Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное учреждение науки
Институт автоматики и процессов управления
Дальневосточного отделения Российской академии наук

На правах рукописи

Шалфеева Елена Арефьевна

Методы, модели и технология обеспечения жизнеспособности интеллектуальных систем с декларативными базами знаний

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

Диссертация на соискание ученой степени
доктора технических наук

Научный консультант:
Грибова Валерия Викторовна,
доктор технических наук,
старший научный сотрудник

Владивосток − 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одним из применяемых на практике направлений искусственного интеллекта являются *системы с базами знаний* (экспертные системы) для поддержки решения задач, требующих анализа обширных объемов знаний и трудоемкого сопоставления их имеющимся данным. Для поддержки решения таких задач за несколько десятилетий были предложены универсальные технологии, и многочисленные *системы с базами знаний* продолжают создаваться. Универсальными средствами являлись специализированные языки программирования, такие как PROLOG, LISP, SMALLTALK, FRL, Interlisp и др., универсальные инструментальные системы и «оболочки» для БЗ: Level5, Object, G2, Clips, Loops, VITAL, KEATS, и др. и более новые: OSTIS, АТ-ТЕХНОЛОГИЯ и др. Существующие различные инструментальные системы различаются наборами предлагаемых разработчику компонентов, уровнем поддержки этапов жизненного цикла создания и сопровождения систем с базами знаний, поддержке различных технологий разработки, используемыми формализмами представления знаний, методами их формирования и отладки, используемым механизмам вывода и пр.

Актуальность темы диссертации и степень разработанности темы. На сегодняшний день актуальным для коллективов разработчиков программных систем является не только создание системы, отвечающей всем текущим требованиям к ней, но и реализация механизмов, обеспечивающих ее неизбежное последующее сопровождение, вызываемое изменениями знаний предметной области, условий эксплуатации или требований пользователей к функциональности и пользовательскому интерфейсу. Процесс сопровождения занимает значительную долю трудозатрат в процессе жизненного цикла программной системы: по оценкам специалистов от 50 до 80%. Поэтому необходимо проектировать программные системы (ПС), закладывая механизмы для упрощения этого процесса.

Известными механизмами, используемыми в программной инженерии для упрощения сопровождаемости ПС, являются: архитектурные решения (разделение на слабо сцепленные компоненты с логически понятными и функциями), декларативное представление компонентов ПС, средства автоматизации построения проектных моделей и автоматической генерации по ним фрагментов кода, средства связывания компонентов

друг с другом, использование типовых архитектурных решений, стандартизация сервисов и интерфейсов (model-driven архитектуры и модельно-управляемый подход к созданию программных систем), разделение компетенций между разными типами разработчиков.

Программная система, которая способна адаптироваться во времени (под воздействием человека или инструментов) и при этом быть адаптивной, называется «жизнеспособной» [119,144,146].

При автоматизации области с интеллектуальной деятельностью со сложными и ответственными задачами принятия разных типов решений, когда ожидается выработка рекомендаций на основе формализованных знаний [147,148], создают программные системы с дополнительным архитектурным компонентом – базой знаний (БЗ). Команда разработчиков при этом пополняется экспертами предметной области (ПрОбл) и когнитологами. Указанная специфика накладывает необходимость использования дополнительных специализированных механизмов обеспечения их жизнеспособности, поскольку требуется управление «жизненным циклом широкого спектра моделей», включая получение и представление знаний [126,147].

Существенный вклад в методы, технологии, инфраструктуру построения систем с базами знаний (СБЗ) внесли российские и зарубежные ученые и основанные ими школы: Артемьева И.Л.; Гаврилова Т. А., Голенков В.В., Грибова В.В., Еремеев А.П., Загорулько Ю.А., Клещев А.С., Кузнецов О.П., Осипов Г.С., Рыбина Г.В., Соснин П.И., Стефанюк В.Л., Финн В.К., Ярушкина Н.Г., Дж. Джарратано, Г. Райли; П. Джексон; Лавилла С.; Musen M., Mortensen J., Noy N.; F. Hayes-Roth; B. Moore; Calero C.; Taboada M.; Emmanuel C. Ogu, Adekunle, Y.; R. M.Sonar, R. Fikes and D. McGuinness; Müller L., Perry J. и многие другие.

Предлагаемые ими методы и технологии позволяют создавать базы знаний и системы, основанные на знаниях, называемые интеллектуальными системами (ИС), тестировать их и вводить в эксплуатацию. Подход с универсальным представлением знаний в виде правил, обрабатываемых единой «машиной вывода», пригоден для областей, где число правил измеряется сотнями. Далее их создание и развитие особенно силами экспертов становится практически невозможным, что подтверждают многочисленные публикации Использование онтологий позволило абстрагировать некоторую «описатель-

ную» часть знаний (чаще как классы сущностей) и начать применять дедуктивный вывод, «используя в посылках и заключениях продукционных правил введенные в онтологиях понятия и, т.е. рассуждать на основе онтологии («ontological reasoning»). Новые среды, сочетающие в себе онтологии и логическое программирование (например, «ontological logic programming»), более эффективны для реализации, но в отношении последующего сопровождения сделали лишь небольшой шаг - декларативное представление некоторой части знаний – той, где для сущностей можно определить свойств, ввести бинарные связи и выявить иерерхии классов таких сущностей [41]. Другой опыт в отношении создания основы для сопровождения СБЗ – через библиотеки повторноиспользуемых sc-компонентов для графодинамических ассоциативных моделей – не стал эффективным, поскольку размещаемые туда семантические компоненты привязаны к частным задачам и не помогают при автоматизации новых задач и деятельностей.

В этих технологиях не уделено достаточного внимания методам и инструментам развития компонентов ИС и особенно БЗ в процессе их эксплуатации. Используемые средства не обеспечивают независимую работу экспертов ПрОбл, когнитологов, программистов с автоматическим согласованием допустимых действий.

Собственный многолетний опыт разработки ИС в различных ПрОбл и множественные публикации показали, что объем знаний не позволяет формализовать базы знаний полностью за приемлемое время силами даже согласованного коллектива экспертов. Кроме того, знания динамичны: они расширяются и корректируются.

Для обеспечения возможности перманентного усовершенствования БЗ вышеперечисленным разработкам не хватает технологически обеспеченного отделения декларативных знаний от процедурных, независимости процессов разработки баз знаний и решателей, механизмов создания, модифицирования и оценивания БЗ, ориентированных на экспертов в течение всего процесса использования ИС без участия программистов и инженеров по знаниям.

Таким образом, несмотря на современные достижения в программной и инженерии и инженерии знаний обострилась необходимость разработки методов, моделей и технологии для обеспечения жизнеспособности интеллектуальных систем, которые могут продолжительное время быть полезными специалистам, решающим сложные и ответственные задачи.

Целью диссертационной работы является разработка моделей, методов и технологии создания интеллектуальных систем (на основе онтологий) с декларативным представлением баз знаний и механизмами эволюционирования.

Для достижения этой цели важно:

- установить спектр задач интеллектуальной деятельности, решаемых на основе формализуемых знаний, формальные характеристики (этих задач), используемые модели знаний, способы их обработки;

- предложить систематический подход к решению задач,

- установить принципиально важный набор компонентов инфраструктуры для построения СБЗ и непрерывной адаптации к развитию ПрОбл.

Поэтому в работе были **поставлены следующие задачи.**

1. Классификация и спецификация задач интеллектуальной деятельности и анализ методов их решения.

2. Разработка модели жизнеспособной системы, основанной на декларативных знаниях.

3. Развитие онтологического подхода к формированию баз декларативных знаний для классов интеллектуальных задач для повышения повторной используемости готовых решений.

4. Разработка методов оценивания и повышения качества баз знаний, семантических описаний входных и выходных данных и их онтологий.

5. Разработка метода декларирования и конструирования онтологических решателей с повторной используемостью готовых решений.

6. Разработка облачной технологии и реализация инструментальных средств (программный комплекс или инструментальная среда развития) поддержки разработки и сопровождения всех компонентов СБЗ.

Объект исследования – процесс разработки систем для поддержки решения на основе баз знаний задач в ответственных народнохозяйственных отраслях и обеспечения их жизнеспособности в условиях непрерывно развивающихся знаний.

Предмет исследования − методы, технология и инфраструктура построения и усовершенствования систем с базами знаний.

Методология и методы исследования. Поставленные задачи решаются с использованием системного анализа, теории информации, теории графов и семантических сетей, теории множеств, технологии объектно-ориентированного программирования, облачных технологий, методологии онтологического инжиниринга и других методов искусственного интеллекта (ИИ).

Научная новизна

В диссертационной работе получены следующие научные результаты: впервые разработаны *методы, модели и технология* обеспечения жизнеспособности *систем с базами знаний,* включающие:

1. новую иерархию постановок задач интеллектуальной деятельности в терминах единых математических абстракций для всех содержательных понятий, открывающую повторное использование готовых решений;
2. модель жизнеспособной системы для поддержки решения задач интеллектуальной деятельности в рамках разработанных постановок;
3. метод непрерывного развития баз знаний экспертами предметной области (без участия инженеров и программистов) на основе потока прецедентов с оцениванием корректности внесенных изменений;
4. новый метод комплексного оценивания корректности, наличия дефектов и несогласованностей в онтологических информационных компонентах на основе широкого спектра графовых моделей онтологий;
5. метод конструирования решателей задач интеллектуальной деятельности с повторным использованием онтолого-ориентированных программных единиц и операций на основе иерархии постановок, предлагающей готовые решения;
6. методология конструирования жизнеспособных систем для поддержки решения задач интеллектуальной деятельности (с учетом их места в иерархии постановок).

Теоретическая значимость работы заключается в развитии методов обеспечения важных свойств систем, основанных на знаниях – адаптируемость и повторная используемость. Результаты диссертационного исследования использованы при выполнении государственных заданий (№ 0262-2014-00025 и 0262-2019-0004) по теме «Интеллектуальные системы обработки данных, знаний и принятия решений» и следующих проектов: РФФИ: № 18-07-01079, 17-07-00299, 16-07-00340, 15-07-03193, 14-07-00270, 13-07-

00024, 12-07-00179; ДВО РАН: 12-I-П15-03, 12-III-А-01И-019; 12-II-УО-01И-001, 12-III-А-01И-016, проект 12-I-ОНИТ-04, 15-I-4-029.

Практическая значимость.

Практическую значимость диссертационного исследования имеют следующие научные результаты:

1. готовая к применению технология конструирования СБЗ на основе онтологий, которая обеспечивает снижение трудозатрат на производство программных систем для интеллектуальной информационной поддержки при принятии решений, способных к развитию по мере развития ПрОбл;
2. специализированная инфраструктура для выполнения всех этапов конструирования СБЗ отдельно взятой предметной области, включающая:
   - совокупность онтологий для формирования портала знаний и сервисов для отдельно взятой предметной области,
   - информационные и программные повторно-используемые компоненты сервисов для поддержки решения задач,
   - комплекс практически-полезных сервисов для этой ПрОбл,
   - инструментальные подсистемы для развития баз знаний и других информационных компонентов в этой ПрОбл и оценивания их качества.

Область применения полученных результатов: разработка практически-полезных развиваемых сервисов консультирования и поддержки принятия ответственных решений в предметных областях, где знания специалистов формализуемы.

Результаты исследований были апробированы при создании сервисов поддержки решений и комплексов сервисов для нескольких специалистов из медицины и вирусологии (Приложение 1), а также сервиса для верификации математических доказательств. Полученные в работе результаты могут быть использованы при построении интегрированных информационных систем в других сферах деятельности.

Достоверность научных и практических результатов подтверждается созданием комплекса инструментальных средств для разработки и сопровождения всех компонентов программных систем с базами знаний. Построенные онтолого-ориентированные компоненты программной системы эволюционируют по мере изменения фактологиче-

ского материала, расширения знаний и выявления на практике новых случаев, требующих корректировки знаний человеком-экспертом или автоматически.

Основные положения, выносимые на защиту:

Многоуровневая классификация интеллектуальных задач, отражающая важные свойства предметных областей.

Спецификация в единой формальной системе понятий классов задач, решаемых на основе формализованных знаний.

Алгоритмы решения практически полезных интеллектуальных задач: запроса дополнительной информации, диагностики развивающегося процесса, планирования воздействий на систему или объект и др.

Модель жизнеспособной системы для поддержки решения задач интеллектуальной деятельности в рамках их постановок.

Структурный подход к оцениванию информационных компонентов систем с базами знаний.

Методология обеспечения качества баз знаний интеллектуальных систем через монотонное повышение «оценки» их правильности.

Принципы архитектурного проектирования онтологического решателя (через композицию из повторно-используемых модулей, обрабатывающих конкретные типы отношений между понятиями).

Методология разработки жизнеспособных систем с базой знаний.

Апробация работы

Основные результаты диссертационной работы докладывались на следующих научных конференциях: конференция по искусственному интеллекту (КИИ) (г. Москва, 2020, г. Ульяновск, 2019, г. Москва, 2018, г. Смоленск, 2016 и др.), "Знания - Онтологии – Теории» (ЗОНТ) (г. Новосибирск, 2019, г. Новосибирск, 2017 и др.), "Открытые семантические технологии проектирования интеллектуальных систем" (OSTIS) (г. Минск, 2019, г. Минск, 2015 и др.), «Russian-Pacific Conference on Computer Technology and Applications» RPC (г. Владивосток, 2010, г.Владивосток, 2017), Крымская конференция «СВЧ-техника и телекоммуникационные технологии» (КрыМиКо) (г. Севастополь, 2012 и др.), международная мультиконференция по инженерным, компьютерным и информа-

ционным наукам SIBIRCON (г. Новосибирск, 2019), «Системный анализ в медицине» (г. Благовещенск, 2018), на научных семинарах ИАПУ ДВО РАН и др.

Содержание диссертации соответствует специальности 05.13.11 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей». Задача 1 связана с методами и алгоритмами проектирования программных систем (п. 1 паспорта специальности); Задача 2 – с обеспечением качества, стандартизации и сопровождения программных систем (п. 10); Задача 3 – с управлением базами знаний (п. 4); Задача 4 – с методами верификации и тестирования, управлением базами знаний и оценкой качества и стандартизацией программных систем (п. 1, п. 4 и п. 10); Задача 5 – с моделями и алгоритмами проектирования программных систем и с сопровождением программных систем (п. 1 и п. 10). Задача 6 – с моделями, методами и программной инфраструктурой для организации территориально распределенной обработки данных и с сопровождением программных систем (п. 9 и п. 10).

По теме диссертационной работы опубликованы 52 научные работы, из них:

15 статей в журналах, рекомендованных ВАК;

2 переводные версии и 14 публикаций, индексируемых в базах SCOPUS, WoS и MathSciNet;

21 статья в научно-технических журналах и сборниках.

2 свидетельства о регистрации программы.

Структура и объем работы. Диссертация состоит из введения, 6 глав, заключения, списка литературы. Объем диссертации составляет 217 страниц, в том числе список литературы из 170 наименований, 24 рисунка, 1 таблица. Диссертацию дополняет 1 приложение.

ГЛАВА 1. Обзор методов и технологий поддержки решения интеллектуальных задач с использованием формализованных знаний

Проблематика систем, использующих формализованные знания для принятия решений, стала актуальной еще в 1970-80-е годы. Множество публикаций (Уотермана, Лената, Хейеса-Рота, Кленси и др.) были посвящены обсуждению того, для решения каких интеллектуальных задач следует разрабатывать так называемые экспертные (или интеллектуальные) системы.

Целью их создания (т.е. автоматизации интеллектуальной деятельности) является повышение качества принятия ответственных решений за счет информационно-интеллектуальной поддержки этого процесса, а в целом – усиление познавательной деятельности человека или умственных способностей человека.

Достижение этой цели связано, во-первых, с поддержкой повседневной интеллектуальной деятельности, во-вторых, с поддержкой процесса управления качеством знаний, используемых для этой поддержки [58].

1.1. Распространенные классификации задач интеллектуальной деятельности

При проведении системного анализа для автоматизации произвольной интеллектуальной деятельности обычно стремятся выделить отдельные подзадачи в автоматизируемой деятельности [105, 151]. Для снижающей трудозатраты повторной используемости решений важно найти нужные готовые постановки известных задач.

Исследования разных авторов, представляющих классификацию разных известных так называемых экспертных задач и их описаний, внесли вклад в инженерию знаний и предназначались для целей обучения [19, 40], а также для инженеров-программистов, нуждающихся в практическом руководстве по экспертным системам, подкрепленном теоретическим материалом [41].

1.1.1 Распространенные задачи и их описания

Различая задачи *анализа* и *синтеза,* авторы делают акценты на разных особенностях получения их решения. В задачах *анализа* требуется [120] определить неизвестные характеристики или свойства сложного объекта (системы). Результатом задач синтеза считается построение модели объекта по заданным условиям [85] или изменение его конструкции [120]. В задачах *синтеза* при конструировании решений из частей множество решений потенциально не ограничено [19] в отличие от множества решений в задачах *анализа*. *Аналитические* системы предполагают выбор решений из множества известных альтернатив [19], а *синтетические* системы – генерацию неизвестных решений [95].

Интерпретация (данных) – это задача *анализа*, нацеленная на определение «смысла данных», результаты которого должны быть согласованными и корректными [19]; формирование описания ситуаций по результатам наблюдений [99]; подготовка объяснений для наблюдаемых данных [40]; определение «сущности» рассматриваемой ситуации [95]. Под это описание попадают два разных случая: когда данные являются невербальными, это визуальные образы, звуковые сигналы и т.п. и когда данные представляют набор признаков объекта (системы) и их значений, т.е. символьные представления проблемной ситуации.

Диагностика (задача *анализа*) определяется как обнаружение у объекта неисправностей [19] некоторого класса или отклонения параметров объекта (системы) от нормативных [80]. Иногда к этому добавляется выявление причин, приведших к возникновению отклонения от нормы [40, 120]. При решении этой задачи используют знания о статических и динамических проявлениях неисправностей, а также о типичных для системы парах сигналов «стимул/реакция» [120] и о влиянии «факторов» на внешние свойства функционирующей системы [95].

Мониторинг (задача *анализа*) описывается как непрерывная интерпретация данных в реальном времени и сигнализация о выходе тех или иных параметров за допустимые пределы [19]. Мониторинг может быть нацелен на обнаружение «отклонения в поведении», получаемом в ответ на поданный сигнал [120]. Иногда мониторинг определяется как измерительная задача слежения за текущей ситуацией [95], сопутствующая экспертным задачам (диагностика, прогнозирование, планирование, коррекция действий) [95].

Прогнозирование – это задача, нацеленная на предсказание хода событий в будущем (на основании модели прошлого и настоящего) [99]; предсказание последствий некоторых событий или явлений (на основании анализа имеющихся данных) [19] или последствий развития текущих ситуаций (на основе математического и эвристического моделирования) [95]. Задача прогнозирования связывает описание некоторых характеристик будущего поведения объекта с описанием прошлого поведения [14]. Иногда прогнозирование относят не к задачам *анализа,* а к «комбинированным» задачам (и анализа, и синтеза) [61, 80].

Проектирование **(**задача *синтеза* [75,94]) – это определение конфигурации объектов [95], или синтез потенциальной структурной конфигурации [167], или построение структурной организации компонентов [120], которая удовлетворяет заданным ограничениям и критериям, или построение по спецификации такого проекта, по которому можно будет произвести объект [162], или выбор конфигурации многокомпонентных систем [41].

Планирование – это задача *синтеза,* нацеленная на выбор последовательности действий по достижению поставленной цели [41, 95], или «нахождение планов действий» для объектов-исполнителей [19], или упорядочение возможных действий для достижения заданного суммарного эффекта [10], или поиск последовательности операторов, которая преобразует начальную ситуацию в конечную [84]. В описаниях этой задачи шаги плана рассматриваются как компоненты синтезируемого решения [41]. План не обязательно рассматривается как последовательность действий; иногда предлагается частичный порядок действий.

Даже эти шесть всеми упоминаемых задач описываются по-разному. А вместо их формальных постановок дается описание характерных особенностей, отличающих их друг от друга.

1.1.2. Описания менее распространенных задач

Управление как «простая» экспертная задача нацелена на определение того, какие входные сигналы следует подать на вход объекта (системы), чтобы получить желаемую реакцию или изменения, опираясь на известные характеристики [120].

Встречаются описания *задачи управления* как функции, поддерживающей определенный режим деятельности сложных систем [80, 96] или как обеспечение выполнения процесса. Заметно, что многие авторы рассматривают поддержку определенного режима функционирования системы как задачу, составляемую из нескольких: интерпретации, прогноза, планирования, моделирования, оптимизации выработанных решений, мониторинга [52,80]. Другой вариант: в управление включены контроль, диагностика, прогнозирование, планирование, слежение за окружающей обстановкой, распознавание происходящих событий [115]. Например, «адаптивное управление поведением сложных человеко-машинных систем» «составлено» из двух задач [99]: прогнозирование появления возможных сбоев и планирование действий, необходимых для их предупреждения. *Отладку* вместе с *диагностикой* иногда рассматривают как единую задачу [95,96], также аналогичные им *диагностику* и *лечение*. Иногда *управление* сводят к *принятию решений* [95].

К «простой» задаче *управления* близка задача *коррекции* – выработка рекомендаций по устранению неисправностей [80] или определение действий по исправлению отклонений от нормального состояния [80], а так называемые *наладочные* системы предназначены для выработки рекомендаций по устранению неисправностей в контролируемой системе. Системы *оказания помощи при ремонте* оборудования выполняют планирование процесса устранения неисправностей в сложных объектах [99], а сам *ремонт* – это устранение неисправностей по уже предписанному плану [96].

Обучение (людей) традиционно рассматривается как составная задача. При обучении проводят анализ знаний студентов по определенному предмету, отыскивают пробелы в знаниях, диагностируют ошибки или слабости в познаниях обучаемых и находят соответствующие средства для их ликвидации, а также планируют акт общения с учеником в зависимости от успехов ученика с целью передачи знаний [19, 99].

Задачу *распознавания* различных ситуаций [95] и *задачу классификации* [16] можно обобщить в *задачу определения принадлежности ситуации к некоторому классу*, сюда же можно отнести «статическую» диагностику – процесс соотнесения объекта с некоторым классом объектов [19]. К задаче *интерпретации* можно отнести задачу *извлечения информации из первичных данных*, а также *структурный анализ сложных объектов* [41]. Задачей *идентификации* в [120] названа достаточно абстрактная *задача ана-*

лиза, отличающаяся от других использованием в качестве анализируемой информации пар сигналов «стимул – реакция».

Иногда при обсуждении спектра решаемых задач упоминается необходимость *определения того, какой информации не хватает*, чтобы получить множество решений. Обычно такую задачу совмещают с той задачей, для решения которой не хватает информации [14, 95]. Задача *доопределения информации* многими упоминается, но никем не определяется.

Концептуальное проектирование – это синтез структурной конфигурации, удовлетворяющей некоторым ключевым ограничениям [167]; ***детальное проектирование*** – выбор и распределение структурных компонентов, удовлетворяющих применимым ограничениям [167]. ***Планирование сборки*** – это поиск плана реализации спроектированной системы из отдельных ее компонентов [120].

Из описания «систем экспертного типа» ясно, что отдельно могут быть рассмотрены задачи ***критики принятых решений***. Постановки таких задач не приводятся, но указывается на их полезность, как минимум, в учебном процессе [67]. Такие системы поддерживают сравнительный анализ гипотез и выбор решения.

Параллельно с разными типами задач анализа и синтеза, решаемыми с использованием заранее построенных баз знаний, авторы определяют и задачи ***построения баз знаний***. Результат решения таких задач – математические модели зависимости свойств объекта от его признакового описания и воздействий на него [14,56]; разделение множества объектов на непересекающиеся классы.

1.1.3. Постановки задач

В работах обычно отсутствуют формальные постановки задач; однако имеются отдельные работы, в которых описывается, что в задаче дано, и что нужно найти [84,162].

Постановка ***задачи поиска плана действий*** применительно к интегральным роботам у [84] заключается в следующем:

дано: начальная ситуация (объект, состояние), конечная, или целевая ситуация (объект, состояние); множество операторов, преобразующих одну ситуацию в другую;

найти такую последовательность операторов, которая преобразует начальную ситуацию в конечную.

Примеры постановок трех задач приводятся в [162].

Задача диагностики

дано ситуация ненормального функционирования; необычные проявления; стандартный набор диагностических тестов;

найти известные категории болезней, объясняющие причины признаков, и рекомендовать методы лечения.

Задача проектирования

дано: спецификация проектируемого объекта или системы; стандартные аналитические тесты на систему и компоненты; возможные компоненты, их свойства и взаимосвязи между ними;

найти объект или систему, которая удовлетворяет этой спецификации.

Задача планирования

дано: спецификация достигаемого результата; отдельные действия;

найти план, который достигает результат, удовлетворяющий этой спецификации.

Постановка задачи медицинской диагностики в [75] такова.

Задача диагностики

дано: медицинские знания о наблюдениях, их нормальных значениях,

знания о причинно-следственных связях между заболеваниями и наблюдениями, между событиями и наблюдениями, между событиями и заболеваниями,

результаты наблюдений пациента (значения наблюдавшихся признаков и анатомо-физиологических особенностей, произошедшие события);

найти: диагноз пациента, причину каждого заболевания и объяснить все полученные значения наблюдений.

Постановка задач поиска знаний по имеющимся наборам данных состоит в следующем.

Детерминированная задача поиска знаний (для фиксированной модели зависимости между значениями признаков объектов и их классами)

дано: обучающая выборка (конечное множество объектов, каждый из которых задан значениями признаков и классом, которому он принадлежит,

найти: решающее правило, которое объекту, заданному значениями признаков, правильно сопоставляет класс, которому этот объект принадлежит [15; 45].

Наличие таких постановок позволяет видеть различие и общие черты разных задач.

1.1.4 Классификации задач

Классифицируя (экспертные) задачи и устанавливая некоторые отношения между ними, авторы чаще всего базируются на дихотомии – *задача анализа* и *задача синтеза*. Эти два «класса» задач различают «по способу формирования решения» [95] (Рисунок 1.1). На **Ошибка! Источник ссылки не найден.** изображена наиболее распространенная иерархия задач.

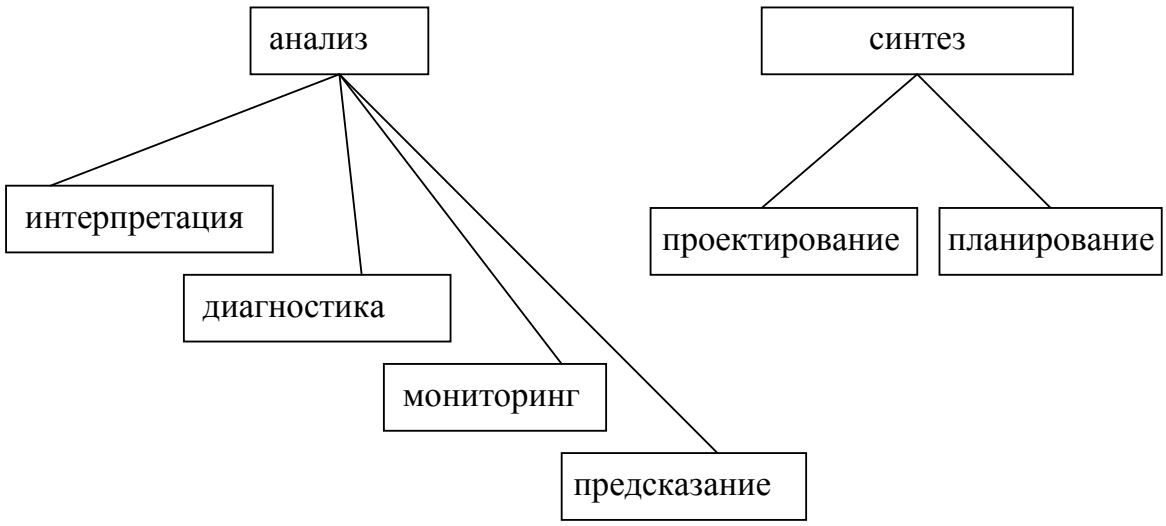


Рисунок 1.1 – Распространенная иерархия задач

Следующий (второй) уровень в классификациях разных авторов отличается. В [19] задачами анализа считаются: *интерпретация данных, диагностика, поддержка принятия решения*, а синтеза – *проектирование, планирование, управление*. Задачи *мониторинга, предсказания* и *обучения* отнесены автором к комбинированным задачам, причем задача *предсказания* отнесена к комбинированным в связи с тем, что при ее решении часто нет возможности выбрать решение из множества известных альтернатив.

В [120] *задачи анализа* различаются «по анализируемой информации». Автор ввел подклассы задач *управления, предсказания* и *идентификации*: в каждом из этих трех подклассов задачи анализа требуется найти значения одного из членов тройки <входные сигналы, реакция, характеристики системы> при известных двух других [120]. А на следующем уровне подклассы задачи идентификации – *мониторинг* и *диагностирование* (Рисунок 1.2) – различаются получаемым результатом.

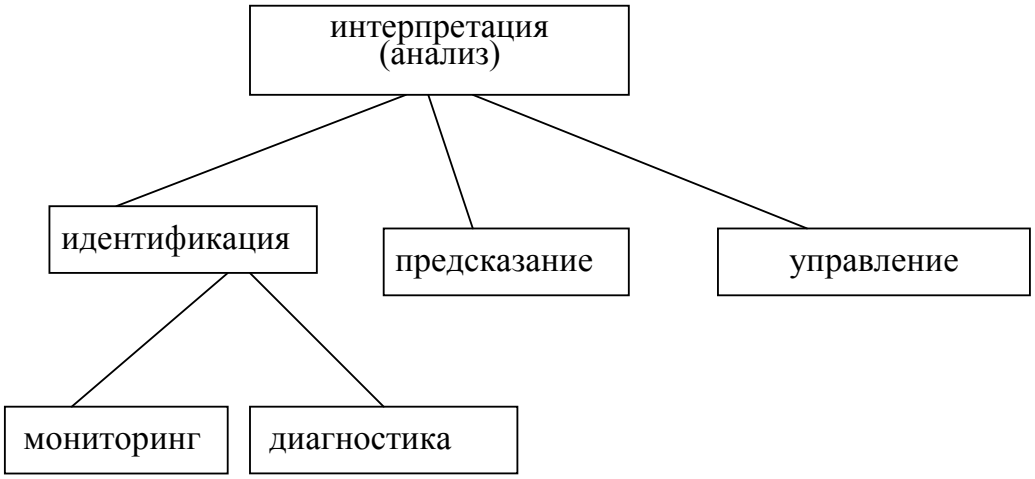


Рисунок 1.2 – Иерархия «аналитических операций» (по Кленси)

К *задачам синтеза* все авторы относят задачи *проектирования* и *планирования* [167], которые принято рассматривать на одном уровне [162,167]. В [19] к ним добавляется *управление*. В [95] к *синтезирующим* (для динамической предметной области) отнесены не только *проектирование* (определение конфигурации), *планирование* (выбор последовательности действий) и *диспетчирование* (составление расписаний), но и *управление*, а также *прогнозирование, мониторинг* [95] (Рисунок1.3).

В [120] *проектирование* дополнено *спецификацией* и *планированием сборки*. Под спецификацией понимается *специфицирование нового состояния существующей системы* (здесь автор проводит аналогию между задачей «специфицировать цели» и задачей «планировать», предшествующей созданию графика работ).

Планирование же поставлено в [120] на следующий уровень классификации (туда же ставят планирование и другие, как отмечается в [41], трактуя *планирование* как «проектирование последовательности операций». Стоящие «под» *проектированием* задачи *конфигурирование* и *планирование* рассматривают либо как альтернативные задачи либо как пару взаимосвязанных задач.

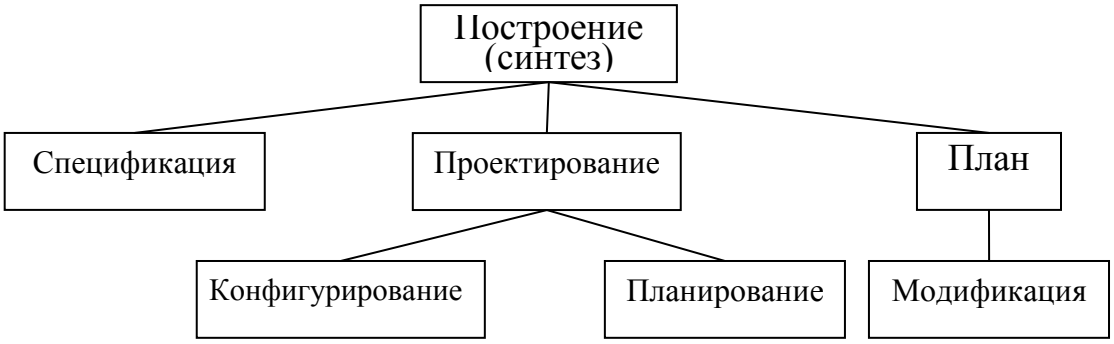


Рисунок 1.3 – Иерархия «операций синтеза» (по Кленси)

Авторы часто отмечают, что реальные задачи являются составными, определяемыми на базе простых, указанных в их классификациях. Но мнения о том, какие задачи и из каких «составляются», весьма различны. Задача *выработки рекомендаций о курсе лечения* в медицинской системе включает мониторинг состояния пациента, диагностирование категории заболевания, идентификацию микроорганизмов и изменение состояния пациента [41]. *Коррекция* – это диагностика, дополненная возможностью оценки и рекомендации действий по исправлению отклонений от нормального состояния [95], а *управление* – это мониторинг, дополненный реализацией действий [52]. *Отладка*, как считают, эквивалентна диагнозу плюс модификации [120]. К *диагностике* относили не только сопоставление случая категории болезней и поиск вероятного объяснения (причины) признаков, но и рекомендации по методам лечения [162], а *мониторинг* рассматривали [99] как часть *диагностической* системы.

Приведенный обзор достижений в области классификации задач показывает, что несмотря на то, что к настоящему времени выполнено немало исследований и обзорных работ по классификации задач, решаемых интеллектуальными системами, почти все авторы основывают свои подходы к классификации на собственной интуиции, не пытаясь апеллировать к тем или иным фундаментальным основам.

Таким образом, до сих пор не предложено общепризнанной стройной классификации известных задач интеллектуальной деятельности.

Более того, задачи описываются в разных терминах, что затрудняет сравнение их содержания. Крайне редко встречаются более или менее формальные постановки задач. Каждый автор описывал некоторый набор задач в своих терминах, чаще всего в виде текста, без формальных постановок. Никто не определил формально в единой терминологии постановки всех известных интеллектуальных (экспертных) задач. Поэтому акту-

альна классификация задач интеллектуальной деятельности на основе общих принципов и в единой терминологии. Точные постановки часто решаемых задач помогут перейти от искусства разработки систем, основанных на знаниях, к технологии.

1.2. Особенности поддержки решения задач интеллектуальной деятельности

Ситуации принятия решений могут возникать как в контексте оперативной деятельности, так и стратегического планирования. Одним из примеров областей, в которой принятие решений играет особенно важную роль, является «поддержка функционирования (безопасности, целостности) какой-либо системы, например промышленного предприятия. Фазы такой деятельности можно описать при помощи цикла управления безопасностью» [21]. Автоматизация дает возможность очень быстро обработать весь объем доступной информации, то есть учесть все предпосылки и возможные следствия [21]. Известны разработки для разных видов деятельности в области медицины [66; 78].

Отличительной особенностью интеллектуальных систем является выделение ранее отсутствовавшего аспекта поддержки решений – способности системы к «пониманию» проблемы, ее решению и объяснению полученного решения. [51; 58]. Это достигается введением в автоматизированную систему специальным образом организованных описаний знаний предметной области (ПрОбл) – базы знаний (БЗ).

Поддержка для исполнителя может состоять в генерации интеллектуальной системой рекомендаций (в том числе возможных альтернативных решений) на основе баз знаний. О*бъяснение* содержит информацию о соответствии гипотез о вариантах решения – информации об объекте (сложной системе, процессе) и базе знаний. «Система также может сформировать нормативное обоснование для всех предложенных ею решений» [21].

О*бъяснение* содержит информацию о соответствии гипотез о вариантах решения – информации об объекте (сложной системе, процессе) и базе знаний.

Реализация *объяснения* может быть выполнена как самостоятельный модуль (блок "объяснения" вывода заключений – один из функциональных блоков системы [53**])** или как попутное сохранение следов процесса вывода. Результаты обработки информации предоставляют пользователю с возможностью «просмотреть логические основания каж-

дого полученного вывода: примененные правила, исходные факты, их происхождение» [21]. Часто построение объяснений – трассировка промежуточных пунктов в ветвях иерархии вывода понятий из понятий более глубокого уровня иерархии (иногда с пометками для читателя символом типа "V"), например, вывода диагнозов из понятий симптомокомплексов и синдромов, а затем их – из симптомов (признаков) [53].

Различают два подхода к построению объяснений: (a) интерпретируемые модели [31], которые полагаются на «системы не-черного ящика, такие как основанные на правилах» («models, which rely on non-black box systems such as rule-based ones») и (b) генерацию объяснений «алгоритма черного ящика» [118; 153]. Ожидание (или требование) от получения на основе баз знаний понятных специалистам объяснений (таких рекомендаций) [159] породило термин «объясняемый искусственный интеллект (XAI)» и понимание в качестве «интеллектуальной системы» – способной объяснить свои рекомендации человеку [153].

На сегодняшний день «бум нейросетей» обеспечил огромное количество инструментов и специализированных библиотек для создания и обучения нейросетей, доступны и готовые, уже обученные, нейросети для решения разного рода задач. Работа над объяснимостью их результатов «в начале пути». В частности, ищется аналогия трассировки всех пунктов в ветвях иерархии вывода понятий в «принципах синаптической передачи возбуждения, определяющим переход в состояние возбуждения нейрона, на теле которого имеются синаптические окончания от других нейронов в нейронной сети» [53; 39]. Пока «непрозрачность» нейросетевой логики производства решений не дает оснований использовать их в ответственных предметных областях.

Сохранение объяснимости работы системы для человека особенно важно для систем поддержки принятия решений [21]. Осуществляемая автоматизация *поддержки принятия решений* означает принятие специалистом решения с учетом объяснения экспертной системы. Окончательное принятие решения при этом остается за человеком, который наряду с формальными основаниями имеет также возможность руководствоваться ценностными установками [21].

Объяснение автоматизированной системы до определенной степени может рассматриваться как аналог консультации. И в том, и в другом случае проводится независимый анализ входных данных на соответствие их тем или иным гипотезам о

решении задачи в свете знаний (консультанта или экспертной системы). Например, в медицине важно объяснение того, какие гипотезы-диагнозы могут быть отвергнуты, а какие гипотезы могут быть приняты для конкретной истории болезни с учетом базы знаний о медицинской диагностике. На рисунке 1.4 показано, что специалист (например, врач), принимая решение (например, ставя диагноз) на основании данных об объекте (например, истории болезни пациента), руководствуется своими знаниями, но получает возможность учесть формируемое системой объяснение (анализ возможных гипотез о диагнозе для этого пациента) [58].

Современные *системы с базами знаний* (СБЗ) могут получать данные от аппаратных средств измерения и часто интегрируются со статистическими и другими программными компонентами [90]. Системы поддержки принятия оперативных решений выполняют:

– сбор информации из разнородных источников (датчики и сенсоры, информационные сообщения от сторонних лиц и организаций, данные исследований и интерпретации их результатов);

– вычисление следствий из имеющихся предпосылок – имеющегося широкого набора фактов и вычисляемых всевозможных следствий) [21].

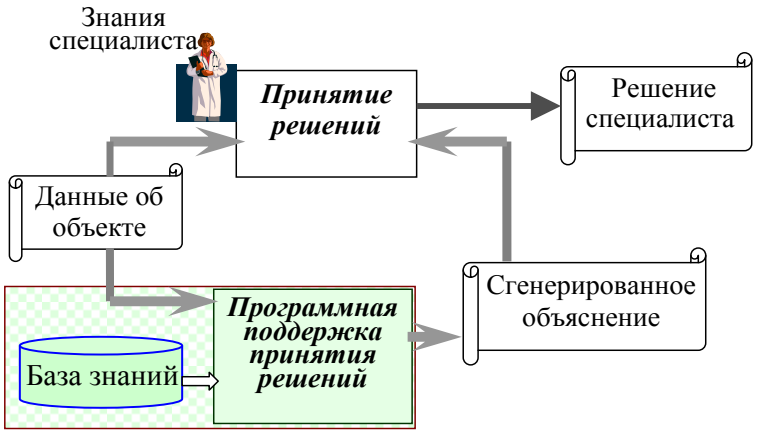


Рисунок 1.4 – Схема принятия решений специалистом с автоматизированной поддержкой

В этих системах задачи решаются «с применением эвристик, включая эмпирическую индукцию, аналогию и дедукцию» [98] часто с использованием более одного метода имитации интеллектуальной деятельности человека.

Спецификация задачи получения решения y ∈ Y (множество значений для результатов решения) на основе баз знаний может быть выражена предикатом:

P(x, k, y), где x ∈ X (множество входных данных или ситуаций), k ∈ Kn (X, Y).

Kn (X, Y) – множество версий формализованных знаний, претендующих на возможность сопоставить входной ситуации ее точное решение.

До недавнего времени применяемые в системах базы знаний, как правило, были сформированы экспертами [51, 86,88,94].

1.3. Проблемы автоматизации решения задач интеллектуальной деятельности

Несмотря на отсутствие единой классификации и формализации таких задач, в сфере искусственного интеллекта были предложены универсальные технологии и продолжают создаваться многочисленные *системы с базами знаний* (экспертные системы) для поддержки решения таких задач.

Универсальными средствами являлись специализированные языки программирования, такие как PROLOG, LISP, SMALLTALK, FRL, Interlisp и др., универсальные инструментальные системы и «оболочки» для БЗ: Level5, Object, G2, Clips, Loops, VITAL, KEATS, и др. и более новые: OSTIS, АТ-ТЕХНОЛОГИЯ и др. Существующие различные инструментальные системы различаются наборами предлагаемых разработчику компонентов, уровнем поддержки этапов жизненного цикла создания и сопровождения систем с базами знаний, поддержке различных технологий разработки, используемыми формализмами представления знаний, методами их формирования и отладки, используемым механизмам вывода, средствам проектирования пользовательских интерфейсов.

Одним из технологических шагов было применение специализированных оболочек, ориентированных на конкретный класс систем с базами знаний. (Создание экспертной системы с использованием оболочки РЕПРОКОД не требует знания программирования для компьютеров и становится самостоятельной задачей, которую может решать сам эксперт или инженер знаний во взаимодействии с экспертом) [53].

Существенный вклад в методы, технологии, инфраструктуру построения СБЗ и улучшения качества БЗ внесли российские и зарубежные ученые и основанные ими

школы: Артемьева И.Л.; Гаврилова Т. А., Голенков В.В., Грибова В.В., Еремеев А.П., Загорулько Ю.А., Клещев А.С., Кузнецов О.П., Осипов Г.С., Попов Э.В.; Рыбина Г.В., Соснин П.И., Стефанюк В.Л., Тельнов Ю.Ф.; Финн В.К., Ярушкина Н.Г., Матвеева Т. О. и Склюева О. Н. (1996); Джарратано Дж., Райли Г.; Джексон П.; Лавилла С.; Уотермен Д., Chandrasekaran B., Musen M., Mortensen J., Noy N.; F. Hayes-Roth; B. Moore; Calero C.; Taboada M.; Emmanuel C. Ogu, Adekunle, Y.; R. M.Sonar, R. Fikes and D. McGuinness; Müller L., Perry J. и многие другие.

Однако, в реальной сфере практической интеллектуальной деятельности очень мало случаев повседневной и продолжительной эксплуатации таких систем для поддержки принятия ответственных решений, поскольку технологии их создания (предлагаемые российскими и зарубежными коллективами) сосредоточены преимущественно на методах разработки («существенно сокращается время разработки систем за счет комбинирования типовых проектных процедур (ТПП) и применения специальных механизмов управления ходом их построения» [66, 68, 69], а не развития.

Довольно часто *интеллектуальная деятельность* при принятии ответственных решений в повседневной деятельности осуществляется в условиях изменчивости знаний предметной области (пример – медицина с регулярным обновлением методов диагностики и лечебных средств). Внедряемые в практику экспертные или интеллектуальные системы) должны обладать принципиально важной способностью адаптироваться во времени к таким изменениям. Требование к возможности их длительной эксплуатации (сохранения полезности) связано и с высокой трудоемкостью создания СБЗ и ответственностью за возлагаемую поддержку деятельности специалистов.

Проблема возможности последующего сопровождения выходит на первый план для большинства сложных программных систем (ПС), усилия на разработку которых измеряются человеко-годами. Процесс сопровождения вносит основной вклад «в совокупную стоимость владения системой» [21] или занимает значительную долю трудозатрат в процессе жизненного цикла программной системы (от 50 до 80% [142]). Актуальным для коллективов разработчиков остается реализация механизмов, обеспечивающих неизбежное последующее сопровождение программных систем, вызываемое изменениями знаний предметной области, условий эксплуатации или требований пользователей к функциональности.

Известными механизмами, используемыми в программной инженерии для упрощения сопровождаемости ПС, являются: архитектурные решения (разделение на слабо сцепленные компоненты с логически понятными и функциями), декларативное представление компонентов ПС, средства автоматизации построения проектных моделей и автоматической генерации по ним фрагментов кода, средства связывания компонентов друг с другом, стандартизация сервисов и интерфейсов, введение в процесс жизненного цикла вида деятельности «управление требованиями», разделение компетенций между разными типами разработчиков.

Полезно использование типовых архитектурных решений. Для программных систем с заранее предопределенной архитектурой (обеспечиваемой фреймворками и оболочками) сопровождение заметно проще, поскольку проблема адаптируемости к устройствам, платформам, гибкость пользовательского интерфейса (ПИФ), формата хранимой и другой информации «перекладывается» на отмеченные инструменты.

Полезен модельно-управляемый подход к созданию программных систем. «Абстрагирование программного кода от привязки к предметной области и конкретным вариациям алгоритмов обработки информации» является целью создания автоматизированных систем, управляемых моделью [21]. Система, построенная по таким принципам, максимально гибка и устойчива к изменению функциональных требований, знаний о предметной области, ситуации на автоматизируемом предприятии и его задач. В идеальном случае, при изменении требований к автоматизированной системе, аналитик должен отражать эти изменения в модели системы, не обращаясь к программистам и не взаимодействуя с программным кодом. А программный код должен автоматически перестраивать свою работу (включая пользовательские интерфейсы) в соответствии с ними. С экономической точки зрения сопровождение и модификация подобных систем оказываются существенно более выгодными [21].

В процессе эксплуатации прикладных ПС (и инструментальных средств для их разработки) часто появляется потребность в:

- исправлении ошибок,
- добавлении пользовательских функций (расширение),
- адаптации (новые приборы, устройства, платформы),

- изменении пользовательского интерфейса (ПИФ) специалистов,

- уточнении формата или состава информации (улучшение).

Что именно потребуется – зависит от назначения ПС, от уникальности (рыночная vs заказная) и от типа обрабатываемой информации.

Сопровождаемость ПС выражается через такие примитивы качества ПС как *расширяемость*, *модифицируемость*, *структурированность*, *модульность* [2, 114,122]. *Расширяемость* обеспечивается возможностями автоматически настраиваться на условия применения ПС по информации, задаваемой пользователем. К таким условиям относятся в т.ч. «требования, которые определяют режим применения ПС или конкретизируют структуру информационной среды». К этим возможностям относят и «возможность добавления к ПС определенных компонентов», например, «прием от пользователя необходимой информации и настройку ПС по этой информации». Важно, что решение о возможности такой расширяемости «принимается в процессе разработки архитектуры ПС». *Структурированность* и *модульность* упрощают ручную модификацию программ ПС. *Модифицируемость* обеспечивается такими свойствами, реализуемыми программным путем, которые облегчают внесение изменений и доработок в документацию и программы ПС ручным путем (возможно, с определенной компьютерной поддержкой). В частности, направления и особенности модифицикации и развития ПС должны быть учтены при разработке архитектуры и модульной структуры ее программ [1,2, 46].

Чаще всего предусматривается развитие ПС в виде последовательности версий, обычно следующие версии предлагают дополнительные функции по работе с информацией. Чтобы все его компоненты ПС на всех уровнях представления оставались согласованными в каждой новой версии ПС, реализуется процесс *управления конфигурацией* (configuration management). Связи и зависимости между документами и их частями описываются в специальной документации по сопровождению (но когда в процессе доработки находится сразу несколько версий ПС, обеспечение согласованности документов в разных конфигурациях требует компьютерной поддержки и специального раздела в базе данных, где фиксируются связи и зависимости между документами и их частями для версий ПС).

В случае с прикладными СБЗ, связанными с решением традиционных интеллектуальных задач (диагностика, планирование, прогноз… и т.д.), акценты смещаются. Здесь реже характерна потребность в добавлении новых пользовательских функций («расширение»), скорее ожидается изменчивость знаний (появление новых методов диагностики, выявление новых факторов, влияющих на ситуации, и т.п.) и методов решения, объяснительной компоненты и пользовательского интерфейса. Причина в том, что постановки многих известных задач – диагностики, планирования и т.д. – достаточно устойчивы, а новые интеллектуальные задачи чаще всего являются конкретизацией или уточнением известных или их комбинацией. Развиваемость (модифицируемость) прикладных ПС также зависит и от инструментария для их разработки.

Устойчивость программной системы к некоторым изменениям окружающей среды и способность адаптироваться во времени к таким изменениям называют *жизнеспособностью*.

По аналогии с тем, как жизнеспособность программных систем проявляется через адаптивность к изменениям в среде и через адаптируемость в ответ на новые требования, проявляется и жизнеспособность для системы, основанной на знаниях [147]. Но в условиях изменчивости среды функционирования и средств пользовательского интерфейса СБЗ оказывается реже, чем в условиях изменчивости знаний предметной области.

Используемые системами формализованные за приемлемое время базы знаний часто содержат упрощенные модели процессов и сущностей. («Элементы знания – отношения понятий, «которые являются, по существу, правилами для вывода, часто «сопряжены с упрощением правил вывода на знаниях по сравнению с реальными на практике» [53]). Реальные задачи неизмеримо сложнее, а объем информации, привлекаемый для их решения, не позволяет формализовать все накопленные вербально представляемые знания так же быстро, как создаются базы данных.

Во многих важных областях нельзя создать БЗ раз и навсегда, а придется регулярно расширять или уточнять формализуемые базы знаний. Это связано с тем, что для задачи на основе баз знаний не является справедливым утверждение о существовании решения [22]. Предполагается, что существует «правильная» база знаний $k^* \in Kn\ (X, Y)$ такая, что $\forall\ x \in X\ \exists\ y \in Y\ \ P(x, k^*, y)$; однако предположение это не может быть доказано. В

общем случае эта «правильная» база знаний неизвестна, а для любой другой базы знаний утверждение о существовании решения не обязано быть справедливым [22].

Начальные версии таких систем принято заполнять знаниями, почерпнутыми из учебников и монографий инженерами знаний, По завершении этого начального этапа разработки к работе привлекают «эксперта с большим опытом: в процессе знакомства с уже работающим прототипом системы такой специалист начинает критиковать конструктивные решения, использованные при построении семантической сети базы знаний, и содержание этой критики и есть знания эксперта, которые инженеру знаний следует использовать для включения в новые версии системы» [53].

По мере увеличения размера БЗ и увеличения прогнозируемой продолжительности жизни БЗ становится все более важным разрешать ввод, модификацию, отладку и обслуживание специалистами в данной области. Создание большой базы знаний – сложная задача даже для людей, «обученных этому искусству», и поскольку люди (которых не обучали искусственному интеллекту) берут на себя большую часть «рабочей нагрузки», для их поддержки необходимо создавать расширенные среды [124].

Поэтому для большинства предметных областей, связанных с решением интеллектуальных задач, подразумевается эволюционируемость знаний. Более того, в предметных областях, где важны влияние факторов и событий на состояние системы (объекта, ситуации), их изменение во времени, влияние индивидуальных характеристик объекта, системы и одних процессов на другие, эволюционируемость баз знаний – главный «вызов» современных «условий» по отношению к СБЗ.

Кроме того, большие задачи с большим сроком службы увеличивают вероятность того, что разработчики и сопровождающие базы знаний сформируют распределенную команду, потенциально с различными взглядами и словарями. Скорее всего, БЗ будут результатом слияния информации от нескольких экспертов в данной области. Это требует разработки среды для поддержки совместного проектирования и обслуживания БЗ, а также для поддержки объединения БЗ из нескольких источников [83,124].

Инструменты и технологии для обеспечения возможности усовершенствования СБЗ в процессе всего жизненного цикла (ЖЦ), как указывается в [123;146,156,163,168], должны обладать следующими основными свойствами:

1) доступность инструментов разработки широкому кругу потенциальных разработчиков,

2) поддержка коллективной разработки и сопровождения,

3) разделение компетенций разработчиков с предоставлением каждому типу разработчиков инструментов, ориентированных на их специализацию (БЗ должны разрабатывать эксперты ПрОбл, пользовательский интерфейс – дизайнеры интерфейса, решатели – программисты),

4) понятность представления знаний, особенно таких как причинно-следственные, временные, пространственные связи, для экспертов ПрОбл,

5) возможность формирования и сопровождения знаний, не влияющих на работоспособность решателей,

6) реализация механизмов оценки баз знаний, сформированных вручную, их обучения и до-обучения,

7) интегрированность со сторонними ПС.

1.4. Обзор реализации требований обеспечения жизнеспособности для систем, основанной на знаниях

Реализация требований 1 и 2 (из семи вышеперечисленных) осуществляется через облачность (или дистанционную доступность) инструментов. Современные технологии Protege, OntoEdit, IACPaaS и технология разработки специализированных интернет-порталов научных знаний коллектива ИСИ СО РАН поддерживают работу по созданию знаний и работе с ними на дистанционно доступных собственных платформах [47,48, 130,156].

На базе инструментального комплекса АТ-ТЕХНОЛОГИЯ разработана веб-ориентированная версия для поддержки построения прикладных веб-ориентированных ИЭС (интегрированных экспертных систем) на основе реализации всех традиционных этапов их создания. Эта версия предоставляет возможности организации веб-ориентированных сеансов интервьюирования экспертов, создания веб-ориентированного пользовательского интерфейса, настройки веб-сервера, управление пользователями и развертывание финального прототипа веб-ИЭС [86,87,89].

Есть web-сервисы для автоматизированного формирования продукционных баз знаний (например, на языке CLIPS) [114].

Для реализации требований 2 и 3 (поддержка совместной работы экспертов, дизайнеров, программистов) важна типовая (специализированная) архитектура с унифицированными внутренними интерфейсами и библиотеками повторно-используемых компонентов.

Применение специализированных оболочек, ориентированных на конкретный класс систем с базами знаний, обеспечило заметное сокращение сроков разработки и усилий на сопровождение систем. Причина в том, что модель представления обрабатываемых знаний максимально приближена к представлению, принятому в предметной области, поэтому не требуется преобразование знаний.

Но у специализированных оболочек "жесткий" решатель и встроенный в него интерфейс, которые, в случае изменения требований трудно модифицировать.

