электронные технологии» – Одесса, 2012. – С. 85.

10. Нестеренко, С. А. Модель онтологии анализа тенденций для диагностики сложных вычислительных систем / С. А. Нестеренко, П. М. Тишин, А. С. Маковецкий // Труды XIV международной научно-технической конференции «Современные информационные и электронные технологии» – Одесса, 2013. – С. 119.

11. Тишин, П. М. Применение нечетких множеств для выявления неполадок в распределенных информационных системах / П. М. Тишин, А. С. Маковецкий, // Труды III Украинско-немецкой конференции «Інформатика. Культура. Техніка» – Одесса, 2015. – С. 7-8.

12. Тишин, П. М. Применение многосортного языка прикладной логики для решения задач диагностики в распределенных информационных системах / П. М. Тишин, А. С. Маковецкий, // Труды XXIII Международной научно-практической конференции «Информационные технологии: наука, техника, технология, образование, здоровье» – Харьков, 2015. – С. 156.

13. Hanemann, A. A Hybrid Rule-Based/Case-Based Reasoning Approach for Service Fault Diagnosis / A. Hanemann // In Proceedings of 20-th International

Conferenceon Advanced Information Networking and Application (AINA2006), includes proceedings of International Symposium on Frontiers in Networking with Applications (FINA 2006). – Vienna : 2006. – P. 734 – 738.

14. L. Lewis. Service Level Management for Enterprise Networks. Artech House, Inc., 1999.

15. H. Wietgrefe, K.-D. Tuchs, K. Jobmann, G. Carls, P. Froelich, W. Nejdl, and S. Steinfeld. Using Neural Networks for Alarm Correlation in Cellular Phone Networks. In International Workshop on Applications of Neural Networks to Telecommunications (IWANNT 1997), Melbourne, Australia, June 1997.

16. W. Hamscher, L. Console, and J. de Kleer. Readings in Model-Based Diagnosis. Morgan Kaufmann Publishers, 1992.

17. G. Jakobson and M. Weissman. Alarm Correlation. IEEE Network, 7(6), November 1993.

18. assureME Assurance Solutions (includes former OSI NETeXPERT), Agilent Technologies Corporation. http://assureme.comms.agilent.com/.

19. D. Meira. A Model for Alarm Correlation in Telecommunication Networks. PhD thesis, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil, 1997.

20. Y. Nygate. Event Correlation Using Rule and Object Based Techniques. In Proceedings of the Fourth IFIP/IEEE International Symposium on Integrated Network Management, pages 278–289, Santa Barbara, California, USA, May 1995.

21. J. Choi, M. Choi, and S. Lee. An Alarm Correlation and Fault Identification Scheme Based on OSI Managed Object Classes. In Proceedings of the IEEE International Conference on Communications (ICC 1999), pages 1547–1551, Vancouver, British Columbia, Canada, June 1999.

22. K. Appleby, G. Goldszmidt, and M. Steinder. Yemanja - A Layered Event Correlation Engine for Multi-domain Server Farms. In Proceedings of the Seventh IFIP/IEEE International Symposium on Integrated Network Management, pages 329–344, Seattle, Washington, USA, May 2001.

23. M. Agarwal, K. Appleby, M. Gupta, G. Kar, A. Neogi, and A. Sailer. Problem Determination Using Dependency Graphs and Run-Time Behavior Models. In Proceedings of the 15th IFIP International Workshop on Distributed Systems: Operations and Management (DSOM 2004), pages 171–182, Davis, California, USA, November 2004.

24. M. Sabin, R. Russell, and E. Freuder. Generating Diagnostic Tools for Network Fault Management. In Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management (IM 1997), pages 700–711, San Diego, California, USA, May 1997.

25. M. Sabin, A. Bakman, E. Freunder, and R. Russell. A Constraint-Based Approach to Fault Management for Groupware Services. In Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (IM 1999), pages 731–744, Boston, Massassuchets, USA, May 1999. IFIP/IEEE.

26. M. Sabin, R. Russell, E. Freuder, and I. Miftode. Using Constraint Technology to Diagnose Configuration Errors in Networks Managed with SPECTRUM. In Proceedings of 8th IEEE International Conference on Telecommunications (ICT 2001), Bucharest, Romania, June 2001.

27. Event Correlation in Spectrum and Other Commercial Products, Aprisma Management Technologies. http://www.aprisma.com/literature/white-papers/wp0551.pdf, September 2000.

28. Nerve Center, Open Service. http://www.openservice.com/ products/nerve-center.php.

29. IBM Tivoli Enterprise Console, International Business Machines Corporation. http://www-306.ibm.com/software/tivoli/products/enterprise-console/.

30. OpenView Event Correlation Services, Hewlett Packard Corporation. http://www.managementsoftware.hp.com/products/ecs/.

31. Simple Event Correlator (SEC), Risto Vaarandi, Tallinn Technical University, Estonia. http://kodu.neti.ee/~risto/sec/.

32. R. Vaarandi. Platform Independent Event Correlation Tool for Network Management. In Proceedings of the 8th IFIP/IEEE International Network and Operations Management Symposium (NOMS 2002), pages 907–909, Florence, Italy, April 2002.

33. Netcool, Micromuse Incorporated. http://www.micromuse.com.

34. H.-J. Kettschau, S. Brueck, and P. Schefczik. LUCAS - an Expert System for Intelligent Fault Management and Alarm Correlation. In Proceedings of the 8th IFIP/IEEE International Network and Operations Management Symposium (NOMS 2002), pages 903–906, Florence, Italy, April 2002.

35. L. Ho, D. Cavuto, M. Hasan, S. Papavassiliou, and A. Zawadzki. Adaptive Network/Service Fault Detection IFIP/IEEE International Symposium on Integrated Network Management (IM 1999), Boston, Massassuchets, USA, May 1999.

36. Q. Zheng, K. Xu, W. Lv, and S. Ma. Intelligent Search of Correlated Alarms from Database Containing Noise Data. In Proceedings of the 8th IFIP/IEEE International Network and Operations Management Symposium (NOMS 2002), pages 405–419, Florence, Italy, April 2002.

37. M. Klemittinen, H. Mannila, and H. Toivonen. Rule Discovery in Telecommunciation Alarm Data. Journal of Network and Systems Management, 7(4):395–423, Dec 1999.

38. L. Burns, J. Hellerstein, S. Ma, C. Perng, D. Rabenhorst, and D. Taylor. Towards Discovery of Event Correlation Rules. In Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM 2001), pages 345–359, Seattle, Washington, USA, May 2001.

39. M. Sloman. Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, 2(4), 1994.

40. G. Molenkamp, H. Lutfiyya, M. Katchabaw, and M. Bauer. Diagnosing Quality of Service Faults in Distributed Applications. In Proceedings of the 20th IEEE International Performance, Computing, and Communications Conference, pages 375–382, Phoenix, Arizona, USA, April 2002.

41. I. Hatzilygeroudis and J. Prentzas. Categorizing Approaches Combining RuleBased and Case-Based Reasoning (to appear). Journal of Expert Systems, Blackwell Publishing, 2007.

42. S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo. A Coding Approach to Event Correlation. IFIP/IEEE International Symposium on Integrated Network Management, pages 266–277, Santa Barbara, California, USA, May 1995.

43. S. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. High Speed and Robust Event Correlation. IEEE Communiations Magazine, 34(5), May 1996.

44. I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik. Real-time Problem Determination in Distributed Systems Using Active Probing. In Proceedings of the 9th IFIP/IEEE International Network Management and Operations Symposium (NOMS 2004), pages 133–146, Seoul, Korea, April 2004.

45. A. Beygelzimer, M. Brodie, S. Ma, and I. Rish. Test-Based Diagnosis: Tree and Matrix Representations. In Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005), pages 529–542, Nice, France, May 2005.

46. EMC Smarts Family. http://www.emc.com/products/software/smarts/-smarts_family/index.jsp.

47. J. Kolodner. Case-Based Reasoning. Morgan Kaufmann Publishers, San Mateo, California, USA, 1993.

48. A. Aamodt and E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AICom - Artificial Intelligence Communications, IOS Press, 7(1):39–59, 1994.

49. L. Lewis. A Case-based Reasoning Approach for the Resolution of Faults in Communication Networks. In Proceedings IFIP/IEEE International Symposium on Integrated Network Management, San Francisco, California, USA, April 1993.

50. L. Lewis. Managing Computer Networks - A Case-Based Reasoning Approach. Artech House, Inc., 1995.

51. G. Jakobson, L. Lewis, and J. Buford. An Approach to Integrated Cognitive Fusion. In Proceedings of the 7th ISIF International Conference on Information Fusion (FUSION2004), pages 1210–1217, Stockholm, Sweden, June 2004.

52. SpectroRx, Aprisma Management Technologies (part of Computer Associates). http://www.aprisma.com.

53. A. Weiner, D. Thurman, and C. Mitchell. Applying Case-Based Reasoning to Aid Fault Management in Supervisory Control. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pages 4213–4218, Vancouver, British Columbia, Canada, October 1995.

54. G. Penido, J.M. Nogueira, and C. Machado. An Automatic Fault Diagnosis and Correlation System for Telecommunications Management. In Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (IM 1999), pages 779–791, Boston, Massassuchets, USA, May 1999.