Альтернатива оболочкам – распространение применения объектно-ориентированного программирования и архитектуры MVC. При их использовании «в коде создаются программные объекты, описывающие свойства моделируемых объектов реального мира и операции» над ними», поэтому «в программном коде прикладных автоматизированных систем все еще содержится много логики, связанной непосредственно с предметной областью и конкретными решаемыми задачами». «Очевидным ограничением этого подхода является необходимость модифицировать код при внесении изменений в структуру модели» [21]. Также в этих технологиях часто «не прилагаются специальные усилия для построения модели предметной области; модель формируется эволюционным путем, без методологии и плана» [21].

Зато использование онтологии во время разработки концептуальной модели и компонентов баз данных и знаний становится “очевидным использованием”, потому что на практике онтология схожа со схемой базы данных [105,141,161].

Требования 3, 4, 5 (понятность представления знаний и ориентированность на специализацию экспертов при их формировании и сопровождении) **реализуются** через представление знаний в привычных терминах, отделение декларативных знаний от процедурных, использование онтологий [19,121,141] для создания БЗ и их сопровождения, семантическое представление знаний.

Привычность терминов и предоставление экспертам ПрОбл инструментов, ориентированных на их специализацию, решается иногда программированием подсистемы приобретения знаний, ведущей опрос эксперта в рамках сценария. (Диалог с пользователем в Системе приобретения знаний SALT ведется либо посредством вопросов-подсказок, либо посредством меню [19, 155]). В этом случае эксперт лишен инициативы в диалоге и поэтому ограничен в выражении всех своих знаний. Опробованы и средства, дающие возможность эксперту ввести список событий предметной области и определить связи между ними или указать, как концептуально могут быть организованы экспертные знания (системы ROGET, MOLE [19, 117]).

В инструментальном комплексе АТ-ТЕХНОЛОГИЯ реализованы: типовые проектные процедуры извлечения знаний из эксперта [86,87,89].

Более современное представление знаний – в виде семантических сетей и иерархий, интерпретация которых частично включает цепочки вывода: «Элементом знания» в такой системе является отношение понятия из одного уровня иерархии к «понятию следующего уровня». Это отношение является, по существу, правилом для вывода понятия из понятий более глубокого уровня иерархии» [53**].** Каждому элементу знаний эксперт приписывает вес, характеризующий величину его вклада (в итоговое заключение) и пороговое значение для суммы весовых значений. При выводе (диагностического) заключения решение принимается по достижению определенного порогового значения (диагностируемый синдром считается установленным, если реализовались: брадикардия, снижение артериального давления, снижение ударного объема сердца, сдвиг КЩС в кислую сторону, эритропения, лейкоциты в моче [53]).

Уже два десятилетия разработчики выбирают онтологии в качестве технологической основы автоматизированных систем [9, 48]. Чаще всего онтологии создаются средствами RDF/RDFS/OWL [113, 114, 121].

Современные технологии обеспечивают онтологический подход к представлению знаний – Protege, OSTIS и IACPaaS. Protege поддерживает объектно-ориентированную модель представления информации [156], а технологии IACPaaS [130] и OSTIS [20] – семантическую модель.

Не только общение с экспертом, но и диалог со специалистами, решающими свои задачи, тоже должен происходить в терминах их предметной области. Это задача поль-

зовательского интерфейса системы, современный подход к построению которых основан на онтологии области [47,48].

В онтологиях, создаваемых по стандартам RDF/RDFS/OWL, «Условия правил помещаются в саму онтологическую модель в виде триплетов, составленных при помощи предикатов специальной метамодели для описания правил» [21]. В SWRL и SPIN аналитик может описывать произвольные правила [21]. При этом понятность пользователям языков SWRL и SPIN для создания правил – вопрос спорный: используются триплеты (состоят из трех частей: субъекта, предиката и объекта), переменные или наборы переменных, унарные и бинарные Предикаты. (В условиях и следствиях SWRL- правил используются Предикаты (примеры: унарный предикат – запись вида СостояниеРегулированияДвижения (state), бинарный предикат –ЗапрещеноДвижение (state, direction)) [21].

Онтологии часто содержат большое число утверждений об индивидуальных объектах (ABox, Assertions box). Отмечают проблемы быстродействия «традиционных машин логического вывода» на таких онтологиях [21] («в промышленных применениях»).

Содержимым онтологии является структура «описания типов объектов, их свойств и связей объектов», а в правилах обработки (информационных объектов) элементы описаний ссылаются на свойства онтологии ПрОбл [21].

К сожалению, многие разработчики отдаляются от понимания онтологии, заложенного основателями этого понятия [139,140] и отождествляют онтологию со знаниями. («Использование онтологических моделей в программном обеспечении, в первом приближении кажется IT-специалистам похожим на подключение нового типа хранилища данных, с которым можно работать, не меняя принципов создания самого программного кода» [21].)

Требования 3 и 5 (инструменты экспертов для формирования и сопровождения знаний**) реализуются** через разделение компетенций разработчиков и возможность формирования знаний, не влияющих на работоспособность решателей. Для этого создаются методы, механизмы, инструменты.

В комплексе АТ-ТЕХНОЛОГИЯ есть: типовые проектные процедуры (ТПП) извлечения знаний (из эксперта; из проблемно-ориентированных текстов и из БД); ТПП конфигурирования компонентов ИЭС; ТПП проектирования БД [86,87,89].

Средства создания хранилищ знаний по стандартам RDF/OWL позволяют создавать разнообразные запросы: стандартизированные (например, выведение факта о том, что объект входит в надкласс, если известно, что он входит в подкласс), или определяемые автором через ПИФ (в нотациях SWRL и SPIN) [21].

Формируются графовые БД, которые хранят триплеты; с графовыми СУБД работают машины логического вывода [21]. Программный продукт АрхиГраф.СУЗ [21] позволяет создавать правила логического вывода**. (**При логическом выводе «правила могут каскадироваться, то есть факты, полученные в качестве вывода одного правила, могут выступать в роли условий для следующего») [21].

Опробованы возможности автоматизированного формирования моделей предметной области на основе визуальных когнитивных моделей (FreeMind – для предметных областей, связанных с материалами, такими как легированная сталь, наблюдаемой кинетикой, деградационными процессами) [92]. Сервис (web) для автоматизированного формирования продукционных баз знаний (на языке CLIPS) загружает визуальные модели и преобразует их во внутренний универсальный формат [114].

О методах и механизмах для реализации этих требований (3 и 5) можно добавить следующее.

Часто разработчики в качестве технологической основы автоматизированных систем выбирают онтологии и машины логического вывода [21]. Машины логического вывода работают с файлами, содержащими онтологические модели, при помощи специального API.

«Основанные на онтологиях системы поддержки принятия оперативных решений» приводят собранную из разнородных источников информацию в соответствие структуре онтологической модели; далее применяют к полученной информации правила логического вывода для вычисления следствий из имеющихся предпосылок; и предоставляют результаты обработки [21].

Онтологическими правилами логического вывода связываются последствия, риски и опасности кризисных событий исследуемого мира с субъектами, ресурсами и процедурами исправления (в примере использования онтологических правил для территориального ситуационного центра с понятиями Crisis(?xc), Danger(?xr), Effect(?xe), Procedure(?xp) и действующим лицом Actor(?xa)) *посылки правил* – формализованное описа-

ние контекста происшествия и его сути: если эффект вызывает опасность (causes(?xe, ?xr)), если кризис вызывает эффект (induces(?xc, ?xe)), если ДЛ предполагает процедуру (assumes(?xa, ?xp)), если процедура снижает опасность reduces(?xp,?xr)) [169].

Проведенные и описанные в литературе эксперименты показывают, что RDF/RDFS/OWL подход применим для задач, где обозреваемой информации не слишком много (определение виновника дорожно-транспортного происшествия, определения диапазона рекомендуемых (или, наоборот, не рекомендуемых) действий, прогнозирование возможных последствий тех или иных вариантов решений). Пример использования онтологии на железнодорожном транспорте: классификация существующих стандартов (для управления 58 стандартами для поездов и подвижного состава (железнодорожной отрасли Великобритании) и выявление пробелов. Если база знаний о британских обязательных стандартах сделать доступной через Интернет как часть семантической сети, это позволит их интерпретировать [154].

Специализированные оболочки позволяют создавать модель знаний в структуре, учитывающей традиции специалистов ПрОбл. Недостатком специализированных оболочек является и ограничения области использования. Такой путь позволяет разработать систему узкого спектра (примеры: диагностика по неонатологии, заболеваниям щитовидной железы, нефрологическим заболеваниям у детей, острым нарушениям мозгового кровообращения или вовсе – диагностика анемии по лабораторным данным [53]). Предлагаемые узко специализированные решения: проблемно-ориентированные средства автоматизации (*технология разработки диагностических систем* [38] или *управление ресурсами* [45] или проблемно-предметно-ориентированные средства автоматизации, например, только диагностика и только в медицине (Doknosis.org).

Отделение декларативных знаний от процедурных – главное, что дает возможность формирования и сопровождения знаний, не влияющих на работоспособность решателей. В ряде технологий при некоторых изменениях модели автоматически меняются и алгоритмы, например, при переносе класса из одного надкласса в другой могут измениться правила обработки объектов этого класса, если они отличаются для двух надклассов.

В рамках продукционного подхода иногда тоже говорят об отделении процедурных и декларативных знаний. Но модель предметной области и модель процессов обработки информации у них технологически неразрывны: содержимым онтологической модели

могут являться не только правила взаимодействия объектов разных типов между собой (в рамках имитационной модели), правила обработки информационных объектов различных типов и формализованные представления расчетных алгоритмов, описания пользовательских интерфейсов, но и правила и регламенты обмена данными, правила формирования информационных сообщений и многое другое [21].

Сервис для формирования баз знаний CLIPS генерирует на основе продукций код для целевой экспертной системы) [114].

В последние годы число систем с продукционной моделью представления знаний снизилось, уступая место системам с другими моделями представления знаний. Тренд на уменьшение продукций можно объяснить расширением доступного инструментария, поддерживающего другие модели представления, наиболее адекватные предметным областям и задачам, для которых продукционное представление не является принятым и удобным в предметной области

А с другой стороны – при использовании универсальных языков «граница между кодом и данными» начала смещаться («по мере развития различных платформ для создания приложений, баз данных и промежуточного программного обеспечения»). «Исполняемый код многих приложений «представляет собой сложно организованную иерархию специализированных и достаточно абстрактных алгоритмов, порядок применения которых в значительной мере определяется настройками, то есть, в сущности, данными» [21].

Требования 5 и 6 (сопровождение и до-обучение баз знаний) реализация механизмов оценки баз знаний, сформированных вручную, их обучения и до-обучения**)** реализуются через автоматизацию процесса управления знаниями. Автоматизированная (экспертная) система применяет базу знаний «правильно» (правильно использует правильный алгоритм решения задачи) и может провести полный анализ любого множества гипотез. Однако результаты такого анализа в значительной степени зависят от качества применяемых в этом анализе знаний.

Методы до-обучения систем (новыми знаниями) существуют, проработаны различные механизмы обучения и пополнения знаний, в том числе индуктивные методы [10,14,15,97], и продолжают создаваться [124, 157]. Процесс управления знаниями

автоматизируется [114], Методология построения систем управления БЗ для интеллектуальных систем включает средства «для индуктивной составляющей» [6].

В инструментальном комплексе АТ-ТЕХНОЛОГИЯ реализованы: типовые проектные процедуры (ТПП) для извлечения знаний из БД и для извлечения знаний из проблемно-ориентированных текстов [87,89].

Однако применение методов формирования знаний (часто называемое data mining или knowledge discovery) пока не стало промышленной технологией разработки баз знаний [17]. Для того чтобы эти методы стали элементами технологии интеллектуальных систем, необходимо «обеспечить механизм сопряжения независимо созданных баз данных, имеющих различные схемы, с базами знаний интеллектуальных систем; установить соответствие между набором полей базы данных и множеством элементов декларативной базы знаний; выполнить преобразование результата работы алгоритма обучения в способ представления, поддерживаемый программными средствами интеллектуальной системы» [17].

Но использование экспертов для повышения качества БЗ продолжает оставаться популярным подходом [50]. Получение экспертных оценок группой специалистов применяется также при оценивании качества баз знаний во многих современных подходах: в подходах, основанных на положениях теории нечетких множеств с моделями аддитивной и мультипликативной свертки, по локальным критериям, при этом требуется сближение мнений экспертов и корректировка результатов экспертизы, и др. [5].

Система управления знаниями «АрхиГраф.СУЗ» позволяет исследовать онтологии, созданной в редакторе Onto.pro, при помощи поисковых запросов, выявлять связи между узлами графа знаний [21].

Для локализации ошибок в БЗ используют тестирование (на заданном множестве тестовых эталонных данных с правильными решениями [44]. С давних пор известны примеры программного инструментария для обслуживания баз эталонных данных (пополнения и модифицирования) и ее использования [85, 93].

Рекомендациями по верификации и отладке БЗ являются: использование встроенных команд верификации и отладки, использование вопросов из тестового сборника вопросов. Принципами и подходами для выявления избыточности и

синонимии, оптимизации набора понятий являются: противопоставление понятий и терминов, выявление потенциальных синонимов в базе знаний, анализ логической схемы или иерархии понятий.

К методам оценки качества баз знаний относятся следующие: методы проверки связности различных онтологических уровней базы знаний, методы оценки числа пар потенциальных синонимов в базе знаний [50].

Изредка встречаются описания процесса управления базами знаний в процессе их эксплуатации со сбором со всех рабочих мест специалистов информации о принятых с помощью ЭС решений («протоколов» с клиническими данными терапии) и передача их «анализатору», чтобы осуществлять по необходимости коррекцию знаний [49].

Но если получаемые новые версии баз знаний выражаются на продукционном языке, то они либо не удобны для оценивания экспертами, либо требуют перепрограммирования компонентов системы.

Для Требования 6 (механизмы оценки и обучения баз знаний) важна также единая терминологическая основа для представления знаний и данных, описания случаев из действительности и прочих документов. Сложные предметный области требуют введения системы синонимов, чтобы интегрировать усилия и результаты работы сообщества специалистов.

«Эксперты в разных областях могут отсылать к одному и тому же понятию и понимать его различным образом. Проблемы возникают и тогда, когда одно и то же слово используется для обозначения различных понятий в каждом отдельном поле» [116,145].

Часто строят тезаурусы. Нередко тезаурусы представляют только иерархические отношения терминов. Однако многие дают информацию об ассоциативных связях терминов, отношениях синонимии (Проекты TOVE (Toronto Virtual Enterprise), Enterprise Project, базы Medical Subject Headings (MeSH), WordNet, РуТез, ProThes). Многие онтологии строятся как дерево понятий, «растущее» от абстрактных ко все более конкретным сущностям (RDF, OWL) [71,74,125]. Иногда разработчики классификатор терминов своей области называют онтологией (приравнивают к ней) [8].

Требование 7 (интегрированность) – через разработку экспортируемых форматов обрабатываемой информации и выбор распространенного языка кодирования программных компонентов, через явную доступность онтологий в качестве информацион-

ного ресурса в процессе разработки и выполнения («явная доступность» онтологий в качестве информационного ресурса является основным элементом интеграции информации, «основанной на посредничестве» [161]).

Проблемная область, представленная с помощью Protégé, экспортируется в формат ИС CLIPS. Базы знаний Protege выгружаются в rdf-формат и совместимы со многими языками, особенно ОО. OSTIS-редактор информации экспортирует sc.g-конструкции в разные форматы изображений. В IACPaaS внешний формат ресурсов – json, язык java.

Каждое из известных универсальных технологий и инструментов для создания ИС – OSTIS, Protege, OSTIS, G2, CLIPS, AT-технология, Ontology-Driven Software development или модельно-управляемый подход, Collaborative Device Modeling Environment (DARPA) – отвечает нескольким из этих требований, но ни одно не поддерживает всех указанных требований одновременно.

Из-за дополнительного архитектурного компонента этих систем (базы знаний) команда разработчиков пополняется экспертами ПрОбл и когнитологами.

Для обеспечения возможности согласованному коллективу экспертов постоянно усовершенствовать БЗ внедренным технологиям не хватает либо понятности представления и отделения знаний от данных, а декларативных знаний от процедурных знаний или декларативного представления причинно-следственных связей, а не только классификации сущностей, либо доступности инструментов коллективной разработки, либо интегрированности со средствами обучения.

Т.к. команда разработчиков имеет разных специалистов (аналитик, эксперт, когнитолог, проектировщик, программист, группа качества) проблема сопровождаемости таких систем стоит особо остро. Это накладывает необходимость использования дополнительных специализированных механизмов обеспечения их сопровождаемости и жизнеспособности.

1.5. Выводы к главе 1

Несмотря на современные достижения в программной и инженерии и инженерии знаний обострилась необходимость разработки методов, моделей и технологии для обеспечения жизнеспособности интеллектуальных систем, решающими сложные и от-

ветственные задачи, которые могут продолжительное время быть полезными специалистам.

Однако не предложено общепризнанной стройной классификации известных задач интеллектуальной деятельности, и нет формальной постановки всех известных экспертных задач в единой терминологии (важной для перехода от искусства разработки систем, основанных на знаниях, к технологии). Предложенные универсальные технологии создания систем с базами знаний сосредоточены на методах разработки и не предлагают инструменты развития, хотя изменчивость знаний типична для многих предметных областей. Внедряемые интеллектуальные системы должны обладать принципиально важной способностью адаптироваться во времени к таким изменениям.

Для создания жизнеспособных программных систем поддержки специалистов, решающих важные для науки и практики интеллектуальные задачи, необходима современная методология обеспечения их жизнеспособности. Поэтому актуальной является разработка моделей, методов и технологии создания эволюционирующих интеллектуальных программных систем с базами знаний, реализующих все требования к СБЗ, описанным выше.

Целью диссертационной работы является разработка моделей, методов и технологии создания интеллектуальных систем на основе онтологий с декларативным представлением баз знаний и механизмами эволюционирования.

ГЛАВА 2. Постановка интеллектуальных задач, решаемых на основе баз знаний, и анализ методов их решения

Приведенный в главе 1 обзор литературы показывает, что несмотря на немалое число исследований и обзорных работ по классификации задач, решаемых интеллектуальными системами, почти все их авторы основывают свои подходы к классификации на собственной интуиции, а задачи описывают в разных терминах, что затрудняет сравнение их содержания. Крайне редко в литературных источниках встречаются более или менее формальные постановки задач.

Для выработки новых и использования апробированных методов автоматизированного решения таких задач нужна классификация задач интеллектуальной деятельности и их единообразное определение. Для точного представления задач необходим математический аппарат и математические абстракции для всех содержательных понятий, используемых в постановках задач [61]. В этих терминах можно предложить постановки известных задач на разных уровнях абстракции. Формально представляя частные свойства предметных областей, можно предложить постановки новых задач, опирающихся на эти частные свойства. И получить т.о. многоуровневую классификацию интеллектуальных задач, открытую для расширения [61].

В главе 2 представлены новые постановки известных интеллектуальных задач и новая классификация задач, позволяющая добавлять и систематизировать новые постановки задач. Постановки задач связываются с онтологией знаний, а для реализации методов и алгоритмов их решения определяются единые вычислительные операции, опирающиеся на онтологию.

2.1 Формальное представление интеллектуальных задач

Будем называть задачей интеллектуальной деятельности задачу, в постановке которой в качестве входных данных и/или результатов присутствуют согласованные с онтологией (Ont) база знаний предметной области (Kn) и ситуация ПрОбл. Для краткости далее задачи интеллектуальной деятельности будем называть просто задачами.

2.1.1.Термины и абстракции для определения задач

Введем математические абстракции для основных терминов, используемых в инженерии знаний:

O – множество предметных символов, для каждого из которых указан тип (сорт) его возможных значений (термины, денотатами которых являются значения);

F – множество функциональных символов, для каждого из которых указаны сорта аргументов и значения функции (термины, денотатами которых являются функциональные соответствия),

Pr – множество предикатных символов, для каждого из которых указаны сорта аргументов предиката (термины, денотатами которых являются отношения).

Примечание. Функции и отношения, являющиеся интерпретациями (денотатами) функциональных и предикатных символов, могут зависеть от времени, координат пространства и других характеристик.

Математической абстракцией концептуализации предметной области (обычно называемой «онтологией», Ont) будем считать пару $<\Sigma, A_\Sigma>$ из сигнатуры $\Sigma = \{O, F, Pr\}$ и множества аксиом A_Σ, представляющих свойства терминов концептуализации (онтологии) и соглашения предметной области [61].

Математической абстракцией понятия формализованные знания (или «база знаний») будем считать непустое и непротиворечивое множество предложений Kn на логическом языке сигнатуры Σ, представляющих свойства и законы предметной области (ПрОбл). Формализованные знания Kn могут быть получены путем выявления закономерностей в базах фактов (т.е. индукции) или путем получения знаний от эксперта предметной области.

Математической абстракцией понятия «ситуация ПрОбл, согласованная с онтологией», будем считать алгебраическую систему AS сигнатуры Σ, относительно которой все предложения из множества A_Σ истинны. Примером ситуации может быть процесс или некоторая система («множество элементов, находящихся в отношениях и связях друг с другом, которое образует определённую целостность» [3]). Абстракция ситуации выражается на языке Σ с учетом соглашений предметной области A_Σ. Т.е. все аксиомы из множества A_Σ истинны относительно этой абстракции ситуации.

Имеющиеся к началу решения задачи сведения, характеризующие ситуацию (факты о существующей ситуации или условия существования новой ситуации) обозначим Sit.

2.1.2 Определения абстрактных задач

Рассмотрим разные классы задач и в зависимости от того, требуется ли найти формализованные знания (т.е. БЗ) или БЗ задана, можно различать *задачи индукции* и *задачи дедукции поддержки решения на основе баз знаний* (Рисунок 2.1).

Рассмотрим разные классы задач и в зависимости от того, требуется ли формализовать знания (т.е. создать БЗ на основе фактов) или БЗ задана, будем различать *задачи индукции баз знаний* и *задачи поддержки решения на основе баз знаний,* т.е. дедуктивные задачи (Рисунок 2.1).

Среди *задач дедукции* выделим задачи *поиска гипотез* и *критики гипотезы* [61].

Задачи поиска гипотез. В задачах поиска гипотез требуется найти все гипотезы, соответствующие Kn.

Дано: БЗ Kn, согласованная с A_Σ;

сведения, характеризующие ситуацию (Sit);

найти: {AS}, удовлетворяющие условию: все предложения из множества $A_\Sigma \cup$ Kn являются истинными относительно каждой из этих AS.

Комментарий: Sit = известные факты о существующей ситуации или ограничения на существование новой ситуации,

искомые AS являются математической абстракцией гипотез о вариантах полного определения ситуации (к какому классу принадлежит или к какому итогу приведет или какая причина создала ситуацию и т.д.).

Задачи критики гипотезы. В задаче критики гипотезы требуется проверить соответствие заданной гипотезы базе знаний.

Дано: AS, Kn (согласованные с A_Σ);

найти**:** те предложения из множества Kn, которые ложны относительно заданной AS_Σ (или показать, что таких предложений нет).

Задачи индукции (баз знаний). В задаче индукции знаний по обучающей выборке (заданному множеству задач с известными решениями) требуется сформировать такую БЗ, относительно которой все задачи обучающей выборки решаются правильно.

Дано: {AS} – множество абстракций полностью определенных ситуаций, согласованных с A_Σ;

найти: Kn, согласованную с A_Σ, такую, что относительно каждого «случая» (каждой задачи) из {AS} все предложения из Kn истинны;

комментарий: {AS} является математической абстракцией множества задач с известными решениями, т.е. абстракцией обучающей выборки.

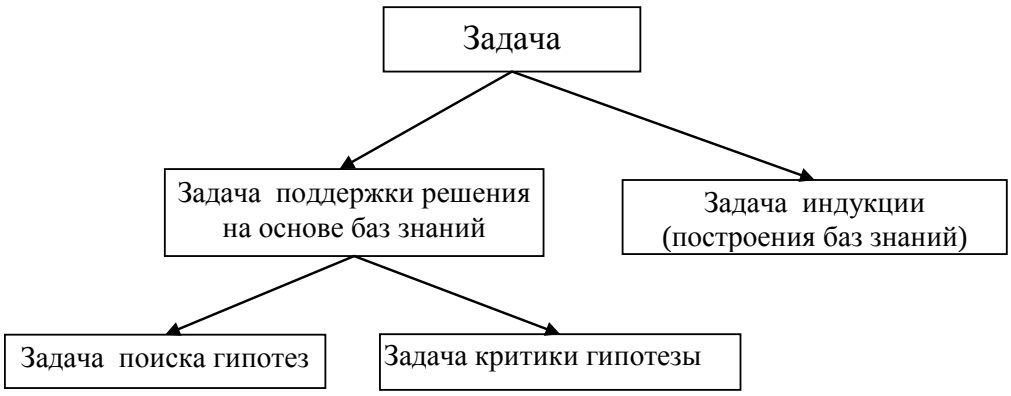


Рисунок 2-1 – Абстрактные задачи интеллектуальной деятельности

Обозначим R подмножество предметных символов $O_1 \subseteq O$ сигнатуры Σ вместе с их значениями (интерпретациями), и подмножества функциональных символов $F_1 \subseteq F$ и предикатных символов $Pr_1 \subseteq Pr$, для которых заданы их частичные (возможно, не на всей области их определения) интерпретации.

Обозначим AS (R) такую алгебраическую систему сигнатуры Σ, что предметные символы из O_1 имеют в AS (R) такую же интерпретацию, что и в R, а интерпретация функциональных символов из F_1 и предикатных символов из Pr_1 в AS (R) является расширением их интерпретации в R, и относительно которой все предложения из множества A_Σ истинны.

Будем считать R математической абстракцией результатов наблюдения ситуации, согласованных с A_Σ, если существует AS (R); при этом AS (R) может рассматриваться в качестве математической абстракции объяснения результатов наблюдения ситуации.

Будем считать Cnd математической абстракцией условий на результат решения задачи, согласованных с A_Σ, такое множество предложений на логическом языке сигнатуры Σ, что множество предложений $A_\Sigma \cup$ Cnd непротиворечиво.

AS (Cnd) может рассматриваться в качестве математической абстракции объяснения проекта (плана), удовлетворяющего Cnd.

Т.е. AS(R) и AS (Cnd) – математическая абстракция полного определения ситуации: согласованных с БЗ объяснений результатов наблюдения действительности или согласованных с БЗ объяснений результата решения – предлагаемого проекта (плана).

В зависимости от того, заданы ли в качестве входных данных результаты наблюдения ситуации R или условия на решение задачи Cnd, можно различать задачи анализа результатов наблюдений и задачи анализа условий на решения (Рисунок 2.2).

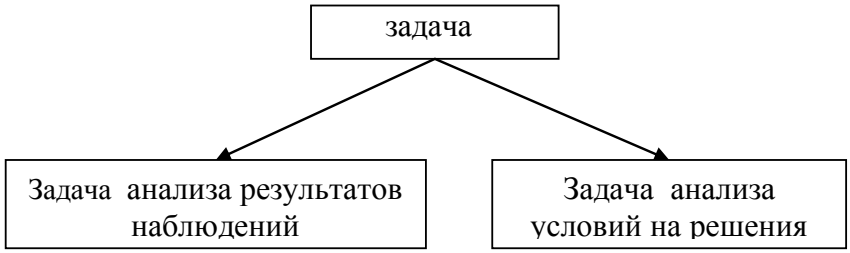


Рисунок 2.2 – Задачи анализа результатов наблюдений и анализа условий на решения

2.1.3. Задачи анализа результатов наблюдений и анализа условий на решения

Задачи, рассмотренные выше, являются слишком абстрактными и, поэтому, мало реалистичными.

Более полезными являются следующие пять групп задач, получаемые путем комбинации вышеописанных абстрактных категорий [61].

Задачи поиска гипотез, объясняющих результаты наблюдений. *Задача поиска гипотез, объясняющих результаты наблюдений*, рассматривается как комбинация *задачи поиска гипотез* и *задачи анализа результатов наблюдений*, в которой кроме БЗ

заданы и результаты наблюдений ситуации: требуется найти все гипотезы, соответствующие результатам наблюдений и формализованным знаниям (базе знаний).

Дано: Kn, согласованная с A_Σ,

R, согласованная с A_Σ;

найти: все такие AS (R), что все предложения из множества $A_\Sigma \cup$ Kn являются истинными относительно каждой из этих AS (R).

Задачи проектирования. *Задача проектирования* рассматривается как комбинация *задачи поиска гипотез* и *задачи анализа условий на решение*, в которой кроме базы знаний, задано непустое множество условий на результат решения задачи: требуется найти все проекты, соответствующие БЗ и условиям на решение задачи.

Дано: Kn, согласованная с A_Σ,

непустое Cnd, при условии: $A_\Sigma \cup$ Kn $\cup$ Cnd непротиворечиво;

найти: все такие {AS}, относительно которых все предложения из $A_\Sigma \cup$ Kn $\cup$ Cnd являются истинными.

Задачи критики объяснения результатов наблюдений. *Задача критики объяснения результатов наблюдений* рассматривается как комбинация задачи *критики гипотезы* и *задачи анализа результатов наблюдений*, в которой кроме базы знаний даны результаты наблюдения ситуации и их объяснение: требуется установить несоответствие объяснения результатов наблюдения базе знаний или подтвердить их соответствие.

Дано: Kn, согласованная с A_Σ,

R, согласованная с A_Σ;

AS (R), согласованное с A_Σ;

найти: те предложения из множества Kn, которые ложны относительно AS (R), или показать, что таких предложений нет.

Задачи критики проекта. *Задача критики проекта* рассматривается как комбинация задачи *критики гипотезы* и *задачи анализа условий на решение*, в которой требуется проверить соответствие проекта базе знаний и условиям, которым он должен удовлетворять.

Дано: Kn, согласованная с A_Σ,

непустое Cnd, при условии: $A_\Sigma \cup Kn \cup Cnd$ непротиворечиво;

AS, согласованное с A_Σ;

найти: те предложения из множества $Kn \cup Cnd$, которые ложны относительно заданной AS (или показать, что таких предложений нет).

Задачи поиска БЗ по обучающей выборке ситуаций, представленных результатами наблюдений. *Задача поиска БЗ по обучающей выборке ситуаций, представленных результатами наблюдений*, рассматривается как комбинация *задачи индукции* и *задачи анализа результатов наблюдений*, в которой каждый элемент обучающей выборки представляет собой результаты наблюдения некоторой ситуации: по такой обучающей выборке требуется сформировать такую БЗ, согласованную с онтологией, что по каждому элементу обучающей выборки может быть построена модель ситуации, согласованная с онтологией и результатами наблюдений, которая соответствует БЗ [56].

Дано: обучающая выборка {R} – множество результатов наблюдений некоторой ситуации, согласованными с A_Σ;

найти: Kn, согласованную с A_Σ, такую, что для каждого R существует AS(R), относительно которой все предложения из Kn истинны.

2.1.4. Задачи, связанные с классификацией

Во многих предметных областях существуют классификации ситуаций и решаются задачи, учитывающие эти классификации. В таких задачах в сигнатуру Σ входит предметный символ «класс», область возможных значений которого состоит из конечного множества значений $\{class_1, \ldots class_i, \ldots, class_n\}$,

а в БЗ Kn входят предложения о свойствах ситуаций каждого класса.

Поставленные выше группы задач имеют свои уточнения; возникает также новая задача, учитывающая классификацию ситуаций (Рисунок 2.3).

Задачи формирования знаний о классах по обучающей выборке. *Задача формирования знаний о классах по обучающей выборке ситуаций, представленных результатами наблюдений*, рассматривается как уточнение *задачи поиска БЗ по*

обучающей выборке ситуаций, представленных результатами наблюдений, в которой элементами обучающей выборки являются не только результаты наблюдения различных ситуаций, но и класс, к которому относится каждая ситуация.

Дано: обучающая выборка {<R, *класс* = $class_i$>}, каждый из элементов которой представлен парой, состоящей из результатов наблюдений некоторой ситуации и известным классом этой ситуации;

найти: Kn, такую, что для каждого элемента обучающей выборки <R, *класс* = $class_i$> существует AS (<R, *класс* = $class_i$>), относительно которой все предложения из Kn истинны.

Задачи распознавания. *Задача распознавания* рассматривается как уточнение задачи *поиска гипотез*, объясняющих результаты наблюдений, в которой требуется найти все гипотезы о классе ситуации, описанной результатами наблюдений.

Дано: Kn и R, согласованные с A_Σ;

найти: множество $H_{R,Kn} = \{class_{i1}, ..., class_{im}\}$, состоящее из всех таких значений, что существуют AS(<R, *класс* = $class_{i1}$>), ..., AS(<R, *класс* = $class_{im}$>), относительно каждой из которых все предложения из множества $A_\Sigma \cup$ Kn являются истинными [63].

Комментарий: AS (<R, *класс* = $class_i$>)} являются математическими абстракциями объяснения принадлежности к различным классам элементов наблюдений R.

Будем говорить, что БЗ Kn удовлетворяет условию разделимости классов, если для любой пары <R, *класс* = $class_i$>, для которой существует AS (<R, *класс* = $class_i$>), относительно которой все предложения из Kn истинны, и в Kn существуют ложные предложения относительно любой AS (<R, *класс* = $class_j$>), при любом $j \neq i$.

Математической абстракцией запроса дополнительной информации, обозначаемого далее Q, для результатов наблюдения R, согласованных с A_Σ, является: либо предметный символ $o \in O \setminus O_1$; либо терм $f(c_1, ..., c_k)$, либо формула $p(c_1, ..., c_k)$, где $c_1, ..., c_k$ – константы, f – функциональный символ сигнатуры Σ, а p – предикатный символ сигнатуры Σ [65].

Математической абстракцией ответа на запрос Q, обозначаемый далее AQ, является либо значение предметного символа o, либо значение терма $f(c_1, ..., c_k)$, либо значение формулы $p(c_1, ..., c_k)$, такие, что существует такая AS (<R, AQ>), что ни одно из предложений A_Σ не является ложным относительно AS(<R, AQ>).

Задачи запроса дополнительной информации для распознавания. Задачей, помогающей *поиску гипотез о классе*, является *задача запроса дополнительной информации для распознавания*, в которой по результатам наблюдений ситуации, для которых существует более одной гипотезы о классе, требуется предложить дополнительное наблюдение ситуации, которое гарантированно позволит сократить множество гипотез [65].

Дано: БЗ Kn, удовлетворяющая условию разделимости классов,

R, такое, что для него множество $H_{R,Kn}$ имеет мощность не меньше двух;

найти: такой запрос *Q* дополнительной информации для результатов R, согласованных с A_Σ, что если R' есть R, объединенное с *AQ*, то множество $H_{R',Kn}$ имеет меньшую мощность, чем множество $H_{R,Kn}$ [63].

Задачи проектирования систем заданного класса. В *задаче проектирования систем заданного класса*, являющейся уточнением *задачи проектирования*, требуется по базе знаний и множеству условий на результат решения найти проект системы заданного класса.

Дано: БЗ Kn, значение $класс_m$ предметного символа «класс», условия на результат Cnd;

найти: все такие {AS (<*класс* = $класс_m$>)}, относительно которых все предложения предложения из $A_\Sigma \cup Kn \cup Cnd$ являются истинными;

комментарий: найденные {AS (<*класс* = $класс_m$>)} являются математической абстракцией множества проектов заданного класса $класс_m$.

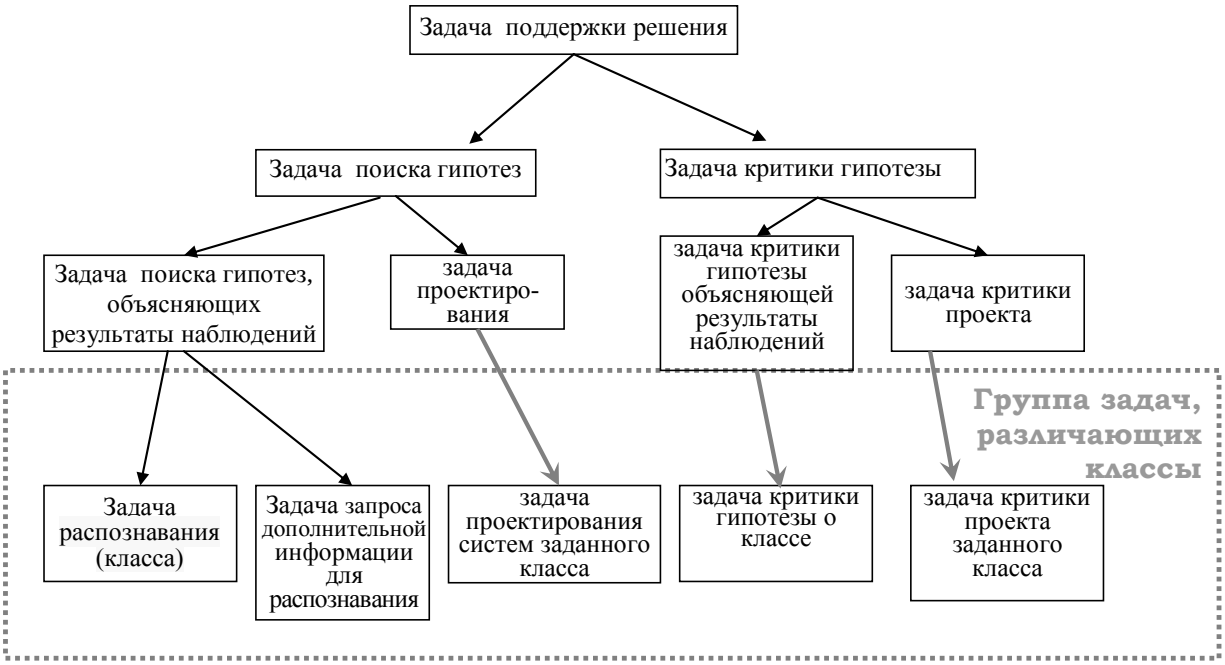


Рисунок 2.3 – Варианты задач, учитывающих классификации ситуаций или систем

Задачи критики гипотезы о классе. *Задача критики гипотезы о классе* рассматривается как уточнение задачи *критики объяснения результатов наблюдений*. В ней требуется проверить соответствие заданной гипотезы о принадлежности ситуации заданному классу базе знаний.

Дано: Kn и R, согласованные с $A_{\square}$, значение $class_i$ предметного символа «класс», AS (<R, *класс* = $class_{ii}$>);

найти: те предложения из множества Kn, которые ложны относительно AS (<R, *класс* = $class_{ii}$>), или показать, что таких предложений нет;

комментарий: AS (<R, *класс* = $class_{ii}$>) является математической абстракцией объяснения принадлежности результатов наблюдения R классу $class_{ii}$ [63].

Задачи критики проекта заданного класса. *Задача критики проекта заданного класса* рассматривается как уточнение задачи критики проекта произвольной системы через указание некоторого класса проектируемых систем.

Дано: Kn, Cnd, согласованные с A_{Σ}, значение $класс_i$ предметного символа «класс», AS (<*класс* = $класс_i$>);

найти: те предложения из множества Kn $\cup$ Cnd, которые ложны относительно заданной AS (<*класс* = $класс_i$>), или показать, что таких предложений нет [63].

2.1.5. Задачи для систем, состоящих из компонентов

Во многих предметных областях анализируются или проектируются системы, состоящие из компонентов, и решаются задачи, учитывающие эти компоненты и отношения между ними (в том числе пространственные) [105]. В таких задачах в сигнатуре S присутствуют символы для обозначения *компонентов* и *отношений* (пространственных и других) между ними; A_Σ описывает множества таких свойств компонентов и отношений, которые определяются соглашениями специалистов предметной области. В базе знаний Kn и в условиях на результат решения задачи Cnd присутствуют предложения о свойствах таких символов, получаемые индуктивным или дедуктивным путем.

Уточнением задач проектирования для этого случая являются задачи конфигурирования (или сборки). В постановке задачи уточняется условие Cnd перечислением компонентов и их свойств, и {AS} описывает множество проектов (спецификаций) конфигураций создаваемой системы (описывает с помощью символов, обозначающих компоненты и отношения).

Аналогичные уточнения имеют место и для задач поиска гипотез, объясняющих результаты наблюдений, критики объяснения результатов наблюдений и критики проекта.

2.1.6.Задачи, в которых существенную роль играет время

Некоторые задачи решаются на моделях, использующих функции и отношения, зависящие от времени. В таких задачах в сигнатуру Σ входят функциональные и предикатные символы, зависящие от *времени.* Эти символы являются математической абстракцией свойств (атрибутов) динамической системы или ситуации, зависящих от времени (одним из которых часто является *состояние*).

Множества таких функциональных и предикатных символов обозначим F_t и Pr_t. Пусть $\{r_{t1}, .. r_{tm}\}$ – подмножество символов из $F_t \cup Pr_t$, для которых заданы их интерпретации в некоторые моменты времени.

Функциональный терм (атомную формулу) с таким символом ($r_{tj} \subseteq F_t \cup Pr_t$), одним из аргументов которой является момент t, обозначим через r_{tj} (…, t, …). Функциональные термы (атомные формулы) символа r_{tj}, аргумент которых, соответствующий моментам времени, имеет значения t_0, …, t_k, обозначим через $\{r_{tj}$(…,

t_0, …), …, r_{tj} (…, t_k, …)}. Если функциональные термы (атомные формулы) {r_{t1} (…,t_0,…), … r_{t1} (…,t_k ,…),.. r_{tm}(…,t_0,…), … r_{t1} (…,t_k ,…)} вместе с их значениями входят в R_Σ, то в этом случае вместо R_Σ будем использовать обозначение $R_\Sigma(t_0, ..., t_k)$. Функциональный терм (атомную формулу) символа q’ с аргументом t’, соответствующим моменту времени, (в который значение терма или формулы неизвестно) обозначим через q’(…, t’, …).

Постановки задач поиска гипотез, объясняющих результаты наблюдений, критики объяснения результатов наблюдений и др. для динамической ситуации не меняются, но уточняются за счет того, что интерпретация некоторых символов зависит от времени [105]. В предметных областях, различающих классы ситуаций, значения которых зависят от времени, такими уточняющими задачами становятся задача распознавания класса ситуации, описанной результатами динамических наблюдений, и задача запроса дополнительной информации для распознавания класса динамической ситуации (Рисунок 2.4).

Задачи проектирования (и *критики проекта*) могут ставиться для динамических систем (а не только для статических систем). Среди символов сигнатуры для проектов динамических систем должны быть функциональные и/или предикатные символы, которые зависят от времени. Эти символы используются при описании динамических свойств компонентов в базе знаний Kn и при задании условий Cnd на конкретную проектируемую систему.

Задачи прогноза. *Задача прогноза* – уточнение *задачи поиска гипотез, объясняющих результаты наблюдений*, в которой по значениям функциональных термов (атомных формул), зависящих от времени, заданным в некоторые моменты времени t_0, … t_k (обычно – по результатам наблюдений за динамической системой) требуется определить значения этого или другого функционального терма (атомной формулы) для заданного момента времени t’, отличного от t_0, …, t_k («до или после текущего момента времени *t*» [68], чаще – для будущего момента) при заданных значениях остальных аргументов (если они есть).

Дано: Kn, согласованная с $A_{\square}$; упорядоченное множество моментов времени {t_0, …, t_k}; R (t_0, ..., t_k); t’ ∉ {t_0, …t_k}; q’ ∈ $F_t \cup Pr_t$;

найти: все такие значения q’(…, t’,…), для каждого из которых существует такая AS (<R (t_0, ..., t_k), q’(…, t’,…)>), что все предложения из Kn истинны относительно нее;

комментарий: если t’ > t_k , то речь идет о прогнозе на будущее; если же t’ < t_k, то речь идет о «ретроспективном» прогнозе [63].

Если в предметной области динамике рассматриваемых ситуаций соответствует конечное множество состояний {$state_1$,… $state_i$, …, $state_n$ } [72], то функциональный символ «состояние», зависящий от времени, начинает «играть роль» предметного символа «класс». При этом частным случаем *задачи прогноза* становится задача *прогноза состояния динамической системы*; в ней требуется найти значение функционального символа «состояние» в интересующий момент времени [54].

Наряду с *задачей распознавания состояния системы*, которая может быть рассмотрена для динамической системы, имеет место задача *распознавания* в реальном времени такого *момента*, когда ситуация требует принятия некоторых мер. Такое состояние системы иногда называют «критическим», отличным от нейтрального, а задачу распознавания такого состояния – *задачей мониторинга*. В этом случае сигнатуру должен входить некоторый выделенный предикатный символ *критическое состояние – criticalState.*

Задачи мониторинга. В *задаче мониторинга* требуется по результатам наблюдения динамической системы определить, является ли состояние системы критическим.

Дано: Kn, согласованная с A_Σ; R(t0, ..., t_k);

найти: такое значение *criticalState* (t_k), для которого существует такая AS_Σ (<R (t_0, ..., t_k), *criticalState* (t_k)>), что все предложения из Kn истинны относительно нее [63].

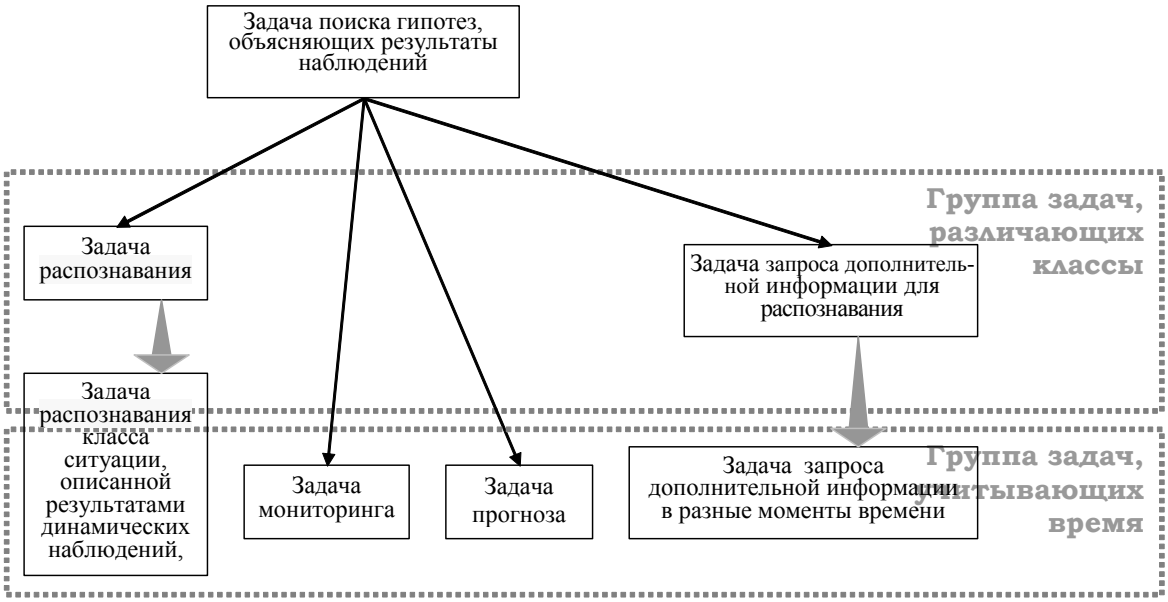


Рисунок 2.4 – Схема изменения спектра возможных задач в зависимости от характеристик предметных областей

Для динамической системы уточнением задачи поиска БЗ по обучающей выборке ситуаций, представленных результатами наблюдений является *задача поиска БЗ по обучающей выборке ситуаций, представленных результатами наблюдений* динамической системы, постановка которой получается заменой {R} на {R (t_0, ..., t_k)).

2.1.7. Задачи, связанные с планированием действий

Во многих предметных областях решаются задачи, рассматривающие действия (упорядоченные хотя бы частично), приводящие к некоторой заданной цели. В таких задачах в сигнатуру Σ входят символы для обозначения действий (из конечного множества названий *действий* в рамках предметной области) [105].

Обозначим D множество *действий*, которые можно выполнять над системой. Выполнение действия изменяет текущее состояние системы, вызывая тем самым переход ее в новое *состояние*. Обозначим S – множество возможных состояний системы, над которой выполняются действия из D. Действие $d \in D$ есть функция, отображающая S в S. Каждое *действие* характеризуется *предусловием* (условием на состояние, в котором его можно применять) и *постусловием* (условием на состояние, возникающее после выполнения действия). Обозначим $\varphi^j_{pred}(s)$ – предусловие для действия d^j, а $\varphi^j_{post}(s)$ – постусловие, соответственно, (аргумент s обозначает состояние, в котором вычисляется значение

соответствующей формулы). В базе знаний каждое действие d^j *представлено парой* $<\varphi^j_{pred}, \varphi^j_{post}>$.

План связывает *действия* с *состояниями*, начиная от *начального* и завершая *целевым* (конечным) состоянием. При планировании «с классическими допущениями», когда мир полностью наблюдаем и статичен, а действия детерминированы, дискретны и недопустимо их параллельное исполнение, (линейный) план представляет собой последовательность действий, которая позволяет достичь цели из начального состояния [10; 84].

Будем называть *линейным планом* кортеж $<s_0, d_1, d_2, ..., d_{n-1}, d_n>$, для которого существует такой кортеж состояний $<s_0, s_1, s_2, ..., s_{n-1}, s_n>$ и AS ($<s_0, s_1, s_2, ..., s_{n-1}, s_n>$), относительно которой справедливы предложения:

$\varphi^1_{pred}(s_0), \varphi^1_{post}(s_1), \varphi^2_{pred}(s_1), \varphi^2_{post}(s_2), ..., \varphi^n_{pred}(s_{n-1}), \varphi^n_{post}(s_n)$.

Кортеж $<s_0, d_1, d_2, ..., d_{n-1}, d_n>$ является математической абстракцией последовательности действий, ведущих к переходу через множество состояний, начиная от начального и завершая целевым (конечным) состоянием.

Задачи линейного планирования. В *задаче линейного планирования* требуется определить множество *линейных планов*, выполнение которых приводит к достижению *целевого состояния*, удовлетворяющего заданному *условию*, начиная от заданного *начального состояния*. *Задача планирования* – расширение *задачи проектирования*, точнее – *проектирования систем заданного класса* (класса «последовательность действий») (Рисунок 2-5).

Дано: $D \subseteq F$; $\{<\varphi^j_{pred}(s) \Rightarrow \varphi^j_{post}(d_j(s))> | d_j \in D\} \subseteq Kn$, $s_0 \in S$; $\varphi_{fin}(s) \subseteq Cnd$;

найти: линейный план $<s_0, d_1, d_2, ..., d_{n-1}, d_n>$, для которого существует АС $AS_\Sigma(<s_0, s_1, s_2, ..., s_{n-1}, s_n>)$, относительно которой справедливы все предложения из Kn, предложение $\varphi_{fin}(s_n)$, а также все другие предложения из Cnd [63].

Постановка *задачи планирования* усложняется в случае невыполнения «классических допущений». Будем называть планом с параллельным выполнением действий ориентированный граф G, метками вершин которого являются состояния из множества S, а метками дуг – действия из множества D; у которого одна начальная вершина, в которую не входит ни одной дуги, одна конечная вершина, из которой не выходит ни одной дуги, и любая вершина этого графа лежит на некотором пути из начальной вершины в конеч-

ную; для которого существует AS (<G>), относительно которой: если в этом графе из вершины с меткой s выходят дуги с метками d_1, ..., d_n, то справедливы предложения $\varphi^1_{pred}(s)$, $\varphi^2_{pred}(s)$, ..., $\varphi^n_{pred}(s)$, если в этом графе из вершины с меткой s_1 выходит дуга с меткой d, входящая в вершину с меткой s_2, то справедливы предложения $\varphi^d_{pred}(s_1)$ и $\varphi^d_{post}(s_2)$, а если в этом графе в вершину с меткой s входят дуги с метками d_1, ..., d_n, то справедливы предложения $\varphi^1_{post}(s)$, $\varphi^2_{post}(s)$, ..., $\varphi^n_{post}(s)$.

Задачи планирования с параллельным выполнением действий.

Дано: $D \subseteq F$; $\{<\varphi^j_{pred}(s) \Rightarrow \varphi^j_{post}(d_j(s))> | \ d_j \in D\} \subseteq Kn$; $s_0 \in S$; $\varphi_{fin}(s) \subseteq Cnd$;

найти: план с параллельным выполнением действий G, у которого начальная вершина имеет метку s_0 и для которого существует AS (<G>), относительно которой справедливы все предложения из Kn, и если конечная вершина имеет метку s_n, то справедливо предложение $\varphi_{fin}(s_n)$, а также все другие предложения из Cnd [63].

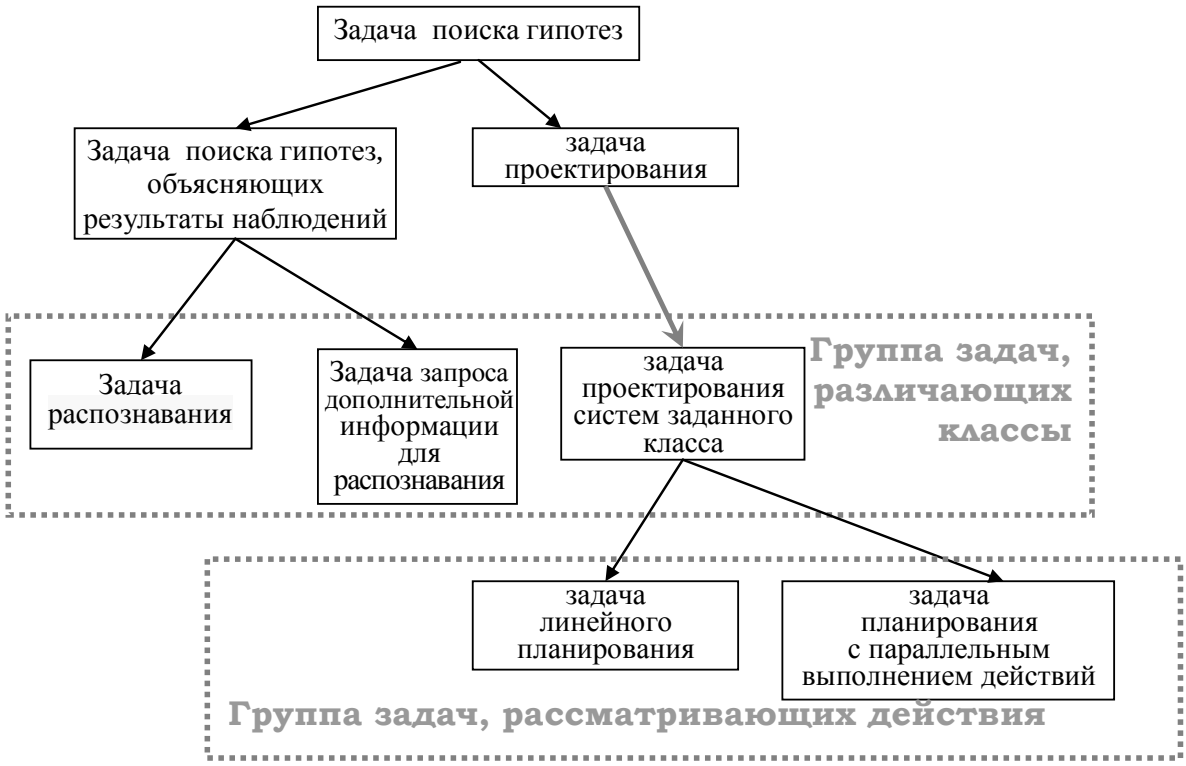


Рисунок 2-5 – Схема возможных задач поиска гипотез, включающая задачи планирования

В предметных областях, рассматривающих планирование, также уточняется и *задача критики плана*.

2.1.8. Группа задач, связанных с причинно-следственными отношениями

В ряде предметных областей решаются задачи относительно систем, в которых во времени протекают процессы, связанные между собой посредством причинно-следственных отношений [105].

Процессы, протекающие в такой системе, условно можно разделить на внешние (*наблюдаемые*) и *внутренние* (которые не могут наблюдаться непосредственно и о которых можно судить лишь по их связям с наблюдаемыми процессами). Наблюдаемые процессы обычно называют *признаками*, признаки имеют значения, которые получаются в результате наблюдения этих признаков и изменяются с течением времени. Помимо признаков наблюдаемыми могут быть постоянные во времени *характеристики* системы и происходящие в различные моменты времени *события*, внешние по отношению к системе, но воздействующие на протекающие в ней процессы (как внешние, так и внутренние) [54].

Причинно-следственные отношения между множеством причин и множеством следствий включают:

- отношения между внутренними и внешними процессами (причинами являются внутренние процессы, а следствиями – признаки);
- отношения между внутренними процессами (причинами являются внутренние процессы, а следствиями – другие внутренние процессы);
- отношения между событиями и внешними процессами (причинами являются события, а следствиями – признаки);
- отношения между событиями и внутренними процессами (причинами являются события, а следствиями – внутренние процессы);
- смешанные отношения (причинами являются внутренние процессы и события, а следствиями – признаки) [63].

В БЗ Kn каждому причинно-следственному отношению соответствует предложение вида $\forall\ t\ p_c(..., t) \Rightarrow \varphi(..., t)$, где $p_c \in P_c$, а $\varphi(..., t)$ – формула (причинная закономерность), устанавливающая соответствие между значениями причин, характеристик системы и значениями следствий этого отношения (аргументов отношения p_c) на некотором интервале времени от начала действия причин и до окончания их действия.

Часть внутренних процессов, протекающих в системе, являются присущими системе и протекают в ней постоянно (определены на всем временном интервале существования системы). Некоторые протекают в системе постоянно; другие процессы – только на некоторых интервалах времени. Система функционирует нормально до тех пор, пока в ней протекают только присущие ей (внутренние) процессы, а происходящие события не приводят к возникновению в ней новых внутренних процессов, не присущих ей. Каждый процесс (внешний и внутренний), не являющийся присущим системе, имеет начало (момент времени, когда он возникает) и конец (момент времени, когда он исчезает).

Для алгебраической системы AS (R), моделирующей некоторую конкретную систему (в которой во времени протекают процессы, связанные между собой посредством причинно-следственных отношений),

выполнены следующие условия:

- для каждого признака существует причинная связь, в которой этот признак является следствием;
- для каждого события существует причинная связь, в которой это событие является причиной;
- для каждого внутреннего процесса, не присущего системе, существует причинная связь, в которой этот процесс является следствием, и существует причинная связь, в которой этот процесс является причиной;
- для каждого внутреннего процесса, присущего системе, существует причинная связь, в которой этот процесс является причиной;
- все предложения из БЗ являются истинными относительно AS (R).

Математическая абстракция результатов наблюдения R охватывает признаки $f_{ex} \in F_{ex}$, характеристики системы $r_O \in R_O$ и произошедшие события $f_{ev} \in F_{ev}$.

Следующие задачи учитывают процессы, связанные между собой посредством причинно-следственных отношений (Рисунок 2-5).

Задачи диагностики. В *задаче диагностики* требуется по результатам наблюдения признаков, характеристик системы и произошедших событий определить возможные причинно-следственные модели системы. При этом диагнозом называется совокупность внутренних процессов этой модели, не являющихся присущими системе [36].

Дано: R (результаты наблюдения признаков, характеристик системы и произошежших событий);

БЗ Kn;

найти: все возможные причинно-следственные модели AS (R) системы, согласованные с результатами наблюдений R, относительно которых все предложения из Kn. При этом диагнозом Δ в каждой такой модели является множество функциональных символов из F_{in} ($\Delta \subseteq F_{in}$), обозначающих внутренние процессы, не присущие системе, интерпретация каждого из которых в AS (R) имеет непустую область определения [63, 153].

Задача критики гипотезы о диагнозе может рассматриваться как уточнение *задачи критики гипотезы о классе* (поскольку множество возможных диагнозов конечно, диагноз может рассматриваться как своего рода класс).

Задачи прогноза результата воздействий. *В задаче прогноза результата воздействий* требуется, зная характеристики системы, диагноз и планируемые события, определить значения признаков в некоторые моменты времени в таких причинно-следственных моделях системы, которые соответствуют БЗ.

Дано: R_O (результат наблюдения характеристик системы); Δ (диагноз); R_{ev} (планируемые события); T (конечное множество моментов времени);

БЗ Kn.

найти: в каждой причинно-следственной модели AS (R_O, Δ, R_{ev}), относительно которой все предложения из Kn истинны, значения всех функций, являющихся интерпретациями функциональных символов из F_{ex}, во все моменты времени из множества T.

Обозначим cond (f_{ex}, t) формулу, являющуюся условием на значение признака $f_{ex} \in F_{ex}$ в момент времени t, а множество таких формул – $Cond_t \subseteq Cnd$.

Задачи планирования управления. В *задаче планирования управления* требуется, зная характеристики системы, диагноз (или класс систем) и условия на значения признаков, определить такую совокупность событий и соответствующие им моменты времени, при которых признаки как функции времени в причинно-следственных моделях системы, соответствующих БЗ, будут удовлетворять этим условиям.

Дано: R_O (результат наблюдения характеристик системы); Δ (диагноз) или class; $Cond_t$ (условия на значения признаков),

БЗ Kn.

Найти: R_{ev} (совокупность событий и моментов времени), при которых признаки как функции времени в причинно-следственных моделях системы, соответствующих Kn, будут удовлетворять этим условиям.

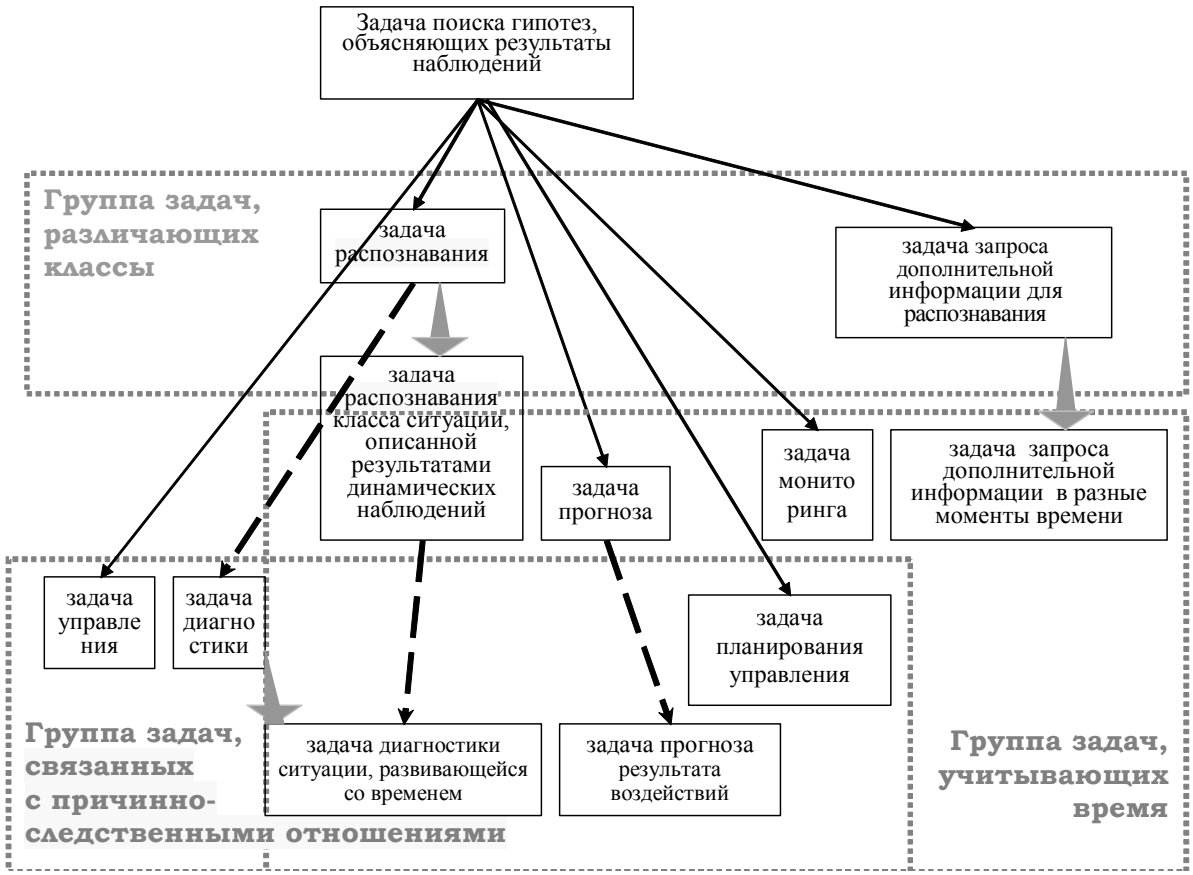


Рисунок 2-6 – Схема спектра возможных задач, учитывающих время

2.1.9. О Составных задачах

В литературе часто отмечают, что экспертные системы реализуют составные задачи, т.е. решают взаимосвязанную комбинацию задач. Повседневная интеллектуальная деятельность часто бывает связана с принятием разных типов решений: сбор информации, диагностика, ремонт, прогноз, и т.п., часть из которых является интеллектуальными задачами. *Составной задачей* интеллектуальной деятельности будем считать совокупность не менее двух совместно решаемых интеллектуальных задач, связанных общей информацией [105]. Примером составной задачи является *задача технической диагностики* автономного робота, решаемая в процессе выполнения им миссии. В процессе

выполнения миссии подсистемы или их отдельные узлы могут выходить из строя. При этом робот в пределах возможного должен продолжать выполнение миссии, возможно, в усеченном виде. Задача *мониторинга технического состояния* робота выявляет внешние проявления неполадок отдельного узла, которые являются входными данными для задачи *обнаружения возможных диагностических ситуаций*, если установлен вышедший из строя узел, то вносятся изменения в план действий – перепланирование миссии с учетом изменившейся модели робота.

Т.о. введение многоуровневой классификации известных интеллектуальных задач с их постановками в едином формализме позволяет обнаруживать сходные задачи, сопоставлять условия задач и виды аксиом в модели знаний, уточнять задачи в зависимости от уточнений свойств предметных областей, в которых они появляются [61]. Классификация вместе с постановками задач позволяет обнаруживать зависимости задач друг от друга (по входным и выходным данным), что облегчает не только. Системный Анализ, но и Системное Проектирование [151].

2.2. Обобщенные модели знаний для определения методов решения задач

Как отмечено выше, в предметных областях со специфическим набором свойств возникают, ставятся и решаются разные задачи анализа и проектирования. Анализ свойств предметных областей, в которых они решаются, позволяет определить обобщенную абстрактную модель знаний для решения нескольких задач (возможно, связанных друг с другом как шаги единой прикладной составной задачи). Например, д*ля распознавания класса критической ситуации* нужны знания о вариантах *признаков класса ситуаций (*внутреннего процесса), о *вариантах возможных стимулов (причин) появления внутреннего процесса, о вариантах реакции на события.*

В ПрОбл, где рассматривается развивающийся процесс (или динамическая ситуация), часто требуется им *управлять, определять стадии* его развития, *прогнозировать* состояние.

Развитие процесса (в т.ч внутри объекта) зависит от некоторых *признаков* или *характеристик* (напр., длина, возраст). Отмечается, что приходится иметь дело с процессами, основным «параметром» которых является время, а структура и «законы поведе-

ния» которых недостаточно изучены для целей принятия решений» [68]; наблюдение их представляют, в частности, временными рядами.

Для решения таких задач *модель знаний* включает предложения о свойствах (каждого класса) процессов ИЛИ о законах изменения свойств ситуаций (каждого класса) с течением времени, в т.ч. под влиянием известных факторов. Например, в [68] рассматривается причинно-следственная связь между последовательностями *интегральных признаков* q_i, описывающих поведение временных рядов на нескольких, следующих друг за другом фрагментах: если в течение $\Delta t = \tau 1$ наблюдается признак $Q = q1$, затем в течение $\Delta t = \tau_2$ наблюдается признак $Q = q_2$, затем ..., то в последующем в течение $\Delta t = \tau_п$ будет $Q = q_n$ (τ_i характеризуют продолжительность этих фрагментов).

Если рассматриваются отклонения функционирования от нормы (обычно это ПрОбл с причинно-следственными отношениями и законами изменения не только с течением времени, но и под воздействием известных внешних событий), то возникают и решаются задачи *диагностики*, задачи *запроса дополнительной информации для распознавания*, «простой» задачи *управления* [99] *и прогнозирования последствий* управляющих воздействий.

Для них модель знаний включает дополнительно предложения о законах изменения свойств, атрибутов ситуаций (каждого класса) при появлении отклонений от нормы.

В ПрОбл, где рассматриваются планы построения статических объектов некоторого класса, или динамических объектов, способных к действиям или саморазвитию, или планирования действий, выполнение которых приводит к достижению *целевого состояния*, удовлетворяющего заданному *условию*, возникают и решаются (и связаны друг с другом) задачи: *планирование, прогноз выполнения плана* и оценка стадии выполнения плана. Возможно, надо *управлять* выполнением плана или ресурсами для его выполнения.

Для них модель знаний включает предложения о свойствах каждого класса компонентов плана (предусловий и результатов отдельных действий), об условиях совместимости таких компонентов в рамках плана или в рамках некоторого его шага (этапа), о правилах построения последовательностей действий из действий (или их последовательностей), о правилах запараллеливания действий (с учетом их предусловий, непроти-

воречивости целей,…), о возможных состояниях объекта (ситуации, системы) на разных этапах выполнения плана.

2.2.1. Модель знаний для решения задач, связанных с анализом динамических ситуаций

Для каждого класса (*class*$_1$,… *class* $_i$, …, *class* $_n$) развивающихся процессов (в том числе внутри объекта) или ситуаций или функционирующих систем или динамических объектов

определяется множество характерных для процесса:

- важных условий (факторов) его протекания в виде требований к характеристикам объекта / системы (метрикам, фенотипам и т.п.):

{<*class*$_k$, {factor$_{kj}$}>},

- *вариантов развития* $Vari_{kp}$ – вариантов последовательностей *периодов развития* процесса, определяемых некоторыми *признаками:*

{<class$_k$, {(period$_i$, interval$_i$), {f^{ex}Name$_{jk}$}>}}},

- вариантов изменения диапазона значений признаков в каждом *периоде развития*:

{<class$_k$, period$_i$, f^{ex}Name$_{jk}$, f$_{ex}$Values$_{jki}$>}}}

или

{<class$_k$, period$_i$, f^{ex}Name$_{jk}$, before-range$_i$ of f^{ex}Val$_{jk}$, after-range$_i$ of f^{ex}Val$_{jk}$>};

- вариантов *реакции признаков на события* (через изменение диапазона значений, в т.ч в виде тренда\направления изменения, с возможной задержкой **τ** начала изменен*ия*):

{f^{ex}Name$_j$, f$^{ev}{}_u$,

f$_{ex}$Values$_{ju}$, f^{ex}Trend, τ_{ju}}.

Несколько разных процессов (в т.ч. внутри объекта или системы) могут иметь схожие проявления, образуя *Группу процессов* Gr (или подгруппу группы):

Gr $_i$, {f$^{ex}_{jik}$}, {□$_{ki}$,}

f$^{ex}_{jik}$ f = (f^{ex}Name$_{jik}$, f^{ex}Values$_{ju}$).

Т.о. каждый *вариант развития* = последовательность (длиной n, не менее 1) диапазонов значений признаков (простых или комплексных (*интегральных)*) в период i

= a1, n; последовательность = множество проявлений; проявление = признак$_j$ с диапазоном Значений$_i$ признака (в каждый период); или проявление нормального *варианта развития* = признак$_j$, интервал одинаковой монотонности$_i$, продолжительность интервала$_i$, величина тренда$_i$ для признака$_j$. Это позволяет описать поведение на следующих друг за другом временных фрагментах [68]; участки монотонного убывания или возрастания признаков и величина тренда на участках (пример – знания о поведении временных рядов на временном интервале как определение тренда для некоторого интервала по известным (наблюдаемым) трендам на других интервалах [68]; закономерности прибавки урожая (*признак* = урожайность) с *диапазоном измененных значений* = «на λ ц/га» для *класс* = сорт С-4727, *фактор* = «высажен на типичном сероземе»; *событие* = внесение удобрений (азотных — 200 кг/га, фосфорных — 140 кг/га, калийных — 100 кг/га), *диапазон задержки* = (сезон урожая – сезон внесения удобрений) [8].

В простых случаях может быть описан закон «развития» в ПрОбл, типа: от *начального состояния объекта (процесса)* в течение $\Delta t = \tau 1$ наблюдается его *признак r* со *Значением* q1; затем в течение $\Delta t = \tau_2$ наблюдается признак r со *Значением* q_2. в последующем в течение $\Delta t = \tau_п$ будет r со *Значением* $q_{n.}$ Тогда для *прогноза*, например, применимы правила: Если в течение $\Delta t = \tau 1$ наблюдается признак r со *значением* q1 и признак r подчиняется описанному закону, *то на интервале* $\Delta t = \tau_п$ будет r со *значением* q_n [68].

В модель заложены временнЫе связи (между периодами) и причинно-следственные (между условиями на факторы и существованием процесса либо вариантом его развития и между событиями и последовавшими изменениями).

Структурированная обобщающая модель знаний о процессах (динамических объектах, функционирующих системах) может быть выражена (языком «content description notation», [160, 170]) и сгруппирована так:

f_{dyn} = [{factor$_i$} +] {[{factor$_k$} +] {(period$_i$ + interval$_i$)

+ {sign$_{jk}$, range$_i$ of sign$_{jk}$ in period$_i$

[+ {event$_u$ + (new-range$_{iu}$ of sign values | sign values trend $_{iu}$)

+ delay$_{kju}$}^{n2}] }}}^{n1}.

2.2.2.Модель знаний для решения задач, связанных с анализом аномальных процессов

Знания описывают законы изменения свойств ситуаций (некоторого класса) с течением времени – «причинно-следственную связь между темпоральными структурами, описывающими поведение до и после текущего момента времени *t*» [68] и законы аномального изменения ситуаций (некоторого класса) с течением времени: диагностические критерии для Δ_{ij} – класса отклонений (из Δ) [36]. К ним относятся:

- множество *вариантов проявления отклонения процесса от нормы* (как единственного или первого из *периодов развития* аномального процесса):

$\{<\Delta_k, \{Vari_{kp}\}>\}$,

$\{<Vari_{kp}, \{f^{ex}Name_{jkp}, f^{ex}Values_{jkp}\}>\}$,

- множество *вариантов последовательностей периодов развития* (<диагноз Δ_k, вариант развития$_i$, признак$_j$, период$_i$, диапазон Значений$_i$ признака в этот период>):

$\{<\Delta_k, \{Vari_{kp}\}>\}$,

$\{<Vari_{kp}, \{(\pi\varepsilon\rho_i, \tau_i), \{(<f^{ex}Name_{jkp}, f^{ex}Values_{ijk} \text{ in } \pi\varepsilon\rho_i\}\}>\}$,

т.е. вариантом «процесса» изменения значений отдельного признака является последовательность (длиной n, не менее 1) меняющихся возможных диапазонов значений признака в период i = a1, n [153];

- множество вариантов *появления аномалии как реакции на варианты событий* (*возможных стимулов / причин появления* аномального процесса)*:*

$\{<f^{ev}_u, \Delta_k, \tau_{ku}>\}$.

А если событие характеризуется в ПрОбл не только моментом совершения, но и некоторой качественной характеристикой (возможно, составной), тогда добавятся характеристики *события:*

$\{<\Delta_k, f^{ev}Name_j, \tau_{ki} [, f^{ev}Value_{ij}]>\}$.

А если процесс вызывается не событием, а «обстоятельствами» (под влиянием факторов без очевидных временных «рамок»):

$\{<\Delta_k, \{factor_{kj}\}>\}$

или $\{<f^{ev}_u, \Delta_k, \{f^{ex}Name_j, \pi\varepsilon\rho_i, f^{ex}Values_{jk} \text{ in } \pi\varepsilon\rho_i, f^{ex}Values^{after}_{jki}\}>\}$ [36].

Как для описания появления аномальных процессов, так и для описания противодействия им, рассматриваются:

- множество *вариантов реакции на воздействие события* – описаний вариантов развития внутреннего процесса, измененного воздействием события, связывающие *диагноз* и *событие* с некоторым *изменением* некоторого *признака*:
  $\{<f^{ev}_{u}, \Delta_k, \{ f^{ex}Name_j, \pi\varepsilon\rho_i, f^{ex}Values_{jk} \text{ in } \pi\varepsilon\rho_i \}>\}$;
- множество *вариантов реакции* процесса при воздействии на него (*как последовательностей изменившихся периодов развития* с изменившимися значениями наблюдаемых признаков процесса (или системы)):

$\{<f^{ev}_{u}, int^{ev}_{uk}, \Delta_k, \{ \pi\varepsilon\rho_i, \{ f^{ex}Name_j, f^{ex}Values_{jk} \text{ in } \pi\varepsilon\rho_i\}\}>\}$,

связывающие значение некоторого события при некотором диагнозе с некоторым значением или изменением значения признака.

Если не все диагнозы / классы *ситуаций* одинаково «критичны» для принятия решения или не для всех экземпляров, то диагноз может быть связан с атрибутом «*критичность» ситуаций / тяжесть состояния*: $<\Delta_k, CriticRate_k>$.

Также степенью критичности может быть охарактеризован период развития:

$\{ \pi\varepsilon\rho_i, StageCriticality_i \{f^{ex}Name_{jk}, f^{ex}Values_{jki}\}\}$.

Структурированная обобщающая модель знаний об отношениях терминов, описывающих аномальные процессы, может быть выражена так:

f_{in} =

$[\{factor_m \mid event_m\} +]\ [CriticRate +]\ \{[\{factor_{nj}\} +]\ \{(period_{ij} + [StageCriticality +]\ interval_i) + \{signName_{jk}, signValuesRange_{ijk}\ [+ \{event_u + \text{ new-range}_i \text{ of } signValues_{jkju} + delay_{kju}\}^{n2}] + signMinQaunt_{ji}\}\}\}^{n1}$.

Часто требуется явно описать нормальные значения признаков:

$\{<Norm, f^{ex}Name_j, f^{ex}Values_j>\}$

или нормальное развитие (или состояние): $\{<Norm, f^{ex}Name_{jk} (\pi\varepsilon\rho_i, \tau_i), f^{ex}ValuesTrend_{jk}>\}$

и

множество вариантов *нормальной реакции признака» на события,* т.е. вариантов *изменения признаков событиями* (вне аномалий и периодов):

{<Norm, $f^{ex}Name_j$, $f^{ev}{}_u$, τ_{iu}, ($f^{ex}Values^{after}{}_{ju}$ | $f^{ex}ValuesTrend_{jk}$)}>}.

2.2.3. Модель знаний для решения задач, связанных с управлением ситуациями

Для решения задач, связанных с управлением ситуациями, знания описывают поведение объектов или над объектами, совокупности действий, выполнение которых приводит к достижению *целевого состояния*, удовлетворяющего заданному *условию*. Часто необходимы *знания* о свойствах (каждого класса) компонентов плана (предусловий и результатов отдельных действий), об условиях совместимости таких компонентов в рамках плана или в рамках некоторого его шага (этапа), о правилах построения последовательностей действий из действий (или их последовательностей), о правилах запараллеливания действий (с учетом их предусловий, непротиворечивости целей,…), знания о возможных состояниях объекта / ситуации на разных этапах выполнения плана оценка текущей стадии выполнения плана как процесса позволяет делать *прогноз выполнения плана.*

Модель включает предложения типа <объект воздействия, {свойство$_i$}$^{i1=1,I1}$,{воздействующий объект, {свойство$_i$}$^{i2=1,I2}$, средство воздействия, {свойство$_{i3}$}$^{i3=1,\ I}$, воздействие$_{i4}$, {характеристики воздействия$_{i4}$}$^{i4=1,\ I}$}$^{j=1,\ J}$> или < инструмент воздействия, объект воздействия, *состояние* $_j$, {действие$_i$}, *состояние* $_{j+1}$>

и выглядит так:

<ObjClass, {$ObjProp_i$}$^{i1=1,\ I1}$,{$Actor_j$, {$ActProp_{i3}$}$^{i3=1,\ I}$, $impact_{i2}$, {$ImpChrct_{i4}$}$^{i4=1,I}$}$^{j=1,J}$>

<$Actor_j$, ObjClass, St_{ij}, $Impact_j$, $St_{i(j+1)}$>.

Модель включает и другие отношения (предложения):

< класс объекта$_j$, {действие$_i$}$^{i=1,\ I}$>

< класс объекта$_j$, {свойство$_{ij}$}$^{i=1,\ I}$>

<исходное состояние, {состояние $_j$}$^{i=1,\ J}$, целевое состояние>

<состояние $_j$, {действие$_i$}, состояние $_{j+1}$>;

<вид действия$_k$, предусловие$_k$, тип результата$_k$>,

<класс компонентов$_j$, свойство компонента$_{jk}$>,

<(не) совместимы, {класс компонентов$_j$} [, условия$_p$ | период$_p$]>,

<вид действия$_k$, может следовать за вид действия$_m$>,

<вид действия, может выполняться параллельно с вид действия$_n$>,

И т.д.

В более общем виде *модель управления* становится такой:

<Цель, Объект управляемый (объект воздействия),

Условия

{свойство$_i$ | *состояние*$_j$ |свойство окружения$_j$ | инцидент$_i$}

План воздействий

{воздействующий объект / инструмент воздействия,

{свойство$_i$},

Средство/материал воздействия, {свойство материала$_{i3}$}, {компонент$_{j\ i3}$},

{воздействие$_{i4}$ \действие$_i$, {характеристика (воз)действия$_{i4}$} }

Ожидаемый эффект

состояние $_{j+1}$, {свойство$_i$ $^{j+1}$} | Прогноз развития>

Структурированная обобщающая модель знаний об отношениях терминов, описывающих управление состоянием, может быть выражена так:

StatCntrPlan = Aim + Obj +

({signName$_j$, signValuesRange$_j$ (t_{w-1}) } | State$_j$) +

({EnvrSign$_j$ } | {event $_{ij}$) +

effectingInstrum+ {InstrumSign$_i$ } +

MaterlSign + {signValuesRange$_s$ } | {Compnt $_{sj}$) +

{effectAct$_v$ + {Character$_{uv}$} } +

({signName$_j$, signValuesRange$_j$(t_{w+1}) } | State $_{j(tw+1)}$))

На модели решаются задачи: *построение плана* изменения ситуации *(планирование)*, *планирование воздействий* на систему или объект (*управление* в смысле [120]), часто на одном из трех уровней: о необходимости вмешательства, о виде вмешательства, о целесообразном способе воздействия [52]. Используются *знания:*

о свойствах компонентов (каждого класса компонентов) плана (часто – действий или воздействий),

о предусловиях и результатах отдельных действий,

об условиях совместимости компонентов в рамках плана или в рамках некоторого его шага (этапа),

о правилах построения последовательностей из действий (или их последовательностей),

о правилах запараллеливания действий (с учетом их предусловий, непротиворечивости целей, и пр.).

На основе знаний о воздействиях на систему или объект (*знания* о возможных состояниях объекта / ситуации на разных этапах выполнения плана применяемых воздействий) решаются *задача оценки текущей стадии* выполнения плана воздействий (как поэтапного во времени, так и комплексного однократного воздействия) или *задача оценки текущего состояния* системы или объекта, а также *задача оценки прогноза выполнения плана* или *прогноза состояния* системы или объекта (в указанный момент в будущем).

2.3.Алгоритмы и операции для решения задач

Алгоритмы решения (или поддержки решения) интеллектуальных задач определяются и зависят от постановки задач (входные данные и база знаний (из «дано»).

Основные алгоритмы на основе обобщенной модели знаний I группы

Алгоритм «общий» распознавания класса критической ситуации (или типа процесса или варианта его развития или этапа процесса),

т.е. поиска $H_{R,Kn} = \{class_{i1}, ..., \}$ для R и построения AS (<R, *класс* = $class_{i1}$>), ... [4, 52, 69], содержит такие шаги, как:

выбор (под)множества гипотез из множества классов (ситуаций), для которых выполнены необх условия;

в цикле по (под)множеству гипотез

сравнение (наблюдаемых) значений статических признаков (в ситуации R) с «портретом» гипотезы $класс_{i1}$

в цикле по предложениям из множества Kn для $класс_{i1}$ ({<$class_k$, {($period_i$, $interval_i$), {$f^{ex}Name_{jk}$, $f^{ex}Values_{ijk}$ >}}})

поиск в ситуации R такого r_{tj} (…, t_k, …), название которого совпадает с названием характерного признака $f^{ex}Name_{jk}$ в Kn-предложении, и сравнение их значений в соответствующие периоды:

если значение признака в ситуации опровергает предложение из Kn, то ложна гипотеза (противоречие «портрету»),

если значения признака в ситуации соответствуют $f^{ex}Values_{ijk}$, то добавляется аргумент в объяснение соответствия «портрету»,

если в ситуации нет $f^{ex}Name_{jk}$, то добавляется аргумент в объяснение невозможности подтвердить гипотезу.

Результат поиска дает для гипотезы h_i три множества: опровергающие признаки, подтверждающие, требуемые для подтверждения – ($\{f^{ex}Name_{ij}{}^{refu}\}$, $f^{ex}Name_{ji}{}^{conf}\}$, $\{f^{ex}Name_{ij}{}^{requir}\}$).

Если множество «опровергающих» ($\{f^{ex}Name_{ij}{}^{refu}\}$) пусто и «требуемые для подтвержд» ($\{f^{ex}Name_{ij}{}^{requir}\}$) пусто, а подтверждающие не пусто, гипотеза-класс считается найденной (подтвержденной).

Если $\{f^{ex}Name_{ij}{}^{refu}\}$ (множество «опровергающих») не пусто, гипотеза-класс считается опровергнутой ($h^{refu}{}_i$) ИЛИ (h_i in H^{refu}).

Результат алгоритма дает три множества: опровергнутые классы (H^{refu} = $\{h^{refu}$-$class_k\}$), подтверждающий класс (h^{conf}-$class_k$), гипотезы, требующие дополнительной информации ($H^{\mathbf{requir}}$ = $\{h^{\mathbf{requir}}$-$class_k\}$),т.е. не отвергнутые. Каждое может быть пусто. Гипотезой-классом для пользователя, решающего задачу, является подтверждающий класс.

Поиск всех гипотез о классе ситуации может быть ускорен, если сформированы группы классов с общими признаками (обычно статическими). Тогда в алгоритме выбор (под)множества гипотез, для которых выполнены необходимые условия, начинается внутри гипотез – классов (ситуаций), для которых подтверждены общие признаки и выполнены необходимые условия для группы;

если в результате множество гипотез содержит более одной, то требуется алгоритм *генерации запроса признаков для сокращения числа гипотез); сводимый к* «*выявления признака-дифференциатора для множества* гипотез»;

также может оказаться полезным алгоритм сортировки гипотез по количеству совпавших признаков.

Поиск всех гипотез о *классе критической ситуации* отличается шагом «выбор (под)множества гипотез, для которых выполнены необходимые условия»; добавляется

условие на критичность класса (в виде атрибута или в форме вхождения его названия в особый список, который может быть разным в разные периоды решения задачи.

Вычислительные операции

Перечислим вычислительные операции, которые являются общими для алгоритмов решения нескольких этих задач и из более частных (согласно классификации) случаев. Операции над БЗ:

Получить множество критических классов ситуаций

(Kn) - > {criticalClass$_i$};

Получить портрет класса ситуаций

(Kn, criticalClass$_k$) - > {<f ^{ex}Name$_j$, f^{ex}Values$_{ijk}$>};

Получить портрет критической ситуации

(Kn, criticalClass$_v$) - > {< f ^{ex}Name$_j$, f^{ex}Values$_{ijk}$ >};

Получить признаки для гипотезы-класса

(Kn, h$_j$) - > { f ^{ex}Name$_{ju}$}.

Операциями над Входными Данными для задач являются:

Получить описание ситуации

Sit - > {(r_{tm}(…,t_0,…)}

Помимо вычислительных операций следует обратить внимание на повторяющиеся процедуры обработки информации (по более чем одной модели знаний или данных). Такими операциями (над данными и БЗ) являются:

Сравнить значения признака с «портретом» одного класса

(r_{tm}(…,t_0,…), Kn, $h_{R,Kn}$) - > (подтв | отверг);

Получить множество гипотез-классов, соответствующих наблюдаемому признаку

(r_{tm}(…,t_0,…), Kn) - > $H_{R,Kn}$;

Найти в ситуации признак, опровергающий гипотезу о классе

(Kn, h$_n$, {(r_{tm}(t_0, …t_k)}) - > f ^{ex}Name$_j$.

Основные алгоритмы на основе обобщенной модели знаний II и III групп

Алгоритм поиска всех гипотез о диагнозе (где требуется найти *объяснени*е всех возможных *диагнозов* как причин наблюдаемых внешне *отклонений* (r_1, r_2, …r_{n1})

$\subseteq R_{ex}$ с учетом анализа применяемых методов диагностики [12, 73, 75] может быть представлен т.о.

Проверить для Sit гипотезу *«ситуация в норме»*.

Если есть отклонения от нормы, то

множество гипотез H_{Kn} = множество всех (или самых актуальных) диагнозов Δ из Kn;

в цикле по Δ из H_{Kn}

сравнить множество *признаков* R_O с *факторами* из необходимого условия (НУ) диагноза Δ, чтобы отвергнуть его;

для не-отвергнутого диагноза Δ:

сравнить множество признаков R_O с НУ всех их Вариантов развития Vari, чтобы отвергнуть Vari;

для не-отвергнутых Vari (Вариант развития диагноза Δ)

сравнить множество признаков (r_1, r_2, …r_{n1}, r_{n1+1}, …r_n) из Sit со всеми Проявлениями из Комплекса, чтобы отвергнуть Vari;

если множество (r_1, r_2, …r_{n1}, r_{n1+1}, …r_n) соответствует комплексу характерных наблюдений **каждого** (очередного) периода, то подтвердить гипотезу-Вариант развития,

если хотя бы один признак из (r_1, r_2, …r_{n1}, r_{n1+1}, …r_n) противоречит указанному характерному наблюдению в **соответствующем** периоде, то опровергнуть Вариант развития;

если опровергнуты все Варианты развития, то опровергнуть гипотезу.

Алгоритм поиска признака для запроса дополнительной информации направлен на сокращение сформированного множества гипотез [29]. Для удобства решения задачи запроса дополнительной информации, направленного на сокращение множества гипотез о классе (или диагнозе), предлагается среди запрашиваемых признаков различать дифференциаторы множества рассматриваемых гипотез и антидифференциаторы (рассматриваемого множества). Дифференциатор – тот признак, каждое значение которого сокращает множество гипотез; антидифференциатор – такой признак, ни одно из значений которого не сокращает множество гипотез; «частичные» дифференциаторы – остальные

признаки (которые могут быть ранжированы, например, по доле значений, которые сократят множество гипотез) [65].

Обозначим t_{mom} текущий момент – точку на временн**о**й оси (от начала наблюдений t_0 либо от начала развития внутреннего процесса t_{beg}), момент принятия решения о запросе дополнительной информации.

Шаг 1) Предобработка БЗ и построение «рабочей» модели

Из Kn выделить фрагмент Kn', содержащий предложения о свойствах ситуаций каждого класса (чаще – диагноза: $class_k \mid \Delta_{ij}$) из множества гипотез о классах $H_{R,Kn} = \{class_k \mid \Delta_{i1}, ..., class_k \mid \Delta_{im}\}$; инвертировать фрагмент Kn', т.е. построить *«рабочую» модель признаков гипотез* Kn'^{Inv} как множество «троек» <признак, «характерные» значения, диагноз>, чтобы в модели рассматривать только подмножество признаков $f_{ex} \in F_{ex}$, которые связаны с гипотезами из $H_{R,Kn}$.

Для некоторых $f_{ex}Name_i$ из Kn' могут существовать варианты развития процесса, измененного воздействием события, связывающие: (Признак, диагноз, Событие, временной интервал начала такого воздействия, Результирующее значение признака) или (Признак, исходное значение признака, Событие, временной интервал начала такого воздействия, результирующее значение признака). Такие предложения должны быть добавлены в «рабочую» модель признаков Kn'^{Inv}.

Шаг 2) Исключение лишних признаков из «рабочей» модели признаков

В «рабочей» модели Kn'^{Inv} удалить (или пометить как «неактивные»):

- каждый признак $f_{ex}Name_i$, который является *статическим* и *его* значение $f_{ex}Value_i$ уже присутствует в R; (подразумевается удаление всех «троек» (признак = $f_{ex}Name_i$, «характ***е***рные» значения, диагноз));
- те признаки, у которых один из вариантов *динамики значений* определен не далее, чем до текущего момента и все его значения $f_{ex\text{-}j}Value_i$ (t_1, ..., t_{mom}), указывающие на этот вариант развития, уже присутствуют в R;
- те возможные факторы и события $f_{ev}Name_w$ ($\in F_{ev}$), информация о существовании которых уже присутствует в R.

Оставшиеся в «рабочей» модели Kn'^{Inv} признаки считаются возможными дифференциаторами.

Шаг 3) Поиск и исключение антидифференциаторов

Искать в Kn'^{Inv} те признаки $f_{ex}Name_i$, у которых

- «тройки» (признак, «характерные» значения, диагноз) присутствуют для каждой из m гипотез, при этом в каждой «тройке» либо Одинаково множество вариантов *значений (статического) признака,*

  либо с каждой гипотезой: Δ_{i1}, ..., Δ_{im} связаны одинаковые варианты *динамики значений признака,*

  либо (при отсутствии *значений динамического признака* до текущего момента t_{mom}) начиная с «текущего момента», с каждой гипотезой одинаковые варианты *динамики значений* во все последующие моменты времени.

Искать в Kn'^{Inv} те возможные события $f_{ev}Name_w$ в момент $t_v <= t_0$, которые присутствуют для каждой из m гипотез.

Такие $f_{ex}Name_i$ и $f_{ev}Name_w$ считать *антидифференциаторами.* Их можно удалить из Kn'^{Inv}.

Шаг 4) Поиск дифференциатора

Искать в Kn'^{Inv} признак $f_{ex}Name_i$, который для всех гипотез имеет разное значение из допустимого множества значений. Типичные поисковые запросы таковы.

А) Искать *дифференциатор – статический* признак с «качественными» з*начения*ми. Среди *статических признаков* $\{f_{ex}Name_i\}$ ((в Kn'^{Inv}), имеющих в качестве ОблВозмЗн множество значений, искать $f_{ex}Name_i$, который для всех m гипотез имеет разное значение (из этого допустимого множества значений $\{f_{ex}Value_{ij}\}$, j=1,n).

Б) Искать *дифференциатор – статический* признак с диапазонами *значений*. Среди *статических признаков* $\{f_{ex}Name_i\}$ (в Kn'^{Inv}), имеющих в качестве ОблВозмЗн один или совокупность числовых диапазонов, найти *признак* $f_{ex}Name_i$, который для всех гипотез имеет непересекающийся поддиапазон значений ($minValue_j$ – $maxValue_j$), т.е. $minValue_{j+1}$ > $maxValue_j$, $minValue_j$ > $maxValue_{j-1}$.

В) Искать *дифференциатор – динамический* признак. Среди *признаков* $\{f_{ex}Name_i\}$, имеющих в любой момент (или период) времени в качестве ОблВозмЗн множество значений, искать $f_{ex}Name_i$, который

- в некоторый момент t_v (или период $<t_v, t_{v+1}>$) времени, >= «текущего» момента t_{mom},

- для всех n гипотез имеет разное значение $f_{ex}Value_i(t_v)$ из допустимого множества значений $\{f_{ex}Value_{ij}\}$.

Среди *признаков* $\{f_{ex}Name_i\}$, имеющих в любой момент (или период) времени в качестве ОблВозмЗн числовые диапазоны, искать признак $f_{ex}Name_i$, который

- в некоторый момент t_v (или период $<t_v, t_{v+1}>$) времени, >= «текущ. момента» t_{mom},
- для каждой из n гипотез имеет такой поддиапазон значений ($minValue_j$ – $maxValue_j$), j=1,n, что множество таких n поддиапазонов не пересекаются.

Искать те возможные события $f_{ev}Name_w$ ($\in F_{ev}$), запрос Q о существовании (или значении) которых в момент $t_v <= t_0$ может иметь ответ (AQ) в «текущий момент» t_{mom} и этот ответ имеет разное значение для всех m гипотез-диагнозов.

Далее оценить дифференциатор, если он был найден. Ориентируясь на предложения $f_{ex}Name_i$, $f_{ex}Value_i(t_0)$, $f_{ev}Name_w$, $f_{ex}Value_i(t_N)>$ в Kn'^{Inv}, понять является ли дифференциатор «подверженным влиянию». Таковым следует считать дифференциатор $f_{ex}Name_i$ (t_v), про который известно, что на его значение влияют каких-либо событий.

Далее оценить наличие информации о таких событиях (для каждого такого «подверженного влиянию» *дифференциатора).* Если события в моменты $t_v <= t_{mom}$ отсутствуют в $R_{ex}(t_0, ..., t_k)$, то $f'_{ex}Name_i$ считать *дифференциатором*.

Если множество *дифференциаторов* = Ø, применить шаг 5. После шага 4 (Если множество *дифференциаторов* = Ø) в Kn'^{Inv} содержатся «*частичные» дифференциаторы*, которые могут *дифференциаторовать гипотезы,* если ответом (AQ) станет «подходящее»/«дифференцирующее» (способное отвергнуть некоторую гипотезу) значение.

Шаг 5) Ранжирование частичных дифференциаторов

Используя Kn'^{Inv}, ранжирование признаков $f_{ex}Name_i$ можно провести по доле значений, которые предположительно сократят множеств*о гипотез* (с какой *вероятностью* выпадет в ответе AQ_Σ значение $f_{ex}Value_{ij}$, способное *исключить k* из *n гипотез).*

Ранжирование *статических* признаков с качественными значениями (*признаки* $\{f_{ex}Name_i\}$, имеющие ОблВозмЗн множество «качественных» *значений*) таково:

признаки $\{f_{ex}Name_i\}$ (в Kn'^{Inv}), имеющие Областью возможных Значений множество «качественных» *значений,* связанные более, чем с одним *диагноз*ом, упорядочить,

например, по величине отношения числа разных значений $f_{ex}Value_{ij}$ признака, связанных только с одним из этих *диагнозов*, к числу возможных Значений. (Далее упорядочить аналогично *признаки*, у которых разные значения связаны только с двумя *диагнозами* (если их число в *рабочей модели* >2) и т.д.)

Ранжирование *статических* признаков $\{f_{ex}Name_i\}$, имеющих ОблВозмЗн множество числовых интервалов, таково:

все такие *признаки* $\{f_{ex}Name_i\}$, связанные более, чем с одним *диагнозом*, упорядочить, например, по величине отношения длины интервала ($f_{ex}maxValue_{ij} - f_{ex}minValue_{ij}$), связанного только с одним из этих *диагнозов*, к диапазону возможных Значений ($f_{ex}maxValue - f_{ex}minValue$).

Если к шагу 5 в KN'^{Inv} множество *признаков* $\neq \emptyset$, то хотя бы один *частичный дифференциатор* будет предложен (признак, значение которого, возможно, сократило множество гипотез).

Т.о. Алгоритм поиска наблюдения для запроса дополнительной информации является алгоритмом поиска и ранжирования дифференциаторов (по времени, влиянию событий и характеристик) [29].

Алгоритм задачи прогноза

Метод решения задачи прогноза в условиях существования формализованной базы *знаний о законах изменения объекта или ситуации во времени* отличается от часто применяемых математических моделей прогноза.

Самый простой случай – в задаче прогнозирования номинальных переменных, относимой к задачам распознавания, поскольку в задаче «прогнозируемая величина принимает значения из множества, содержащего относительно небольшое число элементов» [91]. Аналогично *прогнозирование выходов* оборудования *из строя* на основании различных *критериев* [42]. Это позволяет обойтись без применения «особого метода для прогнозирования, предполагающего, что существует вычислительная модель системы, завершенная на некотором уровне детализации» [99]). На формализованной базе *знаний* решаются задачи п*рогноза развития проблемной ситуации*, когда п*рогноз – это* предполагаемые последствия данной ситуации [99]. В такой задаче для известного класса систем (ситуаций, процессов) требуется по входных данным (признакам) узнать итоговое поведение.

Clancy обращал внимание, что прогнозирование может быть методом, лежащим в основе многих других подзадач/операций (например, генерации гипотез и проверки возможных диагнозов).

Алгоритм поиска гипотез о последствиях (данной ситуации) содержит такие шаги, как:

выбор (под)множества гипотез из вариантов развития, для которых выполнены необх условия;

в цикле по (под)множеству гипотез – вариантов:

выбор (по наблюдаемым признакам в ситуации R) периодов гипотезы-варианта развития, для которых выполнены необходимые условия наступления периода (кроме финального периода),

в цикле по предложениям из множества Kn для period, из *класс*$_{i1}$ ({<class$_k$, {(period$_i$, interval$_i$), {sign$_{jk}$, range$_i$ of sign$_{jk}$ in period$_i$>}}}):

поиск в ситуации R такого r_{tj} (…, t_k, …), название которого совпадает с названием характерного признака sign$_{jk}$ в Kn-предложении, и сравнение их значений в соответствующие периоды:

если значение признака в ситуации опровергает предложение из Kn, то ложна гипотеза (противоречие «портрету»),

если значения признака в ситуации соответствуют range$_i$ of sign$_{jk}$ in period$_{i-}$, то добавляется аргумент в объяснение соответствия «портрету»,

если в ситуации нет sign$_{jk}$, то добавляется аргумент в объяснение невозможности подтвердить гипотезу.

Результат поиска дает для гипотезы-периода три множества: опровергающие признаки, подтверждающие, требуемые для подтверждения.

Если есть гипотеза-период, для которой множество «требуемых для подтверждения» пусто (или гипотеза-$\pi\varepsilon\rho_{-p} \in$ подтвержденные периоды (VARI-$\pi\varepsilon\rho_i^{conf}$),

а ни одна из гипотез-периодов $\pi\varepsilon\rho_j$ (j=1,…p-1) варианта-v не попала в «опровергнутые», то период-(p+1) $\pi\varepsilon\rho_{p+1}$ варианта-v Vari$_v$ добавляется к числу прогнозируемых (Vari$_v\pi\varepsilon\rho_{p+1} \in H^{progn}$).

Часто на практике требуется предупредить о возможной *критической ситуации*, к которой может перейти *процесс*. В модели знаний отражена последовательность смены стадий или последовательности (динамика) изменения некоторых свойств; критической стадии предшествуют менее критичные стадии: *<Δ_k, критичность стадии, {признак$_j$, диапазон значений$_i$ признака, важность признака для стадии или для класса}>*. Понадобится операция определения степени принадлежности имеющегося или полученного значения к ожидаемому значению. Для перехода из одной стадии в другую известны и явно указаны либо временн***ы***е характеристики (например: *если в течение* $\Delta t = \tau 1$ *наблюдается признак* $Q= q1$, *затем в течение* $\Delta t = \tau_2$ *наблюдается признак* $Q= q_2$, *затем ..., то в последующем в течение* $\Delta t = \tau_n$ *будет* $Q= q_n$ (τ_i характеризуют продолжительность этих фрагментов) [68]) либо какие-то факторы или причины. Это позволяет строить прогноз.

Алгоритмы планирования. На основе обобщенной модели знаний для 3-ей *группы* алгоритм решения задач *построения плана* изменения ситуации или планирования воздействий на систему или объект представляет собой последовательные проверки наличия для подцелей, соответствующих преследуемым состояниям, тех воздействий, которые обеспечивают переход к ним (воздействий как элементов выстраиваемой гипотезы); и поиска среди наблюдаемых фактов – возможностей применения таких воздействий (предусловий для таких воздействий, совместимости друг сдругом) либо опровержения элемента (-ов) гипотезы. Пример – задача *построения плана симптоматического лечения* (а также задача *построения плана лечения* при известном диагнозе (с некоторыми ограничениями, связанными с существованием приказов)).

Алгоритм назначения персонифицированного лечения просматривает знания о лечении заболевания (набор схем лечения) и сопоставляет ее входной информации, которой является медицинская карта пациента. Задача состоит в поиске элементов наблюдений, влияющих на назначение лечения и сравнение их значений с диапазонами значений соответствующих наблюдений в базе знаний (сравнение диапазонов значений, допускающих вариант лечения, с наблюдениями пациента). Анализируются модели терапии, их виды, цели, схемы приема на каждом шаге, проверяя соответствующие условия, значения которых анализируются по медицинской электронной карте. В

результате формируется назначение лечения (схемы лечения или некоторой группы веществ в этой схеме или некоторого вещества в этой группе) либо указывается, каких признаков (из анамнеза жизни, лабораторных, инструментальных, объективных методов исследования и др.) не хватает для назначения. В назначении веществ делается указание разовой, дневной, курсовой их дозировки, совместимости и особенностей проведения курса лечения.

Отдельные этапы алгоритмов, упоминаемых в литературе, используют *знания о связях вида:* <инструмент воздействия, {свойство$_{i3}$}$^{i3=1,I}$, объект воздействия, *состояние*$_j$, {действие$_i$}, *состояние* $_{j+1}$> (чтобы «производить отбор и оценку единиц оборудования для проведения планового ремонта»), <воздействующий объект, {свойство$_i$}$^{i2=1,I2}$, {средство воздействия, {свойство$_{i3}$}$^{i3=1,\ I}$}> (делать «*оценку состояния единиц оборудования»),* <объект воздействия, {свойство$_i$}$^{i1=1,I1}$,{воздействующий объект, {свойство$_i$}$^{i2=1,I2}$, средство воздействия, {свойство$_{i3}$}$^{i3=1,\ I}$, воздействие$_{i4}$, {характеристики воздействия$_{i4}$}$^{i4=1,\ I}$}$^{j=1,\ J}$> (делать «прогноз выхода оборудования из строя, который используется при составлении нового плана») [52].

Вычислительные операции

Перечислим общие для перечисленных алгоритмов решения нескольких задач операции над БЗ:

Получить портрет варианта проявления аномального процесса

(Kn, Δ_m, Vari$_{mp}$) - > {<f$_{ex}$Name$_j$, {{$\pi\varepsilon\rho_i$, *int*$_i$, f^{ex}Values$_{ij}$>}}};

Получить все необходимые условия для указанного диагноза

(Kn, Δ_m) - > {<f^{fct}Name$_{mu}$>}$\cup$ {f$^{ev}_{mv}$};

Получить все группы

(Kn) => {GrΔ_{m1},…, GrΔ_{mm}},

Получить все необходимые условия для группы

(Kn, GrΔ_m) => {<f^{fct}Name$_{mu}$>}$\cup$ {f$^{ev}_{mv}$};

Получить список общих признаков для группы

(GrΔ_m) => {<f^{ex}Name$_{jm}$, f^{ex}Values$_{mj}$>};

Получить список видов исследования указанного признака/проявления

(Kn, f^{ex}Name$_j$) => {f^{ex}NameMeth$_{jk}$};

*Построить модель признаков для одной гипотез*ы

*(*Инвертировать связи гипотезы-аномалии с ее признаками для получения множества* Diag-inv = {<$f_{ex}Name_j$, ($\pi\varepsilon\rho_{ij}$, int_{ij}), $f^{ex}Values_{ij}$, h-diag>} **)*

(Kn, h-diag) => {<$f_{ex}Name_j$, {($\pi\varepsilon\rho_{ij}$, int_{ij}), $f^{ex}Values_{ij}$}, h-diag>};

Построить модель признаков и факторов для множества гипотез

*(*Инвертировать связи аномалий с их признаками для получения* Kn-inv – *множества* <признак, период, «характерные» значения, диагноз> и пар <причина, диагноз> и удалить совпавшие по области значений *признаки **)

(Kn, H) => {<$f_{ex}Name_j$, {($\pi\varepsilon\rho_{ij}$, int_{ij}), $f^{ex}Values_{ij}$}, h-diag>} ∪ {<$f^{fct}Name_m$, h-diag>};

Получить воздействия для обеспечения указанного проявления

(Для указанного признака указанного процесса найти в БЗ факторы, которые меняют его значение на указанное значение *)*

(Kn, Δ_m, $f^{ex}Name_j$, $f^{ex}Value_j$) => {$f^{ev}{}_v$};

Получить воздействия для изменений признака

(Для указанного признака указанного процесса найти факторы-воздействия, которые обеспечивают указанный тренд изменения значений*

(Kn, Δ_m, $f^{ex}Name_j$, (увеличение|уменьшение) => {$f^{ev}Values_{jv}$};

Получить признаки, нормализуемые указанным средством

(Найти в БЗ номинальные (качественные) признаки, которые будут приходить к норме из-за действия указанного фактора и некоторого класса/диагноза *)*

(Kn, Δ_m, $f^{ev}{}_v$) => { $f^{ex}Name_j$ };

Получить признаки, которые увеличатся при указанном воздействии

(Для указанных факторов и некоторого класса/диагноза найти в БЗ числовые признаки, которые будут увеличиваться*)*

(Kn, Δ_m, $f^{ev}{}_v$) => { $f^{ex}Name_j$ };

Оценить состояние в следующем периоде

(Определить благоприятность следующего периода развития аномального процесса, следующего за данным периодом данного *)*

(Kn, Δ_m, $\pi\varepsilon\rho_i$) => (улучшение|ухудшение);

Определить воздействия для сложившихся условий

(Kn, $ObjProp_i$, St_{ij}, {$f^{ex}Name_j$}) => {<$Actor_j$, $impact_{ij}$ >};

Оценить длительность периода

(Kn, Δ_m, $\pi\varepsilon\rho_i$) => Int.

Получить предполагаемые периоды ухудшения

(найти в БЗ последовательные возможные переходы к периодам,* критичность которых выше текущего*)

(Kn, Δ_m, $\pi\varepsilon\rho_i$) => {seq ($\pi\varepsilon\rho_{i+1}$, $\pi\varepsilon\rho_{i+2}$,…)}.

Общие для алгоритмов решения задач операции над базами наблюдений или справочниками:

Проверить, в норме ли наблюдение

(r_{n1}, Kn^{Norm}) => bool;

Подтвердить нормальность наблюдения

*(*Подтвердить, что наблюдаемые значения признака соответствуют норме **)

(r_j (t_k,…,t_m), Kn^{norm}) => {true,false}.

Операции над ресурсами с входными данными:

Получить все состоявшиеся события

(R) => {r^{ev}_j (t_{kj})};

Получить все особенности

(R) => {r^{O}_i}.

Процедуры обработки информации таковы.