55. M. Steinder and A. Sethi. The Present and Future of Event Correlation: A Need for End-to-End Service Fault Localization. In Proceedings of the 5th IIIS World Multiconference on Systemics, Cybernetics, and Informatics (SCI 2001), pages 124–129, Orlando, Florida, USA, July 2001.

56. V. Danciu, N. gentschen Felde, and M. Sailer. Declarative Specification of Service Management Attributes. In Proceedings of the 10th IFIP/IEEE International Conference on Integrated Network Management (IM 2007), Munich, Germany, May 2007.

57. J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2). Technical Report RFC 1902, IETF, January 1996.

58. R. Kasseri and B. Stewart. Event MIB. Technical Report RFC 2981, IETF, October 2000.

59. H. Hazewinkel, C. Kalbfleisch, and J. Schoenwaelder. Definition of Managed Objects for WWW Services. Technical Report RFC 2594, IETF, May 1999.

60. CIM Standards, Version 2.13, Distributed Management Task Force. http://www.dmtf.org/standards/cim, September 2006.

61. E. Heidinger. Ein CIM-basiertes Modell der Rechnernetzepraktikum Infrastruktur - in German. Advanced Practical, Technical University of Munich, Department of Computer Science, Munich, Germany, January 2004.

62. Common Information Model (CIM) Infrastructure. Version 2.6.0. Specification, DMTF Standard. – Distributed Management Task Force˜ 2010.

63. Web-Based Enterprise Management (WBEM):Technical report [Text]. – Distributed Management Task Force˜ 2003.

64. Information Technology - Open Systems Interconnection - Structure of Management Information - Part 7: General Relationship Model, IS 10165-7, ISO and International Electrotechnical Committee, 1997.

65. M. Agarwal, K. Appleby, M. Gupta, G. Kar, A. Neogi, and A. Sailer. Problem Determination Using Dependency Graphs and Run-Time Behavior Models. In Proceedings of the 15th IFIP International Workshop on Distributed Systems: Operations and Management (DSOM 2004), pages 171–182, Davis, California, USA, November 2004.

66. S. Shankar and O. Satyanarayanan. An Automated System fo Analyzing Impact of Faults in IP Telephony Networks. In Proceedings of the 10th IFIP/IEEE International Network Management and Operations Symposium (NOMS 2006), Vancouver, British Columbia, Canada, April 2006.

67. B. Gruschke. Integrated Event Management: Event Correlation Using Dependency Graphs. In Proceedings of the 9th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 98), pages 130–141, Newark, Delaware, USA, October 1998.

68. B. Gruschke. Entwurf eines Eventkorrelators mit Abh ¨angigkeitsgraphen - in German. Phd thesis, University of Munich, Department of Computer Science, Munich, Germany, November 1999.

69. A. Keller and G. Kar. Determining Service Dependencies in Distributed Systems. In Proceedings of the IEEE International Conference on Communications (ICC 2001), Helsinki, Finland, June 2001.

70. C. Ensel and A. Keller. An Approach for Managing Service Dependencies with XML and the Resource Description Framework. Journal of Network and Systems Management, 10(2), June 2002.

71. D. Caswell and S. Ramanathan. Using Service Models for Management of Internet Services. Technical Report HPL-1999-43, HP Laboratories, March 1999.

72. S. Ramanathan, D. Caswell, and S. Neal. Auto-Discovery Capabilities for Service Management: An ISP Case Study. Technical Report HPL-1999-68, HP Laboratories, Palo Alto, California, USA, May 1999.

73. S. Bagchi, G. Kar, and J. Hellerstein. Dependency Analysis in Distributed Systems using Fault Injections: Application to Problem Determination in an E-Commerce Environment. In Proceedings of the 12th International IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSOM 2001), Nancy, France, October 2001.

74. P. Marcu. Modellierung von Abh¨angigkeiten in IT-Diensten - in German. Diploma thesis, University of Munich, Department of Computer Science, Munich, Germany, June 2006.

75. A. Hanemann, P. Marcu, M. Sailer, and D. Schmitz. Specification of Dependencies for IT Service Fault Management. In Proceedings of the 1st International Conference on Software and Data Technologies, pages 257–260, Setubal, Portugal, September 2006. INSTICC press.

76. M. Steinder and A. Sethi. A Survey of Fault Localization Techniques in Computer Networks. Science of Computer Programming, Elsevier, 53(1):165–194, 2004.

77. A. Bierman and K. Jones. Physical Topology MIB. Technical Report RFC 2922, IETF Network Working Group, 2000.

78. G. Kar, A. Keller, and S. Calo. Managing Application Services over Service Provider Networks: Architecture and Dependency Analysis. In Proceedings of the 7th IFIP/IEEE International Network and Operations Management Symposium (NOMS 2000), pages 61–75, Honolulu, Hawaii, USA, April 2000.