Проверить наличие указанных факторов в ситуации

(R, {<$f^{fct}Name_{mu}$>}∪ {f^{ev}_{mv}}) - > bool;

Определить соответствуют ли условия предлагаемому воздействиию

(Kn, Sit, $Actor_j$, $Impact_{i2}$, $f^{ex}NameMeth_{jk}$) => {true,false};

Проверить соответствие признака ожиданиям одним из видов исследования

(R, $f^{ex}Name_{jm}$, $f^{ex}Values_{mj}$,{$f^{ex}NameMeth_{jk}$}) - > (bool; *вид исследования*$_j$);

Удалить из модели признаков лишнюю инфо

(*Удалить из модели признаков Kn^{inv} ={<$f_{ex}Name_j$, {($\pi\varepsilon\rho_{ij}$, *int*$_{ij}$), $f^{ex}Values_{ij}$}, h-diag>}

лишнюю инфо об уже подтвержденных значениях признаков ($\pi\varepsilon\rho_{ij}$, *int*$_{ij}$), $f^{ex}Values_{ij}$ и тех, сроки получения которых уже истекли…*)

(R, Tmom, Kn^{inv}) => $Kn^{inv'}$;

Оценить потенциальный фактор

(* Определить по базе знаний степень принадлежности фактора к влияющим факторам для класса (процессов) *)

(Kn, $Class_m$, $фактор_k$) => (важный | возможный | отсутствует);

Определить возможное значение после воздействия

(**определить значение интересующего признака в ожидаемый момент при влиянии* фактора, если в знаниях задана такая закономерность или правило *)

(Kn, $Class_m$, f^{ev}_v, $f_{ex}Name_i$, $\pi\varepsilon\rho_i$, int^{ev}_{ji}) => ($f_{ex}Values_i$);

Определить время изменения значения интересующего признака

(Kn, ($Class_m$), $factor_k$, $\pi\varepsilon\rho_i$, $f_{ex}Name_j$, $range_i$ of $f^{ex}Val_{jm}$) => $\pi\varepsilon\rho_{ii.}$

2.4. Выводы к главе 2

Введение принципа уточнения абстрактных задач по мере рассмотрения свойств ПрОбл и единый формализм для постановки задач на разных уровнях абстракции позволил создать расширяемую классификацию решаемых на практике задач и систематизирующую новые постановки задач.

Определения задач связываются с онтологией знаний, при этом онтологии знаний для решения частных задач являются расширениями более общих, абстрактных за счет определения в них понятий и связей, учитывающих частные свойства ПрОбл.

Алгоритмы решения задач опираются на единые вычислительные операции, определяемые в терминах потока информации от входных данных задачи (в т.ч. и используемых знаний) до выходных результатов.

ГЛАВА 3. Принципы создания жизнеспособных интеллектуальных систем с базами знаний

В главе рассматривается парадигма жизнеспособности. Предлагается модель жизнеспособной программной системы, использующей базы знаний, устанавливаются некоторые ключевые принципы проектирования архитектуры жизнеспособной СБЗ, в том числе онтологический подход к определению компонентов.

3.1. Понятия и парадигма сопровождаемости и жизнеспособности интеллектуальных программных систем

В процессе эксплуатации прикладных программных систем (ПС) появляется потребность в:

- исправлении ошибок,
- добавлении пользовательских функций (расширение),
- адаптации к новым условиям использования (новые приборы, устройства, платформы),
- изменении пользовательского интерфейса специалистов,
- уточнении формата или состава информации (улучшение).

Такое свойство ПС как *сопровождаемость* – возможность исправления, переструктурирования (рефакторинга), адаптации (к аппаратуре и системному программному обеспечению, к новым видам человеко-машинных интерфейсов, к новому типу пользователей) и расширения (новыми функциями по желанию пользователей) – получило заслуженное внимание несколько десятилетий назад.

Позднее обозначилось новое полезное свойство систем – жизнеспособность, – охватывающее сохранение работоспособности и полезности ПС при изменениях окружения [146] и приспосабливаемость к изменчивости требований в течение «жизни» с наименее возможной стоимостью и поддержанием архитектурной целостности [119], т.е. адаптивность и адаптируемость.

Будем называть *жизнеспособностью* устойчивость ПС к некоторым изменениям среды функционирования и способность к развитию в течение жизни. Эти два свойства реализуются через интерактивное изменение компонентов ПС в ответ на новые требования с использованием высокоуровневых механизмов (в отличие от изменения программного кода в рамках сопровождения).

Адаптируемость реализуется через:

- декларативное представление компонентов, обеспечивающее, с одной стороны, программно-обрабатываемое их представление, с другой,

- наглядное и понятное, соответствующее системе терминов (системе понятий, языку общения) разработчика;

- наличие структурных и графических редакторов для поддержки разработки всех компонентов ПС;

- автоматизацию разработки компонентов (генерацию схем СУБД, генерацию программного кода) с проверкой их архитектурной целостности.

В отличие от нее *адаптивность* обеспечивается механизмами:

- машинного обучения, которые на основе наборов входных данных улучшают и совершенствуют алгоритмы обработки данных;

- обратной связи с пользователем, окружающей средой, обеспечивающих настройку интерфейса и сценария взаимодействия с ПС.

Таким образом, жизнеспособность расширяет понятие сопровождаемости, прежде всего, наличием механизмов адаптивности и адаптируемости с использованием высокоуровневых механизмов, минимизирующих изменение программного кода (но не исключающее этот вид деятельности).

Для программных систем с заранее предопределенной архитектурой (обеспечиваемой фреймворками и оболочками) сопровождение и объединение разных компонентов заметно проще, поскольку проблема адаптируемости к устройствам, платформам, гибкость пользовательского интерфейса, формата хранимой и другой информации «перекладывается» на инструменты (фреймворков и оболочек).

В случае с прикладными ПС, связанными с решением интеллектуальных задач, считается, что верхнеуровневая архитектура тоже предопределена. «По сравнению с традиционными ПС поддержки принятия решений (с компонентами: интерфейс лица,

принимающего решение, БД и система управления БД, база моделей с системой управления) в интеллектуальных СППР, добавляются:

- база знаний;
- подсистема объяснения решений;
- подсистема накопления и модификации знаний (система управления БЗ, СУБЗ)» [51].

*Интеллектуальны*е задачи в современных *системах с базами знаний (СБЗ)* решаются «с применением эвристик, включая эмпирическую индукцию, аналогию и дедукцию» [98], часто с использованием более одного метода имитации интеллектуальной деятельности человека [17].

В процессе эксплуатации ПС, связанными с решением традиционных интеллектуальных задач (диагностика, планирование, прогноз… и т.д.) редко возникает потребность в добавлении новых пользовательских функций («расширение»). При разработке ПС для решения интеллектуальных задач первична постановка задачи, знания (Kn (X, Y)) и метод ее решения (метод получения $y \in Y$ при любых $x \in X$).

Чаще обновление потребуется для метода получения результата задачи. Еще скорее ожидается изменчивость знаний (появление новых методов диагностики, выявление новых факторов, влияющих на ситуации, и т.п.) и методов решения, подсистемы объяснения решений и пользовательского интерфейса. Причина в том, что постановки многих известных задач (см. гл 2) достаточно устойчивы, а новые интеллектуальные задачи чаще всего являются конкретизацией (уточнением) уже известных или их комбинацией.

Адаптируемость пользовательского интерфейса СБЗ актуальна в той же мере, что и для остальных ПС: так же, как и прочие ПС, современные СБЗ предлагают удобный интерфейс. Но дополнительно к этому в СБЗ к представлению информации пользователям предъявляется типичное специальное требование: важен привычный специалистам вид исходных данных и результата решения задач и понятность объяснения и аргументации предложенного результата или гипотезы. Вид исходных данных и результата, а также содержание и форма объяснения определяются не столько требованиями к удобству использования (как это принято в программной инженерии), а соглашениями, сложившимися в предметной области.

Адаптируемость к прочим интерфейсам аналогична ситуации для любых ПС: современные СБЗ могут получать данные от современных средств измерения и часто интегрируются со сторонними программными компонентами.

Жизнеспособность для системы с базой знаний, – это сохранение работоспособности и полезности в условиях изменчивости предметной области (чаще – знаний, реже – среды функционирования и пользовательского интерфейса).

Поскольку *жизнеспособность СБЗ* проявляется, прежде всего, в условиях изменчивости знаний предметной области, а «способность к адаптации в соответствии с изменением множества фактов и знаний» считается одним из аспектов интеллекта [98], то для большинства предметных областей, связанных с решением интеллектуальных задач, подразумевается эволюционируемость знаний. Более того, в предметных областях, где важны влияние факторов и событий на состояние системы (объекта, ситуации), их изменение во времени, влияние индивидуальных характеристик системы и одних ее процессов на другие, развиваемость / эволюционируемость баз знаний – главный «вызов» современных «условий» по отношению к СБЗ.

Развиваемость баз знаний позволяет надеяться на получение «эталонной» базы знаний $k^* \in Kn\ (X, Y)$, от актуальности (соответствия современному представлению) и качества которой зависит успех применения СБЗ. Актуальность знаний (базы знаний) достигается адаптируемостью, адаптивностью либо их комбинацией [43,70].

Адаптируемость – интерактивное изменение базы знаний: в процессе решения задач на практике знания нередко уточняются, дополнительным источником совершенствования знаний являются новые научные результаты, относящиеся к этой интеллектуальной деятельности. Адаптивность связана с методами машинного обучения (могут применяться средства индуктивного формирования знаний по отобранным прецедентам, средства выявления знаний из «больших данных»), либо их комбинацией.

3.2. Архитектурная модель системы с базой знаний

Архитектурной моделью СБЗ (KBS-Arch) будем называть представление всех компонентов, обеспечивающих ее функционирование, с указанием связей этих компонентов друг с другом. *Архитектурно-технологической моделью СБЗ* (KBS-Arch) будем

считать представление всех компонентов, участвующих в процессе ее функционирования и в процессе ее адаптации, и связей этих компонентов друг с другом. *Компоненты* системы, решающей на основе баз знаний одну задачу, можно представить так.

KBS-Arch = $<\cup^{i=1,N1} (KB_i), Solv_1, \cup^{m=0,\,N2} (Ui_m), \cup^{p=0,N3} (DB_p)>$.

Здесь совокупность баз знаний обозначена как $\cup^{i=1,N1} (KB_i)$, программный решатель – как $Solv_1$, часто необходимы компоненты, обеспечивающие пользовательский интерфейс (Ui или Gui) и хранилища оперативной или справочной информации и базы фактов (DB_p). Базы знаний содержат предметные знания, которые представляются в явном виде.

Для интеграции компонентов явно специфицируются единые или специфичные правила связывания, т.е. внутренние интерфейсы (обозначим их $\Delta\iota\varepsilon$), между программным решателем и базами знаний или другими хранилищами (API к KB_i и DB_p из $Solv_1$), взаимодействие между основной логикой $Solv_1$ и элементами Ui_m:

KBS-Arch = $<\cup^{i=1,N1} (KB_i), Solv_1, \cup^{m=0,M} (Ui_m), \cup^{p=0,N2} (DB_p)>$

$\cup <\cup^{p=1,N2} (\Delta\iota\varepsilon\,(Solv_1, KB_p))$,

$\cup^{p=1,N3} (\Delta\iota\varepsilon\,(Solv_1, DB_p))$,

$\cup^{m=1,M} (\Delta\iota\varepsilon\,(Solv_1, Ui_m))>$.

База фактов (каждая DB_p) содержит набор фактов (утверждений), наблюдаемых или объективно измеренных в рассматриваемой ситуации, относительно которых решается задача.

3.2.1. Определение основных подзадач для составных задач

В литературе по экспертным системам (ЭС) часто отмечают, что они решают (предназначены для поддержки решения) взаимосвязанную комбинацию задач – интеллектуальных или одной интеллектуальной задачи и нескольких не интеллектуальных, использующих общую информацию. *Системный анализ* ПрОбл, предшествующий началу автоматизации интеллектуальной деятельности, выявляет все задачи, нуждающиеся в информационной поддержке на основе знаний [105,151]. Повседневная интеллектуальная деятельность часто бывает связана с принятием разных типов решений: сбор информации, диагностика, ремонт, прогноз, и т.п., часть которых является интел-

лектуальными задачами. Примеры в медицине: сбор наблюдений о больном, установление диагноза, назначение персонифицированного лечения, прогноз состояния и наблюдаемых значений признаков (Рисунок 3-1). (Примерами неинтеллектуальных задач в медицине можно считать сбор первичной информации, назначение общей схемы лечения при наличии диагноза, мониторинг значений признаков на соответствие их прогнозируемым).

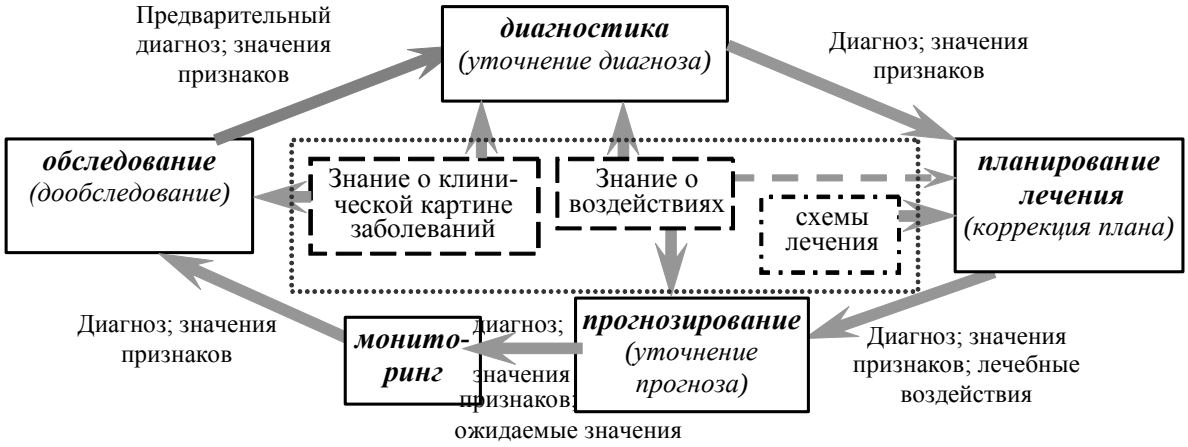


Рисунок 3-1 – Пример совокупности задач, связанных общей информацией

Составной задачей интеллектуальной деятельности будем считать совокупность не менее двух совместно решаемых интеллектуальных задач (и, возможно, нескольких неинтеллектуальных задач), связанных общей информацией.

Примером составной задачи является *задача синтеза реализуемого проекта*. Реализуемость проекта системы означает, что может быть построен план создания системы по этому проекту. Действия и состояния в этом плане связаны с элементами, из которых создается проект. Для *реализуемого проекта* одновременно ставятся *и задача синтеза проекта*, и *задача синтеза плана построения системы* по синтезированному проекту, т.о. задача синтеза реализуемого проекта является составной. Частным случаем составной задачи синтеза реализуемого проекта является *задача синтеза реализуемой конфигурации* (вместе с задачей синтеза плана сборки системы по проекту конфигурации). Составной задачей можно считать *регулярный мониторинг состояния*, когда требуется определять, является ли состояние системы отличным от нейтрального, регулярно в заданные моменты времени или с заданным интервалом (пример –

наблюдение за больными в палатах интенсивной терапии, регулярные медосмотры или диспансеризация сотрудников предприятий).

Наконец, еще одним примером составной задачи является *задача технической диагностики* автономного робота, решаемая в процессе выполнения им миссии (задания). Автономный робот – это система, состоящая из нескольких подсистем – двигательной подсистемы, подсистемы энергопитания, различных сенсорных систем, часто дублирующих друг друга для повышения надежности. В процессе выполнения миссии подсистемы или их отдельные узлы могут выходить из строя. При этом робот в пределах возможного должен продолжать выполнение миссии, возможно, в усеченном виде. Задача технической диагностики состоит в постоянном *мониторинге технического состояния* робота, в ходе которого решается интеллектуальная задача *обнаружения возможных диагностических ситуаций*, связанных с выходом из строя отдельного узла; если обнаруженная диагностическая ситуация не дает однозначного ответа на вопрос, какой узел вышел из строя, то миссия приостанавливается, в соответствие с базой знаний выполняются тестовые действия для этой диагностической ситуации и анализируются их результаты; если установлен вышедший из строя узел, то вносятся изменения в модель робота, если необходимо, вносятся изменения в систему управления роботом (например, переключение с вышедшей из строя сенсорной подсистемы, на дублирующую ее, если таковая имеется и работоспособна) или выполняется перепланирование миссии с учетом изменившейся модели робота.

Вышеописанная многоуровневая классификация задач с их постановками в едином формализме позволяет более четко вычленять, определять основные подзадачи для проблем, решаемых в произвольной предметной области, а также уточнять их в зависимости от (уточнений) свойств предметных областей, в которых они появляются.

Системное проектирование [1,160] распределяет интеллектуальные задачи и их подзадачи и сопутствующие функциональные требования по компонентам (системы) известных типов. В результате СБЗ, предназначенная для решения нескольких подзадач разного типа, имеет в своей архитектуре несколько программных решателей, каждый из которых может обращаться к разным БЗ. СБЗ как разновидность ПС может также:

- включать в себя базы данных для хранения справочной или другой информации,

- принимать на вход файлы или базы данных с оперативной информацией (о ситуации, относительно которой требуется принять решение),

- создавать в качестве результата файлы или базы данных с результатами работы и объяснением принятого решения).

Современные СБЗ часто интегрируются со статистическими и другими программными компонентами, не использующими формализованные знания, но реализующими вычислительные и другие функции (в т.ч. со стандартными пакетами прикладных программ) [88,90].

Системный анализ и системное проектирование – ключевые этапы технологии создания жизнеспособных СБЗ, задающие их архитектуру как совокупность решателей подзадач.

Архитектурная модель для системы *с базой знаний*, интегрированной с (другими) программными компонентами ($Soft_k$), реализующими вычислительные и другие функции, можно представить так.

KBS-Arch = $<\cup^{i=1,N1}(KB_i), \cup^{j=1,N2}(Solv_j), \cup^{m=1,\ N3}(Ui_m), \cup^{p=1,N4}(DB_p), \cup^{k=1,N5}(Soft_k)>\cup<\cup^{1,N6}(\Delta\iota\varepsilon(KB_i, Solv_j)), \cup^{p=1,N2}(\Delta\iota\varepsilon(Solv_j, DB_p)), \cup^{m=1,M}(\Delta\iota\varepsilon(Solv_1, Ui_m)), \cup^{1,N8}(\Delta\iota\varepsilon(Solv_j, Solv_r)),\cup^{1,N9}(\Delta\iota\varepsilon(Solv_j - Soft_k \mid Soft_k\text{-}Ui_m \mid Soft_k\text{-}ExtI_n \mid Soft_k\text{-}DB_p \mid Soft_k\text{-}Soft_s))>$.

Современная парадигма автоматизации интеллектуальной деятельности требует не только дать возможность использовать при решении интеллектуальных задач самые современные и проверенные знания, но и позволить группе экспертов, ответственных за уровень знаний, создавать стандарты знаний. При такой автоматизации единая интеллектуальная программно-информационная система должна «комбинировать» поддержку решения *задач* на основе знаний со средствами формирования и коррекции используемых баз знаний, в частности, позволяющими решать *задачу индукции* БЗ.

3.2.2 Роль онтологии в построении декларативных БЗ

При разработке систем, основанных на знаниях, специальное требование предъявляется к представлению информации – все должно быть в форме, понятной пользователям без посредников: данные, знания, результат (гипотезы о результате решения задач и объяснение гипотез) [37]. Для обеспечения понимаемости (информации) сообществом специалистов должна быть выявлена и формально задана концептуализация ПрОбл (т.е.

ее онтология [9,19,121,143]), охватывающая все виды связей и зависимостей между понятиями. Знания и другая информация должны формироваться в единых понятиях предметной области в соответствии со структурой, ограничениями и соглашениями, определяемыми онтологией ПрОбл.

Онтология, согласно определениям [55, 127,139,140,141,142] – явное описание (на некотором языке) смысла понятий (терминов), определяющих концептуализацию в ПрОбл. В национальном стандарте РФ «Системы искусственного интеллекта. Классификация систем искусственного интеллекта» онтология – формальное представление множества понятий в рамках некоторой области, а также отношения между этими понятиями (*ГОСТ Р 59277-2020).* Часто пишут, что содержимым онтологии могут являться: описания типов объектов реального мира (систем), их свойств и связей объектов; правила получения суждений об объектах на основании имеющихся сведений (правила логического вывода); правила взаимодействия объектов разных типов между собой; правила обработки информационных объектов различных типов со ссылками на свойства предметной области [51,143].

Принципиально важно, что онтология ПрОбл создается как самостоятельный (отдельно хранимый и адресуемый) компонент. При промышленной разработке СБЗ ее окончательная версия должна существовать началу работ по созданию компонентов системы (поскольку все компоненты, реализующие интеллектуальность в системе, разрабатываются на ее основе).

Онтология (Onto) для решения *интеллектуальной задачи* включает *онтологию знаний* (KnOnt) об отношениях понятий, существенных для этой задачи, *онтологию ситуаций* (SitOnt), т.е. входных данных (фактов) и ожидаемых результатов решения; ограничения интерпретации для этих понятий и *онтологические соглашения* (Agreem) о правилах сопоставления фактов знаниям. Номенклатура названий для именования понятий, разрешенных или рекомендуемых в сообществе специалистов, часто формируется в самостоятельную *базу названий* (справочник (Dict), терминологический словарь или классификатор [152]) и может считаться частью онтологии, поскольку является результатом соглашения экспертов.

Onto = KnOnt ∪ SitOnt ∪Agreem ∪ Dict.

Каждая база знаний, формируемая на основе связей понятий предметной области KnOnt с использованием *названий* понятий из Dict, – KB_i(KnOnt, Dict) – является онтолого-ориентированной (или онтологической) базой знаний [136]. Такие БЗ зависят от онтологии, как должны зависеть и данные (DB_j (SitOnt, Dict)) [98], поэтому в отличие от традиционной архитектуры в СБЗ добавится новый компонент – онтология: СБЗ = онтология + онтологическая база знаний + онтологическая база фактов + решатель задачи + интерфейс интеллектуальный:

Модель СБЗ выглядит как: $<\cup^{i=1,N1}$ (KB_i(KnOnt, Dict)), $\cup^{j=1,N2}(Solv_j)$, $\cup^{m=1,N3}$ (Ui_m), $\cup^{p=1,N4}$(DB_p (SitOnt, Dict)), $\cup^{k=1,N5}(Soft_k)>$.

В онтологии предметной области определены типы утверждений, позволяющих решать задачи в предметной области на основе сопоставления входных данных знаниям. К ним относятся причинно-следственные зависимости причин развития процессов разных типов $f_{in,}$, связи этапов развития с проявлениями этапов, описания правил соединения в единый проект взаимозависимых элементов, *действия*, которые можно выполнять для изменения состояния системы, *предусловия* на состояние, в котором действие можно применять и т.д.

В части Agreem определены типы утверждений о соответствии знаний данным, о сопоставлении данных образцам, представленным в знаниях, что необходимо при решении задачи в предметной области. Например, знания о зависимостях между фактами R^{kn}(moments) и условиями Cond и знаниями о развитии процессов разных типов, в том числе внутренних f_{in}. Так, для диагностики процессов одним из самых распространенных типов предложений является утверждение об обязательных проявлениях (с названием каждого признака объекта или процесса и ожидаемого диапазона значений признака) диагностируемого отклонения или варианта проявлений: <класс отклонений-k, признак-j, диапазон значений-kj признака-j>; а также предложения типа «необходимое условие существования процесса». Для управления процессом одним из самых распространенных – <класс-k, признак-j, диапазон значений-kj признака-j; воздействие-m, характеристика воздействия-m, ожидаемый диапазон значений-kj признака-j>; для прогноза развития процесса – <класс-k, признак-j, диапазон значений-kj признака-j в момент времени t1; фактор-m, характеристика фактора-m, момент времени t2, t2>t1, ожидаемый диапазон значений-kj признака-j в момент времени t2> [153].

Пример. Онтология знаний (KnOnt) о «диагностике процессов» содержит следующий набор связей. Каждый аномальный процесс связан с некоторым набором альтернативных вариантов развития (аномалии):

f_{in} = {Вариант развития процесса)}^{n2} +

+ [Условие протекания процесса]

+ {Событие, которое вызывает процесс r_{ev}}^{n3}

+ {Фактор, провоцирующий процесс r_f}^{n4}.

Вариант развития процесса = {Проявления очередного периода развития}^{n4} , число периодов n4 > 0.

Проявления очередного периода = Порядковый номер периода +

+ Интервал длительности периода

+ Комплекс проявлений/наблюдений.

Комплекс проявлений выражается хотя бы одним из наборов – набором необходимого числа характерных признаков и набором признаков разной степени важности:

Комплекс проявлений = [Комплекс характерных наблюдений] +

+ {Признак + модальность}^{n6}, n6>=0.

Комплекс характерных наблюдений = {Признак}^{n7}

+ минимальное количество, n7>1.

Для признака или его элемента/характеристики:

«модальность» = возможно | характерно | необходимо.

Для целостного представления известных симптомов для признака предусмотрена возможность быть комплексным – составленным из элементов или характеристик, также выступающих в роли проявлений процессов; с простым признаком и каждым элементом составного всегда связан диапазон возможных наблюдаемых значений:

<название наблюдаемого признака, диапазон возможных значений>

или {<элемент/характеристика наблюдаемого признака;

диапазон возможных значений>}^{n8}.

Условие протекания процесса =

{(характеристика объекта + значение)

| (событие + временнАя характеристика)}

Фактор, провоцирующий процесс =

(характеристика объекта + значение)

| (событие + временнАя характеристика).

Примечание. В этих отношениях в роли причин выступают f_{in}, также могут выступать: характеристика объекта r^{kn}_{O}, событие (и воздействующее событие) + r^{kn}_{ev}. Следствиями являются остальные понятия.

Для значений признаков и элементов / характеристик предлагаются числовые элементы (поля) с единицами изменения и строковые.

В каждом периоде своей динамики признак (или любой(-ая) элемент / характеристика) в качестве диапазона значений может иметь несколько поддиапазонов или несколько вариантов значений.

Для учета вариантов динамики значений признаков и многообразия вариантов течения процессов каждый период динамики задается верхней и нижней границей длительности периода, единицей измерения границ.

Фактор, провоцирующий процесс = $fc \in R^{kn}_{O} + R^{kn}_{ev}$

Т.е. диагноз связывается с факторами: постоянные характеристики объекта (с их значениями), происходящие события (и момент совершения).

Для возможных значений признака r^{kn}_{ex} (t0, ..., t_k) или отдельных его элементов (например, признака «изменения на листьях» или элемента этого признака «посветление жилок»], могут быть описаны изменения значения этого признака под воздействием некоторых событий r^{kn}_{ev} (внешних воздействий на объект) на разных этапах (аномального) процесса f_{in}:

(f_{in}, название Признака r^{kn}_{ex}

воздействующее событие)

= {номер периода

+ диапазон значений признака или его элемента, возможных для периода,

+ временнАя характеристика события

+ диапазон значений признака после воздействия}.

Соглашения онтологии (Agreem) включают следующие типы отношений (утверждений):

(1) о соответствии фактов из Ситуаций Действительности – проявлениям или необходимым Условиям, важным согласно Знаниям о процессах. Факты из Ситуаций Действительности $\in R(t) = R_{ex}(t_0, ..., t_k) \cup R_O \cup R_{ev}(t_u..., t_v)$. Ожидаемые проявления $\in R^{kn}$ (moments). Здесь moments $\in$ Periods – следующим друг за другом периодам развития рассматриваемого процесса (successive periods of ongoing process).

Примечание. В этих отношениях в роли *посылок* (причин для принятия промежуточных решений) выступают факты из Ситуаций Действительности $R(t) = R_{ex}(t_0, ..., t_k) \cup R_O \cup R_{ev}(t_u..., t_v)$ и ожидаемые проявления из множества R^{kn} (moments). *Следствиями* являются значения предикатов их соответствия Pred (r(t), r^{kn} (mom)).

(2) о соответствии подтвержденных проявлений (условий) рассматриваемых явлений – фактам из Действительности (истинным значениям предикатов Pred (r(t), r^{kn} (mom)) – самим явлениям. В этих отношениях в роли *посылок* (причин для принятия промежуточных решений) выступают ожидаемые проявления из множестваR^{kn} (moments) и значения предикатов. *Следствиями* являются значения предикатов для левых частей формул из набора связей Kn: Комплекс характерных наблюдений, и далее по мере получения истинных значений предикатов – Комплекс Проявлений, Проявления очередного периода, Вариант развития внутреннего процесса, f_{in}.

Количество правил, которые следует «обработать» в процессе решения задачи диагностики, невелико: поиск в документе, описывающем действительность, фактов для получения значений предикатов ожидаемых проявлений, и вывод значений предикатов из модели знаний по полученным значениям предикатов.

Пример поиска:

поиск во входных данных (в документе) значений наблюдений {r^{O}_{k}, r^{O}_{k+1},…, $r^{ev}_{u}(t_{v1})$, $r^{ev}_{u+1}(t_{v1})$, $r^{ev}_{u}(t_{v2})$, …}, которые соответствуют Условию из r^{kn} (mom) для рассматриваемой группы аномальных процессов Gr_{in}, т.е. вычисление Pred (r(t), $r^{kn\text{-}gr\text{-}cond}$(mom));

поиск в документе {r^{O}_{k}, r^{O}_{k+1},…, $r^{ev}_{u}(t_{v1})$, $r^{ev}_{u+1}(t_{v1})$, $r^{ev}_{u}(t_{v2})$, …}, которые соответствуют Условию из r^{kn} (mom) для рассматриваемого Варианта развития рассматриваемого аномального процесса f_{in}, т.е. вычисление Pred (r(t), $r^{kn\text{-}cond}$ (mom)).

Пример вывода:

проверка истинности Pred (r(t), $r^{kn\text{-}cond}$ (mom)) и Pred (r(t), $r^{kn\text{-}cmplx}$ (mom)) для подтверждения Pred (r(t), $r^{kn\text{-}var\text{-}period}$).

Такого рода онтология предметной области является системой понятий, охватывающей рассматриваемые процессы и удовлетворяющей всех участников разработки: в соответствии с KnOnt и Dict формируется БЗ, в соответствии с KnOnt, SitOnt и Agreem строится алгоритм решения (проверки и выдвижения гипотез), в соответствии с SitOnt и Dict – разработка интерфейса пользователя.

3.2.3. Онтологический подход к определению компонентов интеллектуальных систем

«Модель предметной области (онтология) и модель процессов обработки информации в системе технологически неразрывны. При изменении модели предметной области меняются правила обработки и алгоритмы обработки» [51].

С учетом того, что решатель должен правильно сопоставлять представленные в терминах онтологии данные (DB_p (SitOnt, Dict)) онтологической базе знаний, он должен быть онтолого-ориентированным – $Solv_j$(KnOnt,SitOnt, Agreem), также как Ui_m(SitOnt) для взаимодействия с ним) [37,136].

По сравнению с универсальным решателем-рассуждателем, интерпретирующим продукционные правила [98], «специализированный на конкретную онтологию» решатель задачи обрабатывает знания, понимаемые экспертом.

Включение в архитектуру декларативных БЗ (одновременно понятных человеку и программному обработчику) – ключевой принцип создания жизнеспособных СБЗ.

С учетом этого модель жизнеспособной СБЗ выглядит как:

$<\cup^{i=1,N1}$ (KB_i(KnOnt, Dict)), $\cup^{j=1,N2}$($Solv_j$(KnOnt,SitOnt, Agreem)), $\cup^{m=1,N3}$ (Ui_m(SitOnt)), $\cup^{p=1,N4}$(DB_p (SitOnt, Dict)), $\cup^{k=1,N5}$($Soft_k$)>$\cup$<$\cup^{1,*}$ ($\Delta\iota\varepsilon$ (KB_i - $Solv_j$ | | $Solv_j$ - DB_p| $Solv_1$ - Ui_m| $Solv_j$ - $Solv_r$| $Solv_j$ - $Soft_k$ | $Soft_k$-Ui_m| $Soft_k$-$ExtI_n$| $Soft_k$-DB_p| $Soft_k$-$Soft_s$))>.

Онтология знаний чаще соответствует классу решаемых задач или одному классу задач в одной предметной области. Для конструирования прикладных СБЗ может потребоваться взаимосвязанная совокупность из нескольких «самостоятельных» онтологий знаний, например, онтология знаний об изменении процесса, в том числе из-за влияния воздействующих факторов, и онтология знаний о свойствах воздействующих средств, в

том числе о влиянии на развитие процессов. Если строится прикладная СБЗ для поддержки решения нескольких смежных классов задач (например, диагностика неполадок и исправление неполадок), то понадобятся все соответствующие онтологии знаний по каждой задаче.

Такая онтология становится структурной основой не только инструментов для экспертов (средств редактирования), но и программных компонентов СБЗ (решателей интеллектуальных задач и интерфейса) [136].

Поскольку онтология отделяется от собственно базы знаний, решатель задачи реализует специализированный алгоритм на основе множества всех перечисленных типов предложений и отношений из онтологий знаний и действительности (явно представленных) для этой конкретной задачи.

Онтологическая база знаний (версия, удовлетворяющая специалистов) будет заменяемым компонентом («фактическим параметром») для такого решателя [136].

Онтолого-ориентированный алгоритм (ontological reasoner) производит для поиска либо опровержения гипотез обход (декларативной) базы знаний.

К онтологии данных принято относить и форму представления результата решения задачи (как его формирует сам специалист), и структуру аргументированного разъяснения результата, как если бы оно предлагалось специалисту некоторым советчиком, консультантом. Использование онтологии для формирования базы знаний, данных (фактов) и выходной информации дает возможность сгенерировать пользовательский интерфейс в терминах, понятных специалисту. Такой онтолого-зависимый Ui будет адаптируем к изменениям в онтологии данных, знаний, результата, объяснения. В другом случае Ui программируется в соответствии с прочими требованиями к usability и реализует желаемый сценарий диалога.

Таким образом, в составе жизнеспособной СБЗ должны быть:

база знаний, выраженная в терминах онтологии,

онтологический решатель (умеющий выдвигать и объяснять гипотезы),

онтолого-зависимый Ui.

Примечание. В состав жизнеспособной СБЗ входит и онтология ПрОбл; на ней базируются компоненты системы. Часто онтология ПрОбл становится основой для

множества разных СБЗ, создаваемых на Портале (т.е. является фундаментом всего тематического портала).

Характерными свойствами СБЗ, относящимися к жизнеспособности, являются:

- регулярная обновляемость знаний;
- допустимость усовершенствования метода принятия решения;
- допустимость изменения или добавления функций (например, формирования дополнительных результатов);
- адаптивность интерфейса пользователя в связи с изменением входных данных и формируемых результатов функций.

Поэтому создание жизнеспособной СБЗ подразумевает соответствующие механизмы адаптации.

3.3 Архитектурно-технологическая модель развиваемой СБЗ

Конструирование декларативных БЗ обеспечивается онтолого-ориентированным подходом. Онтология задает формат и правила представления всей информации ПрОбл. БЗ формируются вручную или индуктивно, в том числе по обучающим выборкам из архивов и баз данных (индуктивное обобщение в машинном обучении [98], байесовские классификаторы, алгоритмы кластеризации и обучение с подкреплением [79,128].

Создание большой базы знаний – сложная задача, по мере увеличения размера БЗ и увеличения прогнозируемой продолжительности жизни БЗ становится все более важным разрешать ввод, модификацию, отладку и обслуживание специалистами в данной области, для их поддержки необходимо создавать расширенные среды [124].

Естественно, что система управления БЗ (СУБЗ) должна допускать включение новых научных результатов в БЗ. (Для ответственных отраслевых задач такая модификация БЗ будет выполняться экспертами, входящими в особую группу управления базами знаний.) Поэтому требуется инструментарий для добавления и изменения информации в БЗ: формирования в соответствии с онтологией знаний (KnOnt) с учетом ограничений интерпретации и с использованием названий из справочника (Dict) в процессе формирования. От инструментария (редактора БЗ) и его Ui требуется соответствовать требованиям и ожиданиям экспертов предметной области.

«Большие задачи с большим сроком службы увеличивают вероятность того, что разработчики и сопровождающие базы знаний сформируют распределенную команду, потенциально с различными «взглядами» и словарями. Скорее всего, БЗ будут результатом слияния нескольких экспертов в данной области» [124].

«Успешность» интерактивного изменения (адаптируемости) зависит от онтологии предметной области, которая является системой понятий, удовлетворяющей всех участников разработки. Dict, содержащий определения всех понятий {O, F, Pr} и охватывающий допускаемую сообществом синонимию, является наиболее естественным и уместным. (Заставлять специалистов использовать непривычные им термины – значит отказаться от их участия.) В некоторых ПрОбл могут быть единые словари терминов.

Ввиду важности развиваемости и эволюционируемости баз знаний создаваемая СБЗ интегрируется с СУБЗ, т.е. программные компоненты для развития баз знаний, например, средства проверки баз знаний, оценивания их качества на эталонных архивах решенных задач – часть «интегрированной архитектуры».

Таким образом, необходимо разработать среды для поддержки совместного проектирования и обслуживания БЗ, для поддержки объединения баз знаний из разных источников.

3.4. Инструментальная среда для развития

«Элементы» инструментария (требуемого для создания и улучшения СБЗ) должны обладать возможностями и ограничениями для соответствия создаваемых компонентов онтологии предметной области: информационные компоненты имеют заданный формат, а программные умеют работать с любыми информационными, построенными «под управлением» онтологии [34].

Инструментарий для конструирования прикладных систем, обрабатывающих декларативные знания, основан на онтологии предметной области, поскольку алгоритмы обрабатывают знания и другую информацию предметной области. Обрабатываемая информация – формализованные знания о законах предметной области, вводимые через Ui факты, хранимые в базах данных архивные данные и документы). Онтологические соглашения о правилах сопоставления фактов знаниям должны быть известны разработчикам алгоритмов решения задач. Знания и другая информация должны формироваться в единых понятиях предметной области (составляющих ее терминологический справоч-

ник) и в соответствии со структурой, определяемой онтологией ПрОбл, включающей онтологии знаний, входных данных и ожидаемых результатов.

Назовем инструментарий, позволяющий создавать компоненты СБЗ на базе онтологии ПрОбл, *инструментальной средой развития СБЗ* [34]. Такая среда развития должна, как минимум, предоставлять:

- Средства редактирования БЗ (Редактор БЗ),

- Средства ввода данных и/или редактирования баз данных и просмотра данных и результатов,

- Средства формирования структуры результатов решения и их объяснения,

- Библиотеку программных решателей, соответствующих классам решаемых задач, а также средства поиска и выбора повторно используемых программных решателей,

- Средства интеграции БЗ и решателей,

- Специализированный редактор для Ui.

Этот «минимум» должен предоставлять инструментарий, позволяющий создавать и улучшать все компоненты систем с базами знаний.

Для формирования программных решателей понадобятся:

- Средства кодирования новых программных единиц (ПрЕд) для решателя или их новых версий,

- Средства интеграции ПИК и новых ПрЕд в новые решатели или их новые версии.

Примечание. Возможность создания программных единиц для решателя может оказаться излишней (в среде развития), если готовый решатель вырабатывает правильные гипотезы в соответствии с постановкой задачи, а «наращивание» дополнительной функциональности не предполагается.

Тогда среда развития может предоставлять:

- Средства проверки баз знаний,

- Средства оценивания качества баз знаний на эталонных архивах решенных задач.

- Средства индуктивного формирования баз знаний или фрагментов БЗ, интегрируемых с ней.

Вышеперечисленным свойствам жизнеспособной СБЗ соответствуют характерные свойства *инструментальной* среды ее развития:

поддержка обновления знаний;

поддержка изменения компоновки онтолого-ориентированного решателя (в связи с заменой компонента, реализующего другую стратегию принятия решения или метода получения результата или в связи с добавлением компонента, реализующего дополнительную функцию, например, формирования дополнительного результата);

поддержка усовершенствования интерфейса пользователя в связи с изменением функций;

поддержка изменения в онтологии;

поддержка усовершенствования интерфейса эксперта (и пользователя) в связи с обновлением онтологии;

поддержка кодирования новых версий решателя или поддержка изменения кода.

Таковы на современном этапе требования к инструментарию, реализующему онтолого-ориентированный подход к созданию и улучшению БЗ [34].

Таким образом, построение СБЗ с помощью *инструментальных* онтолого-ориентированных сред развития, принципиально увеличивает их жизнеспособность, существенно увеличив роль и долю средств управления (для внесения изменений в декларативные компоненты) по отношению к средствам сопровождения (для внесения изменений в исходный код) [30,133].

Примечание. В ситуации, когда не исключается возможность внесения изменений в онтологию ПрОбл и после построения баз знаний, среда развития СОЗ должна предоставлять инструмент редактирования онтологии. Внесение дополнительных понятий или новых связей между понятиями в онтологию не нарушает работоспособности программных решателей. (Онтологический решатель обладает таким свойством, поскольку обрабатывает в Базе знаний только известные ему типы отношений.)

К характерным свойствам СБЗ, относящиеся к жизнеспособности, можно добавить:

- допустимость расширения онтологии (добавления понятий, отношений).

Изменения в онтологии, как правило, ведут к корректировке всех компонентов СОЗ; в этом случае целесообразно говорить о новой версии СОЗ.

3.5. Выводы к главе 3

Таким образом, в составе жизнеспособной СБЗ должны быть:

- развиваемая база знаний, выраженная в терминах онтологии, и сама онтология,

- онтологический решатель (умеющий выдвигать и объяснять гипотезы по информации из любой базы знаний, выраженной в терминах этой онтологии),

- онтолого-зависимый адаптивный и адаптируемый Ui.

Эти компоненты должны обладать свойствами:

1. БЗ отделена от онтологии, по которой может быть создано целое множество БЗ или их версий. Это обеспечивает возможность корректировать БЗ без необходимости изменения других компонентов СБЗ,

2. онтологический решатель отделен от структурных знаний ПрОбл, для его реализации используется онтология: онтологические соглашения (процедурные знания) о правилах сопоставления фактов знаниям и структурные свойства информации.

Ключевые принципы разработки жизнеспособной СБЗ:

- проведение системного анализа и выбор адекватных проблеме класса(-ов) интеллектуальных задач, позволяющих использовать соответствующие им модели знаний и базовые свойства ПрОбл для формирования онтологии ПрОбл (или определить дополнительные свойства «своей» ПрОбл для расширения или уточнения существующих);

- проведение системного проектирования для определения набора решателей для выделенных подзадач (рассматривая возможные готовые решатели);

- формирование составных частей (разных типов компонентов – силами узких специалистов) на основе выбранной онтологии ПрОбл, с которой связан опыт решений и опробованные методы обработки информации;

- использование инструментальной поддержки, реализующей механизмы адаптивности, т.е. усовершенствования каждого компонента, гарантируя целостность всей СБЗ.

ГЛАВА 4. Методы обеспечения качества декларативных информационных компонентов интеллектуальной системы на основе знаний

В этой главе представлены методы обеспечения качества формируемых декларативных информационных компонентов для СБЗ: метод оценивания и перманентного усовершенствования баз знаний, метод оценивания структурных свойств онтологических информационных компонентов, их влияния на процесс формирования компонентов системы, поиска дефектов или недостатков.

4.1. Обеспечение качества баз знаний

Для построения СБЗ, способных выдавать аргументированный совет или решение, принципиально важны адекватные современному уровню базы знаний [30,133]. Базы знаний (БЗ), содержащие предметные знания, формируются силами экспертов (вручную) или индуктивно (автоматически) по обучающим выборкам из архивов и баз данных. При этом в сложных и ответственных областях есть все основания использовать оба способа формирования БЗ [150].

При решении задачи проводится анализ – перебор множества элементов знаний о связях (гипотез с описаниями допустимых фактов) из БЗ. Результаты (точность или правильность) зависят преимущественно от качества применяемых в этом анализе знаний, поскольку отлаженный решатель будет применять базу знаний правильно (поскольку решатель задачи – это программа, его можно проверить традиционными методами – верификацией и тестированием [160]). Поэтому для СБЗ обеспечение качества «сводится», прежде всего, к корректности знаний.

4.1.1. Важность качества баз знаний

Существенная работа по обеспечению качества СБЗ относится к используемым знаниям. Если база знаний имеет высокое качество, автоматизация позволяет снизить

долю ошибок специалистов из-за нехватки знаний или неправильного применения знаний [30,133,150].

Качество и полезность базы знаний определяются полнотой, точностью и правильностью содержащихся в ней знаний [150]. Концептуальная часть знаний (часто сводимая к определению классов и свойств и называемая TBox, Terminology box) требует проверки формы [17,18] и консистентности [21]. *Качество* знаний (и баз знаний) представляет интерес и с точки зрения правильности результатов применения знаний при автоматизированном решении задач и с точки зрения применения знаний экспертом или пользователем, чтобы быстро найти нужный раздел базы знаний в ходе получения (выявления) знаний или даже наглядно оценить качество базы знаний экспертом [42]. Для применения знаний важно оценивать *достоверность* (к моменту применения БЗ содержит все требуемые записи с актуальным наполнением), требуемую степень точности.

Автоматизируют проверку непротиворечивости [42]. Для БЗ в виде правил предлагаются автоматизированные средства проверки «наличия у аксиом с одним и тем же антецедентом противоречивых последовательностей», «наличия идентичных последовательностей с разными антецедентами», противоречивых следствий из пересекающихся условий-посылок («проверки, что два предыдущих условия, которые не являются непересекающимися, не имеют противоречивых следствий») [124]). Также делают поиск вероятных источниками конфликтов – объектов, которые могут оказаться экземплярами или подклассами двух классов (разделов), которые определены как непересекающиеся в рассматриваемой области, или когда один из двух классов является отрицанием другого [124]. Запускаются тесты на поиск в базе знаний нарушения ограничений «кардинальности» (количественных спецификаторов отношений понятий) [124] .

Созданная экспертом база знаний может быть неполной (некоторые варианты могут быть упущены экспертом или быть ему неизвестны), неточной (может приводить к неоднозначным решениям) или даже неправильной (из-за заблуждений, предубеждений). В литературе отмечают важность оценивания *связности* [29; 42], *полноты, неизбыточности* (как подтвержденная необходимость наличия записи, так и отсутствие дублей), *использование только допустимых значений* (элементы или поля структур содержат только допустимые значения), *использование допустимых ссылок, согласованность формата и единство представления однотипной информации* (формат представ-

ления согласован и един для всех однотипных данных) и способы их оценивания и способы оценки *длины цепочки вывода* [42].

Исследователи стремятся иметь возможность делать некоторые количественные оценки качества БЗ [42,124]. Если *мерой качества* знаний считать то множество задач, которые именно эти знания позволяют решить правильно (подразумеваем, что специалист или их сообщество знают их правильное решение), то можно получать количественную оценку или оперировать «множеством».

Примечание. При оперировании *мерой качества* знаний на практике для конкретных знаний для решения конкретных интеллектуальных задач придется считать множество задач с конкретными входными условиями, для которых найти правильное решение позволяют именно эти знания.

СБЗ станет *полезной* для некоторого специалиста, если *мера качества ее БЗ* «лучше» меры качества знаний этого специалиста. Т.е. с помощью БЗ правильно можно решить б*о*льшее множество задач, чем, полагаясь только на его собственные знания. А для группы специалистов СБЗ можно считать полезной, если множество задач, решение которых известно хотя бы одному специалисту из этой группы, является подмножеством множества правильно (точно) решаемых системой задач. Специалист ПрОбл, принимая решение (например, ставя диагноз) на основании данных об объекте (системе, ситуации), руководствуется своими знаниями, но, имея СБЗ, получает возможность учесть формируемое ею объяснение (анализ возможных гипотез) как консультацию [150].

Определяемая таким способом *мера базы знаний* зависит от *множества задач с известным решением* (базы правильных решений или базы эталонных прецедентов).

В сложных и ответственных областях есть все основания использовать оба способа формирования БЗ: вручную (с помощью инструмента редактирования) или индуктивно (с помощью инструмента обучения). В первом случае качество зависит от человеческого фактора, во втором, от возможности собрать достаточное количество проверенных обучающих примеров.

Онтолого-ориентированный подход к формированию БЗ обеспечивает наглядное и понятное представление, соответствующее системе терминов специалистов. Для формирования БЗ вручную используется онтологический инструмент Ed(KnOnt), он обеспечивает редактирование под управлением онтологии: т.е. в строго предопределенной струк-

туре и с учетом правил и ограничений вносимой информации. Благодаря такому «управлению» сформированная БЗ пригодна для программной обработки.

Для формирования БЗ индуктивно требуется онтологический инструмент обучения TrnSft(KnOnt, PrecB) – для извлечения знаний из базы обучающих решенных задач – прецедентов (под управлением онтологии).

Т.о. онтолого-ориентированный подход к формированию БЗ обеспечивает декларативное представление – программно-обрабатываемое и одновременно наглядное и понятное, соответствующее системе терминов специалистов.

4.1.2. Проблема оценивания качества баз знаний

Основными средствами оценивания баз знаний являются средства контроля формальных свойств построенной БЗ и привлечение экспертов для оценки «решений, предлагаемых системой» [19,93,95].

Если проверка формы онтологии (Tbox) может быть автоматизирована, то полнота терминологии оценивается экспертами и бета-тестировщиками [16]. Традиционно стараются дать возможность экспертам в данной области проверить адекватность и правильность разрабатываемой базы знаний. Инструменты, оценивающие часть формальных свойств Базы Знаний (например, в виде правил) не претендуют на проверку «семантической составляющей» БЗ.

Очевидна необходимость объективного оценивания качества баз знаний. Для «тестирования баз знаний» применяют отдельные диагностические тесты, генерируют тестовые знания и сценарии, используя диагностические инструменты и специализированные средства отладки: инструменты позволяют разработчику описать ситуацию и описать ожидаемый результат, когда он знает, какими будут эти ожидаемые результаты.

Для семантической проверки требуются контрольные примеры (прецеденты). В доступных источниках приводятся примеры программного инструментария для обслуживания базы таких прецедентов (пополнения и модифицирования) и ее использования [85,93].

Для семантической проверки на контрольных примерах (прецедентах) принципиален инструментарий для автоматического сравнения результатов использования БЗ для данных из прецедента (а также для пополнения и модифицирования базы прецедентов).

Введенная выше *мера качества* знаний может быть уточнена с учетом того, что в реальности входные данные часто (а в некоторых ПрОбл почти всегда) неполны. В таких условиях правильное решение – это множество гипотез, содержащее верное решение (а не единственное решение). Это множество будет содержать единственный элемент – решение – при условии полноты входной информации.

Помимо ***контроля*** *полноты, достаточности, правильности знаний*, заслуживают оценивания и такие (менее принципиальные) свойства, например, избыточность:

формальная или явная избыточность – это наличие одинаковых фрагментов базы знаний (семантической сети), «одинаково» связанных с одним и тем же понятием (например, с гипотезой, как например: два одинаковых элемента или подсети во множестве таковых (перечисляемых значений признака или симптомокомплексов диагноза);

«скрытая» избыточность – это наличие схожих фрагментов знаний в описании одной и той же гипотезы (одинаковых вхождений цепочек связей между экземплярами понятий), например, наличие симптомокомплексов одного диагноза,

> который «похожи в главном», или, например, все необходимые и характерные признаки совпадают, наличие симптомокомплексов с одинаковыми признаками, но разными областями значений, но каждая отличная область значений одного является подмножеством области значений другого (их можно объединить).

4.1.3. Проблема непрерывного развития (модификации) знаний

СБЗ должны оказывать поддержку пользователям на уровне совокупного знания специалистов и экспертов (т.е. знаний для решения задач своего класса в БЗ больше, чем у каждого отдельного пользователя, а в сложных и многопрофильных областях – больше, чем у каждого отдельного эксперта). От актуальности (соответствия современному представлению) и качества БЗ зависит успех применения СБЗ (для получения объяснения $y \in Y$ при любых $x \in X$).

Проблемам модификации знаний интеллектуальной системы уделяется значительное внимание более 30 лет. В частности, в литературе описан такой вид работ, как сбор со всех рабочих мест специалистов информации о принятых с помощью системы решений

(«протоколов» с клиническими данными терапии) и передача их «анализатору», чтобы осуществлять по необходимости коррекцию знаний [49].

Зафиксированная БЗ со временем будет устаревать в том смысле, что мера ее качества не будет изменяться, в то время как *база прецедентов* будет расширяться в процессе дальнейшей практики. Расширение в процессе практики *множества прецедентов* означает, что *мера качества* знаний специалиста (группы специалистов) улучшается (они эти случаи поняли и «справились» с ними на практик). Если *мера качества* знаний БЗ не содержит эти случаи/прецеденты как подмножество, то СБЗ с такой БЗ может потерять свою полезность.

Актуальность знаний (базы знаний) достигается: адаптируемостью (интерактивным изменением базы знаний) и адаптивностью (возможностью применять средства индуктивного формирования знаний по обновляемому набору прецедентов).

Для адаптивности, а особенно для адаптируемости (интерактивного изменения базы знаний) принципиально важны программные инструменты развития баз знаний, такие как *средства проверки баз знаний,* и *оценивания их качества* на эталонных архивах решенных задач (AssSft). Онтологии (SitOnt) для представления эталонных задач и документирования решенных задач зафиксированы. Средства оценивания БЗ по эталонным архивам решенных задач ориентированы на онтологии KnOnt и SitOnt. Измененная экспертом и проверенная база (новая версия) есть результат применения таких средств:

$KB_{ver\text{-}2}$ = ($KB_{ver\text{-}1}$(KnOnt) * KBEd(KnOnt)) * AssSft(KnOnt, SamplB(SitOnt).

Примечание. Символом «*» обозначено применение инструмента для усовершенствования БЗ.

Эталонные решения задач поступают из практики (в некоторых областях регулярно). Каждое решение, принимаемое в ходе интеллектуальной деятельности специалистом, в ответственных областях обычно в дальнейшем проходит процедуру верификации. В результате верификации решение специалиста (в том числе принятое с поддержкой системы) признается правильным или ошибочным [59]. Пример. В медицине верификация осуществляется по результату выздоровления пациента в результате лечения, либо по результатам оперативного вмешательства, либо решением анатомов.

Ввиду важности развиваемости баз знаний, следует всерьёз рассматривать только такие СБЗ, которые интегрированы с системой управления ее БЗ (СУБЗ).

4.1.4. Функции системы управления БЗ

Чтобы СБЗ оставалась полезной для специалиста, *мера качества* (как минимум, *мера* правильности, а в идеале – точности) ее БЗ в течение всего времени использования должна оставаться лучше, чем *мера качества* знаний обращающегося к ней специалиста. Поэтому требуется *система управления ее БЗ* [30,133]. Цель системы управления БЗ – обеспечивать полезность СБЗ для специалиста (группы специалистов) в течение всего времени ее эксплуатации.

Для достижения этой цели система управления БЗ должна выполнять следующие функции [59]:

- постоянно накапливать базу прецедентов;
- классифицировать все прецеденты в базе;
- находить все возможные способы модификации БЗ (чтобы новые прецеденты оказались частью множества «мера правильности» БЗ);
- модифицировать БЗ одним из возможных способов (интерактивно или автоматически).

Функция накопления базы прецедентов – получать информацию (как ресурс, доступный для программной обработки) обо всех задачах, решенных специалистами, включая результаты верификации каждого решения задачи - верификации, в результате которой устанавливается правильное решение этой задачи.

Выборки прецедентов (правильно решенных задач) для обучения БЗ формируются по той же онтологии (SitOnt).

Ее естественной реализацией является включение СБЗ в электронный документооборот интеллектуальной деятельности. Для этого информация о принятых специалистами решениях должна включаться в эти документы в форме, допускающей ее обработку системой управления БЗ. Такой электронный документооборот позволяет получать объективные оценки качества решения задач, как отдельными специалистами, так и их группами. Документируется результат решения задачи относительно объекта и все известные данные об объекте (например, для медицины и те, и другие входят в историю болезни (ИБ) пациента): (а) все известные данные об объекте, (б) результат решения задачи специалистом и (в) правильный результат решения задачи. В предметной области, в которой процессы связаны между собой посредством причинно-следственных отно-

шений, прецедентом может стать весь задокументированный процесс наблюдения за объектом и воздействий на него (отчасти так для медицины).