79. A. Bansal and A. Clemm. Auto-Discovery at the Network and Service Management Layer. In Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management (IM 2003), pages 365–378, Colorado Springs, Colorado, USA, March 2003.

80. Systems Management: Application Response Measurement (ARM). Technical report, OpenGroup, July 1998.

81. M. Katchabaw, S. Howard, H. Lufiyya, A. Marshall, and M. Bauer. Making Distributed Applications Manageable through Instrumentation. The Journal of Systems and Software, 45(2):81–97, 1999.

82. A. Brown, G. Kar, and A. Keller. An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Application Environment. In Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM 2001), pages 377–390, Seattle, Washington, USA, May 2001.

83. M. Gupta, A. Neogi, M. Agarwal, and G. Kar. Discovering Dynamic Dependencies in Enterprise Environments for Problem Determination. In Proceedings of the 14th IFIP/IEEE Workshop on Distributed Sytems: Operations and Management, Heidelberg, Germany, October 2003.

84. M. Agarwal, M. Gupta, G. Kar, A. Neogi, and A. Sailer. Mining Activity Data for Dynamic Dependency Discovery in E-Business Systems. eTransactions on Network and Service Management (eTNSM), September 2004.

85. M. Aguilera, J. Mogul, J. Wiener, P. Reynolds, and A. Mutchitacharoen. Performance Debugging for Distributed Systems of Black Boxes. In Proceedings of the 19th ACM Symposium on Operating Systems Principles, pages 329–344, Bolton Landing, New York, USA, October 2003.

86. M. Agarwal, K. Appleby, M. Gupta, G. Kar, A. Neogi, and A. Sailer. Problem Determination Using Dependency Graphs and Run-Time Behavior Models. Technical Report RI04004, IBM Research Report, 2004.

87. C. Ensel. New Approach for Automated Generation of Service Dependency Models. In Network Management as a Strategy for Evolution and Development; Second IEEE Latin American Network Operation and Management Symposium (LANOMS 2001), Belo Horizonte, Brazil, August 2001.

88. C. Ensel. A Scalable Approach to Automated Service Dependency Modeling in Heterogeous Environments. In Proceedings of the 5th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2001), Seattle, Washington, USA, September 2001.

89. K.-D. Tuchs and K. Jobmann. Intelligent Search for Correlated Alarm Events in Databases. In Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM 2001), pages 285–288, Seattle, Washington, USA, May 2001.

90. L. Burns, J. Hellerstein, S. Ma, C. Perng, D. Rabenhorst, and D. Taylor. Towards Discovery of Event Correlation Rules. In Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM 2001), pages 345–359, Seattle, Washington, USA, May 2001.

91. J. Hellerstein, S. Ma, and C. Perng. Discovering Actionable Patterns from Event Data. IBM Systems Journal, 41(3):475–493, 2002.

92. OGC (Office of Government Commerce), editor. Service Support. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2000.

93. The NGOSS Technology-Neutral Architecture. NGOSS Release 4.5, TeleManagement Forum, November 2004.

94. Shared Information/Data (SID) Model, Addendum 4S0 - Service Overview Business Entity Definitions. NGOSS Release 4.0, TeleManagement Forum, August 2004.

95. M. Brenner, M. Garschhammer, M. Sailer, and T. Schaaf. CMDB - Yet another MIB? On Reusing Management Model Concepts in ITIL Configuration Management. In Proceedings of the 17th International IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSOM 2006), pages 269–280, Dublin, Ireland, October 2006.

96. M. Sailer. Towards a Service Management Information Base. In Proceedings of the IBM PhD Student Symposium at the 3rd International Conference on Service-Oriented Computing (ICSOC 2005); IBM Research Report No. 23826, Amsterdam, The Netherlands, December 2005.

97. V. Danciu, A. Hanemann, H.-G. Hegering, and M. Sailer. IT Service Management: Getting the View. In Managing Development and Application of Digital Technologies, pages 109–130, Munich, Germany, June 2006. CDTM, Springer Verlag.

98. Щеглов С. Н. Онтологический подход и его использование в системах представления знаний / С. Н. Щеглов – Таганрог: Известия ЮФУ. Технические науки, 2009. – С. 146-153.

99. W3C, OWL Web Ontology Language Reference. W3C Recommendation / M. Dean, G. Schreiber (Eds.). – 2004.

100. Клещев А. С. Необогащенные системы логических соотношений / А. С. Клещев, И. Л. Артемьева // Научно–техническая информация. Серия 2. – 2000. – Ч. 1.– No 7.– С. 18–28. – Ч. 2. – No 8. – С. 8–18.

101. Kleshchev A. S. A mathematical appara-tus for ontology simulation. Specialized extensions of the extendable language of applied logic / A. S. Kleshchev, I. L. Artemjeva // Inf. Theories and Appl., 2005, vol. 12, No 3. P. 265-271.

102. Артемьева И. Л. Логические модели второго порядка для предметных областей / И. Л. Артемьева, Т. Л., Гаврилова, А. С Клещев // Научно–технич. информация. Серия 2. – 1997.– No 6. – С. 14–30.

103. Клещёв А. С., Москаленко Ф. М., Черняховская М. Ю. Модель онтологии предметной области «Медицинская диагностика». Часть 2.

Формальное описание причинно - следственных связей, причин значений признаков ипричин заболеваний. / А. С. Клещёв, Ф. М. Москаленко, М. Ю. Черняховская // НТИ. Серия 2. – 2006. – No 2. – С.19-30.

104. Соколов С. А. Диагностическая модель для задач выявления закономерностей функционирования компонентов корпоративной IP-сети / С. А. Соколов, А. Л. Стокипный // Информационные технологии. – 2009. – No5/2 (41) .– С. 4–9.

105. EMC Smarts Family. http://www.emc.com/products/software/smarts

106. Борисов В. В. Нечеткие модели и сети / В. В. Борисов, В. В. Круглов, В. В. Федулом – М.: Горячая линия – Телеком, 2007. – С. 284.

107. Рыжов А. П. Элементы теории нечетких множеств и ее приложений / А. П .Рыжов. – М.: 2003. –81 с.

108. Enabling Automatic Reasoning on CIM-based Management Platforms / F. Alonso, R. Fernandex, S. Frutos, J. Soriano // World Academy of Science, Engineering and Technology 14˜ 2006. – P. 123 – 129.

109. Schmidt-Schauss, M. Attributive concept descriptions with complements / M. SchmidtSchauss, G. Smolka // Artificial Intelligence, 48:1–26.˜ 1991. – P. 222 – 228.

110. IBM Tivoli Enterprise Console, International Business Machines Corporation / Available at: http://www-306.ibm.com/software/tivoli/products/ enterprise-console/

111. HP OpenView Event Correlation Services, Hewlett Packard Corporation / Available at: http://www.managementsoftware.hp.com/products/ecs/

112. Appleby, K. Yemanja – A Layered Event Correlation Engine for Multi-domain Server Farms. In Proceedings of the Seventh IFIP/IEEE / K. Appleby, G. Goldszmidt, M. Steinder // International Symposium on Integrated Network Management, Seattle, Washington, USA, 2001. – P. 329–344. doi: 10.1109/inm.2001.918051

ПРИЛОЖЕНИЕ А

Описания введенных понятий и терминов онтологической модели штатных и нештатных ситуаций

Термины знаний и действительности, описывающие штатные ситуации, и онтологические соглашения о соответствии между ними

(35) Термин *знания о штатных ситуациях (ЗШС)* это множество структурных значений с атрибутами следствие, варианты, факторы, каждое их которых описывает знания о конкретной штатной ситуации. Значением первого атрибута является имя параметра, второго – множество вариантов штатных ситуаций для этого параметра, третьего – множество особенностей.

ЗШС ≡ (следствие → *собст_параметры_службы*,

варианты → {} *варианты штатных ситуаций*,

факторы → {}*особенности*)

(36) Термин знаний *варианты штатных ситуаций* это множество структурных значений с атрибутами область значений следствия, условия на факторы, каждое из которых описывает знания о конкретном варианте штатной ситуации. Значением первого атрибута является множество значений параметра в этом варианте, а второго – условие.

варианты штатных ситуаций ≡≡≡ (

область значений следствия → *множества значений*,

условие на факторы → *условия*)

(37) В знаниях о штатных ситуациях в любом варианте нормы область значений следствия – это собственное подмножество возможных значений параметра, являющегося следствием этой штатной ситуации.

(x: *ЗШС*)

(v: варианты(x))

область значений следствия(v) ⊂ возможные значения(следствие(x))

(38) Термин действительности *штатные ситуации* (*ШС*)это множество структурных значений с атрибутами следствие и вариант, каждое из которых описывает наблюдавшую штатную ситуацию. Значением первого атрибута является имя параметра, а второго – *варианты штатных ситуаций*.

ШС ≡ (следствие → *собст_параметры_службы*,

вариант → *варианты штатных ситуаций*)

(39) Если присутствует некоторая штатная ситуация, то в множестве *знания о штатных ситуациях* присутствует такой элемент, у которого следствие совпадает со следствием штатной ситуации, в множестве вариантов *штатных ситуаций* этого элемента содержится вариант *штатных ситуаций* из штатной ситуации и для него выполнено условие на факторы.