Вышеуказанный правильный результат решения задачи относительно объекта (в) сопоставляется (г) объяснению, полученному от СБЗ. Правильное решение (в) может оказаться не таким, как решение задачи специалистом (б). И правильное решение (в) может оказаться не таким, как решение от СБЗ. Если объяснение (г), полученное от СБЗ, не содержит правильного решения (в) – это и есть повод для улучшения БЗ. В случае несовпадения этих решений следует провести анализ сгенерированного объяснения, чтобы оценить, следует ли пару <известные данные об объекте, решение задачи относительно объекта> ((а)+(в)) использовать как прецедент для уточнения или исправления базы знаний системы. Анализ сгенерированного объяснения (и наличия в нем материала для уточнения или исправления БЗ) подразумевает возможность различать несколько разных классов прецедентов.

Функция классификации прецедентов такова. Каждый прецедент должен быть отнесен системой управления БЗ к одному из следующих классов [59]:

1) система предложила правильное и единственное (т.е. точное) решение;

2) система предложила правильное, но неточное решение (несколько возможных альтернатив, среди которых было и правильное решение);

3) система предложила неправильное решение (множество альтернатив, возможно пустое, среди которых не было правильного решения).

С поддержкой эксперта (отвечающего за качество БЗ) возможна более «тонкая» классификация прецедентов класса 2 (2а – входные данные задачи допускают более точное решение; или 2б – допускают единственное; или 2в – не допускают его уточнения) может быть поручена экспертам – управляющим БЗ. Т.е. более полный набор классов для прецедентов таков (Рисунок 4-1): 1) СБЗ предложила правильное и точное решение; 2а – СБЗ предложила правильное, но неточное решение, но входные данные задачи допускают ее точное решение; 2б – предложила правильное, но неточное решение, но входные данные задачи допускают некоторое его уточнение (уменьшение числа альтернатив); 2 в – предложила правильное, но неточное решение, при этом входные данные задачи не допускают

его уточнения; 3) СБЗ предложила неправильное решение. В общем случае решение об отнесении прецедента к классам 2а, 2б или 2в не может быть принято автоматически.

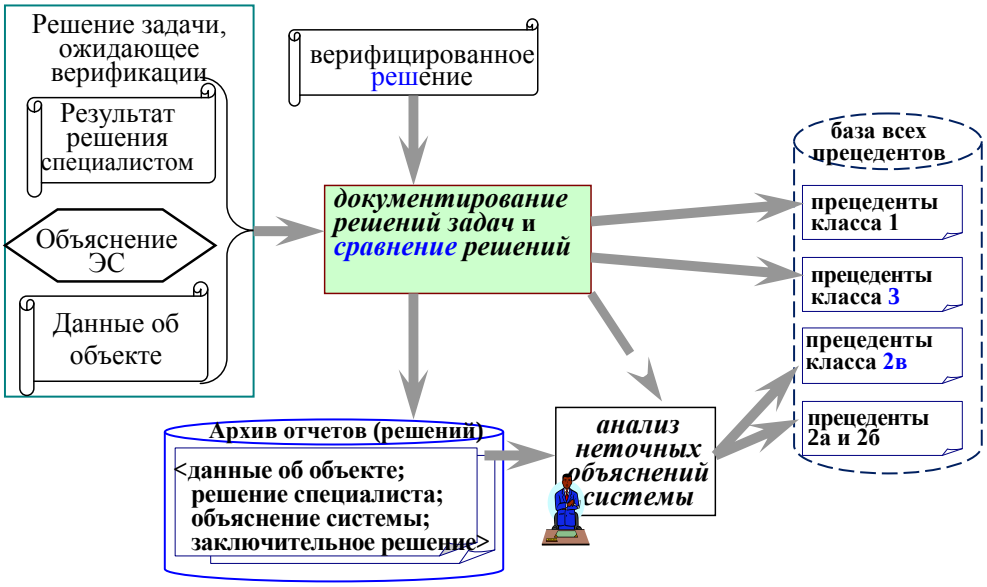


Рисунок 4-1 – Системы управления на основе связи с документооборотом

Множество прецедентов, отнесенных к классам 1 и 2 «входят» в меру качества. Допустимой является такая модификация БЗ, которая не ухудшает ее меру качества. Так, в результате (т.е. после) допустимой модификации новые прецеденты из класса 3 «переходят» в классы 1 или 2. «Переход» означает, что при повторном решении задачи «для» этого прецедента СБЗ предлагает точное решение или правильное, но неточное решение (несколько возможных альтернатив, среди которых есть и правильное решение).

Новые прецеденты из классов 2а и 2б (выделенных с помощью экспертов по качеству) используются для уточнения БЗ, т.е. такой ее допустимой модификации, при которой прецеденты из класса 2а переходят в класс 1, а прецеденты из класса 2б переходят в классы 1 или 2в. Это позволит увеличить меру точности БЗ или несколько уточнить БЗ.

Привлечение к такой работе экспертов не *всегда оправдано, часто оценка «требуемая степень точности» обеспечивается обработкой* прецедентов из классов 1,2 и 3.

В поиске всех возможных вариантов таких допустимых модификаций БЗ для всех новых прецедентов (поиске возможностей включения новых прецедентов в оценку БЗ) и состоит функция поиска возможностей включения новых прецедентов в оценку БЗ

(«находить все возможные способы модификации БЗ»). Эти варианты допустимых модификаций БЗ должны вычисляться системой управления БЗ автоматически.

В результате выполнения этой функции может быть получено несколько вариантов допустимых модификаций БЗ для новой группы прецедентов, либо такие варианты могут вообще отсутствовать.

Поэтому функция модификации БЗ состоит в выборе одного такого варианта допустимой модификации (если они есть) и его выполнения.

Не выбранные варианты (не использованные допустимые) сохраняются вместе с «состоянием эталонной базы» и множеством новых прецедентов текущего этапа управления БЗ (в базе конфигураций СБЗ).

Если вариантов допустимых модификаций БЗ нет, необходим пересмотр некоторых ранее принятых решений при модификации БЗ (в т.ч. рассмотрение неиспользованных более ранних версий). Реализация этой функции должна осуществляться экспертами, входящими в группу управления БЗ при поддержке системы управления.

4.1.5. Требования к реализации системы управления БЗ

Среда развития баз знаний должна обеспечить, чтобы онтология ПрОбл (с определением терминов и связей и онтологическими соглашениями о правилах интерпретации и сопоставления фактов знаниям и системой профессиональных терминов) создавалось самой первой, т.е. до формирования компонентов СБЗ. Для апробации онтологии (достижения такого «уровня устойчивости», когда согласованы единые термины для знаний и данных, структура знаний полна, формат описания данных достаточен) и для создания БЗ может использоваться один инструмент – редактор знаний. И по одной онтологии создается произвольное множество БЗ.

Развиваемость баз знаний позволяет получить в перспективе «эталонную» базу знаний $k^* \in Kn(X(SitOnt), Y(SitOnt))$ путем последовательного приближения к этому «идеалу».

От проектируемых систем автоматизации **управления БЗ** требуется поддержка получения формализованных новых знаний (Рисунок 4-2):

удобные для эксперта средства формирования *обучающей выборки* из прецедентов (класса 3, по возможности – классов 2а, 2б и 3, или с избыточностью – классов 2 и 3),

средства автоматического формирования очередного варианта модификации БЗ TrnSft (KnOnt, PrecB (SitOnt)) по *обучающей выборке* на основе онтологии.

Примечание. На практике монотонное улучшение целесообразно планировать и вручную (интерактивно) с помощью такого инструмента, если (в ПрОбл при эксплуатации СБЗ) ошибочные прецеденты появляются редко (штучно).

Наряду с поддержкой получения новых версий формализованных знаний от систем автоматизации управления требуется поддержка оценивания этих версий знаний.

Включение новых научных результатов в БЗ, выполняемое экспертами, входящими в «группу управления БЗ» (Рисунок 4-2) также требует автоматизированной поддержки оценивания БЗ (на основе эталонной базы). Оценивать вариант изменяемой (по результатам научных исследований) БЗ, надо чтобы убедиться, что оценка базы знаний улучшилась (или хотя бы не ухудшилась на существующей базе эталонов) из-за включения новых научных результатов в БЗ.

Каждому прецеденту ((а)+(в)) – паре <известные данные об объекте, решение задачи относительно объекта> – класса 1 и классов 2а и 2б место в *базе эталонов* - эталонной базе для проверки качества (Рисунок 4-3) Т.е. вновь обнаруженные прецеденты могут быть рекомендованы к добавлению к ней (при условии, что их там нет). Прецеденты, отнесенные к классам 2а и 2б особенно нужны в базе эталонов в том случае, если охватывают редкие варианты связей при описании объектов.

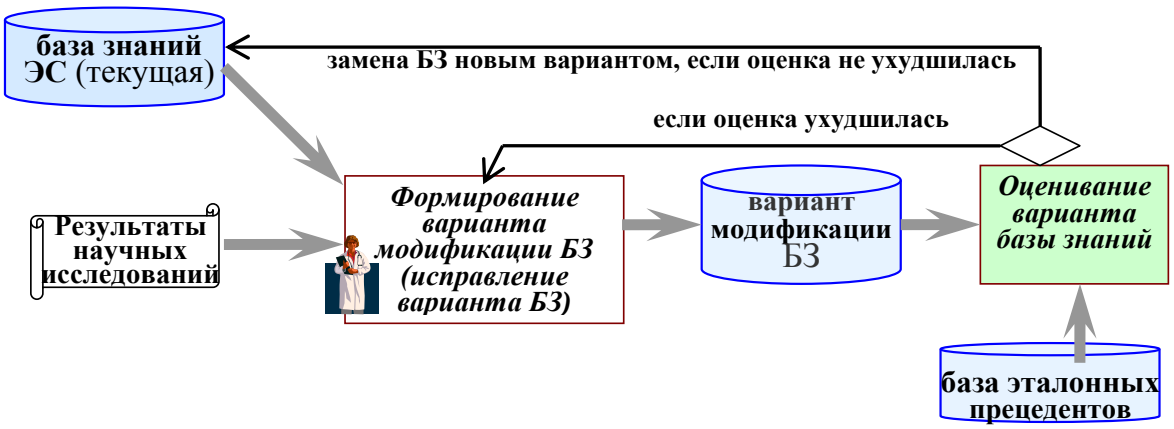


Рисунок 4-2 – Включение в базу знаний результатов научных исследований.

Если *мера качества модифицированной БЗ* становится не хуже при включении в нее новых научных знаний, то такой вариант модификации становится новым компонентом действующей СБЗ.

Средства обучения (индуктивного формирования баз знаний или фрагментов БЗ) также должны быть ориентированы на онтологии KnOnt и SitOnt.

Это дает возможность иметь всегда актуальную версию БЗ:

KBver = KB_{ver-1} (KnOnt) * (TrnSft(KnOnt, PrecB(SitOnt)) | (KBEd(KnOnt)) * AssSft(KnOnt, SamplB(SitOnt)).

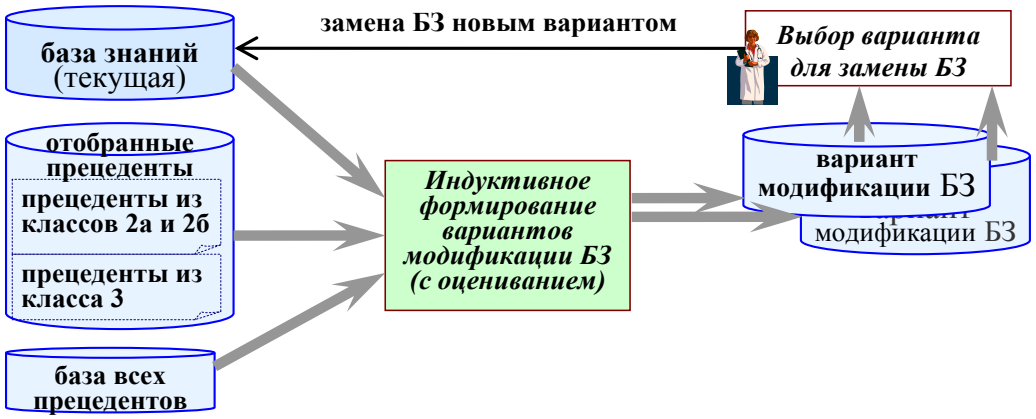


Рисунок 4-3 – Монотонное улучшение базы знаний на основе классификации прецедентов

Для оценивания варианта модификации БЗ применимы как средства оценивания баз знаний по эталонным базам-архивам, так и средства сравнительного оценивания двух версий баз знаний по критерию меры качества их знаний (на общем эталонном архиве или коллекции конкретных эталонных или тестовых задач) [59,60].

Удовлетворительный результат означает, что получен вариант БЗ с оценкой не хуже оценки специалиста (или не хуже совокупной оценки специалистов), поэтому он становится новой БЗ ЭС (вместо «текущей БЗ», используемой на этот момент).

Средства оценивания баз знаний по эталонным базам-архивам дают принципиально другой уровень качества по сравнению с традиционными средствами контроля формальных свойств построенной базы знаний и привлечением экспертов для оценки «решений, предлагаемых системой».

Требуемая система управления БЗ должна дать возможность убедиться, что по БЗ можно получить правильную гипотезу для отдельного прецедента (т.е. в объяснении она есть). Либо дать возможность на этом случае модифицировать БЗ так, чтобы СБЗ стала решать эту задачу правильно. СУБЗ должна дать возможность накапливать базу

прецедентов, чтобы потом модифицировать БЗ так, чтобы СБЗ стала решать задачу на этих новых ситуациях правильно.

Каждое обновление БЗ

Conf (KB-j-ver) = KB_{ver} (KnOnt, $Dict_{ver}$) ∪ (TrnSft-j (KnOnt-j, PrecB(SitOnt)) ∪ (KBEd-j(KnOnt)) ∪ AssSft-j(KnOnt-j, $SamplB_{ver}$(SitOnt))

приводит к очередной конфигурации СБЗ:

Conf (KBS)ver = {Conf (KB-j-ver)} ∪ {Conf (Solv-i-ver)} ∪ (Gui_m(SitOnt)) ∪ {$Soft_{k-ver}$} ∪ **{**DB_p (SitOnt, Dict)} ∪

<$\cup^{i=1,N1}$ (KB_i(KnOnt, Dict)), $\cup^{j=1,N2}$($Solv_j$(KnOnt,SitOnt, Agreem)), $\cup^{m=1,N3}$ (Gui_m(SitOnt)), $\cup^{p=1,N4}$(DB_p (SitOnt, Dict)), $\cup^{k=1,N5}$($Soft_k$)>.

Не только наборы тестовых прецедентов (PrecB(SitOnt)), но и отчеты об оценивании (KB&SolvTestReport) являются компонентами конфигурации СБЗ (аналогично традициям программной инженерии).

При формировании системы управления знаниями строятся сервисы для их оценивания. И редактор знаний, и сервисы для оценивания знаний работают на основе онтологии. Чтобы экспертам, отвечающим за качество БЗ, было удобно работать, важно учесть их требования к структуре отчета об оценивании БЗ. Среда развития (инструментарий) должна обеспечить возможность определять структуру отчета, которая (как часть онтологии) станет требованием к формату выходных данных сервисов, таких как «анализатор соответствия базы знаний эталонному прецеденту».

Сервис проверки (компонент системы автоматизации управления БЗ) сопоставляет известные правильно решенные задачи некоторого типа (прецеденты) текущему состоянию БЗ (для таких задач) и создает требуемый отчет (с заданной структурой) о результатах оценивания.

Примеры.

Анализатор соответствия Истории болезни (ИБ) клинической картине заболевания по каждому найденному прецеденту дает результат, фрагмент структуры которого:

Идентификатор прецедента (История болезни)

- ▼ Проверяемая гипотеза-диагноз$_i$
- ▼ Проверяемая гипотеза-Симптомокомплекс$_j$
  - ► Признаки, подтверждающие гипотезу-симптомокомплекс
  - ► Признаки, отвергающие гипотезу-симптомокомплекс
  - ► Признаки для симптомокомплекса, значения которых не найдены в ИБ.

Анализатор соответствия назначенного лечения заболевания знаниям о лечении с учетом особенностей

по каждому найденному прецеденту дает результат, фрагмент структуры которого:

Идентификатор прецедента (История болезни)

▼ Проверяемая гипотеза-ЛС$_i$

▼ Проверяемая гипотеза-схема лечения$_j$

► Признаки и факторы, подтверждающие гипотезу-схему лечения

Проверяемая гипотеза-вид терапии

► Признаки и факторы, подтверждающие гипотезу-вид терапии

Проверяемая гипотеза-фармГруппа

Признаки и факторы, подтверждающие гипотезу

Проверяемая гипотеза-ЛС

Признаки и факторы, подтверждающие лс

Признаки и факторы, отвергающие ЛС

Признаки и факторы, отвергающие гипотезу

► Признаки и факторы, отвергающие гипотезу

► Признаки и факторы, отвергающие гипотезу-схему

Важно, чтобы в структуре отчета было место для записи множества элементов (экземпляров понятий и их связей), которые отвергают известное правильное решение (например, найденную причину-диагноз или построенный план-лечение).

Когда среди отвергающих элементов присутствует элемент из правильного прецедента, он указывает на ошибку в БЗ, которую надо исправлять. Некоторые (например, непустое множество «Признаки, значения которых не найдены») означает неточность БЗ, которую целесообразно корректировать.

На этапе обучения или до-обучения БЗ на эталонных архивах должны использоваться средства индуктивного формирования баз или их фрагментов. Например, в технической диагностике фрагментами баз будут типы отказов и типы причин, приведших к отказу; в медицине фрагментами будут отдельные диагнозы (на этапе обучения) и отдельные симптомокомплексы конкретных диагнозов (на этапе до-обучения). Типы значений элементов информации (например, проявляемых признаков), подготовленной к обучению, могут быть: числовые, строковые («качественные») или интервальные, т.е. отклонения от числовой нормы (понижение, повышение).

Обучающие прецеденты (рисунок 4-4) отбираются автоматически или вручную (в примере для БЗ диагностики это – «Признаки, отвергающие гипотезу-симптомокомплекс» и «Признаки для симптомокомплекса, значения которых не

найдены в ИБ», которые не пусты). Все такие случаи используются для формирования новой версии БЗ.

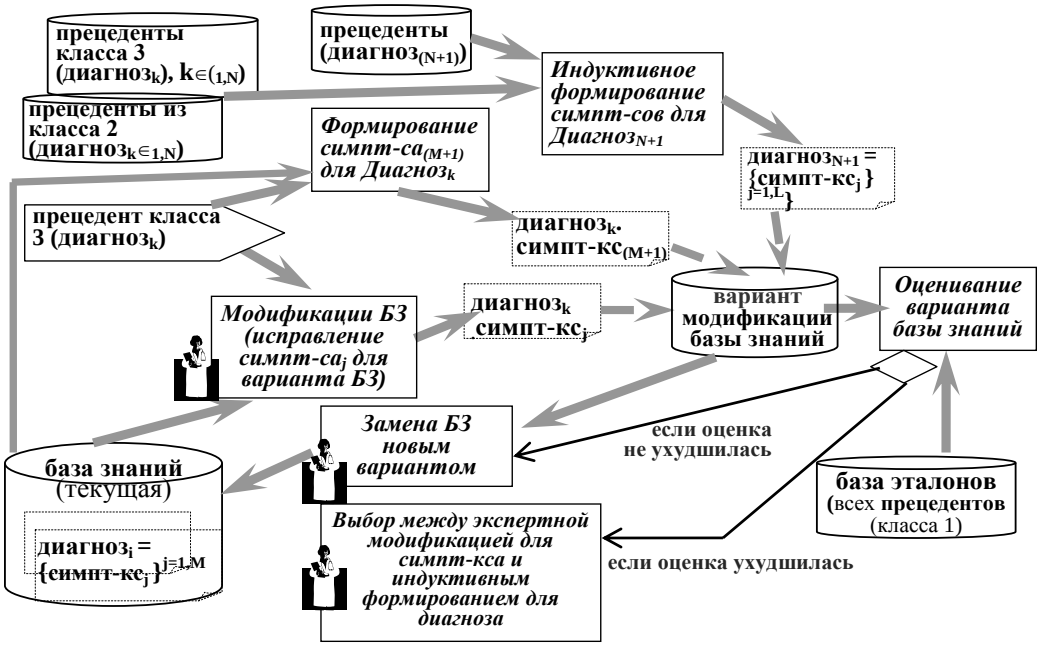


Рисунок 4-4 – Прецеденты для обучения, использования, оценивания

Т.о. следующим ключевым принципом методологии создания жизнеспособных систем становится создание (под)системы управления качеством всех БЗ с инструментами для оценки БЗ, для накопления прецедентов, формирования базы эталонов, обучения базы знаний с помощью прецедентов.

Автоматизация интеллектуальной деятельности подразумевает поддержку принятия решений задач одного класса (например, поддержка диагностики без поддержки планирования лечения) или нескольких. Для каждой задачи интеллектуальной деятельности разрабатывается своя БЗ (иногда некоторые базы знаний – общие). Важно иметь средства оценивания и усовершенствования для всех используемых баз знаний.

4.1.6. Применение метода модификации знаний (обновления БЗ) с помощью прецедентов разных классов

Методы модификации и корректировки знаний имеют разный уровень сложности и зависит от сложности онтологии знаний, что определяется метриками числа понятий, длины цепочек и дугими (определяемыми в пар 4.2). Наиболее трудными могут оказаться методы корректировки знаний для задач о процессах,

протекающих во времени, и для задач проектирования объектов или систем, учитывающих несколько типов отношений между элементами, а не только их пространственные связи [138].

Работу с временным рядом одного признака по поиску соответствия момента измерения периоду(-ам) развития процесса упрощенно можно описать так. Для измеренного признака отсчитывается порядковый номер дня (или другой единицы, привычной для типа процессов) dj в последовательности дней от начала наблюдения процесса.

Для знаний о признаке по БЗ, где указаны min и max длительности временных интервалов для каждого j периода (contj-min – contj-end), j=1,n, вычисляются min- и max-возможные начальные и конечные порядковые дни для всех периодов: t_{1b} =1; t_{2b} = 1+$cont_{1\text{-}min}$; t_{3b} = 1+$cont_{1\text{-}min}$ + $cont_{2\text{-}min}$; t_{4b} = 1+$cont_{1\text{-}min}$ + $cont_{2\text{-}min}$+ $cont_{3\text{-}min}$; t_{1e} = $cont_{1\text{-}max}$; t_{2e} = $cont_{1\text{-}max}$ + $cont_{2\text{-}max}$; и т.д.; по вычисленным номерам упорядочиваются начальные и конечные порядковые дни для всех периодов. Если для простоты принять, что любая t_{je} будет меньше $t_{(j+2)b}$ и любая t_{jb} будет больше $t_{(j-2)e}$) и n=4, то последовательность моментов развития заболевания: <1, t_{2b}, t_{1e}, t_{3b}, t_{2e}, t_{4b}, t_{3e}, t_{4e}>. Для отсчитанного порядкового номера дня (часа) надо найти «место» в этом ряду [138].

Рассмотрим метод усовершенствования БЗ на основе прецедентов класса 2 (неточное решение) и класса 3 (неправильное решение) (на примере медицинской диагностики).

Средство анализа соответствия базы знаний эталонному прецеденту (такое, как *анализатор соответствия* истории болезни – клинической картине заболевания) выдает информацию для отнесения единичного прецедента некоторому классу прецедентов: даст прецедент класса 3, если решение опровергнуто; и даст прецедент класса 2, если не подтверждено. А *средство поиска всех гипотез* даст прецедент класса 3, если решение не вошло в список гипотез; и даст прецедент класса 2, если кроме этого решения есть другие гипотезы. С помощью этих прецедентов проводится корректировка знаний (БЗ надо переобучить).

Если получено множество прецедентов этих классов, то начинать исправлять надо по прецедентам класса 3. (Отчет-объяснение, сгененрированнный анализатором *соответствия,* содержит список опровергнутых признаков $\{f^{ex}Name_j^{Rej}\}$). Когда диагноз имеет в точности один *вариант последовательности периодов развития* признаков (симптомокомплекс), то по каждому прецеденту (с номером 1,…m,…) класса 3 (этого симптомокомплекса этого диагноза)

$<\Delta_k, Vari_{kp}, \{(\pi\varepsilon\rho_i, \tau_i), \{(<f^{ex}Name_{jkp}, Value^{ijm} \text{ in } \pi\varepsilon\rho_i\}\}>$

в БЗ этого симптомокомплекса $Vari_{kp}$ этого диагноза Δ_k

$<Vari_{kp}, \{(\pi\varepsilon\rho_i, \tau_i), \{(<f^{ex}Name_{jkp}, f^{ex}Values_{ijk} \text{ in } \pi\varepsilon\rho_i\}\}>$

– расширить ОВЗ этих признаков: $<\Delta_k, Vari_{kp}, \{(\pi\varepsilon\rho_i, \tau_i), \{(<f^{ex}Name_{jkp}, (Value^{ijm} \cup f^{ex}Values_{ijk}) \text{ in } \pi\varepsilon\rho_i\}\}>$.

(Другой вариант модификации – понизить модальность признаков, сделать признаки возможными). Оговоримся, что без участия человека можно обойтись в простом случае (например, в некритической прОбл).

Обычно есть множество вариантов последовательностей периодов развития признаков $\{<\Delta_k, \{Vari_{kp}\}>\}$, $\{<Vari_{kp}, \{(\pi\varepsilon\rho_i, \tau_i), \{(<f^{ex}Name_{jkp}, f^{ex}Values_{ijk} \text{ in } \pi\varepsilon\rho_i\}\}>\}$. А в базах знаний, формируемых экспертами, как правило, каждый такой вариант имеет свое необходимое условие.

Отчет-объяснение анализатора позволяет из всех симптомокомплексов $\{Vari_{kp}\}$ (необходимому условию которого прецедент удовлетворяет) выбрать (автоматически) один – по критерию меньшего количества опровергнутых признаков $\{f^{ex}Name_j^{Rej}\}$ или по близости значений признаков с наибольшей модальностью.

Пример усовершенствования БЗ по *прецеденту класса 3* для признака с «качественным» значением (или с интервальной шкалой):

Анализатор соответствия опроверг гипотезу «Хронический холецистит (обострение)», которая для некоторой ИБ была подтверждена документально, с объяснением причины опровержения:

Хронический холецистит (обострение) [Заболевание проверяемое].[Симптомокомплекс]

Признаки, отвергающие гипотезу: Боль в животе (Локализация = правое подреберье и правый мезогастрий), 14.03.14 10:31 [момент].

Алгоритм корректировки БЗ (отвергнутого симптомокомплекса, который был единственным по «необходимым условиям» или самым вероятным по мнению эксперта):

1) вычислить в модели развития признака период(ы), которому соответствует моменту времени из прецедента; момент (номер дня в последовательности от начала заболевания) может соответствовать в БЗ двум соседним периодам (i-1-му и i-му) или только одному.

2) Если значения признака из прецедента не было в БЗ среди вариантов значений этого признака (с модальностью «необходимость») вообще (а не только в соответствующий период не было),

то {Если момент соответствует в БЗ только одному периоду,

то добавить обнаруженный на практике вариант значений (который указан в объяснении) рассматриваемого наблюдения (признака с характеристикой) в вычисленный период} [138].

В примере – когда дата 14.03.14 соответствует i-му периоду, к вариантам значений Боль в животе.Локализация для i-го периода добавить значение «правое подреберье и правый мезогастрий».

Если момент соответствует в БЗ двум соседним периодам (i-1-му и i-му), то выполнить:

А) добавить значение в оба периода;

Б) добавить в оба, смягчив модальность этой характеристики;

В) создать новый симптомокомплекс, повторяющий все признаки, кроме исправляемого, а этот признак определить заново на основе временного ряда в этом прецеденте;

Г) переопределить периоды – (i-1-й) уменьшить на вычисленный интервал, (i-й) уменьшить, между ними вставить период, в котором будут множество значений будет объединением значений (i-1)-го и i-го периодов и (обнаруженного) варианта значений.

3) Если в БЗ среди вариантов значений только в вычисленный период i его не было, но было в какой-то из соседних, тогда изменить их длительности, а именно:

А) обнаруженный вариант значений соответствует только соседнему периоду слева, (i-1)-му, или обоим, но момент ближе к (i-1)-му, чем к (i+1)-му то расширить правую границу длительности (i-1)-го периода так, чтобы «захватить» указанный момент;

Б) соответствует соседнему периоду справа, (i+1)-му, то уменьшить левую границу длительности текущего периода.

Пример усовершенствования БЗ по Прецеденту класса 3 для признака с числовым значением:

Анализатор соответствия опроверг гипотезу «О. холецистит». Признаки, отвергающие гипотезу: Лейкоциты (клинический анализ крови) [признак (или событие)]

15.1 [значение]

07.11.15 22:10 [момент времени]

Алгоритм корректировки:

1) вычислить в модели развития признака период(ы), которому соответствует момент из прецедента.

Выполнить корректировку (одним из способов):

А) расширить диапазон значений рассматриваемого признака в периоде (i-ом), например, так:

если ближайший отличаются НЕ более, чем на допустимый процент (10% для двух-трех-значных, 5% для попадающих в диапазон до 10, 2% для температуры), то заменить границу этим вариантом,

Б) расширить правую границу длительности (i-1-го) периода, если значение ему соответствует;

В) уменьшить левую границу длительности текущего периода, если значением соответствует следующему периоду.

Пример усовершенствования БЗ по *прецеденту класса 2*:

Анализатор соответствия не подтвердил гипотезу «Острый холецистит», которая для ИБ 1003/14 была подтверждена документально, с объяснением причины:

Признаки, необходимые для подтверждения диагноза, но не найденные (в соответствующий период времени) в истории болезни:

СОЭ в ПД3 (т.е. III период),

Температура тела в ПД 1, 2, 3,

Щелочная фосфатаза в ПД1,

Боль в животе в ПД1 (Характер, Интенсивность, Выраженность),

Напряжение мышц передней брюшной стенки в ПД2 (Локализация),

Симптом Ортнера-Грекова.

Алгоритм корректировки:

1) вычислить в модели развития признака период(ы), которому соответствует момент из прецедента.

2) Модальность «необходимость» заменить на более слабую модальность («характерность» или при участии и согласии эксперта на «возможность»).

После корректировки в соответствии с алгоритмом требуется использовать Сервис проверки БЗ на Эталонной Базе Прецедентов (или на полном архиве законченных ИБ из учреждений), чтобы убедиться, что все случаи диагностируются правильно.

На основе этого метода БЗ диагностики, первоначально сформированная экспертом, проходит непрерывное улучшение (уточнение), адаптируясь к изменению вариантов проявления заболеваний у определенных категорий пациентов, в определенных условиях пребывания и т.д.

В некоторых ПрОбл качество знаний определяется еще и *актуальностью* (срок, прошедший с даты проверки знаний) [42]. Метод непрерывного усовершенствования знаний на основе поступающих прецедентов «заботится» о такой актуальности.

4.2. Оценивание свойств информационных компонентов

Помимо правильности баз знаний, на характеристики создаваемых СБЗ влияют и другие свойства информационных компонентов. В зависимости от типа информационного компонента (знания, справочные данные, оперативные данные) могут быть важны размер (объем) или формальная полнота, соотношения количества (числа экземпляров) одних элементов по отношению к другим, соответствие значений ограничениям или смыслу предметной области и др. [48]. Успешность оценивания людьми полноты информации (особенно знаний), внесения дополнений, исправлений зависят от структурных свойств их онтологии [109].

4.2.1. Структурное оценивание информационных компонентов

В случае формирования информационного компонента человеком (знаний – экспертом, данных – специалистом или инженером), качество результата этого процесса зависит в значительной мере от предоставленных средств, например, от свойств средства редактирования знаний [98].

Когда редактор информации (или его интерфейс) генерируется по онтологии, его свойства зависят от свойств онтологии [62,149].

При разработке СБЗ имеет место и проверка полноты и корректности знаний экспертом. Возможность и удобство (и затраты) деятельности эксперта, проверяющего полноту и корректность знаний, тоже зависят от свойств редакторов знаний (следовательно зависят и от свойств онтологии) [98]. Поэтому для информационных компонентов СБЗ необходимы дополнительно: методы оценки онтологий ПрОбл (завершенность, морфологические характеристики, неизбыточность или фактическая использованность ее элементов при формировании информации) [109].

Удобство интерфейса с пользователем, предназначенного для ввода исходных данных для решаемых задач и отображения результатов, в большей степени зависит от свойств онтологии действительности (составной части онтологии предметной области). Проведение диалога как с пользователем, решающим задачи, так и с пользователем-экспертом, редактирующим знания, зависит от сложности связей между терминами предметной области, наличии отношения частичного порядка между терминами, облегчающего проектирование сценария, с числом терминов-сущностей, с числом терминов-атрибутов, с числом связей и от онтологических соглашений в ПрОбл [98].

Принято проводить «структурные сравнения объектов и баз знаний для поддержки идентификации» аналогичных объектов или сценариев [16,18,124]. Онтолого-ориентированный подход к формированию БЗ позволяет распределить ответственность за качество между экспертами, отвечающими за смысл, полноту и точность, и когнитологами, отвечающими за форму и достаточность системы понятий. Можно сказать, что при онтологическом подходе анализ свойств готовой к использованию (утвержденной) онтологии знаний, является источником для деятельности по обеспечению качества компонентов разрабатываемой интеллектуальной системы и источником информации для планирования затрат на их разработку [109].

Измерение используемых онтологий предметных областей дает возможность улучшить их до того, как на их основе будут построены редакторы знаний и решатели, выявить и исправить существенные недостатки [102,104].

Свойства структуры онтологии наиболее естественно определяются в терминах *графовых моделей,* и разные графовые модели отражают различные структурные свойства онтологии [64]. Структурные свойства онтологии определяются в терминах соответствующих графов через подсчет числа вершин или дуг, числа одноименных дуг, разветвлений, числа и доли дуг с определенными метками и т.д.

Подгруппы графовых моделей онтологий – графы *синтаксических связей, стандартных связей, концептуальных связей* и *графов проблемно-ориентированных связей* – учитывают выразительные способности языков онтологий (owl и oil, kif, языка прикладной логики, языка описания информации и др. [55,158].

Наиболее информативные свойства с точки зрения оценивания влияния онтологий на особенности основанных на онтологиях редакторов и программных систем определятся по следующим графам [103,106]:

граф зависимости терминов, граф связи предложений и *граф структуры предложения* (из группы *графов синтаксических связей*), которые применимы независимо от языка формализации онтологии;

граф таксономии с*ущностей* или *граф «теоретико-множественных» связей* или *граф стандартной партономии* (из группы *графов стандартных связей*) – в зависимости от языка формализации онтологии (owl, oil или kif, ЯПЛ) обычно используется тот или иной граф;

граф предметно-ориентированных связей из группы *графов концептуальных связей.*

4.2.2. Оценивание онтологий по графам структуры синтаксических связей

Для *графов синтаксических связей* характерно, что их вершины соответствуют некоторым синтаксическим компонентам текста (их имена или обозначения становятся метками вершин), а направленные дуги – «синтаксическим» связям между такими компонентами. Рассматриваемые синтаксические компоненты могут быть как одного, так и разных «уровней рассмотрения» (например, используемые функции, их аргументы, отдельные операнды в выражении для аргумента функции), а названия связей составляют

конечное множество (они могут рассматриваться как метки для дуг) [48].

При построении порталов знаний часто онтология ПрОбл содержит несколько онтологий знаний, некоторые из них, в сою очередь могут быть составлены из частей-моделей. Для анализа зависимости терминов из разных онтологий можно построить *граф межмодульных связей*, также относящийся к *графам синтаксической структуры*.

Граф связей определяемых понятий – графовая модель <V, E>, где вершины V = {v-i}, v-i – вершина, соответствующая *понятию онтологии*, а дуги E = {e-j}, e-j – связь *использования* от вершины – определяемого понятия к вершине, соответствующей понятию, используемому при определении. Вершина имеет *метку* – имя понятия, частью которого может быть префикс, содержащий ссылку на другую онтологию, в которой определено это понятие [102,103].

Значения свойств, определяемых по *графу связей определяемых понятий* онтологии: *глубина зависимости терминов, число вершин, число дуг*, влияют на свойства редактора знаний а они связаны с затратами на возможность и удобство деятельности эксперта, проверяющего правильность или полноту знаний [106].

Пример определения свойства по этому графу [62,149]):

Используемость понятия – число *дуг,* входящих в *вершину-понятие,* в графе связей определяемых понятий.

Область значений: целое неотрицательное число.

Неоднозначность имен – множество *меток вершин*, которые в графе связей определяемых понятий встречаются более одного раза (или множество *вершин*, *метки* которых совпадают).

Область значений: множество идентификаторов.

«*Неоднозначность имен*» (разные вершины онтологии названы одинаково) может иметь серьезные последствия. Пример выявления этого факта: наличие термина-множества «качественные значения» и термина-объекта «качественные значения» с различной структурой создает трудность при анализе, чтении *онтологии заболеваний*.

Это свойство относится к категории *дефектов*.

Следующие свойства обеспечивают допустимые значения и ссылки в порождаемой онтологической информации [106]:

число определений значений понятий через конечные множества *терминов*

число определений значений понятий через числовые диапазоны

число определений значений понятий через множества *сочетаний символов ограниченного количества;*

число определений понятий через *ссылки на другие понятий*,

число (или множество*) понятий* с *неопределенными значениями.* Если таких понятий не обнаружено в онтологии, то для указания значимых областей значений (терминальных) понятий в БЗ определены допустимые значения, а также допустимые ссылки на другие определения (т.е. обеспечены свойства БЗ – *использование только допустимых значений и использование допустимых ссылок*).

С удобством редактирования знаний, с минимизацией двусмысленности, трудностью при анализе/чтении онтологии, с выбором подходящего термина для обозначения некоторой сущности в диалоге с экспертом связаны свойства [106]:

число понятий, имеющих синонимы,

наличие переопределений терминов.

С популярным свойством *связности онтологии* или совокупности определяемых понятий связаны свойства:

связность понятия (количество ссылок на данное *понятие* и из него) [42],

число неиспользуемых понятий,

число обращений к переопределяющему понятию.

На практике, при обеспечении качества редакторов знаний и данных, такие свойства позволяют на ранних этапах обнаруживать потенциальные проблемы [107].

Например, при разработке редактора для системы, решающей задачу построения характеристического рентгеновского спектра термин «характеристические линии» определен в онтологии как новое название для термина «радиационные переходы». При этом наличие трех *обращений к переопределяющему термину «*радиационные переходы» (для терминов-сортов «начальное положение электрона при радиационном переходе», «конечное положение электрона…», «относительная интенсивность») и двух – к *переопределяемому термину «*характеристические линии» (для терминов-сортов сечение возбуждения и вероятность излучения) свидетельствует о неочевидном выборе термина для проектирования сценария и может стать поводом изменить саму онтологию.

Граф связей определяемых понятий позволяет установить частичные отношения порядка между понятиями, а такой порядок важен при организации диалога с экспертом, (логичный и естественный сценарий диалога с экспертом обеспечивает вклад в качество баз знаний).

При изменении онтологии знаний обеспечение проверки корректности производимых изменений напрямую зависит от размеров цепочек взаимосвязанных понятий.

При использовании в некоторой онтологии знаний терминов другой онтологии (что определяется по *графам межмодульных связей*) важно значение *глубины зависимости терминов,* которые используются.

Граф связей предложений – графовая модель <V, E>, где вершины V = {v-i}, v-i – вершина, соответствующая *предложению онтологии*, характеризующаяся *типом*, принимающим одно из значений {«определение сущности», «определение области значений», «определение связи», «соглашение»} и меткой – именем определяемого понятия или номером соглашения или именем используемого понятия (если оно в тексте другой онтологии, то метка, как правило, с префиксом); а дуги E = {e-j}, e-j – связь *использования* от вершин-соглашений к вершинам-понятиям и между вершинами-понятиями (от предложения, в котором использовано имя *понятия,* определяемого в другом предложении, к вершине-используемому-понятию) [102, 103].

Пример определения свойства по этому графу [62, 103,149]:

Число понятий/терминов, не связанных с другими – число вершин, не имеющих никаких входящих дуг и выходящих дуг из вершин, имя которых без префикса.

Область значений: целые неотрицательные.

Это свойство относится к категории дефектов.

Не связанные онтологическими соглашениями понятия/термины – множество вершин, не имеющих никаких входящих дуг из вершин (соответствующих *предложению онтологии*), характеризующаяся *типом* «соглашение».

Область значений: множество идентификаторов.

Для терминальн*ых понятий* становится важным такое свойство как наличие определения области значений. Их наличие обязывает реализовать проверку соответствующих соотношений между вводимыми значениями в самом редакторе, а их отсутствие – повод для повторной проверки онтологии: действительно ли произвольны значения таких элементов знаний [107].

По *графу связей предложений* можно определить *связанные онтологическими соглашениями понятия.* Нулевое значение свойства *связанные соглашениями понятия* мо-

жет рассматриваться как признак неполноты онтологии и послужить поводом для проверки самой онтологии [109].

4.2.3. Оценивание онтологий и порожденной по ним информации по графам структуры стандартных связей

Для *графов стандартных связей* характерно, что их вершины соответствуют терминам онтологии (их имена становятся метками вершин), а направленные дуги – стандартным видам связей (типично употребляемым при формализации онтологий).

Традиционно *стандартными связями* являются часть-целое и частное-общее. Названия связей составляют конечное множество уточняющих «подвидов» стандартных связей, они могут рассматриваться как метки для дуг или как специальные графические обозначения [64].

Граф партономии онтологии – графовая модель <V, E>, где вершины V = {v-i}, v-i – составные (нетерминальные) и простые сущности (в том числе строковые константы) онтологии, а дуги E = {e-j}, e-j – связи сущностей следующих видов: «*состоит из»* (разнотипных терминов) *(является «парой», «тройкой»...), включает часть*, *является последовательностью* (термин определен как *последовательность* (однотипных) *терминов,* длина последовательности задана или не определена), *включает (под)последовательность, является множеством*, *включает подмножество***,** *представлено альтернативным понятием* [102,104]. Составные (нетерминальные) сущности обычно имеют названия (как повторяемые, так и уникальные). Дополнительно партономическая связь (оба ее конца) характеризуется кардинальностью (1 или *) и модальностью (0 или 1). Примечание. Язык определения понятий графа может позволять при порождении вершины указать, является ли она нетерминальной (составной) или простой.

Граф партономии становится популярной моделью для анализа структуры онтологии, поскольку наглядным «инструментом» для понимания особенностей понятий предметной области и глубинных отношений между ее концептами служит поле знаний (описание основных объектов ПрОбл, их атрибутов и отношений между ними, выявленных из некоторого источника знаний) [11,18]. Этот «скелет» базы знаний (с «полями знаний») носит иерархическую структуру, что обусловлено иерархичностью понятийной структуры знаний человека [10,18]. Успешным примером представления онтологий

являются иерархические семантические сети, являющиеся развитием языка представления вербальных моделей «content description notation» [170].

Примеры свойств:

число составных вершин (*в партономии) с указанным именем*;

число составных вершин, от которых не определены дуги к их частям (*«включает подмножество», «состоит из», «включает часть»*, *«включает последовательность», «представлено альтернативным понятием»)*;

глубина партономии;

ширина партономии;

максимальная длина цепочки дуг от *корневой* вершины к вершине с указанным именем [110].

Пример определения свойства по этому графу (в каталоге свойств):

Глубина партономии – максимальная *длина цепочки дуг* от *корневой* вершины к *листовой*.

Область значений: целое неотрицательное число.

Возможное практическое применение: характеризует сложность моделируемой области знаний. Влияет на *длину цепочки вывода,* если их посылки (и/или следствия), задаваемые *онтологическими соглашениями*, являются сложносоставными, представляемыми «глубокими партономиями».

Недоопределенные части – число/множество составных вершин, из которых выходит одна дуга типа «*состоит из» (или «включает часть», «представлено альтернативным понятием»)* [106]. Это одно из свойств, возможно, связанных с полнотой (не определены остальные части).

Избыточные части – число цепочек (из дуг типа «*состоит из» или «включает часть»*), в которой каждый узел имеет только одну выходящую дугу и одну входящую дугу, с числом узлов более двух. Это одно из свойств, отражающих простоту структуры и влияющих на успешность оценивания людьми полноты информации (особенно знаний), внесения дополнений, исправлений.

Число нетерминальных (*составных) вершин, от которых не определены* дуги к их частям. Это одно из свойств, связанных с формальной полнотой.

Формальная полнота отражает минимальное требование к готовности информаци-

онных компонентов быть использованными программными компонентами [109].

Например, если при решении задачи относительно объекта важно знать нормальные значения или допустимые значения его атрибутов (характеристик, наблюдений), то до начала эксплуатации потребуется проверить полноту определения значений для норм в базе признаков (как важную метрику полноты):

Число незаданных норм = Число составных вершин (*в партономии) с именем «Норма», от которых не определены* дуги *включает подмножество*.

Область значений: целое неотрицательное число.

Возможное практическое применение: характеризует полноту фрагмента знаний. Наиболее полезна для классификаторов и словарей.

По *графу партономии* возможно выявление «*неоднозначности имен*» как наличие вершин с одинаковыми именами, но имеющие разные части-«потомки». Как отмечено выше, значение такого свойства чрезвычайно важно [62,64,149].

Граф таксономии* сущностей *онтологии – графовая модель <V, E>, где вершины V = {v-i}, v-i соответствует *сущности* онтологии или конкретному экземпляру некоторой сущности (индивиду), а дуги E = {e-j}, e-j – связь одного из двух видов: «*является представителем*» и «*является потомком*». Дуги «*является представителем*» выходят из *вершины-экземпляра* и входят в *вершину-класс*. Дуги «*является потомком*» выходят из *вершины-класса* и входят в *вершину-класс* [102,104].

Пример определения свойства по этому графу (в каталоге свойств):

Число листов – число вершин с полустепенью захода дуг «является потомком», равной *нулю.*

Область значений: целое неотрицательное число.

При допустимости множественного наследования могут представлять интерес свойства:

число сущностей, являющихся подклассами разных классов (типов),

число экземпляров сущностей разных типов/классов.

Примечание. При использовании иерархических семантических сетей исключается возможность порождения в качестве целевой информации экземпляров разных типов (классов, сущностей) [131].

Сложность редактирования знаний зависит, в частности, от *глубины иерархии терминов* (по *графу таксономии*, а иногда и *партономии*). При вводе значений терминов переход от одного термина к другому включает в себя последовательность шагов «подъема» до некоторого термина-предка в иерархии и последовательность шагов «спуска».

Глубина иерархии терминов может быть оценена на основе одного из вышеупомянутых графов. Значения, превышающие 10, обычно означают, что ожидается относительно высокая трудоемкость редактирования знаний и проверки таких знаний [100].

Пример – *глубина партономии = 16* для *онтологии заболеваний* в медицинской диагностике. Это означает, что для каждого компонента диагностических знаний (здесь – для каждого заболевания) для указания требуемых значений признаков может понадобиться до 16 шагов перехода от термина к термину. (При формировании вручную базы заболеваний (в области офтальмологии») в соответствующем редакторе «путь» от названия заболевания к заданию значений нетривиальных признаков довольно длинный). Связывание нужного *наблюдения* с его *характеристиками* и их *значениями,* которые лежат глубоко, отражается на скорости редактирования знаний.

Сложность редактирования знаний зависит от *числа вершин, числа дуг, числа листов* (в рассматриваемой модели). Чем больше эти значения, тем больше затраты на редактирование и проверку, а в последнем случае – тем больше компонентов знаний (в примере: для каждого заболевания) должны быть описаны при редактировании [107].

Большие значения *ширины иерархии понятий*(измеряемой как *ширина таксономии* либо как *ширина партономии*) свидетельствуют о проработанности предметной области, о ее тщательной структурированности, о потенциальной возможности адекватно представить знания [106]. С другой стороны, они свидетельствуют о сложности знаний (сложнее разобраться, дольше проверять, тщательнее тестировать). *Ширина партономии без учета дуг-альтернатив* характеризует минимум ширины партономии знаний, которые могут быть созданы в рамках рассматриваемой онтологии [100]. *Полустепень исхода для дуг-компонентов* характеризует число компонентов знаний, которые должны быть описаны при редактировании. А чем больше *полустепень исхода для дуг-альтернатив*, тем чаще пользователю предоставлен выбор типов данных (или значений) при редактировании знаний [110].

Число циклов характеризует «потенциал» для увеличения глубины партономии знаний. Если нет циклов, то глубина партономии базы знаний соответствует глубине партономии онтологии. В другом случае использование цикличной связи терминов приводит к углублению партономических связей в модели знаний [100]. В *банке медицинских знаний*многоцелевого банка знаний [20, 21] хранится *онтология наблюдений* (всего 19 терминов), в ее партономическом графе есть три цикла, за счет циклов между этими терминами определяемая и редактируемая в этих терминах «база наблюдений в области офтальмологии» гораздо сложнее, она имеет большую глубину партономического дерева [107].

Свойствами таксономической структуры, позволяющими эксперту убеждаться в полноте и корректности онтологии, лежащей в основе знаний, являются:

Число прямых потомков (рассматриваемой сущности),

Число прямых предков (рассматриваемой сущности),

Множество «собратьев» (рассматриваемой сущности).

Иерархичность знаний и показатели размера иерархии отражают возможность полноценной проверки введенных знаний [100].

4.2.4. Оценивание онтологий по графам структуры концептуальных связей

Для *графов концептуальных связей* характерно, что их вершины соответствуют сущностям онтологии (их имена становятся метками вершин), а направленные дуги – бинарным связям одного типа, для которых предусмотрено произвольное множество названий (названия могут рассматриваться как метки для дуг) [100].

Граф предметно-ориентированных связей онтологии **(простой)** – графовая модель <V, E>, где вершины V = {v-i}, v-i - сущности онтологии, конкретные экземпляры (представители) сущностей, или фиктивные (подразумеваемые сущности) без имени, а дуги E = {e-j}, e-j – бинарные связи (отношения, функции, предикаты), определяемые в онтологии с некоторыми предметно-ориентированными названиями, дуги имеют метку с названием связи [102,165].

Примеры определения свойств по этому графу (в каталоге свойств) [64]:

Число вершин (сущностей и элементов их описаний)

Область значений: натуральные.

Число пар вершин, связанных разными дугами,

Область значений: целые неотрицательные.

Множество пар вершин, связанных разными дугами,

Область значений: Множество кортежей.

Максимальное связанность/сцепление пары сущностей = максимальное число дуг *между парой* вершин *в* графе предметно-ориентированных связей

Область значений: целые неотрицательные.

С характеристиками онтологии по числу сущностей, по числу связей, по кардинальности этих связей (один-ко-многим, многие-ко-многим) связано планирование требуемых для заполнения и хранения форм, списков, таблиц, семантических сетей (знаний и данных) [110].

Свойство *число элементов описаний* отражает

для SitOnt – сложность (и в некоторых случаях объем) *описаний* наблюдаемых или создаваемых *объектов,*

для KnOnt – сложность и потенциальный объем знаний;

в обоих случаях это влияет на сложность алгоритма решателя.

Свойства:

число временных, пространственных и других отношений (связей),

длина цепочек причинно-следственных отношений (в т.ч. между указанными вершинами)

отражают число *типов аксиом (предложений)* KnOnt (онтологии знаний) и сложность (*типов) аксиом (предложений)* онтологии, и влияют на *сложность алгоритма решателя*, например, в задачах анализа при проверке гипотез по элементам описаний объектов; в задаче проектирования при учете условий и свойств конструируемого объекта; влияют на число ограничений или целевых условий на объект (например, в задаче лечения ограничение = «объект должен стать здоров») [165].

Сложность алгоритма решателя и сложность предоставляемого пользователю объяснения напрямую зависит от *длины цепочек вывода* (при решении задачи), определяемых свойством *длина цепочек причинно-следственных* отношений [109].

Для каждого вводимого пользователем (по SitOnt) *набора экземпляров сущностей, наборов значений их атрибутов* или связей между разными экземплярами разных или однотипных сущностей важно наличие своевременной проверки соответствия таких значений, если они явно существуют и описаны. По *графу предметно-ориентированных связей* можно установить *недоопределенные сущности* [64].

Есть необходимость контроля следующей информации: соответствует ли вводимые *характеристики* и *значения* характеристик тем *характеристикам* и допускаемым *значениям* характеристик, которые предусмотрены (например, в *словаре терминов*) [165].

Например, важно своевременно обнаруживать, не введены ли пользователем в качестве жалоб новые виды *наблюдений,* если они не совпадают с описанными в *базе наблюдений*, то для них нет информации в *знаниях о заболевании*).

Становятся важными следующие свойства:

число/множество вершин, которые являются только «отправителями» связей;

число/множество связей с неопределенным «получателем».

На основе измерения этих свойств можно установить те термины-сущности, при формировании знаний о существовании которых и вводе названий эксперт может вносить опечатки, вводить множества не полностью или создавать избыточность в списках. Достоверность вводимых значений повлияет на результаты решения задачи с использованием введенных знаний [109].

Так, на практике, при обеспечении качества редакторов знаний и данных системы, решающей задачу медицинской диагностики было обнаружено, что в онтологии действительности для связей *наблюдение, характеристика, качественное значение результата наблюдения* определены без «получателя». Поэтому при вводе данных (в архив истории болезней) не только имя жалобы может быть указано в произвольной форме, но и *название наблюдения* может быть дано не из хранимого списка наблюдений, а также выбор типа значений может быть сделан несоответствующий (например, вместо качественных может быть назначено целое значение) [152].

Эти же свойства, определяемые по онтологии объяснения решения задачи, связаны с возможностью пользователю системы поддержки решений на основе знаний за приемлемое время увидеть и понять важные для него советы. Онтология в данном случае является языком написания информации для человека. Внимание к этому имеется и у раз-

работчиков СБЗ, им приходится разрабатывать «средства для предоставления объяснений выводов, которые понятны экспертам. Объяснения должны быть представлены на языке, который является приемлемым для пользователя, они должны иметь «эргономичный» размер, состоять из отдельных фрагментов объяснения, которые могут быть самостоятельными и иметь смысл [124].

Выбор способа и средств формализации (представления) знаний влияет, в частности, на затраты на проверку качества [62,149].

Так, для иерархических семантических сетевых моделей IACPaaS имеются встроенные средства проверки, обеспечивающие, например, невозможность *нарушения ограничений «кардинальности»* в БЗ, т.к. редактор управляется онтологией, в которой заданы эти ограничения как количественные спецификаторы отношений понятий [98,131];

невозможность *нарушения согласованности формата и единства представления однотипной информации:* формат представления знаний согласован и един для всех однотипных фрагментов, поскольку редактор знаний управляется онтологией, задающей формат для каждого фрагмента знаний *(для всех* и структурных и *причинно-следственных* отношений понятий) [165].

Редактор, управляемый онтологией, в которой определены возможные скалярные значения всех (терминальных) понятий, контролирует указание значимых областей значений в БЗ, не выходящих за пределы возможных, в результате чего *только допустимые значения* и *допустимые ссылки на другие определения* будут содержаться в БЗ [165].

4.3. Выводы к главе 4

Для поддержки работы специалистов (и обучения или повышения квалификации) необходимы программные системы, знания в которых актуальны. Меры, направленные на контроль используемых знаний, естественно связаны с мерами, направленными на приведение в соответствие знаний специалистов с обновляемыми (наиболее современными и актуальными) знаниями (особенно когда происходят коренные изменения в системе знаний) через применение метода перманентного монотонного усовершенствования баз знаний и оценки с помощью эталонных задач.

Предложен подход к регулярной проверке информационных компонентов ~~(ресурсов)~~ перед применением, (или перед компоновкой СБЗ). Он позволяет получать объек-

тивные значения структурных свойств, выявлять дефекты, задавать субъективные показатели качества ресурсов, основываясь на структурных свойствах. Ранний контроль качества прочих онтолого-ориентированных информационных компонентов (хранимой информации) предлагается обеспечивать через оценивание корректности, наличия дефектов и несогласованностей в них и регулярную проверку после каждого изменения и расширения.

Установлена зависимость некоторых внешних свойств онтологии (свойств, проявляющихся при использовании онтологии в системе, основанной на знаниях) от внутренних (структурных) свойств, определяемым по графовых моделям из предложенного набора графовых моделей онтологий.

Выявлены те структурные свойства, которые влияют на процесс редактирования (требуя повышенной внимательности от эксперта или пользователя), на сложность реализации некоторых компонентов системы, на дополнительные затраты при обеспечении контроля вводимой информации, на создание увеличенных контрольных списков для проведения экспертиз. Наличие точных определений свойств в терминах графовых моделей и метода построения графовых моделей для онтологий, написанных на различных языках, позволяет применять такой подход для широкого диапазона интеллектуальных систем.

Оценивание структурных свойств онтологий на основе единого подхода является наиболее «экономически целесообразным» подходом к анализу онтологий, поскольку на ранних стадиях разработки дает полезные показатели.

Проектирование архитектуры СБЗ, интегрированной с подсистемами для управления качеством БЗ – ключевой принцип жизнеспособности. Поэтому к основным элементам модели конфигурации жизнеспособной СБЗ (в среду ее развития) добавляются средства, реализующие методы контроля их качества.

ГЛАВА 5. Онтолого-ориентированный подход к разработке решателей систем поддержки решений на основе знаний

В главе рассматривается подход к разработке решателей, функциональностью которых является выдвижение и объяснение гипотез о решении задачи, и метод их конструирования из программных компонентов, обрабатывающих информацию, порожденную в соответствии с онтологией.

5.1. Место решателя в СБЗ

Онтология ПрОбл (Onto = KnOnt ∪ SitOnt ∪ Agreem ∪ Dict) определяет правила представления и обработки информации (анализ, формирование, модификация) в ПрОбл. KnOnt определяет структуру профессиональных знаний о решении задачи, SitOnt определяет структуру информации об объектах действительности, Agreem – онтологические соглашения о правилах обработки, т.е. сопоставления фактов и знаний в процессе рассуждения и принятия решений.

Онтолого-ориентированный (или онтологический) решатель Solv (KnOnt ∪ SitOnt ∪ Agreem) в составе СБЗ должен быть способен выдвигать гипотезы по результату сопоставления входной информации (об объекте) – предложениям из БЗ КВ (KnOnt, Dict) на основе онтологических соглашений Agreem о связи данных и знаний и с учетом знания структуры предложений (связей терминов) KnOnt и ограничений на интерпретацию смысла терминов.

«Сопоставляемость» входной информации – предложениям из БЗ обеспечивается тем, что и данные, и знания формируются на единой базе терминов ПрОбл (Dict).

Для наиболее понятного отражения ПрОбл (и наиболее полно охватывающего понятия и связи из ПрОбл) KnOnt представляют семантическими сетями (сложно устроенную входную информацию SitOnt – тоже). Для некоторых задач в некоторых ПрОбл семантическая сеть, представляющая БЗ, может естественным образом разбиваться на подсети (модуляризоваться) [166]. Примеры таковы. Для задачи лечения в медицине знания состоят из БЗ о фармГруппах и ЛС и БЗ о схемах лечения. Для диагностики не-

исправностей АНПА БЗ содержит модуль о диагностических ситуациях и архитектуре (комплектации) различных видов/типов/модификаций подводных аппаратов.

Задача онтологического решателя $Solv_j$ ($KnOnt_{j1}$, …, $KnOnt_{jw}$, $SitOnt_{jy}$) – предложить (путем «рассуждения») одно или несколько обоснованных решений-гипотез, используя входную информацию и предложения из Базы знаний KB_{iu} ($KnOnt_u$).

Обоснование каждой гипотезы – явное указание ее связи с теми предложениями Базы знаний, которые ее подтверждают или не отвергают.

Онтологический Решатель (средство «рассуждения на основе онтологий (ontological reasoning)» [17]),

(«неразрывная» пара <Onto, онтологический «рассуждатель» (reasoner)>),

позволяет делать рассуждение (вывод) на декларативных (онтологических) базах знаний, построенных по Onto.

Онтологический решатель $Solv_j$ – программно реализованное «рассуждение», алгоритм, проводящий обход каждой Базы знаний KB_{iu} (на основе знания ее структуры $KnOnt_{iu}$ и онтологии в целом) и обращение к фактам Sit и сопоставление фактов цепочкам связей БЗ с учетом Agreem..

С учётом разнообразия (под)задач, которые могут определяться по общей онтологии, и ожидаемой повторяемости их более мелких подзадач/процедур/функций важно выделить виды программных единиц, которые обеспечат более прозрачную и менее трудоемкую архитектуру решателей.

5.2. Представление и роль онтологических соглашений

Формализованные в KnOnt сведения о причинно-следственных, временных, пространственных и др. связях терминов с ограничениями интерпретации сопровождаются онтологическими соглашениями Agreem – явно прописанными договоренностями о правилах интерпретации описанных связей (оставшихся за рамками ограничений интерпретации) и о правилах обработки информационных объектов [21], формирования фрагментов решения или объяснения.

Agreem (зафиксированные онтологические соглашения) включают соглашения о правилах сопоставления фактов – знаниям, а также о правилах выдвижения, подтвер-

ждения, опровержения гипотез, содержащихся в знаниях или создаваемых с их помощью.

Например, для задач анализа процессов применяются следующие типы отношений (утверждений):

(1) о соответствии фактов (ситуаций действительности) – необходимым условиям существования процессов, согласно Знаниям о процессах. Факты из ситуаций действительности $\in R(t) = R_{ex}(t_0, ..., t_k) \cup R_O \cup R_{ev}(t_u..., t_v)$;

(2) о соответствии фактов – внешним проявлениям внутренних процессов в некоторые периоды их развития (согласно Знаниям). Ожидаемые проявления $\in R^{kn}$ (moments), moments $\in$ Periods – следующим друг за другом периодам развития рассматриваемого процесса (successive periods of ongoing process),

(3) о соответствии промежуточных заключений о прошедших периодах сделанных на основе подтвержденных проявлений периодов (истинных значений предикатов Pred (r(t), r^{kn} (mom)) – знаниям о следующих друг за другом периодах развития рассматриваемого варианта течения процесса (successive periods of ongoing process),

(4) о соответствии подтвержденной последовательности периодов динамики процесса (истинных значений предикатов Pred (r(t), r^{kn} (Var)) – допустимому варианту его течения согласно Знаниям о процессах.

Для задач синтеза – следующие:

(5) о соответствии заданных условий (Cnd) на конструируемую Систему/План/Проект Cnd_O или на компоненты Проекта $Cnd_O^{comp}(p_0, ..., p_k)$ – свойствам планируемых элементов согласно $F_{ex}^{comp}(p_0, ..., p_k)$ – Знаниям о конструируемых Системах/Проектах;

(6) о соответствии промежуточных выводов о свойствах планируемых / конструируемых Проектов или их частей – подтвержденным свойствам планируемых элементов;

для задач управления:

(7) о соответствии заданных условий Cnd на состояние Системы или на отдельные внешние проявления $Cnd^O_{comp}(t_0, ..., t_k, p_0, ..., p_k)$ – необходимым условиям перехода

в состояние s_n и | или вариантам воздействий <d_0, … d_m> на признаки для изменения их проявлений [164].

Формализация соглашений может осуществляться как на некотором логическом, математическом языке, так и на структурном языке описания информации.

Соглашениям также могут быть сопоставлены правила формирования фрагментов решения или объяснения [164]. Но обычно они являются следствием соглашений и становятся частью спецификации программных единиц (ПрЕд), реализующих обработку (в соответствии с правилами программной инженерии).

Пример описания онтологического соглашения на языке ЯПЛ [55, 57]:

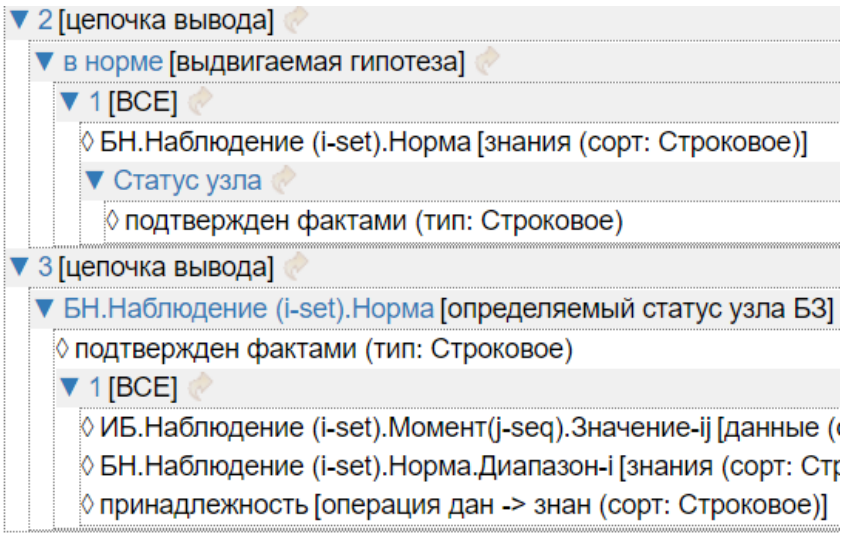
(нормальная реакция: *нормальные реакции)*

(v (знания о нормальной реакции: *знания о нормальных реакциях) следствие(знания* о нормальной реакции) =

= *следствие (*нормальная реакция) && *вариант (*нормальная реакция) ∈ *варианты(знания* о нормальной реакции) && *выполнено (условие на воздействующие факторы(вариант(*нормальная реакция))));

Пример соглашения в текстовом виде: «Если в ситуации присутствует некоторая нормальная реакция, то в множестве *знания о нормальных реакциях* присутствует такой элемент, у которого следствие совпадает со следствием нормальной реакции, в множестве вариантов нормы этого элемента содержится вариант нормы из нормальной реакции и для него выполнено условие на воздействующие факторы.

Пример соглашения в виде структурной записи:



Каждое соглашение может определять один шаг логического вывода процесса решения задачи. При построении решателя (реализации решения задачи) шаг логического вывода может быть определен в вызываемой ПрЕд, которая читает БЗ и модифицирует нелокальную структуру данных, отображающую этапы рассуждения (например дерево вывода).

Пример спецификации ПрЕд для вычисления функции *здоров ли пациент* (история болезни, База знаний о Нормальных Значениях признаков) => {здоров, имеется патология}:

если каждый признак в ситуации (в истории болезни (ИБ) пациента имеет значение, попадающее в диапазон нормальных значений признака, указанный в Базе знаний о Нормальных Значениях признаков (БНормЗн), то установить диагноз пациента:= «здоров». Примечание. Для абсолютного заключения о том, что «пациент здоров», пришлось бы применить более строгое правило: если для каждого признака в БНормЗн установлено, что признак у пациента имеет значение, попадающее в диапазон нормальных значений, указанный в БНормЗн, то пациент здоров. Однако ввиду сложности полного обследования (десятков тысяч наблюдений) на практике применима более простая процедура.

5.3. «Онтологический характер» рассуждения

Решатель для выдвижения и объяснения гипотез обрабатывает входную информацию (обход семантических сетей – БЗ) и строит выходную информацию, содержащую гипотезы о решении с их объяснением (фиксируя в ней шаги принятия или отклонения возможных логических заключений) [111]. В итоге получается семантическая сеть, отображающая этапы его рассуждения, которые привели к заключениям по каждой Гипотезе Hyp-j (объяснение Expl-Hyp-j). Решатель Solv формирует такой «отчет» (по каждой Гипотезе) [166].

Обход Kn проводится с учетом структуры описываемых знаний KnOnt и ограничений на интерпретацию смысла терминов.

Примером причинно-следственных и временных связей терминов (KnOnt) является фрагмент онтологии знаний диагностики для описания множества аномалий $F_{in} = \{f_{in}\}$ (см. гл 2).

Для некоторых ПрОбл семантическая сеть может иметь явно выраженный иерархический вид.

Факты относятся к объекту действительности или условиям (ограничениям), которым он должен удовлетворять.

Обращение (доступ) к ним проводится с учетом структуры описания фактов SitOnt и ограничений на интерпретацию (смысла используемых терминов).

Обход Kn и Обращение к фактам проводится в рамках рассуждения и принятия решений, поиска и анализа информации, устроенной предписанным образом.

Процесс рассуждения «опирается на» зафиксированные онтологические соглашения: часто одно соглашение может определять один шаг логического вывода при построении решения задачи [164].

Онтологические соглашения Agreem и все сведения из KnOnt ∪ SitOnt должны быть известны (предоставлены) разработчикам алгоритмов решения задач.

В процессе рассуждения и принятия решений, поиска и анализа информации, алгоритм совершает Обход Базы знаний Kn, Обращение к фактам Sit и сопоставление фактов цепочкам связей БЗ с учетом предложений из Agreem. (Обход решателем Базы знаний в виде иерархической семантической сети – частного случая семантической – представляет собой проход по цепочкам связей БЗ, начиная с корневой вершины) [111].

При применении соглашений о соответствии фактов – знаниям о рассматриваемых явлениях связываемые понятия используются для промежуточных заключений. Часто в роли *посылок* (причин для принятия промежуточных решений) выступают факты из Ситуаций Действительности R(t) и ожидаемые проявления из множества R^{kn} (moments). *Следствиями* являются значения предикатов их соответствия Pred (r(t), r^{kn} (mom)).

При применении соглашений о соответствии подтвержденных (в процессе вывода заключения) «фрагментов» – знаниям о рассматриваемых явлениях (например, подтвержденных периодов динамики или подтвержденной последовательности периодов) в роли *посылок* (причин для принятия промежуточных решений) выступают $_{ожид}$ подтверждаемые проявления из множества R^{kn} и значения предикатов. *Следствиями* являются значения предикатов в утверждениях БЗ (KnB) (в левых частях формул, определяющих явление через его составляющие). Примеры левых частей формул из KnB – Комплекс характерных наблюдений, и далее по мере получения истинных значений предикатов –

Комплекс Проявлений, Проявления очередного периода, Вариант развития внутреннего процесса, f_{in} (см. гл.3 «Модель знаний для решения задач, связанных с анализом аномальных процессов»).

Решатель, способный выдвигать гипотезы на основе онтологических соглашений, структуры предложений и ограничений на интерпретацию, сохраняет (**объясн**яет) анализ гипотез о решении. Его логический вывод, в отличие от логических выводов, реализованных как исчисления, для задач диагностики, прогноза, распознавания класса создается как алгоритм последовательного подтверждения или опровержения элементов знаний. Это подтверждение или опровержение делается через: сопоставление фактов элементам знаний на основе онто-соглашений и перехода к анализу (с целью подтверждения, опровержения) очередного элемента знаний в соответствии со структурой знаний.

Логический вывод решателя для задач планирования, проектирования, управления создается как алгоритм последовательной проверки подцелей, соответствующих элементам знаний, и поиска соответствующих им элементов или действий, полная совокупность которых, согласованная с онтологией, становится гипотезой (если в процессе конструирования не будет отвергнута отсутствием возможности найти полную совокупность). Логический вывод решателя для задач планирования, проектирования, *построение плана* изменения ситуации, планирования воздействий на систему или объект создается как алгоритм последовательной проверки наличия для подцелей, соответствующих преследуемым состояниям, воздействий, обеспечивающих переход к ним (как элементов выстраиваемой гипотезы); и поиска среди наблюдаемых фактов – возможностей применения таких воздействий (предусловий для таких воздействий, совместимости друг сдругом) либо опровержения элемента (-ов) гипотезы.

Выдвигая гипотезы или формируя заключения (на основе указанной онтологии) Решатель «попутно» запоминает и сохраняет информацию из «посылок», «привязывая» к промежуточным выводам. Цепочка вывода соответствует совокупности установленных в онтологии связей понятий, например: (по онтологии диагностики) от диагноза к множествам значений признаков в указные периоды, (по онтологии планирования управления) от целевых свойств объекта к воздействиям на составные части объекта или на свойства объекта или на явления, изменившие объект, и т.п.

5.4. Обработка онтологических информационных компонентов и Онтологический подход к формированию объяснения

При создании алгоритмов обработка информации включает работу над БЗ, работу с фактами, описывающими действительность, работу по фиксации аргументов в пользу решений, принимаемых на каждом шаге рассуждения.

Работа над БЗ – это выбор узлов-посылок для узлов-заключений (в том числе промежуточных); работа с фактами аналогична, если они являются сложным документом (как, например ИБ).

Работа по фиксации аргументов – формирование объясняющей (процесс вывода и анализа причинно-следственных связей) информации в сохраняемом ИнфРес «Объяснение получения решения» (объяснительная, обосновывающая, «причинно-следственная» модель). Часто «Объяснение» по сути имитирует фрагмент использованной БЗ – семантической сети с разметкой узлов (как подтвержденных или отвергнутых или требующих дополнительной Информации).

Поэтому структура ИнфРес «Объяснение» может копировать или инвертировать структуру БЗ, часто предлагая «наверху» список гипотез, связанных с «цепочками» их объяснения по типу «Следствие имело следующие посылки».

Для задачи диагностики, согласно постановки которой (см. гл.2), требуется найти: все возможные причинно-следственные модели ($AS_\Sigma(R_\Sigma)$), объясняющие $\Delta \subseteq F_{in}$ результатами наблюдения *признаков, характеристик и событий* в последовательности, соответствующей причинно-следственным цепочкам связей из БЗ, структура объяснения предполагается следующей.

Множество возможных *процессов-диагнозов* (провоцируемых указанными *событиями*),

множество вариантов развития (каждого) процесса (при указанных *характеристиках*),

последовательность периодов,

связанных с фактами (результатами наблюдения *признаков*, возможно, изменяемыми некоторыми *событиями*).

Для задач прогноза, согласно постановки которых, требуется найти все такие значения признаков в указанной перспективе (q'(…, t',…)), для которых существует причинно-следственная модель (AS_Σ ($R_\Sigma(t_0, ..., t_k)$, q'(…, t',…))) системы, объясняющая их в рамках нормального процесса либо известного отклонения ($\Delta \subseteq F_{in}$) в соответствующей причинно-следственным цепочкам связей из БЗ, структура объяснения предполагается такой.

Множество возможных *признаков* (с их значениями),

А) связанных с последовательностью периодов динамики признака нормального процесса,

подтверждающие их факты-в-норме

или

Б) обусловленных начавшимся аномальным процессом,

подтверждающие их факты-отклонения.

(Структура объяснения может быть спланирована иначе:

множество возможных *периодов процесса* (нормального или обусловленного начавшимся процессом),

подтверждающие их факты,

связанные с характерными для *периодов* значениями признаков.)

Такое содержание объяснения Expl имеет цепочки, связывающие $Expl^{Hyp}_i$ с фактами, совпавшими с диапазонами наблюдаемых или планируемых значений описателей рассматриваемого объекта/системы («реализующие» Agreem). Но в общем случае для структуры «Объяснение» (семантической сети) может браться подсеть структуры БЗ, которая «покрывается» набором Agreem для решаемой задачи.

На практике разумно учитывать интересы специалистов – ожидаемых пользователей. Поэтому на практике формат объяснения (выходного инфоресурса) корректируется или определяется разработчиками сервиса с участием специалистов – ожидаемых пользователей. При наличии у пользователей СБЗ собственных требований к структуре «Объяснения решения», который они будут использовать в реальной практике, разработчик решателя должен будет удовлетворить требования (через анализ требуемой структуры, проектирование ее, проектирование ПрЕд и т.д. по ТРПО). Т.о. запись (построение) объяснения производится в ИнфРес, структура которого (онтология объясне-

ния) соответствует структуре (онтологии) знаний и информационным потребностям пользователя СБЗ. Для задач, результатом поддержки которых является нбор гипотез, обычно формируются три подсети: «Подтвержденные гипотезы», «Отвергнутые», « Не опровергнутые», внутри которых содержится аргументация. Например, при объяснении лечения струкура аргументации такова.

Набор Целей терапии

{Диагноз,

Ссылка на приказ Минздрава по Диагнозу,

Набор Схем терапии

{Название схемы,

{Симптом как критерий,}

{Вариант Группы используемых НММ/ЛС,

{НММ/ЛС,

Целевой уровень терапии,

Эффективность НММ/ЛС}}}}.

Таким образом, Объяснение может быть «спланировано» как отчет с уникальной структурой или как более универсальный отчет об анализе всех связей понятий, относящихся к решаемой задаче или к анализируемому типу знаний.

Пример. Результатом рассуждения и объяснения процесса прогнозирования наступления критической стадии становится отчеты, где (в соответствии с онтологическим-соглашением «если у заболевания есть сценарий, некоторый «период-j» которого подтвержден для анализируемого входного объекта (ИБ пациента) и «период (j-1) не отвергнут и «период-j+1» имеет высокий *критический уровень*, то считать его прогнозируемым, если «провоц факторы» для «периода-j+1» не отвергаются) описана пара (гипотеза – текущая стадия; Гипотезы – следующий стадии), и по Гипотезе–текущей стадии создано «подтверждение» (факторы и признаки, которые подтверждены), а по каждой Гипотезе–следующей стадии собраны «возможное» («провоц факторы», которые подтверждены или не отрицаются) и «невозможное» («провоц факторы», которые отрицаются).

Результатом рассуждения и объяснения процесса диагностики является вся аналитическая информация о результатах сопоставления исходных данных пациен-

та симптомокомплексу из клинической картины предполагаемого заболевания. С участием медицинских специалистов определен Формат объяснения, где сгруппированы отчеты по каждой Гипотезе и по каждому ее Симптомокомплексу. Внутри него собраны: «Признаки, подтверждающие гипотезу», «Признаки, необходимые для подтверждения гипотезы, но не найденные», «Признаки, отвергающие гипотезу».

«Признаки, подтверждающие гипотезу», сгруппированные как «Признаки необходимые», «Признаки характерные» и "Признаки возможные», являются доказательством гипотез на основе онтологических соглашений и позволяют понять, почему такая гипотеза имеет право на существование. Для подсказки врачу о дальнейшем обследовании сформированы группы «Признаки, необходимые для подтверждения диагноза, но не найденные в истории болезни». А если множество «Признаки, отвергающие гипотезу» не пусто, то оно является аргументацией против рассмотренного предварительного диагноза.

СБЗ чаще создаются для *задач поиска гипотез* (где требуется найти все гипотезы, соответствующие Kn) или *критики гипотез* (требуется проверить соответствие заданной гипотезы базе знаний Kn), в них БЗ дана на входе. Системы с БЗ «на выходе» (для решения задачи индукции по обучающей выборке, как определено в гл.2) чрезвычайно актуальны. Для получаемой БЗ (или варианта ее модификации) по тем же принципам генерируется объяснение. Как правило, здесь структура ИнфРес «Объяснение» близка к структуре БЗ, дополняя некоторые узлы полезной информацией, например, узлы-классы (диагнозы) – информацией о количестве элементов выборки, узлы-значения признаков – информацией о множестве идентификаторов или количестве элементов выборки, по которым значения сформированы, узлы-модальности – информацией о доле элементов выборки, по которым значения сформированы.

Т.о. онтологический решатель $Solv_j$ ($KnOnt_{jy1}$, …, $KnOnt_{jyw}$, $SitOnt_{jy1}$, …, $SitOnt_{jyx}$)), способен объяснять свой выходной результат (составляя достаточное для понимания объяснение Expl).

Объяснение по каждой выдвигаемой Гипотезе $Expl^{Hyp}$ – это аналитическая информация о результатах сопоставления исходных данных имеющимся знаниям (по Hyp) на основе онтологических соглашений о связи данных и знаний [164]. Решатель $Solv_j$ форми-

рует заключения с объяснением $Expl_j$ (в «отчет» по каждой Гипотезе $Expl^{Hyp}_i$).

Построение логики решателя через пошаговое подтверждение следствий через посылки требует (или может сводиться к) последовательного заполнения узлов структурированного объяснения.

5.5. Особенности алгоритма решателя

Сложность алгоритма [13] и его Размер (длина исходного кода) зависят от выбранных структур данных. Если алгоритм онтологический, Сложность определяется числом и сложностью типов аксиом (предложений) онтологии, длиной цепочек причинно-следственных отношений (связей) между искомыми (проверяемыми) гипотезами и элементами описаний объектов ПрОбл, количеством временных, пространственных и других отношений (связей). Особенно если для выдвижения гипотезы или создания решения по задаче требуются все (а не выборочные) понятия и отношения онтологии. Также влияет число элементов описаний наблюдаемых или создаваемых объектов, т.е. объем (и в некоторых случаях сложность) SitOnt. Дополнительно влияет число ограничений или целевых условий, например, в задаче проектирования условия будут касаться свойств конструируемого объекта (моста, аппаратного комплекса и т.д); в задаче лечения ограничение = «объект должен стать здоров».

Применение правил для «обработки» в процессе решения задачи выражается в связывании *посылок* со *следствиями* и поиска информации для подтверждения посылок. Для задач, являющихся уточнениями *задачи анализа результатов наблюдений* (определенных в гл.2.) количество предложений/правил, которые следует «обработать» в процессе общего решения (безотносительно к предметной области), невелико:

предложения о связи условий с возможностью появления или варианта течения процесса;

предложения о связи варианта течения процесса с признаками-проявлениями варианта;

предложения о связи процесса, воздействия и измененного варианта течения;

предложения о влиянии воздействия на признак;

соглашения о сопоставлении условий течения процессов – фактам (свидетельствующим о наличии внутренних свойств или внешних обстоятельств).

Обычно обработка – это поиск (в документе, описывающем действительность) фактов для подтверждения условий течения процессов (и «вывод» – получение значений соответствующих предикатов); вывод значений предикатов (на множестве условий, на множестве проявлений) – предикатов существования гипотез из элементов модели знаний по полученным значениям предикатов-«посылок», выдвижение гипотез на основе соответствующих вычисленных предикатов (подтвержденных или (не-) отвергнутых посылок) [164]. Например, количество предложений для решения общей задачи диагностики, невелико [36]: поиск (в документе, описывающем действительность) фактов для подтверждения условий для аномалий, фактов для подтверждения ожидаемых проявлений аномалий («посылок» в правилах вывода), вывод значений предикатов подтверждения проявлений, выдвижение гипотез на основе подтвержденных предикатов.

При реализации онтологического «рассуждателя» принимается выбранная (проектировщиком решателя) стратегия обхода (или эвристика). Например, в зависимости от цели диагностики: либо для некоторого аномального процесса f_{in} должен быть полностью подтвержден некоторый *вариант развития процесса*, либо должны быть найдены все неотвергнутые *варианты развития* всех аномальных процессов.

При создании алгоритмов работа (обработка информации) может распределяться на работу над БЗ, работу с фактами, работу по фиксации аргументов-объяснений в сохраняемом ИнфРес. Например, для создания алгоритма (реализуемого в ПрЕд) *проверки гипотезы о варианте развития и обоснования варианта* одна ПрЕд может работать над причинно-следственной связью «Вариант развития – set of (признак)» и обращаться к ПрЕд *проверки гипотезы о признаке* (и к ПрЕд *подтверждения необходимых факторов для варианта). В свою очередь* ПрЕд *проверки гипотезы о признаке* может работать над отдельными узлами БЗ («*признак*») и над ИБ в соответствии с онтологическими соглашениями, связывающими факты в ИБ с *признака*ми в БЗ, обеспечивая установление *статуса узла-признака в «*объяснении».

5.6. Проектирование онтологического решателя из программных единиц

Решатель (умеющий выдвигать гипотезы и объяснять их) планируется к реализации из программных единиц (ПрЕд), среди которых единицы $Unit_m$ разных типов, с доступом и без доступа к ИнфКомп, для вычислений, для связи с внешним окружением.

ПрЕд для поиска фактов

- получает список условий на искомые факты, ищет в документе названия таких наблюдений, сравнивает с условиями и дает ответ: данные соответствуют условию или данные не соответствуют или данные отсутствуют;

- получает значения наблюдений, ищет в документе названия таких наблюдений, сравнивает с условиями и дает ответ: {данные соответствуют; противоречат; отсутствуют}.

ПрЕд для анализа знаний и вывода (в т.ч. вывода промежуточного заключения):

- запрашивает и получает информацию о подтверждении или отрицании (фактами) элементов отношений или их цепочек, прописанных в Kn (KnOnt);
- связывает информацию (о подтверждении или отрицании) с элементами-посылками возможных правил формирования промежуточных заключений (для потенциального шага логического вывода процесса рассуждения).

Например, *проверить подтвержденность варианта развития для подтверждения существования процесса* (ответ: {данные соответствуют; противоречат/ не соответствуют; отсутствуют}). .

Программная единица, обрабатывая такие виды связей между элементами информации, делает промежуточные заключения процесса логического вывода. (Они же делают поиск элементов знаний и их связей).

ПрЕд ($Unit_g$), которые делают некоторое заключение о соответствии подмножества фактов – знаниям некоторого типа (причинно-следственных или других структурных связей между понятиями), «имеют дело» с онтологическими соглашениями (Agreem).

В совокупности такие ПрЕд, как правило «покрывают» все утверждения из KnOnt ∪ Agreem. Решатель, создаваемый из ПрЕд, программируемых по такому принципу, – онтологический.

В алгоритме решателя происходит поочередный вызов ПрЕд – процедур вывода следствий из обработанных посылок, которые записывают результат проверки посылок в объяснение. Например, ПрЕд *проверить вариант развития процесса* – процедура вывода заключения о подтверждении / опровержении присутствия *комплекса признаков* из обработанных результатов подтверждения или опровержения всех обязательных *Признаков* и всех дизъюнктивных наборов признаков) активируется после активации ПрЕд

проверить гипотезу о процессе и после активации ПрЕд *проверки необхУсловия процесса*. А ПрЕд *проверить дизъюнкцию признаков* (процедура вывода заключения о подтверждении признаков из «обработанных» признаков и минимал их числа) вызывается после ПрЕд проверить *вариант развития;* процедура *проверить признак* (процедура вывода заключения о подтверждении ожидаемых значений признака) вызывается после активации процедуры *проверить вариант развития* и процедуры *проверить Дизъюнкцию признаков* …и т.д.

ПрЕд для поиска фактов и ПрЕд для подтверждения/вывода обращаются с запросами к той БЗ, где содержатся знания об известных связях, типы которых зафиксированы в онтологии знаний. Например, ПрЕд для поиска методов для воздействия на признак для изменения его значений в заданном направлении *getMethodsForTrendSign* реализует отношения «состояние – воздействие – результат» (между методом воздействия, признаком(-ами) до начала воздействия и признаком(-ами) желаемыми/целевыми. Другими словами, реализует правила вида «если наблюдаемые значения признака (одно в момент t_k или временной ряд значений в моменты t_1, …t_k) соответствуют *ОВЗ наблюдаемых* в связях <метод воздействия r-i, признак, ОВЗ наблюдаемых, ОВЗ целевых> и заданный *тренд нормализации* соответствует ОВЗ целевых, то выдать все такие воздействия r-i)».

В составе решателя также могут быть и другие *ПрЕд*, не нуждающиеся в доступе к ИнфКомп, – вычислительные, интерфейсные (для связи с внешним окружением).

В зависимости от того, «спланировано» ли Объяснение как отчет с уникальной структурой или как более универсальный отчет об анализе всех связей понятий, относящихся к решаемой задаче или к анализируемому типу знаний, реализация компонентов решателя, формирующих объяснение, будет специализированной (под пользователя) или универсальной (под структуру знаний). Т.е. степень повторного использования кода будет разной.

5.7. Операции над онтологическими информационными ресурсами как вид программных единиц

При разработке алгоритма для каждого шага вывода требуется либо обращение к элементам описания модели (знаний), либо к фактам действительности, либо к тому и другому. Так, при поиске фактов для подтверждения условий существования искомых явлений или признаков их проявления (т.е. «посылок» для вывода) требуется обращение к фактам (к возрасту, к размеру, к температуре…).

Каждое такое обращение за фактом или их комплексом – «кандидат» на самостоятельную операцию доступа Op_i к документу с данными.

Операции доступа к содержимому онтологических Инфоресурсов, такие как: *прочитать значения экземпляров понятий, найти все экземпляры понятий, связанных с данным элементом*, *найти метрики объекта; найти временной ряд указанного наблюдения* и т.д. наиболее связаны со структурными и причинно-следственными связями используемых понятий (определяемыми KnOnt и SitOnt). Поэтому являются онтологическими. И с конкретной онтологией связывается множество таких операций, чтобы применяться затем к любой информации (знаниям, данным), управляемой этой онтологией.

$Solv_j$ ($KnOnt_{jy1}$, … $KnOnt_{jyw}$, $SitOnt_{yx}$)) = {$\cup^{m=1,M}$ ($Unit_m$ ($KnOnt_{m1}$, …, $KnOnt_{mw}$, $SitOnt_m$, Op_{m1s} ($KnOnt_{m1}$), .., Op_{mzs} ($SitOnt_{mz}$), …)}.

Op_i осуществляют запросы к элементам структуры ИнфРес в соответствии с правилами интерпретации и соглашениями. ПрЕд ($Unit_g$), которые делают некоторое заключение о соответствии подмножества фактов знаниям некоторого типа, сопоставляют результаты выполнения *операций* над информацией Sit и над Базами знаний КВ. *Операции* могут получать на вход список параметров, на выходе обеспечивать вычисленные значения либо модификацию ИнфРес, например, объяснения или создаваемого плана/проекта (породить экземпляр указанного понятия, записать значение указанного элемента).

Набор операций отражает востребованные запросы, возникающие в переборных алгоритмах решения классов задач.

Так же, как в программной инженерии, проектировщик-архитектор решателя, создаваемого по онтологии, планирует на роль онтологических операций прежде всего те,

которые более одного раза требуются для реализации полного рассуждения. (Особая ответственность – на проектировщике первого решателя, создаваемого по единой для многих задач онтологии).

Формирование и использование операций доступа Op_i для управляемого программного доступа к целевым базам знаний и др. хранимой информации – один из принципов реализации решателя.

5.8 Программные единицы разных уровней

Для процесса рассуждения на основе анализа семантических связей между элементами информации, представляющей знания и данные, требуется обработка фрагментов онтологических ИнфРес (ОИР) [111]. Для этого нужны онтологические ПрЕд. Другие ПрЕд решателя могут не зависеть от онтологии и работать с другой информацией, с источниками / потребителями.

Можно выделить программные единицы $Unit_m$ двух «уровней» (с точки зрения «масштабности» реализуемой подзадачи и необходимой для этого обработки):

- *анализатор-и-конструктор-подсетей (компонент общего процесса решения)* - способен провести некоторый анализ фрагмента одной или нескольких ОИР и добавить фрагмент результата к другому ОИР (например, провести анализ соответствия описания объекта каждому комплексу общих признаков группы диагнозов; провести анализ соответствия описания объекта каждому комплексу условий для диагноза из указанной группы диагнозов; провести анализ соответствия описания объекта каждому симптомокомплексу признаков для указанного диагноза; провести анализ соответствия описания объекта каждому уточняющему комплексу признаков для стадий развития указанного диагноза и т.д.);
- *атомарный компонент* – функционально-независимая [46,160] *программная единица* (unit, модуль) для работы с информацией, которая близка к простым операциям над данными.

$Unit_m$ типа «анализатор-и-конструктор-подсетей» работает с несколькими ОИР, «знает правила» и позволяет выводить следствия из посылок в процессе анализа или синтеза. В роли анализатора-и-конструктора-подсетей может оказаться готовый решатель, если он обеспечивает решение задачи (соответствующей классификации из гл.2) или «слож-

ной» (под)задачи. Для понятности, прозрачности формирования и снижения сложности сопровождения желательно такие $Unit_m$ формировать из готовых компонентов (этого же и/или следующих уровней).

Виды $Unit_m$ – *атомарных компонентов* с учётом повторяемости или уникальности подзадач таковы:

- *вычислитель условия (предикатная программная единица) Prd* - частный случай о*бработчика параметров,* у которого передаваемый атомарный фрагмент информации относится к логическому (true-false) или перечисляемому типу;
- *обработчик параметров (параметрической информации) Prc* – получает, вычисляет или передаёт некоторый атомарный фрагмент информации, не обращаясь вовне за информацией;
- *операция над онтологическим инфоресурсом* SemOp – выполняет обработку (чтение/запись) фрагмента одного информационного ресурса (семантической сети), сформированного под управлением некоторой конкретной *онтологии* (например: вычислить минимальный интервал начала указанного периода от начала процесса; вычислить все периоды развития процесса, в которые попадает признак, наблюдаемый в указанный день в описания объекта);
- *аппаратно-интерфейсная операция* HrdOp – получение/передача данные через интерфейсы с сторонним аппаратным обеспечением;
- *человеко-интерфейсная операция* UiOp – получение/передача данных от/к человеку.

Любая из атомарных $Unit_m$ (в составе решателя) может получить на вход список параметров. Параметры могут быть любой сложности – от скалярных типов данных до семантических подсетей. «*Вычислительная» ПрЕд* (*обработчик параметров* и *вычислитель условия*) обрабатывает только получаемые параметры и вычисляет некоторое значение; *интерфейсные* операции работают с данными через некоторый внешний интерфейс.

В общем случае решатель задач создается из (процедурных) ПрЕд (модулей) разных типов:

$Solv_j$ ($KnOnt_{jy1}$, …, $KnOnt_{jyw}$, $SitOnt_{yx}$)) = {$\cup^{m=1,M}$ ($SemUnit_m$ ($KnOnt_{m1}$, …, $SitOnt_{mq}$)} U {$\cup^{m=1,M}$ ($SemOp_m$ ($KnOnt_{m1}$)…, … $SemOp_k$ ($SitOnt_{mq}$))} U {$\cup$ (Prc_x; Prd_y, $HrdOp_u$, $UiOp_v$)}.

Среди программных единиц, из которых структурно планируется решатель, *ПрЕд* двух типов являются *онтологическими* (т.е. “программирующими” обработку информации в соответствии с конкретными типами причинно-следственных и других связей в онтологии решаемой задачи):

- $SemOp_m$, *операции* (или наборы операций) *над ОИР*, созданной по конкретной онтологии;

- $SemUnit_m$, *ПрЕд для обработки множества ОИР* и для их модификации; они получают на вход список их имен ИнфКомп, на выходе обеспечивают вычисленные значения (передаваемые через параметры) либо модифицированный ОИР.

Каждый ОИР (онтологическая Информация) явл Информационным компонентом (ИнфКомп) создаваемой СБЗ.

Предложенная архитектурная классификация соответствует потребностям создания сложных эволюционирующих систем, основанных на знаниях.

Онтологический решатель СБЗ обязательно имеет в своем составе онтологические *ПрЕд* (процедурные), “обрабатывающие» элементы информации в соответствии с конкретными типами связей между ними.

Пример. Решатель задачи планирования управления (от которого требуется по характеристикам объекта ObjProp, результатам наблюдения ObjDescr (и Diag) и целевым показателям (Mom, {SignName, SignValue}) предложить совокупность воздействий Actions = {(Mom, Action}) (и объяснить их), может быть реализован через ПрЕд, среди которых есть:

getActionsForModifyingSign

(* инициировать запрос воздействий для изменения признака*)

(KBname, SignName) => Actions;

compareSignValueWithAim

(* сравнить текущее не-нормальное значение признака с целевым условием на значения и определить нужный тренд *)

(ObsKBname, SignName, SignValueCur) => Trend;

getMethodsForTrendSign

(* для признака с указанным трендом нормализации найти группу методов для воздействия на признак в нужном направлении-тренде *)

(MethodsKBname, SignName,Trend) => Methods;

chooseMethodForObject

(*выбрать метод из ГруппыМетодовВоздействия или ЛС, который соответствует характеристикам индивида *)

(MethodsKBname, Methods, ObjDescrName) => Method.

5.9. Создание программных оболочек для систем с базами знаний

В общем случае при решении задачи рассуждение и формирование обоснованных гипотез связано с обработкой хотя бы одного ИнфКомп, содержащего знания и хотя бы одного, содержащего объяснение гипотез о результате решения. Зависимость некоторых частей решателя (типа $SemUnit_m$, $SemOp_k$) от $KnOnt_{jy1}$, …, $KnOnt_{jyw}$, $SitOnt_{jy1}$, …, $SitOnt_{jyx}$, ExplOnt и независимость частей всех типов ($SemUnit_m$, $SemOp_k$, Prc_x; Prd_y, $HrdOp_u$, $UiOp_v$) от баз знаний определяет способность решать задачу по любым базам знаний, созданным под управлением $KnOnt_{jy1}$, …, $KnOnt_{jyw}$. Т.о. онтологический решатель является оболочкой для компоновки произвольного множества СБЗ.

Для удобной в использовании СБЗ программным компонентом станет не только решатель, но и пользовательский интерфейс (ПИФ). Его предназначение – это прежде всего отображение для просмотра и редактирования входной и выходной информации СБЗ. В составе жизнеспособной СБЗ ПИФ нужен адаптивный и адаптируемый, для этого он создается на основе онтологии. Его онтолого-зависимость – это использование информации обо всех связях сущностей из онтологии для отображения входной и выходной информации. Это делает возможной его автоматическую генерацию (хотя бы для прототипов СБЗ). При создании СБЗ для конечных пользователей обычно создается специализированный улучшенный ПИФ, настраиваемый на набор терминов. В его разработке используются две онтологии – ПрОбл (SitOnt ∪ Dict) и онтология ПИФ (часть онтологии технологии разработки ПрогОбес). Наличие онтологии описания информации, а также структуры объяснения дает возможность генерировать пользовательский интерфейс (ПИФ) для ввода фактов в привычных специалистам терминах и генерировать сце-

нарий работы с объяснением результатов (предлагая просматривать списки или подробности подтвержденных или отвергнутых гипотез).

В результате применения этого подхода онтологический решатель вместе с ПИФ (программная составная часть) является пригодным для конструирования мно-жества систем (сервисов). Он не зависит от наполнения баз знаний (использует ее как параметр), поэтому их полнота и качество могут улучшаться бесконечно, интегриру-ясь с решателем в новые и разные версии полезных сервисов. Тем самым обеспечивается возможность усовершенствования сервисов без участия программистов, т.е. в условиях уже существующих программных компонентов.

Для возможности решателю задач (с ПИФ) быть оболочкой, т.е. быть использованным как часть множества разных сервисов), Решателю должен соответствовать специфицирующий его документ. В нем важно точное указание онтологии обрабатываемых знаний и данных – все $KnOnt_{jy1}$, …, $KnOnt_{jyw}$, $SitOnt_{jy1}$, …, $SitOnt_{jyx}$ и ExplOnt. Специфицирование онтологии (метаинформации) обрабатываемых Решателем знаний и данных является основой интегрируемости решателя задач с другими компонентами сервисов).

Документ-спецификация Решателя может быть декларативным (интерпретируемым человеком и машиной). Кроме спецификации форматов знаний и данных могут быть определены все $Unit_m$ и даже порядок их выполнения и связей друг с другом. Такой специфицирующий интерпретируемый документ (декларация Решателя) может использоваться а) автоматически для контроля сборки, для динамической (run time) загрузки частей по мере потребности в них, б) человеком – для сборки СБЗ из Решателя и всех обрабатываемых ИнфКомп. А также для его сборки из ПрЕд, если для них создаются свои специфицирующие документы (для SemUnit и SemOp явно указывается формат / онтология обрабатываемой Информации). Проектирование архитектуры решателей задач из декларируемых программных компонентов разных типов – шаг в направлении создания жизнеспособных систем (и еще один ключевой принцип проектирования).

Декларирование решателей задач, проектирование архитектуры решателей задач в соответствии с типом решаемой задачи и набором рассматриваемых структурных и причинно-следственных связей между сущностями, сборка решателя из заменяемых программных компонентов, – обеспечивают прозрачность его архитектуры и сопровождаемость. Это особенно важно в условиях коллективной разработки и развития.

5.10. Особенности архитектуры Решателя IACPaaS

Реализация оболочек для СБЗ на облачной платформе IACPaaS характеризуется следующим [24, 26, 27]. Решатель IACPaaS создается из программных агентов IACPaaS, которые (по классификации http://www.aiportal.ru/) являются реактивными агентами, и взаимодействуют друг с другом через сообщения установленных форматов [26,28]. В исходном коде таких агентов программируется функциональность – обработка информации, связанная с конкретными типами причинно-следственных и других связей, характерных для решаемой задачи [28]. Вышеупомянутая работа с фактами, с выводами и объяснениями производится в иерархических семантических сетях – ИнфРес IACPaaS (играющих роль ИнфКомп сервиса) [130, 166].

При реализации на облачной платформе IACPaaS архитектурная модель программных решателей имеет «слои» таких типов:

I. API работы с иерархическими семантическими сетями [131],
II. операции доступа к иерархическим сетевым ИнфРес, управляемым их метаинформацией, реализованные через API;
III. программные агенты, проводящие анализ или построение фрагментов ИнфРес, в т.ч. через операции доступа к ИнфРес.

Использование операций доступа к ИнфРес IACPaaS (запрограммированных с учетом онтологии – метаинформации IACPaaS, управляющей этими ИнфРес) призвано повысить прозрачность и понятность программируемых агентов IACPaaS (и снизить сложность) [26,27].

Чтобы использовать решатель как оболочку, создается IACPaaS-декларация решателя [26, 28, 166]. Этот документирующий информационный ресурс содержит список ссылок на онтологии всех обрабатываемых ИнфКомп, читаемых и формируемых.

Для повышения прозрачности решателя задач в его декларации указываются его модули (ПрЕд): как минимум, указывается агент, начинающий работу. Другие ПрЕд могут явно не перечисляться, а быть указаны внутри “первого” агента при передаче им управления. Другой вариант – указание в декларации очередности работы ПрЕд и связывания их с нужными информационными ресурсами – внешними и локальными**.** Очередность работы ПрЕд, умеющих принимать решения по результатам анализа фактов и

знаний некоторого типа отношений, определяется управляющей конструкцией (на платформе IACPaaS для этого создается «управляющий граф»).

Для интегрируемости онтологических ПрЕд/модулей (SemUnit и SemOp) их внешние связи (метаинформация обрабатываемых знаний и данных, форматы входных и выходных сообщений) тоже специфицируются (в соответствующей декларации этой ПрЕд – агента). Декларация ПрЕд содержит формат обращения к ней со списком типов параметров. Декларация онтологической ПрЕд содержит среди параметров имена обрабатываемых ИнфРес и дополнительно содержит список ссылок на онтологии этих обрабатываемых ИнфРес. Декларирование включает и описание назначения ПрЕд, что способствует их повторному использованию.

В отличие от декларации решателя (на облачной платформе IACPaaS) декларация агента связывается с байт-кодом этой ПрЕд (прикрепляется ссылкой) [130].

На платформе IACPaaS генерируется автоматически «стандартный» ПИФ непосредственно по онтологии. Он достаточен и удобен для прототипов, позволяя пользователю работать с входной и выходной информацией в виде иерархических семантических сетей (ИерСемСет) [111,131].

5.11. Апробация Методов конструирования и декларирования решателей

Метод конструирования и декларирования опробован на платформе IACPaaS на решателях ряда практических задач (диагностика, запрос дополнительной информации для сокращения числа гипотез, планирование воздействий на систему или объект; выявление признаков возникновения чрезвычайных ситуаций).

Решатели задач диагностики (как выдвижения гипотез о диагнозах), запроса дополнительной информации (как поиска одного или нескольких признаков, значения которых необходимо запросить для дифференциации между несколькими гипотезами) сконструированы из *SemUnit*, примеры спецификаций которых представлены ниже.

MakeHypSignsModel

(*построить модель общих признаков для множества гипотез-диагнозов*)

((DiagKnOnt); (DiagKnName: path, Hyp: setOfString)

=> HypSignsModel: setOfTriples);

CheckNormalState

(*проверить гипотезу-объект-в-норме*)

((ObjDescrOnt, ObsKnOnt); (ObjDescrName: path, ObsKnName: path)

=> Answer: {norm, fail}.

ConfirmConditionsForDvlpmVari

(* *подтвердить НеобУсл варианта процесса-диагноза*)

((ObjDescrOnt, KnOnt);

(ObjDescrName: path, KnB: path, DvlpmVari: string, (Charact, Value): Pair)

=> Answer: {confirm, unknown}).

ConfirmVariForDiag (* *подтвердить у объекта вариант для процесса-диагноза* *)

((ObjDescrOnt, KnOnt);

(ObjDescrName: path, KnB: path, Diagns: string, DvlpmVar: string

=> Answer: {confirm, unknown}).

SemUnit сконструированы из операций (SemOp), например, SemUnit *CheckNormalState* обращается к операциям:

getStaticSigns

(* получить все статические признаки-наблюдения*)

((ObjDescrOnt); (ObjDescr: path) => ObjSigns: setOfPairs),

getTemporalSigns

(**получить все динамические признаки-наблюдения* *)

((ObjDescrOnt); (ObjDescr: path) => ObjSigns: setOfTriples),

compareStaticSignValueWithTheNorm

(*«сравнить значение указанного статического признака с нормой» *)

(KBtermOnt); ((KBterm: path, (Sign, Value): Pair

=> Answer: {norm,fail}),

compareTimeSeriesValuesWithTheNorm

(*сравнить каждое значение указанного динамического призн с нормой *)

(KBtermOnt); (KBterm: path, (*Sign*, *Value*, Tmom): Triple

=> Answer: {norm,fail}).

При разработке первых версий СБЗ для диагностики были использованы средства стандартного ПИФ: ввод информации пользователем и просмотр объяснения результата производится через штатный редактор и просмотрщик ИерСемСетей. При переходе от

бета-тестирования к экспериментальным исследованиям для конечных пользователей создавался специализированный ПИФ, настраиваемый на набор терминов (Dict).

Например, создается декларация для IACPaaS-Решателя диагностики включающая его название, описание, онтологию (метаинформацию) базы диагностических знаний, Знаний о нормах признаков, онтологию (метаинформацию) историй болезни, структуру объяснения: (DiagnOnt, NormOnt, HistOnt, DignExplOnt). Такой решатель диагностики стал частью систем диагностики нескольких медицинских профилей, будучи интегрирован с соответствующими профильными БЗ.

5.12. Сопровождение и повторное использование программных единиц

Эффективное конструирование (снижение трудозатрат) решателей ряда практических задач основано на повторном использовании программных единиц и операций. Одни и те же виды структурных и причинно-следственных связей могут приниматься во внимание при решении разных классов задач (прогноз, диагностика, планирование и др.) по единой модели течения процесса. В частности, построение заключения о соответствии подмножества фактов - знаниям некоторого типа причинно-следственных связей, характерного для задач разных классов (например, диагностики и прогноза) является часто выполняемой обработкой. Для Решателей таких задач (или интегрирующих их составных задач) шанс повторно использовать ПрЕд, которые делают такое заключение, увеличивается. Даже при создании одного Решателя задачи частые (повторяемые) варианты поиска, сравнения, выбора, добавления Информации тоже должны оформляться в самостоятельные ПрЕдиницы, обрабатывающие ИнфРес (для повторного использования).

Кроме того, сами решатели, реализующие поддержку своей задачи на основе обхода Базы знаний, способны стать повторно-используемыми, поскольку, во-первых, набор традиционно решаемых задач (диагностики, прогноза, управления, планирования….) известен [41,97, 120], во-вторых, традиционный набор типов связей в предметных областях с ПричСлОт (для решения задач диагностики, прогноза, управления состоянием объекта) ограничен.

Повторное использование решателя выполняется так:

- выбор задачи в соответствии с классификацией задач (см. гл.2) и с уточнениями в зависимости от свойств предметной области (дополнительных типов отношений в онтологии),

- поиск готового решателя задачи (например, в доступной библиотеке),

- возможно, адаптация его к формату результата работы (объяснению) через замену ПрЕд (операций над этим ИнфРес), которые его формируют (делают запись в «объяснение»).

Когда объяснение «спланировано» как отчет с уникальной структурой, многие компоненты решателя будут специализированы, а повторное использ-е будет меньше. Созданные ранее для других решателей SemUnit (фрагменты алгоритмов обработки данных и знаний) могут быть использованы при условии модификации SemOp (части кода), которые делают вывод в объяснение.

Когда объяснение «спланировано» как универсальный отчет об анализе всех связей понятий, относящихся к решаемой задаче или к анализируемому типу знаний, компоненты решателя, формирующие объяснение, создаются и используются по аналогии с известной парадигмой сборочного программирования [160].

С использованием SemUnit и SemOp (программных единиц) некоторого решателя могут создаваться специализированные инструменты обеспечения качества баз знаний, т.к. ориентированы на те же виды связей.

Например, (для диагностики, прогноза, оценки риска) и в решателе, и для проверки правильности знаний может быть использована одна ПрЕд *оценки соответствия признаков варианту развития*. ПрЕд для *оценки соответствия признаков объекта варианту развития* формирует фрагмент ИнфРес, перечисляя *признаки* – подтверждающие и опровергающие рассматриваемый *вариант развития*.

Повторной используемости при создании решателей способствует «послойная» стратегия формирования этих программных компонентов СБЗ [122]. Такое «послойное» их формирование повышает шансы расширять функциональность СБЗ эффективно.

В процессе сопровождения при расширении функциональности СБЗ (сопутствующими подзадачами) к решателям могут добавляться ПрЕд, обрабатывающие потребовавшиеся типы связей понятий, описанные в той же онтологии ПрОбл (или даже при ее расширении дополнительными отношениями).

Пример. Если для практического применения в решатель диагностики потребуется добавить рекомендации по выбору исследований для сокращения числа гипотез, то архитектура решателя расширится одним из следующих способов. 1) После окончания работы совокупности ПрЕд, умеющих в совокупности выдвинуть все гипотезы, управление будет передано к ПрЕд запроса дополнительной информации. ПрЕд будет связана ее с ИнфКомп с объяснением гипотез и с БЗ диагностики. 2) *ПрЕд оценки соответствия* будет расширена *процедурой формирования* для проверяемого комплекса всех обязательных признаков*, которые могли бы подтвердить гипотезу.*

Отметим, что более широкое использование программных компонентов потенциально обеспечивается на основе более общих (абстрактных) онтологий. Если решатель ориентирован на универсальную онтологию задачи, то и сам решатель и его ПрЕд (понятные и прозрачные) могут использоваться для разработок в произвольной области, где решаются частные случаи этой задачи. «Другое использование» - решатель по универсальной онтологии может быть и расширен (функционально) для некоторой ПрОбл, если ее онтология является расширением универсальной онтологии диагностики процессов, включая дополнит-е в этой ПредмОбл связи.

Пример для ПредмОбл «медицина»: стандартную онтологию диагностики желательно «углубить» (1) или расширить (2) для учета в качестве характеристик объекта индивидуальных особенностей. В связи с этим онтологию объяснения потребуется расширить, и для подтверждения или опровержения подклассов, потребуется передавать управление аналогичной ПрЕд *оценки соответствия*, но проверяющей симптомы «на уровне» подкласса.