(x: *ШС*)(∨ (z: *ЗШС*) следствие(z) = следствие(x)&& вариант(x) ∈ варианты(z)&& выполнено(условие на факторы(вариант(x))))

Термины знаний и действительности, описывающие нештатные ситуации, и онтологические соглашения о соответствии между ними

(40) Термин знания о нештатных ситуациях (*ЗНШС*) это множество структурных значений с атрибутами причина, период развития, следствие, варианты, факторы, необходимое условие(*ну*), модальность, каждое из которых описывает знания о конкретной нештатной ситуации. Значением причины является имя причины отклонения, значением периода развития – номер периода развития причины, значением следствия – имя параметра, значением вариантов – множество вариантов ншс, значением факторов множество особенностей, значением необходимого условия – множество условий, значением модальности – необходимость или возможность.

ЗНШС ≡ (причина → *причины отклонения*,

период развития → I[1, чпр(причина)],

следствие → *собст_параметры_службы*,

варианты → {}*варианты ншс*,

факторы → {}*особенности,*

ну → условия,

модальность → *приоритеты*)

(41) Термин *варианты ншс* это множество структурных значений с атрибутами число периодов динамики(*чпд*), описание динамики, условие на факторы, каждое из которых описывает знания о конкретном варианте нештатной ситуации. Значением первого атрибута является положительное целое число. Второй атрибут является функцией, которая номеру периода динамики сопоставляет период динамики. Значение третьего атрибута – множество условий.

варианты ншс ≡ (чпд → I[1, ∞),

описание динамики → (I[1, чпд] → периоды динамики),

условие на факторы → условия)

(42) Термин действительности *нештатные ситуации* (*НШС*)это множество структурных значений с атрибутами причина, период развития, следствие, вариант, динамика значений, модальность, каждое из которых описывает некоторую нештатнную ситуацию. Значением причины является имя причины отклонения, значением периода развития - номер периода развития, значением следствия – имя параметра, значением варианта – *вариант ншс*, значением динамики значений – разбиение, значением модальности – необходимость или возможность.

НШС ≡ (причина → *причины отклонения*,

период развития → I[1, чпр(причина)],

следствие → *собст_параметры_службы*,

вариант → *варианты ншс*,

динамика значений → *разбиения*,

модальность → {возможность, необходимость})

(43) Если присутствует нештатная ситуация, на некотором периоде ее развития, то начало его динамики значений совпадает с моментом начала периода развития, а конец – с концом этого периода.

(x: *НШС*) el(динамика значений(x), 0) =

= el(развитие(причина(x)),период развития(x) – 1) &

& el(динамика значений(x),length(динамика значений(x))) =

= el(развитие(причина(x)),период развития(x))

(44) Если в ситуации присутствует нештатная ситуация, на некотором периоде ее развития, то в модель знаний входит элемент множества знания о нештатных ситуациях, у которого:- причиной является эта причина отклонения с тем же периодом развития,- следствие совпадает со следствием нештатной ситуации,- выполнено необходимое условие,- модальность совпадает с модальностью нештатной ситуации,- множеству вариантов ншс принадлежит вариант ншс нештатной ситуации, причём такой, что количество интервалов в динамике значений нештатной ситуации равно чпд этого варианта и выполнено условие на факторы.

(x: *НШС*)(∨(z: *ЗНШС*)

причина(z) = причина(x) &

& период развития(z) = период развития(x) &

& следствие(z) = следствие(x) &

& выполнено(необходимое условие(z))&

& модальность(z) == модальность(x) &

& вариант(x) ∈ варианты(z) &

& length(динамика значений(x)) – 1 = чпд(вариант(x))&

& выполнено(условие на факторы(вариант(x))))

Общие термины и онтологические соглашения, используемые для описания причинно-следственных связей

(45) Особенность, входящая в условие на факторы некоторого варианта причинно-следственной связи, является фактором этого отношения.

(x: *ЗШС* ∪*ЗНШС*) (вариант: варианты(x)) (УВФ: условие на факторы(вариант)) особенность(УВФ) ∈ факторы(x)

(46) У каждого элемента множества *знания о нештатных ситуациях*, в любом его варианте область значений следствия в каждом периоде динамики является подмножеством возможных значений параметра, являющегося следствием этого элемента знаний.

(x: *ЗНШС*) (вариант: варианты(x)) (n: I[1, чпд(вариант)])

(период динамики: описание динамики(вариант)(n))

область значений следствия(период динамики) ⊆

⊆ возможные значения(следствие(x))

(47) Вспомогательный термин причинно-следственные связи (*ПСС*) есть множество значений терминов *ШС* и *НШС*.

$ПСС \equiv ШС \cup НШС$

(48) Если присутствует нештатная ситуация то длительность каждого периода динамики из динамика значений этого причинно-следственного отношения принадлежит интервалу допустимых длительностей этого периода динамики для варианта этого причинно-следственного отношения.

(x: *НШС*) (n: I[1, *чпд*(вариант(x))])

element(динамика значений(x), n) – element(динамика значений(x),n –1) ∈

∈ I[нижняя граница(длительность(описание динамики(вариант(x))(n))), верхняя граница(длительность(описание динамики(вариант(x))(n)))]

Термины и соглашения, описывающие интервалы развития параметра и причины отклонений

В настоящем разделе вводятся термины, описывающие причины значений параметров на интервалах их развития, а также термины, описывающие диагноз.

Термины и соглашения, описывающие интервалы развития параметров

В этом разделе приводятся соотношения, задающие причинно-следственные связи, действующие на некотором интервале разбиения оси времени, связанном с параметром, которые определяют значения этого параметра, а также условия, налагаемые на границы разбиения оси времени, связанного с параметром.

Описание причин значений параметра на интервале его развития и их свойств

(49) Термин *возможные причины значений параметра* обозначает функцию, аргументом которой является интервал развития параметра, а значением - множество всех таких причинно-следственных связей (штатных и нештатных ситуаций), присутствующих в ситуациях, которые протекают на протяжении всего интервала развития параметра и, таким образом, могут определять значения этого параметра на этом интервале.

возможные причины значений параметра ≡≡

(λ (x: *U*){(y: *ШС*) следствие(y)= параметр(x)} ∪ {(z: *НШС*)

следствие(z) = параметр(x) &

& interval(развитие(параметр(x)),номер интервала(x)) ⊆

⊆ I[el(динамика значений(вариант(z)),0), el(динамика значений(вариант(z)), length(динамика значений(z)))] &

& interval(развитие(параметр(x)),номер интервала(x)) ⊆

⊆ I[el(развитие(причина(z)), 0), el(развитие(причина(z)), length(развитие(причина(z)))]})

(50) Термин *приоритет причин значений параметра* обозначает двуместный предикат, аргументами которого являются возможные причины,

которые могут определять значения параметров на интервалах их развитий, и который истинен тогда и только тогда, когда обе причины относятся к одному и тому же параметру и интервалу его развития и первая причина «приоритетнее» второй, то есть:

а) либо причинная связь, соответствующая значению первого аргумента, есть нештатная ситуация, входящая в диагноз, с приоритетом более высоким чем причинная связь, соответствующая значению второго аргумента;

б) либо причинная связь, соответствующая значению первого аргумента, есть нештатная ситуация, входящая в диагноз с приоритетом соответствующим модальности необходимость, а причинная связь, соответствующая значению второго аргумента, есть:- штатная ситуация.

приоритет причин значений параметра ≡

≡ (λ(П1: *ПСС*)(П2: *ПСС*) (∨ (u: *U*) П1 ∈

∈ *возможные причины значений параметра* (u) &

& П2 ∈ *возможные причины значений параметра* (u) &

& (П1 ∈ *НШС* & модальность(П1)=необходимость &

& (П2 ∈ *ШС* ∨ П2 ∈ *НШС* & модальность(П2) = возможность))))

(51) Термин *причины значений параметра наивысшего приоритета* обозначает функцию, аргументом которой является интервал развития параметра, а значением – только те причинно-следственные связи из множества возможные причины значений *параметра*, которые имеют наибольший приоритет.

причины значений параметра наивысшего приоритета ≡

≡ (λ (u: *U*) *возможные причины значений параметра*(u) \

\ {(П1: *возможные причины значений параметра*(u))

(∨ (П2: *возможные причины значений параметра*(u))

приоритет причин значений параметра(П2, П1))})

(52) Термин *причина значений параметра(PU)* это функция, которая *интервалу развития параметра* сопоставляет причинно-следственную связь, присутствующую в ситуации, и определяющую значения этого *параметра* на этом интервале развития *параметра*.

сорт PU: $U \rightarrow$ *ПСС*

(53) *Причина значений параметра* на интервале его развития имеет наивысший приоритет.

(u: *U*) *PU*(u) $\in$ *причины значений параметра наивысшего приоритета*(u)

(54) Если причиной, определяющей значения параметра на некотором интервале его развития, является штатная ситуация, то значения этого параметра в любой момент его наблюдения, принадлежащего этому интервалу, принадлежат области значений следствия у варианта этой штатной ситуации.