При создании универсального решателя используется и универсальная онтология (формат) описания действительности. Например, для задач анализа это динамическое признаковое описание объекта (множество троек). Тогда для ПрОбл при необходимости решатель будет обращаться к дополнительной ПрЕд, добавляющей к универсальному признаковому описанию специфичные характеристики объекта.

Если же для ПредмОбл стандартная онтология нуждается только в адаптированных терминах (например, использовать вместо термина «отклонение» и «воздействие» – термины предметной области болезнь и лечение) – достаточно в копии ПрЕд заменить

только названия узлов-прототипов (терминов онтологии), по которым совершается обход базы знаний.

5.13. Среда декларирования и разработки решателя

Для обеспечения сборки Решателя из заменяемых компонентов и повышения прозрачности его архитектуры актуальны средства декларирования всех составных частей - ПрЕд (и порядка их запуска), контроля доступности используемых ими ИнфРес, стандартизация их взаимодействий (шаблоны обращений), что особенно важно в условиях коллективной разработки и развития.

Следовательно, важна инструментальная поддержка декларирования решателя (произвольной задачи) и декларирования его компонентов. Кроме средств (инструментальных сервисов) декларирования нужны средства создания заготовок исходных кодов по декларациям, средства кодирования новых ПрЕд, средства каталогизации ПрЕд как потенциально *повторно используемых компонентов (ПИК)* и средства интеграции ПИК и новых ПрЕд в новые решатели или их новые версии. Таков инструментарий для декларативно-компонентного метода формирования (*программных) решателей.*

Комплекс таких Инструментальных сервисов может быть создан на основе онтологического подхода. Тогда технология разработки рассматривается как прОбл, а онтология процесса разработки станет основой онтологических *программных* средств разработки (инструментальных сервисов). В частности, инструментальные сервисы декларирования решателя и ПрЕд – это Редакторы (соответствующей информации), генерируемые по части *онтологии технологии разработки СБЗ*, описывающей содержание (и формат) декларации.

На основе этого онтологического подхода в зависимости от потребностей разработчиков можно по-разному организовать «среду разработки и развития» для СБЗ.

Минимальная среда разработки и развития СБЗ представляет собой:

- онтологию ПрОбл (включая базу терминов),
- сгенерированные редакторы ИнфРес (БЗ и баз данных),
- инструменты декларирования решателя и ПрЕд,
- среда разработки программ (IDE) для кодирования ПрЕд.

Типичная среда разработки и развития дополняет *минимальную* множеством подготовленных и проверенных ИнфКомп (БЗ, архивов и баз данных – DB, KN_{iu}); библиотеками подготовленных и протестированных ПрЕд ($KnOnt_{m1}$, $SemOp_m$, Prc_x, Prd_y, $HrdOp_u$, $UiOp_v$), обрабатывающих ИнфРес; множеством подготовленных решателей задач $Solv_j$, созданных из подготовленных ПрЕд.

В состав *полной среды* разработки и развития СБЗ включаются средства оценивания БЗ и средства индуктивного формирования БЗ и другие инструменты усоверш-я баз знаний.

Инструментарием разработчиков программных частей обеспечиваются свойства СБЗ, относящиеся к жизнеспособности:

- допустимость усовершенствования метода принятия решения;

- допустимость изменения или добавления функций (например, формирования дополнительных результатов);

- адаптивность ПИФ пользователя в связи с изменением входных данных и формируемых результатов функций.

В зависимости от роли системы (прототип, программа для одного профессионального коллектива, программа для применения в профессиональной среде) коллективу разработчиков обеспечивается *минимальная* либо *типичная* либо *полная среда разработки и развития*.

Среда разработки и развития СБЗ IACPaaS формируется из накопленных в облачном хранилище единиц разных типов (информационных и программных) [130]. Методы и инструменты декларирования решателя нацелены на обеспечение прозрачности его архитектуры, а декларирование ПрЕд – на поддержку создания их исходных кодов и динамическое подключение в процессе выполнения.

5.14. Создание программных компонентов для инструментов управления качеством баз знаний

Поскольку *качество базы знаний (*БЗ*)* проявляется ее соответствием текущим знаниям сообщества специалистов, то реализуются такие функции управления качеством, как уточнение БЗ и адаптация БЗ к новым случаям (не «охваченным» теорией). Система

управления базой знаний «вынуждена» быть ориентирована на ту же онтологию: для типа задачи (диагностика, прогноз, управление…) или для проблемы в ПрОбл (медицинская диагностика, лечение,…).

Для *базы знаний управления состоянием объекта* адаптация к новым случаям - это:

А) появление новых случаев (в реальной истории), когда к требуемому состоянию привело некоторое воздействие, еще не описанное в текущей БЗ;

Б) появление новых отклонений в функционировании или состоянии, - для которых не написаны руководства по исправлению или улучшению.

Пример. Для *базы знаний диагностики* адаптация к новым случаям - это:

А) появление новых случаев, когда (один) окончательный диагноз (в реальной истории, случае) не соответствовал известным его проявлениям (описанным в текущей БЗ) и поэтому был отвергнут как гипотеза;

Б) появление новых отклонений в функционировании – диагнозов, - для которых не написаны документы-диагностические руководства.

Для *базы знаний* диагностики уточнение БЗ – это процесс усовершенствования по случаям, для которых *правильно выставленный на практике диагноз* (и только один), оказывался в категории «не подтвержден, но и не отвергнут» по БЗ или в категории «один из множества выдвинутых гипотез».

Для предложенного ранее (гл. 4.) управления качеством баз знаний на основе прецедентов (рисунок 5-1) предложен *метод онтологической реализации* системы управления. Онтологическая реализация *системы управления качеством баз знаний* означает использование онтологии действительности SitOnt, определяющей структуру прецедентов и структуры отчета ExplOnt [35]. В качестве структуры *отчета о неполноте и неточности БЗ* для *системы управления качеством* БЗ используется онтология отчета о несоответствиях с их объяснением.

Например, для управления диагностическими базами удобен вариант структуры «отчета с объяснением» со следующими связями:

{Диагноз$_i$ (не выполненное условие | [, заключение о невыполнении] {вариант проявления$_{ij}$ (не выполненное условие | <Подтверждающие признаки {название признака$_{ijk}$ <Значение признака$_{ijk}$, Момент$_{ijk}$ >}, Опровергающие признаки {название

признака$_{ijm}$ <Значение признака$_{ijm}$, Момент$_{ijm}$, период$_n$>}, Не найденные признаки {название признака$_{ij}$, период$_n$>}>)}}.

Программное средство проверки правильности БЗ работает с ExplOnt, в котором собрано и объяснено несоответствие утверждений БЗ – описанию ситуации в эталонных примерах [35].

Это и другие специализированные инструменты обеспечения качества используют наборы единых онтологических *ПрЕд и* операций, что и решатель на основе знаний, т.к. ориентированы на те же виды связей. Т.о. применяется принцип повторно использования программных единиц [122] и операций.

Например, для диагностики в решателе и для проверки правильности знаний о диагностике может быть использована одна ПрЕд *оценки соответствия признаков симптомокомплексу*. ПрЕд для *оценки соответствия признаков объекта очередному симптомокомплексу* формирует фрагмент объяснения, перечисляя *признаки* – подтверждающие и опровергающие *симптомокомплекс*.

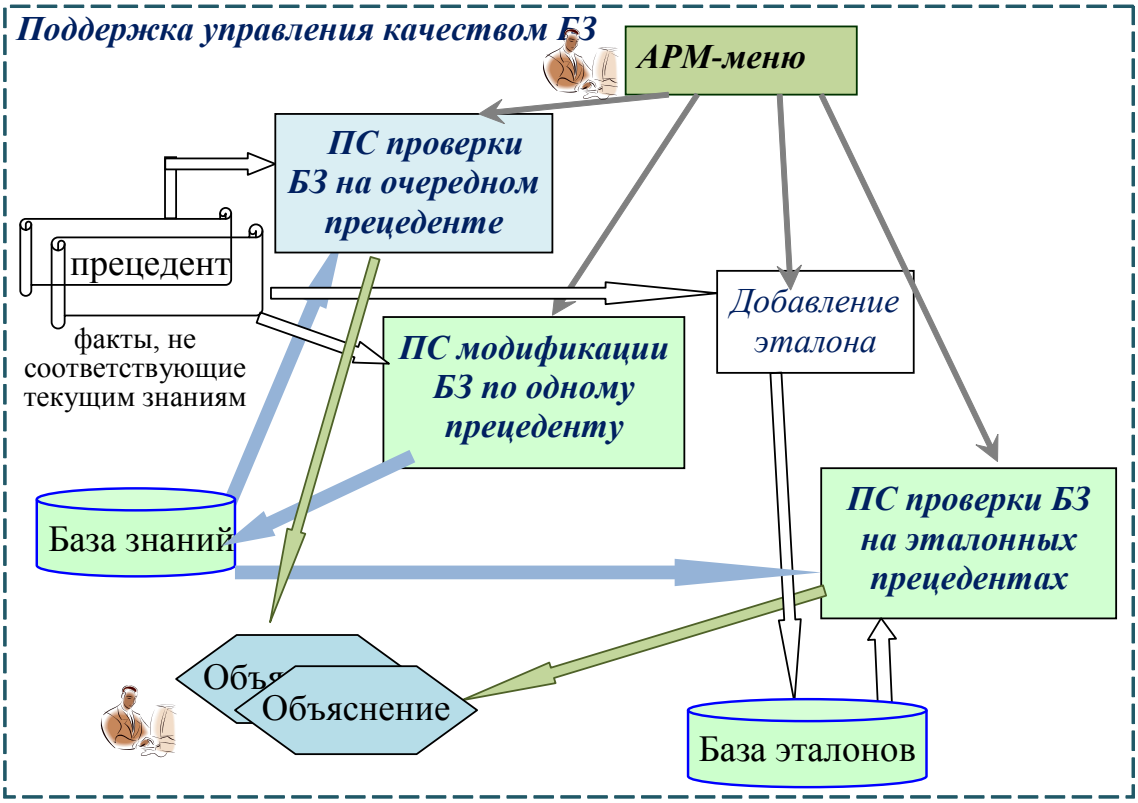


Рисунок 5.1 – Инструменты *системы управления качеством* БЗ

ПрЕд обрабатывают факты из прецедентов и конкретные знания определенного вида причинно-следственных связей. В совокупности они выводят следствия из причин, если причины, описанные в БЗ, найдены среди фактов, указанных в прецедентах, или среди следствий, выведенных другими ПрЕд.

Для реализации *системы управления качеством* БЗ в качестве онтологии эталонных прецедентов и онтологии обучающих прецедентов используется онтология описания решенных задач (законченных случаев). А для осуществления процесса *управления качеством этих* БЗ, в т.ч. их обучения, в качестве эталонных прецедентов используется архив (проверенных) случаев с известным правильным решением.

Например, в медицинской ПрОбл используются Онтология истории болезни (ИБ) или архива таковых и архив проверенных историй бол (ИБ) одной нозологии.

В проектной (design) модели *системы управления качеством* (как *среды оценивания и модификации БЗ)* отчет с «объяснением» (Report) является нелокальным ИнфРес, к которому имеют доступ все ПрЕд подсистем (рисунок 5.2).

Пример. Архитектурный проект (design) подсистемы оценивания и подсистемы модификации баз знаний диагностики процессов использует такие ПрЕд: CheckPrincipalConditions (Проверка в прецеденте выполнения НеобхУсловия диагноза и *вариантов проявления* диагноза), FindPrincipalSigns (проверка наличия в прецеденте важных *наблюдений* одного *варианта проявления* диагноза), CheckPrincipalFacts (проверка наблюдения или его временного ряда на соответствие ожидаемым значениям), FindComplexAndSignsForModidication-v-reject (Выбор *варианта проявления* диагноза, наиболее соответствующего прецеденту, и выявление опровергающих признаков в прецеденте), ПрЕд FindComplexAndSignsForModidication-v-nonconfirm (Выбор *варианта проявления* диагноза, наиболее соответствующего прецеденту, и выявление отсутствующих в прецеденте *наблюдений*), ModidicateValueOfSymptom-v-reject (корректировка *значений признака одного варианта проявления одного отвергнутого* диагноза *в БЗ),* ModificateValueOfSymptom-v- nonconfirm (корректировка *Значений признака одного неподтвержденного варианта проявления одного* диагноза *в БЗ),* ModificatePeriodDurationOfSymptom-v-reject (*корректировка длительностей периодов для варианта проявления), AddPrecedentToSamplesBase (Добавить очередной обучающий прецедент в эталоны для*

одного диагноза*), ConstructNew*ComplexAsGenerated*Precedent (Построить новый вариант проявления* диагноза *как обобщенный прецедент).*

ПрЕд CheckPrincipalConditions ((MedHisOnt, KnOnt, ReportOnt); (MedHis: path, KnName: path, Des: string, ReportName: path) => Res: {fulfilled, not-fulf}) принимает ссылку на описание одной нозологии в БЗ и ссылку на прецедент (во множ-ве прецедентов). ПрЕд анализирует причинно-следственные связи: «все необходимые условия диагноза - гипотеза-диагноз», «гипотеза-диагноз – любой вариант проявления (заболевания)», «все «необходимые условия» – вариант проявления». В «отчете» (нелокальном ИнфРес), создатся фрагмент - {Диагноз (нозология)$_i$ (не выполненное условие | [, заключение о невыполнении] {вариант проявления$_{ij}$ (не выполненное условие)})}. Т.е. на выходе ожидаются множества невыполненных условий и названия неотвергнутых-по-НУ вариантов проявления (СимптКсов), которые далее будут проверяться.

ПрЕд FindPrincipalSigns ((MedHisOnt, KnOnt, ReportOnt); (MedHis: path, KnName: path, Des: string, ReportName: path) => Res: {conf, not-rej, reject}) принимает на вход ссылку на описание одного *варианта проявления* в БЗ и ссылку на прецедент. ПрЕд анализирует причинно-следственную связь: «*вариант проявления* - все важные *«признаки» этого варианта».* В «отчете», который является нелокальным ИнфКопм, создается множество фрагментов {вариант проявления$_{ij}$ <Не найденные признаки {название признака$_{ij}$, период$_n$>} и, возможно, временный узел «список найденных признаков».

ПрЕд CheckPrincipalFacts (проверка факта или временного ряда на соответствие необх значениям) принимает ссылку на описание ожидаемого наблюдения в БЗ и ссылку на прецедент (во мн-ве прецедентов). ПрЕд анализирует причинно - следственные связи: «важный *«признак»* - все *периоды развития этого признака», «период развития признака – область возможных значении (ОВЗ) признака для этого* периода *развития», «ОВЗ признака* – любое значение из ОВЗ». В «отчете» для процесса (диагноза)$_i$ создается множество фрагментов «вариант проявления»$_{ij}$: <Подтверждающие признаки {название признака$_{ijk}$, <Значение признака$_{ijk}$, Момент$_{ijk}$>}, Опровергающие признаки , {название признака$_{ijm}$, <Значение признака$_{ijm}$, Момент$_{ijm}$, период$_n$>}>.И т.д.

Использование прецедентов для проверки (правильности и/или точности) БЗ, реализуется через подобный набор ПрЕд. Для повышения качества БЗ (приведения ее в соот-

ветствие уровню совокупных знаний специалистов) рассматриваются следующие виды прецедентов (из потока решенных и проверенных задач):

прецедент, который «отвергался», т.е. при проверке БЗ созданные гипотезы не охватили записанное в нем решение

(для прогноза – гипотезы-варианты состояний в рассматриваемый момент не совпали с реальностью;

для БЗ диагностики это: вариант прецедента, когда (один) окончат диагноз (в ИБ) не соответствовал известной клинич картине в текущей БЗ, т.е. необходимые значения признаков вошли в противоречие;

для конфигурирования – множество гипотез-вариантов сборки не содержало созданный в реальности,

для управления – гипотезы-варианты воздействий на состояние не содержали примененного в реальности),

прецедент, который не подтвердился по БЗ (но, по мнению ответственных экспертов, фактов в ИБ для подтверждения было достаточно).

Все новые прецеденты могут быть использованы для расширения эталонной базы, полученная версия которой может быть использована для автоматической индукции новой версии БЗ индуктивными инструментами из состава *полной среды разработки и развития* СБЗ.

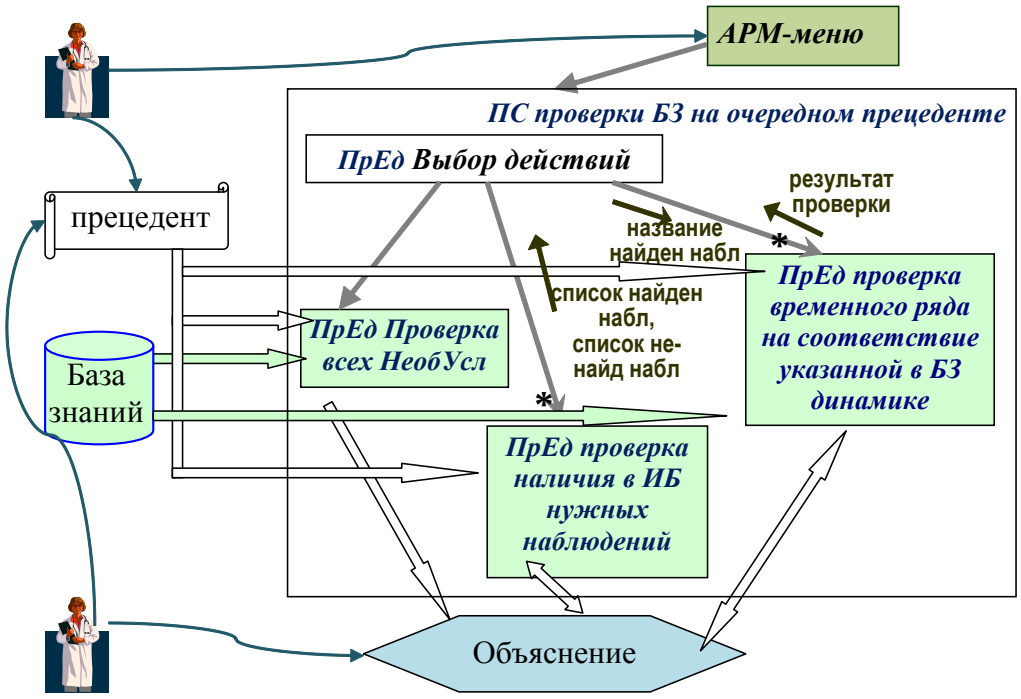


Рисунок 5-2 – Повторно используемые ПрЕд инструментов *системы управления качеством* БЗ

5.15. Выводы из главы 5

Онтолого-базированный решатель в общем случае обрабатывает несколько информационных ресурсов (ИнфРес), сформированных или формируемых по онтологии. В процессе рассуждения и принятия решений алгоритм опирается на формализованные связи и зафиксированные онтологические соглашения и использует онтологию объяснения. Он проектируется с учетом повторного использования программных единиц двух уровней (в т.ч. онтологических, обрабатывающие формализованные связи).

Особенности, отличающие онтологический решатель от других решателей, таковы:

- алгоритм формируется по онтологии и не зависит от самих БЗ, что соответствует современному подходу к разработке интеллектуальных систем;
- в нем «естественно» реализуется подробное и понятное специалисту объяснение предлагаемого решения (структура объяснения является «отражением» онтологии знаний).

Совокупными ключевыми **принципами** технологии создания жизнеспособных систем являются:

- проектирование декларативных БЗ по онтологии знаний,

- проектирование архитектуры решателей задач в соответствии с онтологией знаний и данных (конструирование из онтологических программных единиц);
- создание подсистемы управления качеством всех БЗ с проектированием ее компонентов.

В соответствии с этим модель жизнеспособной СБЗ уточнена. Онтология ПрОбл, включающая онтологию знаний для решения одного или нескольких смежных классов задач, становится структурной основой для конструирования сопровождаемых интеллектуальных программных сервисов, а также инструментов управления их качеством.

Инструментальной средой для создания таких решателей задач является платформа IACPaaS – единая среда для построения проблемно-ориентированных онтологий и формируемых в их терминах (и под их управлением) баз знаний и решателей соответствующих классов задач. В ней же создаются средства обеспеч качества БЗ. Реализация решателя и средств проверки качества БЗ производится по единым семантическим моделям: по онтологии данных, знаний и структуре отчета о сопоставлении данных и знаний.

Онтолого-ориентированная среда развития реализует механизмы адаптивности таких систем. Она развивается и эволюционирует, когда создаваемые в ней прикладные СБЗ нуждаются в адаптации к условиям функционирования и уточнению знаний ПрОбл. Она развивается и эволюционирует также, когда «базовый инструментарий» нуждается в адаптации к новой аппаратуре и системному ПрОбесп, к новым видам человеко-машинных интерфейсов. Эти механизмы адаптивности обеспечивают возможность усовершенствования каждого компонента силами узких специалистов, гарантируя целостность всей СБЗ.

ГЛАВА 6. Технология разработки и развития СБЗ и ее практическое использование

В главе представлена облачная технология разработки жизнеспособной СБЗ, требования к комплексу инструментальных средств разработки и сопровождения всех компонентов конфигурации СБЗ, метод реализации такого комплекса инструментальных средств, и метод формирования специализированных технологических *сред создания и развития* БЗ и СБЗ. Обобщен опыт применения предложенной технологии в отдельно взятой предметной области. Проведена сравнительная оценка затрат на создание и развитие СБЗ разными технологиями.

6.1.Процесс разработки облачных сервисов для решения интеллектуальных задач

Для создания сервисов поддержки решения интеллектуальных (экспертных) задач в некоторой ПрОбл (в соответствии с принципами жизнеспособности) необходима технологическая среда, позволяющая выполнить все этапы Жизненного Цикла сервиса. Требуется сформировать на основе онтологии ПрОбл все компоненты сервисов и все средства управления ими.

Начальные этапы Жизненного Цикла интеллектуальных сервисов ожидаются ёмкими [108], особенно при формировании Портала знаний для решения нескольких задач в одной ПрОбл [59,60]. Этап Системного анализа нацелен как на выявление задач, требующих автоматизации, так и на проработку ПрОбл, в которой решаются эти задачи [54]. Обеспечивается основа задел для разумной модуляризации всех ресурсов многопользовательской среды.

При формировании нового Портала *онтология ПрОбл* создается в самом начале (в идеале создается один раз). Набор понятий и отношений в ней по меньше мере должен быть достаточен для решения интеллектуальных задач, т.е. для представления всей необходимой информации, обрабатываемой в процессе их решения. При апробации *онтологии ПрОбл* проводится прототипирование БЗ и, возможно, методов или алгоритмов

обработки и использования БЗ и БД. Достигается такой уровень устойчивости, когда согласованы единые термины для знаний и данных, структура знаний полна, формат описания данных достаточен. Примечание. Со временем онтология может потребовать развития, расширения (могут добавляться новые термины, связываемые с уже существующими).

После получения устойчивой онтологии эксперты могут коллективно создавать нужные БЗ и другие ИнфРес. Как правило, эта работа – коллективный процесс: каждый участник формирует свой фрагмент, или одни формируют, другие уточняют, добавляют, проверяют [59,60]. Часто одна готовая БЗ применяется для разных задач. (Пример – Фармакологический Справочник как БЗ об отношениях между Лекарственными Средствами и нозологиями / симптомами применяется при решении задач лечения, прогноза и даже диагностики.)

Проектировщики и программисты могут создавать новые программные компоненты. Системный анализ обычно подразумевает декомпозирование проблемы на совокупность интеллектуальных и обычных подзадач [59]. Для автоматизации обычных реализуются методы программной инженерии (реализуются с учетом интерфейсов и API прочих компонентов).

Для поддержки решения интеллектуальных подзадач могут создаваться новые программные компоненты (параллельно с работой экспертов) [108]. Но по мере развития Портала и предметно-независимых сред разработки повышается вероятность находить готовые для использования компоненты решения интеллектуальных подзадач. Так, (поиск или) определение для (под)задачи (в составе поставленной проблемы) типа (вида) этой (под)задачи по единой классификации задач открывает возможность использовать опыт и наработки для найденного типа (вида) задач: использовать готовые для них решатели или их компоненты (и другие элементы конфигурации). Т.е. открывается перспектива использования готовых решений «как есть» или (для более общей онтологии и абстрактной задачи) путем уточнения алгоритма перебора элементов информации (с учетом свойств ПрОбл). Некоторая библиотека уже может содержать набор решателей или программных единиц (модулей) для выбранной задачи в ПрОбл.

После автоматизации интеллектуальных и «обычных» подзадач может последовать их интеграция в единый комплекс или систему, что особенно важно для объединения с

документооборотом автоматизируемой организации. Решатели на основе знаний будут подсистемами создаваемой СБЗ [59,60].

При нахождении (под)задачи в единой классификации задачи отсутствии готовых решателей или их компонентов возможность использовать только онтологию соответствующих знаний (типы понятий и отношений для решения такой задачи) тоже дает экономию затрат.

Разработка нового решателя по онтологии – это типичный процесс модульной разработки ПС. Каждый модуль (среди которых есть онтолого-ориентированные) декларируется, что позволяет автоматизировать создание его кода. Решатель создается через организацию последовательности работы модулей. Одним из средств организации последовательности (или распараллеливания) может быть выбран «управляющий граф» - декларативный компонент Решателя, обеспечивающий его прозрачность. Используемые или создаваемые программные модули (компоненты решателей) для поиска и манипулирования информацией работают в соответствии с установленными в онтологии ПрОбл отношениями между ее элементами. Онтологическая основа обеспечивает совместимость компонентов в рамках формируемых порталов. Декларирование и каталогизация программных компонентов обеспечат удобство их использования в последующих разработках в этой ПрОбл.

При нахождении(под)задачи в единой классификации задач и наличии готового решателя только для более абстрактной задачи экономию затрат даст путь адаптации компонентов абстрактного решателя. Компоненты абстрактного решателя обрабатывают информацию в соответствии с типами понятий и отношений более абстрактной онтологии. А для решения «специализированной» задачи, учитывающей больший спектр отношений или более длинные цепочки связей, понадобится обработка дополнительных *посылок* для принятия промежуточных решений и сопоставление их фактам из Ситуаций Действительности. Это будет добавлено либо непосредственно в код компонентов абстрактного решателя, либо в дополнительные программные единицы (ПрЕд) для *формирования* промежуточных решений, которые будут вызываться из компонентов, реализуя адаптации абстрактного решателя к конкретной онтологии.

Т.о. технология СБЗ предусматривает последовательность работ:

1) создание онтологии ПрОбл, включая поиск готовых онтологий знаний об отдельной задаче и онтологий данных);
2) параллельная разработка по этой онтологии нового решателя и баз знаний,
3) конструирование сервиса (или нескольких - для пользователей с разными требованиями к интерфейсу),
4) разработка средств управления БЗ (с использованием онтологии и компонентов решателя).

6.2. Характеристики среды для поддержки технологии разработки облачных систем с БЗ

Поддержка представленной технологии и Жизненного Цикла требует обеспечения доступности инструментов для выполнения этих видов работ и обеспечения коллективно-параллельной работы разных участников, как описано в гл. 3 (п. «Инструментальная среда для развития») [129].

Инструментарий призван обеспечить «планируемый» и более линейный «путь построения», что отличается от ранних технологий создания «экспертных систем», где предусмотрен многократный возврат к начальным этапам для добавления новой порции знаний-правил к проверенному на тестах прототипу [129].

Однако потребности в инструментах могут быть разными в зависимости от роли СБЗ и от требуемого уровня сервисов:

для одного профессионального коллектива или для применения в профессиональной среде;

прототип или программа на заказ;

сервис или комплекс сервисов;

СБЗ для одной задачи или сервис для спектра задач.

Для пользователей, которые имеют цель наполнять базы знаний и данных вручную и создать СБЗ достаточно минимальной *инструментальной среды:* редактор онтологии ПрОбл, редакторы ИнфРес для наполнения базы знаний, средства специфицирования решателя, его модулей (ПрЕд) и их программирования (или взять готовую онтологию ПрОбл с базой терминов и готовый решатель, то использовать редакторы ИнфРес для наполнения базы знаний и инструмент сборки серрвиса). *Типичная среда* имеет готовые

решатели задач, библиотеку подготовленных и протестированных ПрЕд с понятной функциональностью, версии БЗ и БД. *Полная среда* разработки и развития СБЗ включает средства оценивания БЗ и средства индуктивного формирования БЗ. Перечисленные варианты организации среды создания и развития СБЗ составляют трех-уровневую «аддитивную» модель процесса формирования среды разработки: предыдущий вариант является фундаментом для следующего. Когда в облачном хранилище разворачивается портал, коллективу разработчиков обеспечивается *типичная* либо *полная* среда разработки и развития СБЗ.

Реализующий требования к *типичной среде* разработки и *полной среде* инструментарий призван снизить технические риски, в т.ч. за счет использования готовых решений.

Для воплощения принципов жизнеспособности необходимы средства реализации, отвечающие требованиям, представленным в гл 1 (и связанным с ними «вызовам»):

- доступность коллективу участников разработки,

- возможность создания онтологий ПрОбл, управляющих формированием информации и ее обработкой,

- доступность/наличие проблемно-независимых онтологий («технологических» шаблонов), нужных всем,

- доступность, накапливаемость проблемно-ориентированных онтологий, нужных (разным пользователям или коллективам), а также (при этом) и «обособляемость» онтологий - для использования «общих решений»,

- доступность (и обособляемость) специализированных предметных онтологий,

- удобное внесение и естественное отображение вербальной информации,

- прозрачность конструируемых онтологических решателей,

- повторной используемость информационных и программных компонентов, создаваемых под управлением проблемно-ориентированных или предметных онтологий [GriboShalfOntologicalApproach].

Возможность создания требуемого набора средств обеспечила современная облачная платформа IACPaaS (http://iacpaas.dvo.ru) [130].

6.3. Инструментарий технологии разработки жизнеспособных СБЗ

На платформе IACPaaS реализована двухуровневая модель представления информации, так что «нижний» уровень (метаинформация) предписывает правила формирования содержательного уровня (т.н. «целевой» информации). Такая модель информации (вместе с инструментами IACPaaS для ее формирования - Редактором метаинформации и Редактором информации, - РедМет и РедИнф [76]) позволяют с помощью базового языка формировать KnOnt, SitOnt, используемые для управления представлением следующего уровня – БЗ и баз фактов в форме ИерСемСет.

Для доступа к элементам этих информационных единиц IACPaaS предоставляет API (application programming interface) к сетевым ИнфРес, для поиска и манипулирования информацией [76].

Обеспечивается послойная реализация функциональности: «нижний слой» - это API доступа к элементам иерархических семантических сетевых моделей, следующий слой - операции доступа (поиска и манипулирования) к целевым ИнфРес (базам знаний и фактов), сформированным под управлением онтологии, следующий слой - ПрЕд (агентов), работающих с информацией через эти операции; далее «слой» решателей задач, построенных через ПрЕд. Сервисы («верхний» слой) конструируются из баз знаний, данных, решателей.

Инструментами IACPaaS для формирования онтологии ПрОбл (в виде совокупности семантических сетевых моделей с корнем и циклами) являются РедМет и РедИнф (IACPaaS-Редактор метаинформации и IACPaaS Редактор информации) [22,25, 76]. Первый позволяет создать KnOnt и SitOnt, DictMeta и AgreemMeta второй – Agreem и Dict (под управлением DictMeta, AgreemMeta).

Реализация *инструментальной* поддержки технологии создания СБЗ имеет тоже онтологическую основу – правила формирования специфицирующих документов и шаблонов для всех компонентов являются едиными и явно формируемыми [108, 112, 164]. Среди них: Структура декларирования ПрЕд (UnitDeclarOnt), Структура декларирования решателя задач (SolvDeclarOnt), Структура декларирования сервиса (ServDeclarOnt), а также Онтология базы терминологии (DictMeta) и Структура онто-соглашений (AgreemMeta). Все они – «общие» онтологии и необходимы разработчикам. Их редакти-

рование обеспечивает РедМет (IACPaaS-Редактор метаинформации). Такие «общие» онтологии созданы один раз разработчиками технологии и единообразно используются в процессе создания СБЗ.

После того, как сформирована онтология ПрОбл, технологическая среда становится специализированной.

Для формирования БЗ применяется инструмент РедИнф (IACPaaS-Редактор информации) с само-адаптирующимся (под управлением KnOnt) пользовательским интерфейсом (Ui) [22]. Аналогично для формирования других информационных компонентов СБЗ применяется РедИнф (интерфейс управляется SitOnt) (Рисунок 6.1).

Для формирования декларативной части ПрЕд и других программных компонентов СБЗ – IACPaaS Редактор целевых ИнфРес под управлением метаинформации UnitDeclarOnt (Структура агента), SolvDeclarOnt (Структура решателя задач), ServDeclarOnt (Структура сервиса) (https://iacpaas.dvo.ru/fund).

Помимо IACPaaS-Редакторов комплекс инструментов IACPaaS также содержит [22,23,24]:

- Генератор заготовок программного кода новых IACPaaS-агентов (или новых версий) по декларативной части агента (целевому ИнфРес, созданному по UnitDeclarOnt),

- Загрузчик байт-кода декларативную часть IACPaaS-агента (IACPaaS-агенты кодируются на языке Java, для написания исходных кодов и создания байт-кода агента рекомендована среда разработки (IDE) JavaDevelopmentKit (JDK) для версии Java SE 8),

- Средства тестирования ПрЕд (IACPaaS-агентов) и подготовки их для повторного использования [77].

Т.о. на платформе IACPaaS технологическая среда создания и развития СБЗ формируется из накопленных единиц разных типов – информационных, программных, инструментальных [25].

Формирование Solv (решателя задачи) поддерживается средством декларативного определения (в формате SolvDeclarOnt), и (автоматическим) предоставлением само-адаптирующегося интерфейса для ввода входных данных и для просмотра результата, построенного этим решателем [101, 132]. При этом в IACPaaS-среде предоставляется три способа их визуализации, один из которых графовый [76].

Перечисленные инструменты поддерживают послойное распределение функциональности при планировании архитектуры. В технологии IACPaaS, обеспечиваются (соответственно) единый архитектурный принцип СБЗ и многоуровневая концепция хранения ресурсов (в т.ч. компонентов) для их конструирования (сборки) [22,23,24].

Инструментарий специализированной технологической среды позволяет управлять размещением готовых ПрЕд в папках хранилища для обеспечения их поиска и повторного использования в процессе создания программных решателей или их новых версий [101]. В *среде разработки и развития СБЗ* обеспечивается многообразие вариантов повторного использования готовых решений: как решателей, так и их частей (для создания решателей).

Облачная среда делает доступными (при наличии Интернета) все компоненты разработчикам, заинтересованным в создании сопровождаемых интеллектуальных программных сервисов (для поддержки решений специалистов).

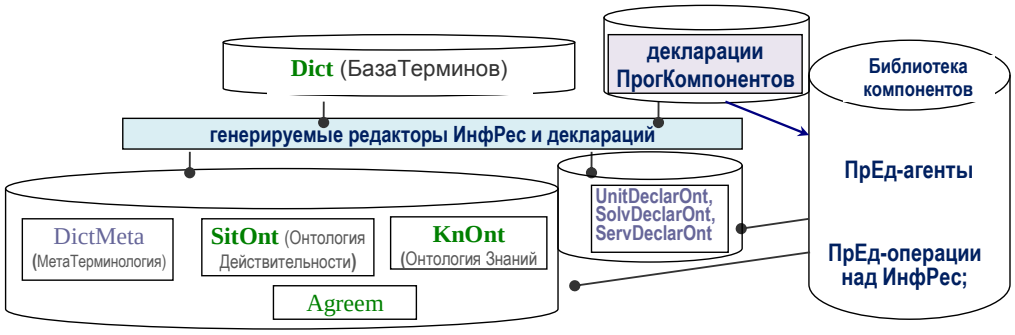


Рисунок 6.1 – Модель специализированной технологической среды

Т.о. IACPaaS – единая инструментальная среда для формирования Порталов знаний и создания на них эволюционирующих (развиваемых) сервисов поддержки решения интеллектуальных задач. После построения проблемно-ориентированных онтологий под их управлением (в их терминах) начинается работа по формированию баз знаний и решателей соответствующих проблеме классов задач. При построении более узкой онтологии - по формированию баз знаний и решателей для конкретной задачи.

6.4. Обеспечение жизнеспособности СБЗ инструментами сред развития IACPaaS

Жизнеспособность для системы, основанной на знаниях, определена в работе как адаптивность к изменениям в среде и адаптируемость в ответ на новые требования. На протяжении эксплуатации СБЗ или целого их комплекса может появляться потребность в регулярной обновляемости знаний, в усовершенствовании Решателя (замене метода принятия решения или в изменении или добавлении функций); в адаптивности интерфейса пользователя. При этом СБЗ оказываются чаще в условиях изменчивости знаний предметной области, реже - в условиях изменчивости среды функционирования и средств пользовательского интерфейса.

А.Потребность в обновлении баз знаний происходит в связи с получением новых знаний через науку или практику.

Б.Потребность в изменении компоновки Решателя происходит в связи с заменой компонента, реализующего другую стратегию принятия решения или метода получения результата или в связи с добавлением компонента, реализующего дополнительную функцию (например, формирования дополнительного результата).

В.Потребность в обновлении / адаптивности интерфейса пользователя происходит в связи с изменением входных данных и формируемых результатов функций, а также в связи с обновлением онтологии.

Предложенная методология и IACPaaS-платформа обеспечивают жизнеспособность создаваемых СБЗ через «свойства» специализированных сред их создания и развития:

механизмы поддержки обновления знаний;

поддержки изменения компоновки онтологического Решателя;

поддержки усовершенствования интерфейсов эксперта и пользователя.

А.Рассмотрим свойство специализированной среды (т.е. сформированного Портала) «поддержка обновления знаний» и свойство СБЗ «обновляемость знаний»**.**

Если требуется обновление базы знаний в связи с получением новых знаний (утверждений), наиболее естественно модифицировать БЗ вручную – заменив устаревшие «предложения» или добавив новые варианты связей. Для этого в специализирован-

ной среде создания и развития имеется Редактор онтологической БЗ. (Его генерирует штатный инструмент IACPaaS по соответствующей онтологии.) (Чтобы получаемые новые знания загрузить в БЗ автоматически, можно воспользоваться (сторонним) онтологическим парсером для перевода текста в утверждения в соответствии с KnOnt и Dict в виде Json-файла и загрузчиком-импортером Json-файла в присоединяемый «модуль» БЗ.)

Для проверки качества новой версии БЗ имеется инструмент получения по версии БЗ *объяснения результата* решения эталонной задачи и сравнения *объяснения результата* с эталонным *результатом.* Этот инструмент – компонент комплекса автоматизации процесса проверки не-ухудшения знаний при замене версии БЗ (все задачи из увеличивающегося множества эталонных задач на новой версии БЗ решаются правильно).

Если требуется обновление базы знаний в связи с получением «фактов» (прецедентов, решений), не согласуемых с утверждениями в БЗ (найден прецедент, с помощью которого знания вообще и БЗ в частности должны быть откорректированы), то эффективно индуктивными методами формировать новую версию БЗ.

Пример такого обновления знаний «от практики»: из медучреждения через определенный промежуток времени «приходит» правильный результат диагностики или лечения, он сравнивается с результатом (объяснением гипотез) от СОЗ: не противоречат ли они друг другу.

Комплект программных инструментов для реализации процесса монотонного усовершенствования баз знаний (медицинского портала) включает:

- сервис индуктивного формирования симпт-в заболеваний (диагностика),
- сервис индуктивного формирования индивидуального плана лечения,
- сервис поддержки выбора прецедентов для свойства правильности БЗ диагностики,
- сервис поддержки выбора прецедентов для свойства точности БЗ диагностики,
- сервис поддержки выбора прецедентов для свойства правильности БЗ лечения (т.е. новых вариантов лечения),
- сервис проверки правильности и полноты новой версии БЗ диагностики,
- сервис проверки правильности новой версии БЗ лечения,
- сервис проверки качества БЗ о течении заболеваний с учетом воздействий.

В полной мере свойство «Обновляемость знаний» присуще СБЗ, создаваемым в *полной среде развития* на платформе IACPaaS. Однако и *типичная среда* позволяет используемым знаниям обновляться: генерируемый Редактор знаний всегда в наличии, остальные инструменты могут быть добавлены в среду развития по мере их готовности.

Технология обеспечения качества БЗ реализует процесс монотонного улучшения баз знаний по новым фактам, поступающим из реальной практики.

Б.Рассмотрим свойство специализированной среды «поддержка изменения компоновки онтологического Решателя» и свойство СБЗ «сопровождаемость Решателя» (если появились причины для улучшения метода получения результата или добавления дополнительных выходных данных)**.**

Помимо обновляемости знаний для жизнеспособности важны, как отмечено выше: возможность усовершенствования метода принятия решения (внесение изменения в процесс принятия решения); допустимость изменения или добавления функций; адаптивность интерфейса пользователя (в связи с изменением входных данных и формируемых результатов функций).

В рамках традиционных потребностей в развитии программных компонентов обеспечивается поддержка: кодирования новых версий программных единиц (онтологического) решателя или изменения их ПрЕд такого Решателя, усовершенствования интерфейса пользователя в связи с изменением функций.

Технологическая среда развития предлагает, в частности, для кодирования новых ПрЕд (в т.ч. операций над ИнфРес и операций в java-классах IACPaaS-агентов) библиотеку IACPaaS-API (работа с информационным наполнением - понятиями и отношениями информационных ресурсов), Gui (построение интерфейса) и др). Библиотека добавляется в «среду» кодирования, компиляции, создания jar, которая может быть локальной для программиста (JDK для Java SE 8) или размещенной на платформе IACPaaS.

Т.о. инструментарий сред развития, реализуемых на платформе IACPaaS, обеспечивает конструирование СБЗ (интеллектуальных сервисов) с жизнеспособной архитектурой: во-первых, с обновляемостью знаний, во вторых, с развиваемостью програмных компонентов и интерфейсов.

6.5. Развитие онтологии

Дополнительно к регулярной обновляемости знаний, сопровождаемости Решателя и адаптивности Ui для исследовательских (т.е. не промышленных, не коммерческих) разработок (с созданием прототипов СБЗ) важна адаптируемость к изменениям в онтологии.

Различаем три случая обстоятельств появления изменений в онтологии: при апробации, в процессе эксплуатации, в рамках развития Портала знаний. Изменения в онтологии могут появиться в связи с накоплением опыта или «знаний» в ПрОбл: при апробации они ожидаются и принимаются, в процессе эксплуатации они ограничены, в рамках развития Портала они контролируются. Обеспечение адаптируемости компонентов СБЗ к изменениям в онтологии является одним из свойств специализированной технологической среды развития IACPaaS.

Примечание. Для практического применения онтология (как основа Портала и среды развития) обычно апробируется до начала ее реального использования при разработке СБЗ. После апробации онтологии допустимы только некоторые «контролируемые» изменения: в частности, внесение дополнительных понятий или новых связей между понятиями (это не нарушает работоспособности программных решателей). Если изменения в онтологии появляются «в связи с» эксплуатацией СБЗ, то в этом случае целесообразно говорить о новой версии СБЗ, т.к. изменения ведут к корректировке всех компонентов СБЗ.

Для обеспечения адаптируемости компонентов к изменениям в онтологии в Среде развития СБЗ поддерживается добавление операций над ИнфРес, в которых понятий или связей стало больше, кодирование (с помощью JDK) новых версий агентов с использованием добавленных операций, когда в процессе принятия решения используются эти новые термины или связи терминов. Помимо добавления (кодирования) новых операций над ИнфРес может производиться и расширение функции существующих операций над ИнфРес. Т.о. для IACPaaS-Решателя адаптация обхода декларативной Базы знаний в связи с обновлением онтологии знаний производится через обновление операций над ИнфРес, используемыми IACPaaS-агентами.

Пример. При расширении онтологии знаний (KnOnt) в медицине был добавлен дополнит тип провоцир факторов для начала заболевания - узел типа «метрика», для учета узла в процессе Решения изменена операция *дать Множ-во факторов для Заболевание* над ИнфРес «БЗ», которая при формировании множества *факторов обойдет узлы «событие», «патология» и* «метрика», а в java-коде IACPaaS-агента новой операции не понадобится.

В рамках продолжения автоматизации ПрОбл и развития Портала знаний естественно добавление понятий и/или их связей. Часто это необходимо для представления знаний, относящихся к еще одной задаче. Добавление понятий и/или их связей может оформляться в виде новых «модулей» к онтологии ПрОбл - модулей онтологии знаний о понятиях, позволяющих решать дополнительные задачи. (Например, после получения готовой среды для решения задачи распознавания объектов возникла потребность в конструировании желаемых объектов или в воздействии на них с целью изменения их свойств.)

Обеспечение возможности обновления онтологии (внесения изменений и создания новых «модулей») предоставляет штатный инструмент редактирования метаинформации IACPaaS, а генерируемость редакторов БЗ по онтологии является свойством всех специализированных технологических сред создания и развития БЗ и СБЗ на IACPaaS.

6.6 Опыт формирования среды создания и развития БЗ и СБЗ в предметной области

Одним из практических применений подхода и методологии развиваемой среды было использование инструментального комплекса для коллективной разработки жизнеспособных СБЗ на медицинском облачном Портале знаний.

Процесс формирования Портала знаний и интеллектуальных сервисов в предметной области «медицина» был начат с формализации современной версии медицинской онтологии, номенклатура, типы сущностей и связи которой являются результатом много-десятилетнего опыта проведения системного анализа с участием экспертов, когнитологов, аналитиков и проектировщиков.

Медицинская онтология - основа управляемых интеллектуальных сервисов в предметной области «медицина», обеспечивающая разработку онтологических баз знаний,

онтологических программных компонентов для решения требуемого спектра задач и компонентов специализированных СУБЗ.

Разработка онтологии ПрОбл охватила работы:

- создания словаря медицинских терминов (около 15000), прежде всего для представления наблюдений за состоянием пациентов, а также описания внутренних процессов (заболеваний) и лечебных воздействий на них,

- создания онтологии истории болезни (ИБ) и онтологии электронной медицинской карты (около 200 понятий - узлов семантической сети) как онтологию действительности,

- адаптацию онтологии диагностики к медицинской диагностике (добавлено около 30 понятий и столько же отношений) [134],

- специализацию онтологии воздействий до онтологии схем лечения заболеваний с учетом особенностей пациентов (добавлено около 40 понятий),

- разработку онтологии знаний о номенклатуре и воздействиях лекарственных средств на организм человека с различными нарушениями функционирования (около 80 понятий),

- разработку форматов подробных «объясняющих» отчетов для врачей; это иерархически устроенные документы, «ведущие» от предложенных гипотез (для решаемых задач) к подробному их объяснению на основе знаний и анализа данных. По такой модели обоснования принятого решения интеллектуальный решатель формирует структурированный отчет в терминах специалистов

Полученная онтология истории болезни обладает свойством полноты видов информации об особенностях организма, фактах, событиях и наблюдаемых проявлениях происходящих/рассматриваемых заболеваний. Под ее управлением были внесены реальные истории болезни, соответствующие интересам специалистов, а именно: относящиеся к заболеваниям пищеварительной системы [81], к специфическим для региона лихорадкам, вызываемым грызунами, к мукополисахаридозам.

Онтология знаний о диагностике, позволила формализовать разные заболевания как многовариантные развивающиеся внутренние процессы [32,33,134]. Всего определено семь групп заболеваний, в каждой из которых формализовано по несколько нозологий [32,33]. Онтологии схем лечения заболеваний и воздействий Лекарственных

Средств на организм позволили определять схемы лечения и составлять справочник Лекарственных Средств.

Онтология знаний (как наиболее требовательная к качеству) была оценена по ряду структурных свойств. Для иерархических семантических сетевых моделей характерны структурные свойства графов партономии (и для их частного случая - ИерСемСет IACPaaS). Среди оцененных свойств было «*избыточные части»* в исходном определении, а также свойство *избыточные прокси-узлы»* определяемое по графу иерархических семантических сетей IACPaaS (язык которых предлагает proxy-узлы) как число узлов типа proxy, таких что: узел является единственной частью единственного «составного» узла. Контроль такого типа позволяет создавать базы знаний более компактной и читаемой структуры. Выбор штатного средства формализации знаний со встроенными средствами проверки [31] обеспечил невозможность *нарушения ограничений «кардинальности»* в БЗ (т.к. редактор управляется онтологией, в которой заданы эти ограничения как количественные спецификаторы отношений понятий); невозможность *нарушения* указания значимых областей значений в БЗ, не выходящих за пределы возможных, в результате чего *только допустимые значения* и *допустимые ссылки на другие определения* содержатся в БЗ.

Помимо Онтологии ПрОбл и баз Знаний разрабатывались компоненты решателей задач диагностики и лечения - программные ресурсы IACPaaS [32,33,82,113].

Для каждого онтологического решателя формировалась своя «онтология объяснения», по которой работают его (программные) компоненты.

Из онтологического решателя диагностики и конкретной Базы знаний о диагностике заболеваний конструируется *специализированный решатель*. Его предназначение: тщательно сопоставить все особенности организма пациента и наблюдаемые обстоятельства болезни – «диагностической» картине всех (или группы) заболеваний или нескольких явно указанных предполагаемых заболеваний, а также «дифференцирующих» детальных диагнозов (уточнений по этиологии, тяжести и пр.) [81,134].

В зависимости от потребности пользователя в дифференциальной диагностике для конструирования *специализированного решателя* используется либо База знаний одной группы заболеваний, либо интегрирующая несколько групп (или включающая все сформированные диагностические знания) [82].

В результате «соединения» каждого такого решателя с конкретным архивом истории болезни и с файлом для сохранения результата был получен облачный сервис по дифференциальной диагностике заболеваний. Он позволяет как проводить диагностику отдельных заболеваний, так и осуществлять поиск гипотез о заболеваниях, в том числе сочетанную патологию.

Другой специализированный решатель был построен путем интеграции двух решателей с базами знаний диагностики и лечения вирусных заболеваний. В результате «соединения» его с новой БД (для ИБ) и двумя видами объяснений, а также единым интерфейсом, настроенным на подключаемую БзНабл, получен облачный сервис [135]. Сервис готов для практического использования на облачной платформе IACPaaS (Рисунок 6.2).

Пользователь сервиса создает новую историю болезни или выбирает из БД (архива) конкретную ИБ, относящуюся к исследуемому пациенту (по ее шифру), добавляет, если имеется, свежую информацию, запускает процесс и получает результат ее анализа: одну или несколько гипотез о диагнозе с объяснением или конструктивное объяснение недостаточности обследования или объяснение подтверждения диагноза. Пользователь может добавить свою гипотезу (предварительный диагноз) и, запустив процесс, получит объяснение обнаруженного несоответствия признаков пациента этому заболеванию либо объяснение подтверждения диагноза. После подтверждения предоставляется возможность поддержки этапа лечения.

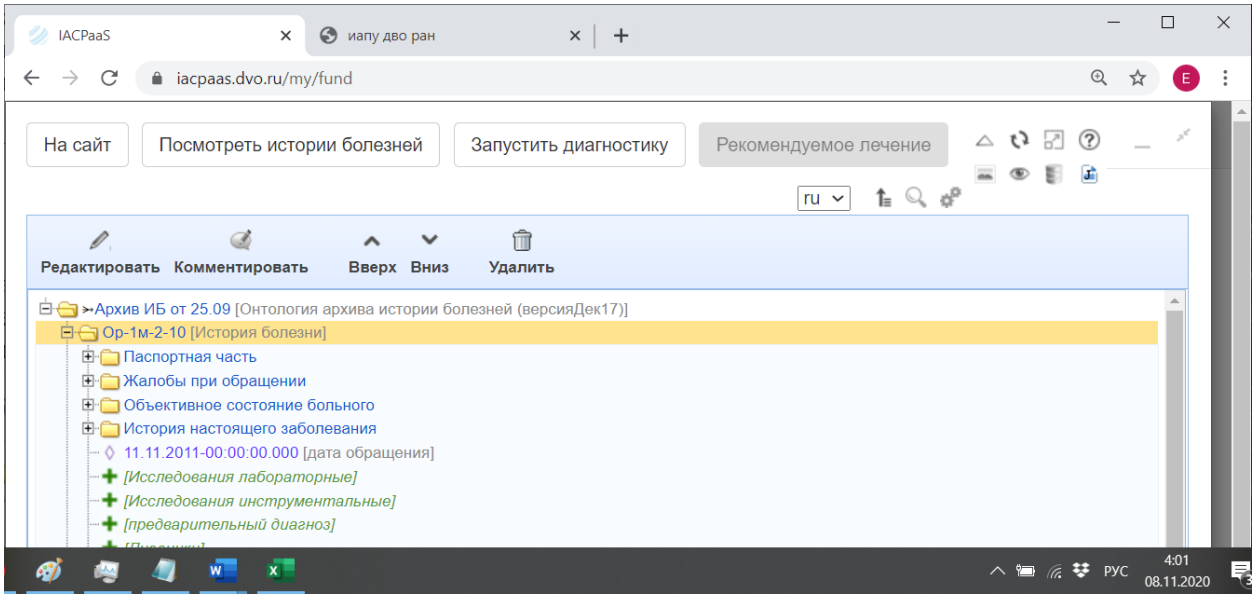


Рисунок 6.2 – Интерфейс работы с медицинским облачным сервисом в общем доступе пользователей

Стандартный пользовательский интерфейс [132] для работы с отчетом (объяснением) таков, что каждый раздел можно видеть в концентрированном содержании и с разной степенью детализации (см. рисунок 6.3).