(u: *U*)(t: моменты(параметр(u)) $\cap$ interval(развитие(параметр(u)), номер интервала(u))) *PU*(u) $\in$ *ШС* $\Rightarrow$

$\Rightarrow$ параметр(u)(t) $\in$ область значений следствия(вариант(*PU*(u)))

(55) Если причиной, определяющей значения параметра на некотором интервале его развития, является причинно-следственная связь *НШС*, то значения этого параметра в любой момент его наблюдения, принадлежащего рассматриваемому интервалу развития параметра и некоторому периоду развития в действующем варианте этой причинно-следственной связи, принадлежат области значений следствия для этого периода динамики этого варианта рассматриваемого причинно-следственного отношения.

(u: {(uu: *U*) PU(uu) $\in$ *НШС*})(p: I[1, чпд(вариант(PU (u)))])

(t: моменты(параметр(u))$\cap$

$\cap$ interval(развитие(параметр(u)), номер интервала(u)) $\cap$

$\cap$ interval(динамика значений(PU(u)), t))

параметр(u)(t) $\in\in$ область значений следствия(

описание динамики(вариант(PU (u)))(p))

(56) Термин одна и та же причинная связь (*одна ПСС*) это предикат, аргументами которого являются две причинно-следственных связи, и который истинен тогда и только тогда, когда их следствия совпадают и при этом эти причинно-следственные связи:

- либо являются *ШС*;

- либо являются *НШС* одной и той же причины на одном и том же периоде развития;

одна ПСС ≡ (λ (П1: *ПСС*)(П2: *ПСС*)

((П1 ∈ *ШС* & П2 ∈ *ШС*)∨

∨ (П1 ∈ *НШС* & П2 ∈ *НШС* & причина(П1) = причина(П2) &

& период развития(П1) = период развития(П2))) & следствие(П1) = следствие(П2))

(57) Если в ситуации для одного и того же параметра его значения на двух интервалах его развития определяются одной и той же причинной связью, то они определяются одним и тем же вариантом этой причинной связи.

(u1: *U*)(u2: *U*) параметр(u1) = параметр(u2) &

& *одна ПСС*(*PU*(u1), *PU*(u2)) ⇒

⇒ вариант(*PU*(u1)) = вариант(*PU*(u2))

(58) Термин *соседние интервалы развития параметра* (*SU*) это предикат, аргументами которого являются два интервала развития параметра, и который истинен тогда и только тогда, когда у параметра не менее двух интервалов развития и первый указанный интервал предшествует второму.

SU≡ (λ (u1: *U*)(u2: *U*) параметр (u1) = параметр (u2) &

& интервал(u2) – интервал(u1) = 1)

(59) Причины, определяющие значения параметра на двух соседних интервалах его развития, различны.

(u1: *U*)(u2: *U*) *SU*(u1, u2) ⇒ *PU* (u1) ≠ *PU* (u2)

(60) Если причины, определяющие значения параметра на двух соседних интервалах его развития, не являются *ШС*, то граница между этими интервалами является либо концом динамики значений причинно-следственной связи на первом интервале, либо началом динамики значений причинно-следственной связи на втором интервале.

(u1: *U*)(u2: *U*) *SU*(u1,u2) & *PU* (u1) ∉ *ШС* & *PU* (u2) ∉ *ШС*) ⇒

⇒ el(развитие(параметр(u1)), номер интервала(u1))∈

∈ {el(динамика значений(*PU*(u1)), length(динамика значений(*PU*(u1)))), el(динамика значений(*PU*(u2)),0)}

(61) Если причиной, определяющей значения параметра на некотором интервале его развития, является *ШС* , то граница между этим и следующим интервалами является началом динамики значений причинно-следственной связи, являющейся причиной значений параметра на втором интервале.

(u1: *U*)(u2: *U*) *SU*(u1,u2) & *PU* (u1) ∈ *ШС* ⇒

⇒ el(развитие(параметр (u1)), номер интервала(u1))=

= el(динамика значений(*PU*(u2)), 0)

(62) Если причина, определяющая значения параметра на некотором интервале его развития, является *ШС*, то граница между этим и предшествующим интервалами является концом динамики значений причинно-следственной связи, являющейся причиной значений параметра на первом интервале.

(u1: *U*)(u2: *U*) *SU*(u1, u2) & *PU*(u2) ∈ *ШС*) ⇒

⇒ el(развитие(параметр(u1)), номер интервала(u1)) =

= el(динамика значений(*PU*(u1),length(динамика значений(*PU*(u1)))

ПРИЛОЖЕНИЕ Б

Акт внедрений результатов диссертационной работы

ПРИЛОЖЕНИЕ В

Справка о внедрении результатов научных исследований в учебный процесс