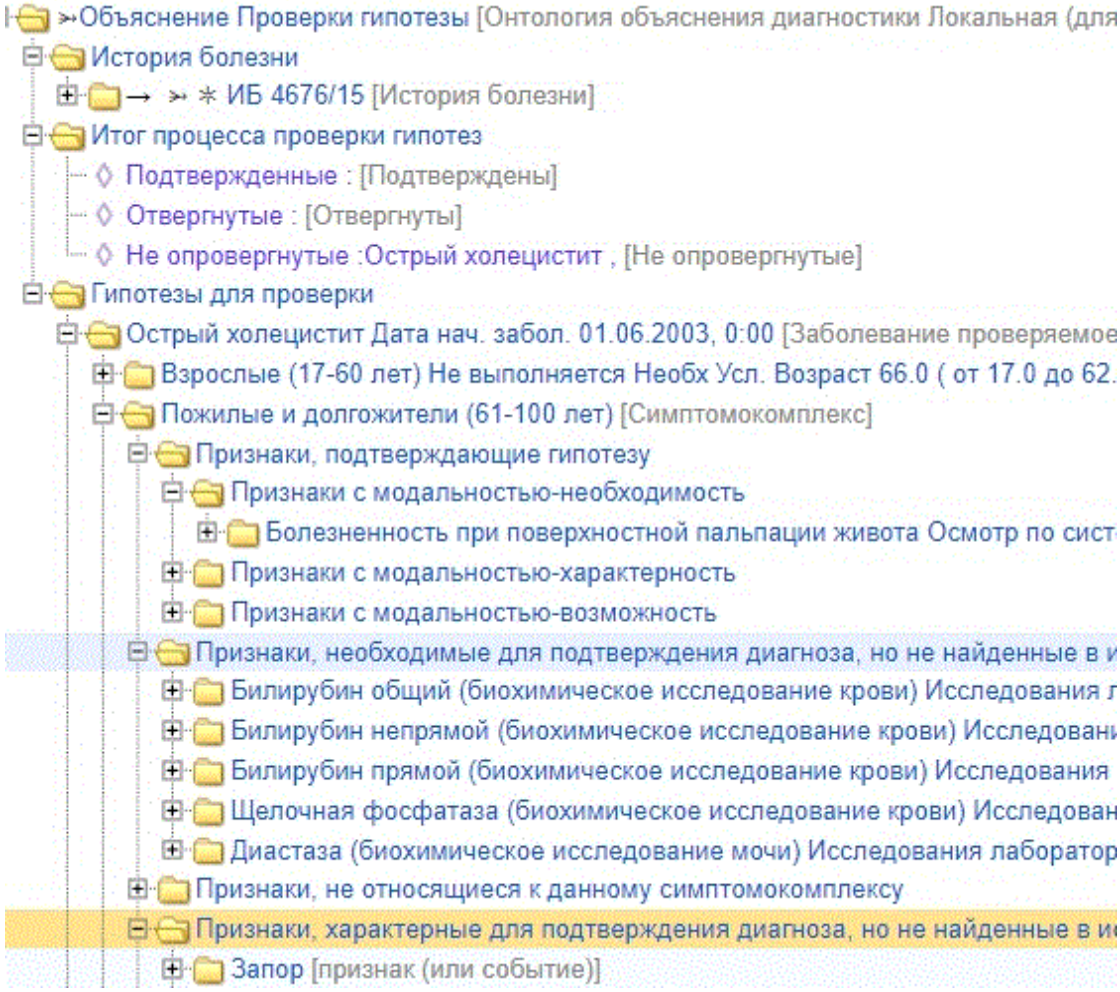


Рисунок 6.3 – Фрагмент отчета о диагностике для указанной истории болезни пациента

Для проведения дифференциальной диагностики пользователь в истории болезни пациента указывает предполагаемое уточнение нозологии и сразу получает результат проверки: подтверждение такого дифференциального диагноза (Рисунок 6.4) или критику его с конструктивным объяснением недостаточности обследования или обнаруженного несоответствия либо объяснение подтверждения диагноза [81].

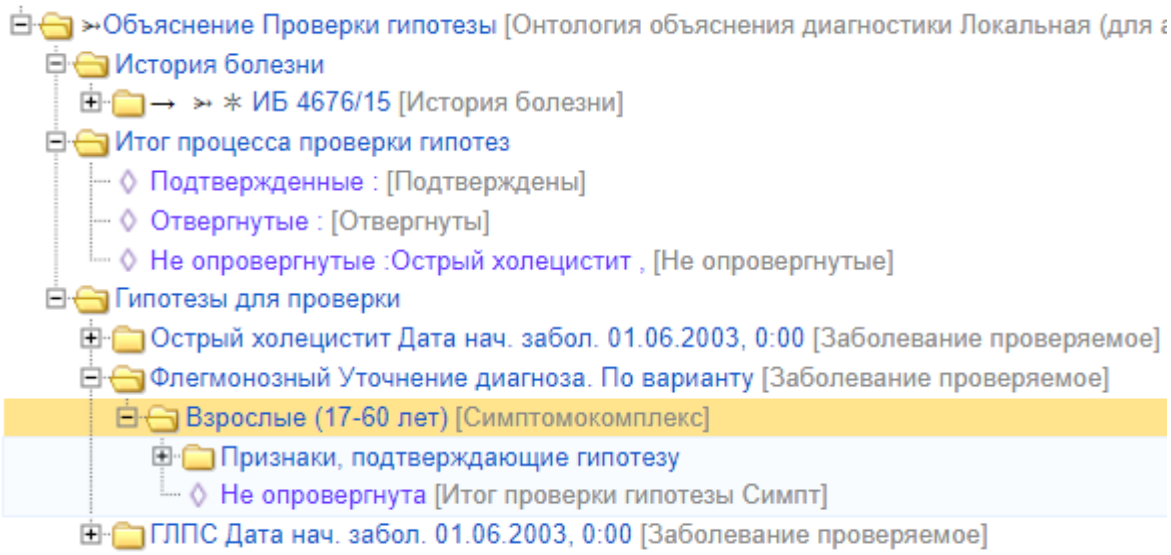


Рисунок 6.4 – Фрагмент отчета о дифференциальной диагностике для пациента.

6.7. Опыт развития пользовательских сервисов

В течение трех лет длится использование медицинских сервисов разными коллективами, в основном для апробации уникальных медицинских знаний по методам диагностики изучаемых заболеваний. Когда в начале 2020 г. началось распространение COVID-19: то для помощи в постановке предварительного дифференциального диагноза на основе клинических данных был создан сервис по диагностике и лечению вирусных заболеваний. Дополнение имеющейся БЗ описанием (нескольких) вариантов течения и способов диагностики нового заболевания заняло несколько дней. И на базе готового решателя IACPaaS был развернут востребованный облачный сервис поиска гипотез о возможном вирусном заболевании пациента и дифференциальной диагностики среди множества респираторных заболеваний [135]. Сервис предоставляет обоснование предлагаемых решений и выдаваемых рекомендаций (Рисунок 6.5). Оно содержит информацию не только о подтвержденных и отвергнутых системой заболеваниях, но также и детализированное обоснование принятых решений: какие признаки заболевания входят/не входят в клиническую картину рассматриваемых заболеваний, а также, в случае, если для подтверждения/опровержения необходима дополнительная информация, сервис подсказывает, значения каких признаков необходимо получить дополнительно.

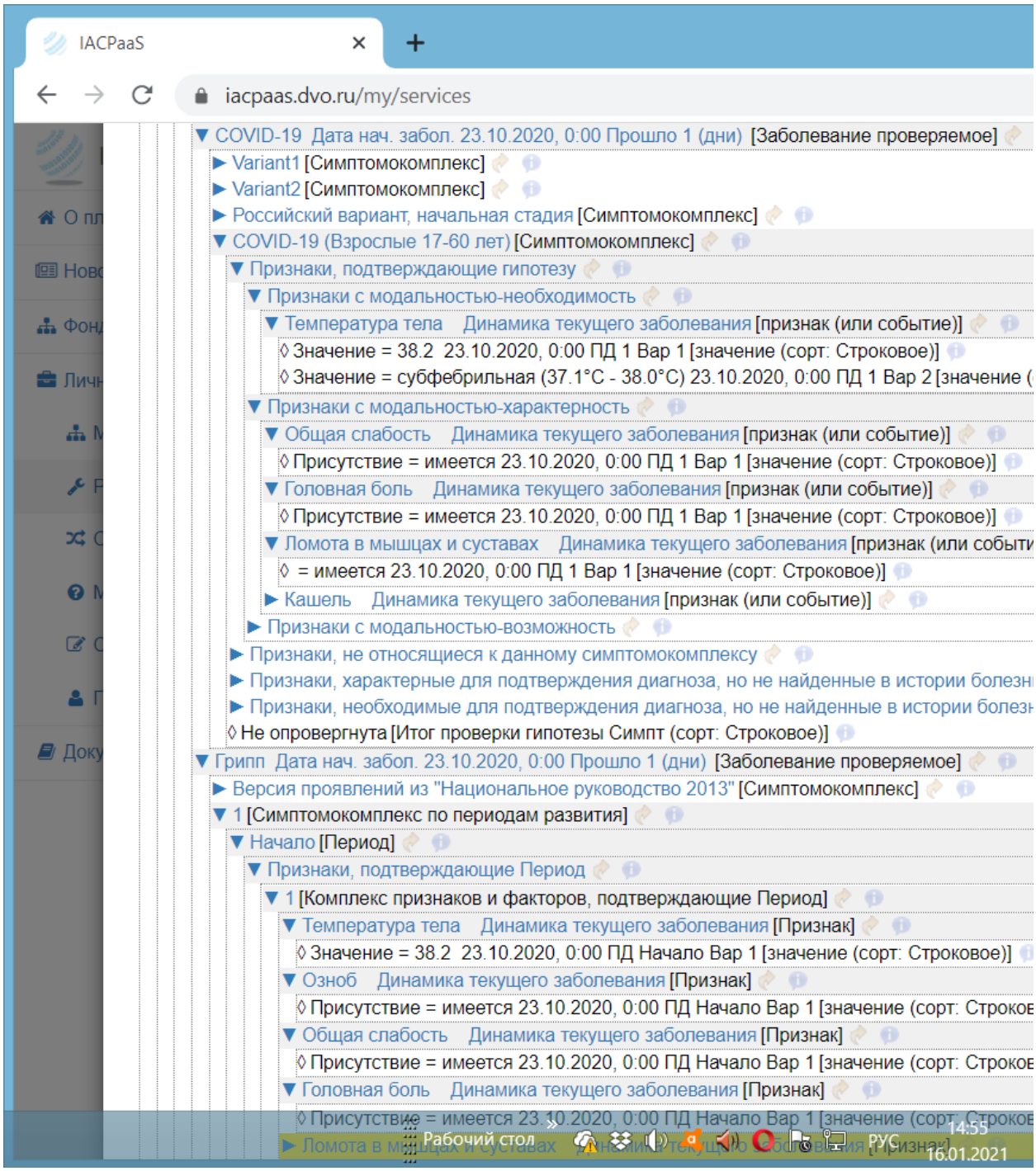


Рисунок 6.5 – Фрагмент работы с сервисом поддержки врача (на этапе диагностики)

Продолжающееся использование медицинских сервисов разными коллективами и с использованием разных версий баз знаний дало материал для сравнения правильности и точности диагностики в условиях разной комплектации сервисов [81]. В процессе использования возможностей генерации объяснений на основе знаний у специалистов, которые подключались позже, была возможность оценить объем знаний, составленный

предшественниками, и добавить свои уникальные знания (опубликованные в научных статьях) к базе, либо предложить базу случаев, не описанных в учебниках, но накопленных в рамках своей практики.

С учетом опыта коллективного *развития медицинских* баз знаний, конструирования *специализированных медицинских решателей*, их интеграции для получения ряда облачных сервисов был проведен сравнительный анализ трудозатрат на автоматизацию медицинской деятельности для двух вариантов реализации:

в случае облачной реализации СБЗ с разделением компетенций разработчиков

и в случае традиционной разработки СБЗ (для каждого заказчика-учреждения).

Рассмотрим N условных медучреждений, работающих по *P* медпрофилям. Пусть *N = 10* учреждений, *P=5* профилей (и *K* команд разработчиков, вовлеченных в автоматизацию, предположительно работающих по разным технологиям).

На графике (Рисунок 6.6) сверху – оценка затрат при традиционной разработке, снизу – при автоматизации отраслевой деятельности через «облачные» системы с декларативными БЗ и универсальными решателями.

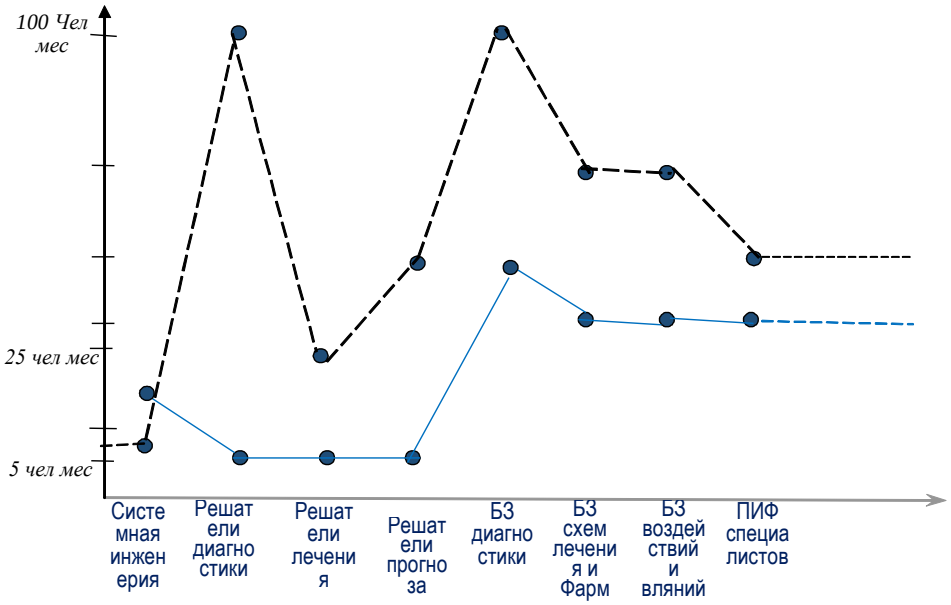


Рисунок 6.6 – Графическое представление сравнения трудозатрат для разных подходов к автоматизации интеллектуальной деятельности

Графики на рис.6.6 получены путем измерения трудозатрат при выполнении работ по построению компонентов интеллектуальных систем для нескольких медицинских

коллективов и сообществ. По горизонтали представлены некоторые принципиальные этапы работ, по вертикали – оценки на их выполнение в человеко-месяцах, например:

Один решатель диагностики потребовал около 10 чел/мес *vs* затрат на множество таких решателей (для K*P), составляющих 100 чел/мес (10/5 * 5 = 10) = 10*10).

Аналогично:

Один Решатель лечения = 5 чел/мес *vs* M решателей лечения (для K*P = 10/5 * 5/2 = 5) 5*5 = 25 чел/мес.

1 Решатель прогноза = 10 чел-мес, но M решателей (K*P = 10/5 * 5/2 = 5) 5*10 = 50 чел-мес.

База знаний диагностики по 1 профилю = 10 чел-мес, P баз = 50 чел-мес, но L баз = (N*P = 10/5*5) = 100 чел-мес.

На диаграмме (Рисунок 6.7) штриховкой показано оценочное число специалистов (по вертикали), участвующих в развитии (поддержании актуальности и усовершенствовании) СБЗ при традиционном подходе, нижний уровень показывает ситуацию при облачной реализации СБЗ с разделением компетенций. По горизонтали представлены некоторые принципиальные этапы и результаты работ, выполняемые разными категориями участников.

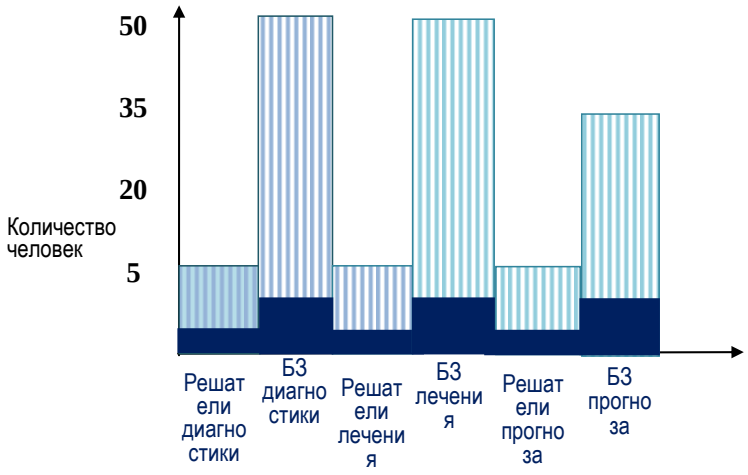


Рисунок 6.7 – Графическое представление сравнения усилий для разных подходов к автоматизации деятельности

Усовершенствование баз знаний диагностики = P чел = 5 чел

N*P специалистов = 10*5 = 50 чел

Усовершенствование баз знаний лечения = 5 чел-мес

N*P специалистов = 5*10 = 50 чел

Усовершенствование Решателя диагностики = 2 чел

K*P = 10/5 *5 = 10 чел.

Более подробные расчеты типичных трудозатрат (на системную инженерию, на программное обеспечение и на базы знаний, включая управление ими), а также на вспомогательные и организационные процессы показан в виде таблицы (таб 1).

Таблица 1. Сравнительная оценка затрат

Виды работ	традиционная разработка для N учреждений (работающих по P профилям) систем с БЗ	автоматизация отраслевой деятельности через «облачные» системы с декларативными БЗ
на программное Обеспечение:		
Создание Решателей диагностики	M штук решателей, 1<M<N или даже 1<M<N*P. Чаще M ближе к N или N*P. В идеале M = K (числу компаний-разработчиков ПС и МИС), но скорее M = K*P (если решатель не универсален). Поскольку не стандартизована терминология специалистов, мало шансов на M = K.	1
Создание Решателей планирования лечения	M штук, 1<M<K*P (можно ожидать стандартизованности терминологии специалистов разных учреждений в области лечения).	1
Создание Редактора знаний диагностики	M штук, P<M<N*P. Возможно, 1, если универсальный, тогда необходимы N*P инженеров по знаниям,.	1 (или 0, если генерируемый по онтологии)
на базы знаний:		
Создание Базы знаний диагностики	L штук баз, 1<L<N*P. L = S (числу компетентных экспертов знаний).	P
Создание Базы знаний планирования лечения	P (баз схем лечения от МинЗдрава) +1 (ФармСправ)	P (баз схем лечения от МинЗдрава) +1(ФармСправ)
Усовершенствование	N*P специалистов	P специалистов

БЗ диагностики		
Усовершенствование баз знаний планирования лечения	N*P специалистов – каждый вносит изменение в свою БЗ или проверяет автоматически сгенерированные фрагменты БЗ	P специалистов
Усовершенствование Решателя диагностики	M = K*P	1

Трудозатраты на системную инженерию (обычно выполняемую организацией-разработчиком) складываются из:

- проведения системного анализа профессиональной деятельности и формирования онтологии предметной области;
- концептуального проектирования системы автоматизации, документирования требований на разработку системы;
- интеграции всех подсистем и БЗ в единую систему (т.е. комплексирования, верификации, валидации всей системы автоматизации и передачи ее в среду функционирования);
- процессов комплексирования, верификации, технического обслуживания и других, традиционно относящихся к системной инженерии.

Трудозатраты на программное обеспечение состоят в разработке, а затем сопровождении следующих видов программных средств:

- решатель;
- редактор знаний;
- подсистема документирования;
- подсистема формирования обучающей выборки;
- подсистема модификации БЗ;
- подсистема оценивания БЗ.

Трудозатраты на формирование базы знаний складываются из затрат:

- на разработку первого варианта каждой базы знаний (с помощью редакторов знаний и подсистем оценивания);
- на сопровождение и управление базами знаний (с помощью редакторов знаний, АРМов, подсистемы формирования вариантов модификации БЗ и подсистемы оценивания БЗ).

Автоматизация одного учреждения состоит в разработке СБЗ для всех классов решаемых там задач интеллектуальной деятельности (например, в больнице, как основном звене медицинской отрасли) и обеспечивает эффект в получении преимуществ в повседневной деятельности и в совершенствовании знаний только специалистами одного учреждения (т.е. в зависимости от сложности задач, которые этим специалистам приходится решать).

Трудозатраты (для m профилей деятельности):

- на базы знаний и системы управления ими – их n * m штук,
- на интеграцию n штук решателей с подсистемой документооборота и подсистемами управления качеством n * m баз знаний.

«Облачная» реализация системы означает, что для всей отрасли предлагается использование общей базы знаний по всем классам задач и профилям (все ее модули размещаются на защищенных центральных серверах отрасли), там же устанавливаются универсальные решатели и система управления БЗ и накапливается единый архив правильно решенных задач по каждому профилю. Сборка сервисов (СБЗ) из базы знаний и соответствующих решателей осуществляется на рабочих местах в зависимости от потребностей специалистов. Имея доступ в Интернет, специалисты из основных звеньев отрасли используют «облачные» СБЗ, а коллектив экспертов каждого профиля (в идеале - специально выделенные высококвалифицированные специалисты) управляет качеством баз знаний посредством «облачно» доступных инструментов. Эффект от этой парадигмы состоит в получении всех преимуществ в повседневной деятельности и в достижении наивысшего качества БЗ, т.к. оно определяется полным спектром решаемых по всей отрасли задач – от типичных до самых сложных.

6.8. Выводы к главе 6

Разработанная технология создания жизнеспособных СБЗ воплощена в виде облачной технологии разработки систем, БЗ которых создаются в виде иерархических семантических сетей. Сформирована специализированная технологическая *среда развития* СБЗ с комплексом инструментальных средств разработки и сопровождения всех компонентов конфигурации СБЗ. С ее использованием произведено несколько медицинских СБЗ, БЗ которых продолжают развиваться силами медицинских экспертов.

ЗАКЛЮЧЕНИЕ

Основные результаты, полученные в диссертационной работе, таковы.

1. Разработана новая многоуровневая классификация задач интеллектуальной деятельности, основанная на принципе усложнения свойств ПрОбл, позволяющая определить место решаемой задачи ПрОбл среди множества известных задач и готовых решений.

2. Разработаны формальные постановки задач интеллектуальной деятельности, обеспечивающие основу для поиска готовых решателей и для конструирования повторно используемых компонентов систем.

3. Разработаны алгоритмы решения важных для практики задач интеллектуальной деятельности (запроса дополнительной информации, диагностики развивающегося процесса, выработка вариантов решений по воздействию на систему или объект и др.) и метод архитектурного проектирования онтолого-ориентированного решателя из повторно-используемых программных единиц.

4. Предложена модель жизнеспособной системы для поддержки решения задач интеллектуальной деятельности в рамках их постановок.

5. Предложен метод монотонного улучшения баз знаний и структурный подход к оцениванию всех информационных ресурсов для их проверки на ранних этапах разработки интеллектуальных программных систем.

6. Разработана технология формирования среды построения и развития СБЗ и коллективного создания программных систем для поддержки деятельности специалистов на основе формализованных баз знаний и их развития, преимущественно за счет развития БЗ и обеспечения их качества БЗ.

Перспективами дальнейшей разработки темы являются создание методов интеграции компонентов СБЗ с подсистемами, реализующими иные методы искусственного интеллекта, и подготовка инфраструктуры для декларативного формирования таких гибридных систем.

СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ Р ИСО/МЭК 12207-2010. «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств» (дата введения в действие: 2012-03-01).
2. ГОСТ ИСО/МЭК 9126-2001. «Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению» (дата введения в действие: 02.11.2001).
3. Большой Российский энциклопедический словарь. М.: БРЭ. 2003
4. Антипов С.Г., Вагин В.Н., Фомина М.В. Методы диагностики динамических объектов на основе анализа временных рядов // ИТНОУ: информационные технологии в науке, образовании и управлении, 2017, №2 (2).
5. Балан В.П., Душкин А.В., Новосельцев В.И., Сумин В.И. Введение в системное проектирование интеллектуальных баз знаний / Под ред. В.И. Новосельцева – М.: Горячая линия – Телеком, 2016. – 107 с.
6. Башлыков А.А. Методология построения систем управления базами знаний для интеллектуальных систем поддержки принятия решений // Программные продукты и системы. 2013. № 2. - С. 131-137.
7. Бекмуратов Т. Ф., Мухамедиева Д. Т., Бобомурадов О. Ж. Нечеткая модель прогнозирования урожайности // Проблемы информатики, 2010, вып.3. С. 11-23
8. Бледжянц Г.А., Саркисян М.А., Исакова Ю.А. и др. Ключевые технологии формирования искусственного интеллекта в медицине // Ремедиум. 2015. № 12. С.10–15.
9. Боргест Н.М. Ключевые термины онтологии проектирования: обзор, анализ, обобщения // Онтология проектирования, 2013. № 3(9). С. 9-31.
10. Братко И. Алгоритмы искусственного интеллекта на языке PROLOG, 3-е издание. Пер. с англ. — М.: Издательский дом "Вильямс", 2001. 640 с.
11. Брунер, Дж. Психология познания. М.: Прогресс, 11067. 413 с.
12. Верткин А.Л., Тополянский А.В. Алгоритмы диагностики: боль в грудной клетке // РМЖ, 2016 №14. с. 913-916.
13. Вирт Н. Алгоритмы и структуры данных // ДМК-Пресс, 2016

14. Воронцов К. В. Математические методы обучения по прецедентам. 2012. http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_(курс_лекций,_К.В.Воронцов).
15. Воронцов К. В. Вычислительные методы обучения по прецедентам. Введение. 2007 г. // http://www.ccas.ru/voron/download/Introduction.pdf
16. Гаврилова Т.А., Горовой В.А., Болотникова Е.С., Горелов В. В. Субъективные метрики оценки онтологий // Материалы Всероссийской конференции с международным участием. Знания — Онтологии — Теории (ЗОНТ-09), 2009. С. 178-187.
17. Гаврилова Т. А., Кудрявцев Д. В., Муромцев Д. И. Инженерия знаний. Модели и методы: Учебник. СПб.: Издательство «Лань», 2016. 324 с.
18. Гаврилова Т.А., Страхович Э.В. Визуально-аналитическое мышление и интеллект-карты в онтологическом инжиниринге // Онтология проектирования, 2020, том 10 № 1, С. 87-99.
19. Гаврилова Т. А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. СПб: Питер, 2000. 384 с.
20. Голенков В.В., Гулякина Н.А. Проект открытой семантической технологии компонентного проектирования интеллектуальных систем. Часть 2: унифицированные модели проектирования // Онтология проектирования. 2014. №4(14). С 34-53.
21. Горшков С. В. Онтологическое моделирование предприятий: методы и технологии: монография; отв. ред. С. В. Горшков; Екатеринбург: Изд-во Урал. ун-та, 2019. 236 с.
22. Грибова В.В., Клещев А.С., Крылов Д.А., Москаленко Ф.М., Тимченко В.А., Шалфеева Е.А. Базовая технология разработки интеллектуальных сервисов на облачной платформе IACPaaS. Часть 1. Разработка базы знаний и решателя задач // Программная инженерия. 2015. № 12. С. 3-11.
23. Грибова В.В., Клещев А.С., Крылов Д.А., Москаленко Ф.М., Тимченко В.А., Шалфеева Е.А. Базовая технология разработки интеллектуальных сервисов на облачной платформе IACPaaS. Часть 2. Разработка агентов и шаблонов сообщений // Программная инженерия. 2016. №1. С. 14-20.
24. Грибова В.В., Клещев А.С., Москаленко Ф.М., Тимченко В.А., Федорищев Л.А., Шалфеева Е.А. Базовая технология разработки интеллектуальных сервисов на об-

лачной платформе IACPaaS. Часть 3. Разработка интерфейса и пример создания прикладных сервисов // Программная инженерия. 2016. Т. 7. № 3. С. 99-107.

25. Грибова В.В., Клещев А.С., Москаленко Ф.М., Тимченко В.А., Федорищев Л.А., Шалфеева Е.А. Методы и средства разработки жизнеспособных интеллектуальных сервисов // Вестник ДВО РАН Владивосток: Изд. "Дальнаука", 2016. №4 с.133-141.
26. Грибова В.В., Клещев А.С., Москаленко Ф.М., Тимченко В.А., Федорищев Л.А., Шалфеева Е.А. Облачная платформа IACPaaS для разработки оболочек интеллектуальных сервисов: состояние и перспективы развития // Программные продукты и системы. 2018. Т. 31. № 3. С.521-526.
27. Грибова В.В., Клещев А.С., Москаленко Ф.М., Тимченко В.А., Федорищев Л.А., Шалфеева Е.А. Платформа для разработки облачных интеллектуальных сервисов // Сборник «Пятнадцатая национальная конференция по искусственному интеллекту с международным участием». Труды конференции: в 3 томах. 2016. С. 24-32.
28. Грибова В.В., Клещев А.С., Москаленко Ф.М., Тимченко В.А., Федорищев Л.А., Шалфеева Е.А. Управляемая графовыми грамматиками разработка оболочек интеллектуальных сервисов на облачной платформе IACPAAS // Программная инженерия. 2017. Т. 8. № 10. С. 435-447.
29. Грибова В.В., Клещев А.С., Шалфеева Е.А. Метод решения задачи запроса дополнительной информации // Онтология проектирования. 2017. Т.7. N3(25). С. 310-322.
30. Грибова В.В., Клещев А.С., Шалфеева Е.А. Управление интеллектуальными системами // Известия РАН. Теория и системы управления. 2010. № 6. С. 122-137.
31. Грибова В.В., Окунь Д.Б., Петряева М.В., Шалфеева Е.А. Инфраструктура IACPAAS для формирования интерпретируемых баз диагностических знаний по заболеваниям произвольной направленности // Национальная конференция по искусственному интеллекту с международным участием. КИИ-2019. Сборник научных трудов: в 2-х томах. С. 81-89.
32. Грибова В.В., Петряева М.В., Окунь Д.Б., Шалфеева Е.А. Онтология медицинской диагностики для интеллектуальных систем поддержки принятия решений // Онтология проектирования. 2018. Т. 8. - №1(27). С. 58-73.

33. Грибова В.В., Петряева М.В., Шалфеева, Е.А. Облачный сервис поддержки принятия диагностических решений в гастроэнтерологии // Врач и информационные технологии, Москва, ООО Издательский дом «Менеджер здравоохранения», 2019. № 3. С. 65–71.
34. Грибова В.В., Шалфеева Е.А. Комплекс средств поддержки процессов разработки и сопровождения решателей для систем с онтологическими базами знаний / // Информационные и математические технологии в науке и управлении, 2020. № 4(20). С. 34-43.
35. Грибова В.В., Шалфеева Е.А. Модель управления качеством баз знаний с оцениванием // Сборник «Знания - Онтологии - Теории (ЗОНТ-2019)». Материалы VII Международный конференции. 2019. С. 138-147.
36. Грибова В.В., Шалфеева Е.А. Онтология диагностики процессов // Онтология проектирования. 2019. Том 9. № 4(34). С.449-461.
37. Грибова В.В., Шалфеева Е.А. Системы на основе онтологических баз знаний как основа для создания современных систем искусственного интеллекта // Восемнадцатая Национальная конференция по искусственному интеллекту с международным участием КИИ-2020. Труды конференции. Под ред. В.В. Борисова, О.П. Кузнецова. Москва, 2020. С. 12-19.
38. Грищенко М.А., Дородных Н.О., Коршунов С.А., Юрин А.Ю. Разработка интеллектуальных диагностических систем на основе онтологий // Онтология проектирования. 2018. Т.8. №2(28). С.265-284.
39. Гусев А.В., Добриднюк С.Л. Искусственный интеллект в медицине и здравоохранении // Информационное общество, 2017. №4-5. С. 78-93.
40. Джарратано Дж., Райли Г. Экспертные системы. Принципы разработки и программирование, 4-е издание. М.: Издательский дом "Вильямс", 2007. 1152 с.
41. Джексон П. Введение в экспертные системы: Уч. пос. М.: Издательский дом "Вильямс", 2001. – 624 с.
42. Дзенгелевский А.Е. Способы визуализации качества базы знаний по автоматизации предприятий на примере ситуационной инструментальной экспертной системы // Научная визуализация. 2016. № 8(5). С. 59-73.
43. Додонов А.Г., Ландэ Д.В. Живучесть информационных систем. К.: Наукова думка. 2011. - 256 с.

44. Долинина О.Н. Метод генерации тестов для отладки баз знаний экспертных систем // Программная инженерия. 2011. № 5. С.40-48.
45. Жиляев А.А. Онтологии как инструмент создания открытых мультиагентных систем управления ресурсами // Онтология проектирования. 2019, том 9, №2(32). с. 261-281.
46. Жоголев Е. А. Технология программирования. - М. научный мир, 2004, 216 с.
47. Загорулько Ю.А. Технология разработки порталов научных знаний // Программные продукты и системы. 2009. № 4. С. 25-29.
48. Загорулько Ю.А., Загорулько Г.Б., Боровикова О.И. Технология создания тематических интеллектуальных научных Интернет-ресурсов, базирующаяся на онтологии // Программная инженерия. 2016, Т. 7, № 2. - С. 51-60.
49. Зильбер А.П., Шифман Е.М., Павлов А.Г. Пре-эклампсия и эклампсия: клинико-физиологические основы и алгоритмы диагностики // Петрозавод. ун-т. Петрозаводск, 19106. 52 с.
50. Ивашенко В. П. Семантическая технология проектирования баз знаний // Доклады БГУИР. - 2009. № 7 (45). С. 44 - 51.
51. Карпов О.Э., Клименко Г.С., Лебедев Г.С. Применение интеллектуальных систем в здравоохранении // Современные наукоемкие технологии. 2016. № 7 (часть 1) – С. 38-43.
52. Кизим А.В. Постановка и решение задач автоматизации работ // Доклады ТУСУРа, 2009. № 2 (20), с 131-135.
53. Киликовский В. В., Олимпиева С. П. Компьютерные медицинские консультативные системы, основанные на представлении знаний эксперта в виде семантической сети // Медицинский научный и учебно-методический журнал. 2001. № 2. с. 17–27.
54. Клещев А.С., Артемьева И.Л. Математические модели онтологий предметных областей. Часть 1. Существующие подходы к определению понятия «онтология» // Научно-техническая информация. 2001. № 2. С. 20-27.
55. Клещев А.С., Артемьева И.Л. Необогащенные системы логических соотношений // Научно–техническая информация. Серия 2. 2000, Часть 1. № 7, С. 18–28. Часть 2 - № 8. С. 8–18.

56. Клещев А. С., Смагин С. В. Задачи индуктивного формирования знаний для онтологии медицинской диагностики // Научно-техническая информация. Серия 2. Информационные процессы и системы. М.: ВИНИТИ РАН. 2012. №1. С. 9-21.

57. Клещев А.С., Черняховская М.Ю., Москаленко Ф.М. Модель онтологии предметной области «Медицинская диагностика». Часть 1. Неформальное описание и определение базовых терминов // Журнал НТИ - Серия 2. № 12, 2005.

58. Клещев А.С., Черняховская М.Ю., Шалфеева, Е.А. Особенности автоматизации интеллектуальной деятельности // Научно-техническая информация. Сер.2, 2015, № 1, с. 10–20.

59. Клещев А.С., Черняховская М.Ю., Шалфеева, Е.А. Парадигма автоматизации интеллектуальной профессиональной деятельности. Часть 1. Особенности интеллектуальной профессиональной деятельности // Онтология проектирования. Самара: “Новая техника”, 2013. № 3(9). 53-69.

60. Клещев А.С., Черняховская М.Ю., Шалфеева, Е.А. Парадигма автоматизации интеллектуальной профессиональной деятельности. Часть 2. Парадигма автоматизации отрасли // Онтология проектирования. Самара: “Новая техника”, 2013. №4(10). С. 28-40.

61. Клещев А.С., Шалфеева Е.А. Онтология задач интеллектуальной деятельности // Онтология проектирования. Самара: “Новая техника”, 2015. № 2 (16), с. 179-205.

62. Клещев А.С., Шалфеева Е.А. Определение структурных свойств онтологий // Изв. РАН. Теория и системы управления, 2008. №2. С. 69–78.

63. Клещев А.С.., Шалфеева Е.А Постановки практически полезных задач интеллектуальной деятельности // Дальневост. матем. журн., 2016. № 16(1). С. 44–61.

64. Клещев А.С., Шалфеева Е.А Принципы организации каталога свойств онтологий // Научно-техническая информация, серия 2, 2007, No 6. С. 7-15.

65. Клещев А.С., Шалфеева Е.А. Решение задачи запроса дополнительной информации на основе декларативной темпоральной базы знаний для дифференциальной диагностики // Сборник «Знания – Онтологии – Теории (ЗОНТ-2017). Материалы Всероссийской конференции с международным участием». 2017. С. 6-14.

66. Кобринский Б.А. Ретроспективный анализ медицинских экспертных систем // Новости искусственного интеллекта. №2. 2005. С.6-17.

67. Кобринский Б. А. Системы поддержки принятия решений в здравоохранении и обучении // Врач и информационные технологии: Научно-практический журнал. 2010. № 2. С. 39-45.
68. Ковалев С.М. Гибридные нечетко-темпоральные модели временных рядов в задачах анализа и идентификации слабо формализованных процессов // Интегрированные модели и мягкие вычисления в искусственном интеллекте. Сборник трудов IV международной научно-практической конференции. В 2-х томах. Т.1 М.: Физматлит, 2007. – 354 с.
69. Корноушенко Е. К., Простой алгоритм номинальной классификации по качественным признакам, Пробл. управл., 2017, вып 1, 2–9.
70. Кряжич О.А. Обеспечение жизнеспособности информации во времени при ее обработке в СППР // Математичні машини ісистеми. – 2015. №2. С. 170-176.
71. Кудрявцев Д. В. Практические методы отображения и объединения онтологии // Труды 11-й Национальной конференции по искусственному интеллекту с международным участием (КИИ-08), г. Дубна, Россия. Том 3. М.: URSS, 2008. С. 164-173.
72. Литвиненко В.И. Модели обработки неполной и противоречивой информации в диагностических системах: дис. На соиск. ст. канд. техн. наук. Херсон, 19106. 207 с.
73. Лянденбурский В. В., Тарасов А. И., Судьев В. В. Алгоритм поиска неисправностей дизелей // Молодой ученый. 2015. №4. С. 214-217.
74. Митрофанова О.А., Константинова Н.С. Онтологии как системы хранения знаний / Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению "Информационно-телекоммуникационные системы", 2008. - 54 с.
75. Москаленко Ф.М. Задача медицинской диагностики и алгоритм её решения, допускающий распараллеливание // Информатика и системы управления. № 2(10), 2005. С. 52-63.
76. Москаленко Ф.М., Тимченко В.А., Шалфеева, Е.А. «Административная система (версия 1.0) интернет-комплекса IACPaaS» Свидетельство о регистрации программы для ЭВМ №2012618861. (Зарегистрировано в Реестре программ для ЭВМ 28 сентября 2012 г.).

77. Москаленко Ф.М., Тимченко В.А., Шалфеева, Е.А. "Инструментальное средство мониторинга состояния программных агентов интеллектуальных многоагентных сервисов". Свидетельство Федеральной службы по интеллектуальной собственности о государственной регистрации программы для ЭВМ № 2013619809. (Зарегистрировано в Реестре программ для ЭВМ 16 октября 2013 г.).
78. Назаренко Г.И., Осипов Г.С. Медицинские информационные системы и искусственный интеллект. Москва. Издательский центр «Медицина XXI». 2003. 240 с.
79. Николенко С. И., Тулупьев А. Л. Самообучающиеся системы. М.: изд-во МЦНМО. – 2009. – 288 с.
80. Павлов С. Н. Системы искусственного интеллекта: учеб. пособие. В 2-х частях. Ч. 1. Томск: Эль Контент, 2011. — 176 с.
81. Петряева М.В., Лифшиц А.Я., Шалфеева Е.А. Медицинские ресурсы IACPaaS для дифференциальной диагностики заболеваний желчного пузыря // Информатика и системы управления. 2018. № 3(57). С. 81-92.
82. Петряева М.В., Шалфеева Е.А. Сервис подтверждения предварительного диагноза на основе формализованных знаний // Сборник «Материалы XII международной научной конференции "Системный анализ в медицине"» (САМ 2018). 2018. С. 50-53.
83. Подлипский О. К. Построение баз знаний группой экспертов // Компьютерные исследования и моделирование, 2010. Т. 2 № 1 С. 3–11.
84. Попов Э.В., Фирдман Г.Р. Алгоритмические основы интеллектуальных роботов и искусственного интеллекта. Главная редакция физико-математической литературы изд-ва «Наука», М., 11066. - 456 с.
85. Попов Э.В., Фоминых И.Б., Е.Б. Кисель, Шапот М.Д. Статические и динамические экспертные системы, учебное пособие. М: «Финансы и статистика», 1996. 318 с.
86. Рыбина Г. В. Интеллектуальная технология построения обучающих интегрированных экспертных систем: новые возможности // Открытое образование. 2017. №4. URL: https://cyberleninka.ru/article/n/intellektualnaya-tehnologiya-postroeniya-obuchayuschih-integrirovannyh-ekspertnyh-sistem-novye-vozmozhnosti (дата обращения: 06.12.2020).
87. Рыбина Г. В. Интеллектуальные системы: от А до Я. Серия монографий в трех книгах. Книга 3. Проблемно-специализированные интеллектуальные системы. Инстру-

ментальные средства построения интеллектуальных систем. М.: ООО "Научтехлитиздат", 2015.

88. Рыбина Г.В. Обучающие интегрированные экспертные системы: некоторые итоги и перспективы // Искусственный интеллект и принятие решений. 2008. № 1. С. 22–46.
89. Рыбина Г.В. Рабочее место для построения интегрированных экспертных систем: комплекс АТ-ТЕХНОЛОГИЯ // Новости искусственного интеллекта. 2005. №3. С. 69-87.
90. Рыбина Г. В., Блохин Ю. М., Иващенко М.Г. Интеллектуальная технология построения интегрированных экспертных систем // Искусственный интеллект и принятие решений. 2011. № 3. – С. 48-57.
91. Сенько О.В. Математические основы теории прогнозирования. Лекция I // http://www.machinelearning.ru/wiki/index.php?title=МОТР
92. Сигов А.С., Нечаев В.В., Кошкарёв М.И. Архитектура предметно-ориентированной базы знаний интеллектуальной системы // Образовательные ресурсы и технологии. 2015. № 1 (9). С. 156-163.
93. Соловьев С.Ю., Соловьева Г.М. Методы отладки баз знаний в системе ФИАКР. // Сб. Автоматизация и роботизация призводства с применением микропроцессорных средств. Кишинев, 1986. С.36-37.
94. Сутягин И. В. Методы формализации экспертных знаний для наполнения базы знаний // Молодой ученый. 2012. №1. Т.1. С. 151-153.
95. Тельнов Ю.Ф. Интеллектуальные информационные системы. М.: МЭСИ, 2004. - 246 с.
96. Уотермен Д. Руководство по экспертным системам: Пер. с англ. М: Мир, 1989 . - 388 с.
97. Финн В.К. Индуктивные методы Д.С. Милля в системах искусственного интеллекта // Искусственный интеллект и принятие решений. 2010 №3, с. 3-21.
98. Финн В.К. Об интеллектуальном анализе данных // Новости Искусственного интеллекта, № 3, 2004. С. 3-18.
99. Хейес-Рот Ф., Уотермен Д., Ленат Д. Построение экспертных систем. М: Мир, 1987. - 441 с.

100. Шалфеева, Е.А. Измерение структурных свойств для оценивания, сравнения и выбора онтологий // Информатика и системы управления. - Благовещенск: АмГУ, 2008. - №4(18), С. 106-115.

101. Шалфеева, Е.А. Использование семантических сетей для поддержки проектирования повторно используемых программных единиц / Шалфеева, Е.А., Родионов А.Е. // Сборник материалов XXXVIII Дальневосточной математической школы-семинара имени академика Е.В. Золотова. тезисы докладов. Владивосток: ИАПУ ДВО РАН, 2014. С. 474-483.

102. Шалфеева, Е.А. Каталог структурных свойств онтологий // Искусственный интеллект, 2006, т.4, с. 66-74.

103. Шалфеева Е.А. Каталог структурных свойств онтологий. Свойства синтаксической структуры // Владивосток: ИАПУ ДВО РАН. Препринт. 2007. 28 с.

104. Шалфеева Е.А. Каталог структурных свойств онтологий. Свойства структуры стандартных связей // Владивосток: ИАПУ ДВО РАН. Препринт. 2007. 38 с.

105. Шалфеева, Е.А. Классификация для задач, выявляемых при системном анализе интеллектуальной деятельности // Материалы IV Междунар. научн.-техн. конф. "Открытые семантические технологии проектирования интеллектуальных систем" (OSTIS-2015) – Минск: БГУИР, 2015. С. 187 – 192.

106. Шалфеева, Е.А. Классификация структурных свойств онтологий // Искусственный интеллект, 2005, т.3, с. 67-77.

107. Шалфеева, Е.А. Метод оценивания структуры онтологий для реализации редакторов знаний и данных // Вестник компьютерных и информационных технологий, Москва, "Издательство Машиностроение", 2010, № 2 (68). С. 23-32.

108. Шалфеева, Е.А. Метод построения проектных представлений интеллектуального решателя задач по моделям начальных стадий жизненного цикла // Искусственный интеллект («Штучний інтелект»). Донецк: Изд-во ИПИИ "Наука і освіта", 2013. № 4. С. 51-61.

109. Шалфеева, Е.А. Методы оценивания структуры связей онтологий различных уровней общности // Искусственный интеллект, 2007, т.4, с. 11-19.

110. Шалфеева, Е.А. Мониторинг информационных ресурсов жизнеспособной интеллектуальной программной системы // Программная инженерия, 2012. №1, С. 10-15.

111. Шалфеева, Е.А. Онтология моделирования программных компонентов, работающих с семантическими сетями // Материалы Всероссийской конференции с международным участием "Знания – Онтологии – Теории" (ЗОНТ-2013). Новосибирск: ЗАО «РИЦ Прайс-Курьер», 2013. Том 2. – С. 213-221.

112. Шалфеева Е.А. Преобразование моделей анализа требований в проектные представления интеллектуального решателя // Информатика и системы управления. 2013. №4(38). С. 100-110.

113. Шалфеева Е.А., Новеселов А.С. Конструирование облачного медицинского сервиса по технологии IACPaaS // Стратегии устойчивого развития мировой науки — Москва: «Евразийское Научное Объединение», 2016, № 5 (17). С. 25-29.

114. Юрин А.Ю., Дородных Н.О. Web-сервис для автоматизированного формирования продукционных баз знаний на основе концептуальных моделей // Программные продукты и системы. 2014. №4. – С.103-107.

115. Ясенев В.Н. Информационная технология экспертных систем / Глава (3.5) в «Информационные системы и технологии в экономике: учебное пособие». Издательство: Юнити-Дана, 2012. 560 с.

116. Agirre E., Ansa O., Hovy E., Martinez D. Enriching very large ontologies using the WWW. In: The Proc. of the Workshop on Ontology Learning, ECAI 2000, pp. 37–42.

117. Bennet J. S. A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System //Journal of Automated Reasoning. 1985. No. 1. P. 49-74.

118. Biran O., Cotton C. Explanation and justification in machine learning: a survey // Workshopon Explainable AI (XAI). 2017. P. 8-13.

119. Breivold H.P., Crnkovic I., Eriksson P.J. Analyzing software evolvability // Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International. – IEEE. 2008. P. 327-330.

120. Clancey W.J. Heuristic Classification // Artificial Intelligence, 1985, №27, с. 289-350.

121. Chandrasekaran B., Josephson J. R., Richard B.V. Ontology of Tasks and Methods // The Eleventh Workshop on Knowledge Acquisition, Modeling and Management, 18-23 April, Banff, Alberta, Canada. 1998.

122. Cogan B., Shalfeeva E.A. A generalised structural model of structured programs for software metrics definition // Software Quality J., 2002. № 10. P. 147–165..

123. Dehaghani S.M.H., Hajrahimi N. Which factors affect software projects maintenance cost more? // ActaInformaticaMedica. – 2013. № 21(1). P. 63-66.

124. Fikes R., McGuinness D. Creating, Maintaining, and Integrating Understandable Knowledge Bases; DARPA RKF Program Kickoff Meeting; New Orleans, Louisiana; June 7, 2000.

125. Gangemi A., Pisanelli D. M., Steve, G. Ontology Integration: Experiences with Medical Terminologies, in N. Guarino, ed., 'Formal Ontology in Information Systems', IOS Press, 1998. pp. 163–178.

126. Gartner Identifies the Top Strategic Technology Trends for 2021. STAMFORD, Conn., October 19, 2020. https://www.gartner.com/en/newsroom/press-releases/2020-10-19-gartner-identifies-the-top-strategic-technology-trends-for-2021.

127. Gerbaux F., Gruber T. Library of Ontologies, 2000, http://www.loacnr.it/medicine/

128. Golenkov V.V., Gulyakina N.A., Grakova N.V., Nikulenka V.Y., EremeevA.P., Tarasov V.B. From training intelligent systemstotrainingtheirdevelopmentmeans. Proceedings of the International Conference OSTIS-2018. Minsk. 2018. P. 81–98.

129. Gribova V, Kleschev A, Krylov D, Moskalenko P, Timchenko V., Shalfeeva, E. A. Cloud Computing Platform for Lifecycle Support of Intelligent Multi-agent Internet-services // International Conference on Power Electronics and Energy Engineering (PEEE 2015) Hong Kong, People's Republic of China, 2015. P. 231-235.

130. Gribova V.V., Kleschev A.S., Moskalenko Ph.M., Timchenko, V.A., Fedorischev L.A., Shalfeeva E.A. The IACPaaS Cloud Platform: Features and Perspectives. In Proc. of 2017 Second Russia and Pacific Conference on Computer Technology and Applications (RPC) (25-29 Sept. 2017). Vladivostok. 80-84.

131. Gribova V., Kleschev A., Moskalenko Ph., Timchenko V., Fedorischev L., Shalfeeva E. The methods and the IACPAAS platform tools for semantic representation of knowledge and development of declarative components for intelligent systems // Open Semantic Technologies for Intelligent Systems (Research Papers Collection). 2019. Issue 3. P. 21-24. (OSTIS-2019).

132. Gribova V., Kleschev A., Moskalenko Ph., Timchenko V., Fedorischev L., Shalfeeva E., Zamburg E. Technology for the development of intelligent service shells based on extended

generative graph grammars // Proceedings of the 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC 2018).

133. Gribova, V.V., Kleshchev, A.S., Shalfeeva, E.A. Control of intelligent systems // Journal of Computer and Systems Sciences International, 2010. № 49(6), p. 952-966.

134. Gribova V., Okun D., Petryaeva M., Shalfeeva E., Tarasov A. Ontology for differential diagnosis of acute and chronic diseases // Computer and Information Science, 2018. V 934, p. 152-163.

135. Gribova V.V., Okun D.B., Shcheglov B.O., Shchelkanov M.Yu., Shalfeeva E.A. Cloud service for the differential clinical diagnostics of acute respiratory viral diseases (including those associated with highly conta-gious coronaviruses) with an application of methods of artificial intelligence // Yakut Medical Journal, 2020. N 2. P. 44–47.

136. Gribova V., Shalfeeva E. Ontological Approach to Viable Decision Support Services Development // Advances in social science, education and humanities research, 2020. Том 483. p. 274-277.

137. Gribova, V., Shalfeeva, E. Ontology of anomalous processes diagnosis // International Journal of Intelligent Systems. 2021. Volume 36, Issue 1. p. 291-312.

138. Gribova V.V., Shalfeeva, E.A. The Model of Knowledge Base Quality Control with Assessment // International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), Novosibirsk, Russia, 2019, IEEE Xplory. P. 0872-0877.

139. Gruber T.R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing // International Journal of Human and Computer Studies. 1993. V.43(5/6). p. 907-928.

140. Gruber T.R., Olsen G. R. Ontology for Engineering Mathematics, 1994. http://www-ksl.stanford.edu/knowledge-sharing/papers/fn19.

141. Guarino N. Formal Ontology and Information Systems. In Proceeding of International Conference on Formal Ontology in Information Systems (FOIS'98), Trento, Italy. Amsterdam, IOS Press, pp. 3-15.

142. Guarino N., Giaretta P. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In N. Mars (ed.) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, 1995. IOS Press, Amsterdam. p. 25–32.

143. Guarino N., Schneider L.: Ontology-Driven Conceptual Modelling: Advanced Concepts. ER 2002. Pre-Conference Tutorials. Available in: http://www.loa-cnr.it/odcm.html

144. Herring C., Kaplan S. Viable Systems: The Control Paradigm for Software Architecture Revisited // Australian Software Engineering Conference, 2000. pp. 97–105.

145. Hovy E., Knight K., Junk M. Large Resources. Ontologies (SENSUS) and Lexicons // URL: www.isi.edu/natural-language /projects/ONTOLOGIES.html]. (9.12.2020)

146. Islam M., Katiyar V. Development of a software maintenance cost estimation model: 4th GL perspective // International Journal of Technical Research and Applications. 2014.Vol. 2, Issue 6. P. 65-68

147. Izurieta C., Bieman J.M. A multiple case study of design pattern decay, grime, and rot in evolving software systems // Software Quality Journal.-2013. Vol. 21, № 2. P. 289-323.

148. Kleshchev A.S. Chernyakhovskaya M.Yu., Shalfeeva E. Features of the automation of intellectual activities // Automatic Documentation and Mathematical Linguistics, 2015, V. 49, Issue 1, p. 10-20.

149. Kleshchev A.S., Shalfeeva E.A. Defining structural properties of ontologies // Journal of Computer and Systems Sciences International, 2008. № 47(2), p. 226-234.

150. Kleshchev A.S., Shalfeeva E.A. Knowledge bases' control for intelligent professional activity automatization // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2015. V 9227, p. 519-525.

151. Kleshchev A.S., Shalfeyeva Ye.A. Features of the system analysis at automation of intellectual professional activity // 22nd International Crimean conference microwave and telecommunication technology (CRIMICO 2012), Conference proceedings. 2012. P. 419-420.

152. Klyshinsky E., Gribova, V.V., ShakhgeldyanC., Shalfeeva, E.A Okun D.B., Geltser B.I., Gorbach T.A., Karpik O.D. Formalization of medical records using an ontology: patient complaints // Communications in Computer and Information Science. 2020. Vol. 1086. P. 143-153.

153. Lamy J.-B., Sekar B., Guezennec G., Bouaud J., Séroussi B. Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach // Artificial Intelligence in Medicine, 2019. vol. 94, pp. 42-53.

154. Lu J., Roberts C., Lang K., Stirling A., Madelin K. (2006) The Application of Semantic Web Technologies for Railway Decision Support. In: Gupta, Jatinder N.D. &Forgionne&Guisseppi A. & Mora T., Manuel (eds.) Intelligent Decision-making Support Systems. Decision Engineering. 2006. Springer, London. P. 321 – 337. https://doi.org/10.1007/1-84628-231-4_17

155. Marcus, S., McDermott, J. SALT: A knowledge acquisition language for propose-and revise systems // Artificial Intelligence. 1989. V.39. P. 1-37.

156. Musen M. The Protégé Project: A Look Back and a Look Forward // AI Matters. 2015. Jun; № 1(4). – P. 4-12.

157. Müller L., Gangadharaiah R. et. al. An open access medical knowledge base for community driven diagnostic decision support system development // BMC Medical Informatics and Decision Making, 2019. vol.19. P 1-7.

158. Noy N. F. and C.D. Hafner: The State of the Art in Ontology Design, AI Magazine, 1997. 18(3). P. 53-74.

159. Patil R. S., Szolovits P. and Schwartz W. B. "Modeling Knowledge of the Patient in Acid-Base and Electrolyte Disorders" Chapter 6 in Szolovits, P. (Ed.) Artificial Intelligence in Medicine. WestviewPress, Boulder, Colorado. 1982.

160. Pressman R.S. Software Engineering: Practitioner's Approach. 7th ed.McGraw-Hill, - 2010. - 930 p.

161. Ruiz F., Hilera J.R. (2006) Using Ontologies in Software Engineering and Technology. In: Calero C., Ruiz F., Piattini M. (eds) Ontologies for Software Engineering and Software Technology. Springer, Berlin, Heidelberg. 2006. pp 49-102. https://doi.org/10.1007/3-540-34518-3_2.

162. Rychener M.D. Expert systems for engineering design // Expert Systems. - 1985. vol. 2. № 1. P. 30-44.

163. Selby R. W. Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research. Wiley-IEEE Computer Society Pr, 2007. 832 p.

164. Shalfeeva E. Modeling of an intellectual problem solver by transformation of Semantic models // International Journal "Information Models and Analyses", 2013, Vol.2, N.3, p. 217-222.

165. Shalfeeva E.A. Monitoring of conceptual informational resources for intelligent software systems // RPC 2010 - 1st Russia and Pacific Conference on Computer Technology and Applications, p. 140-144.

166. Shalfeeva E. Semantic networks for intelligent task solvers // The 9th International Conference on Information Technology and Applications (ICITA2014), 1-4 July, 2014, Sydney, Australia.

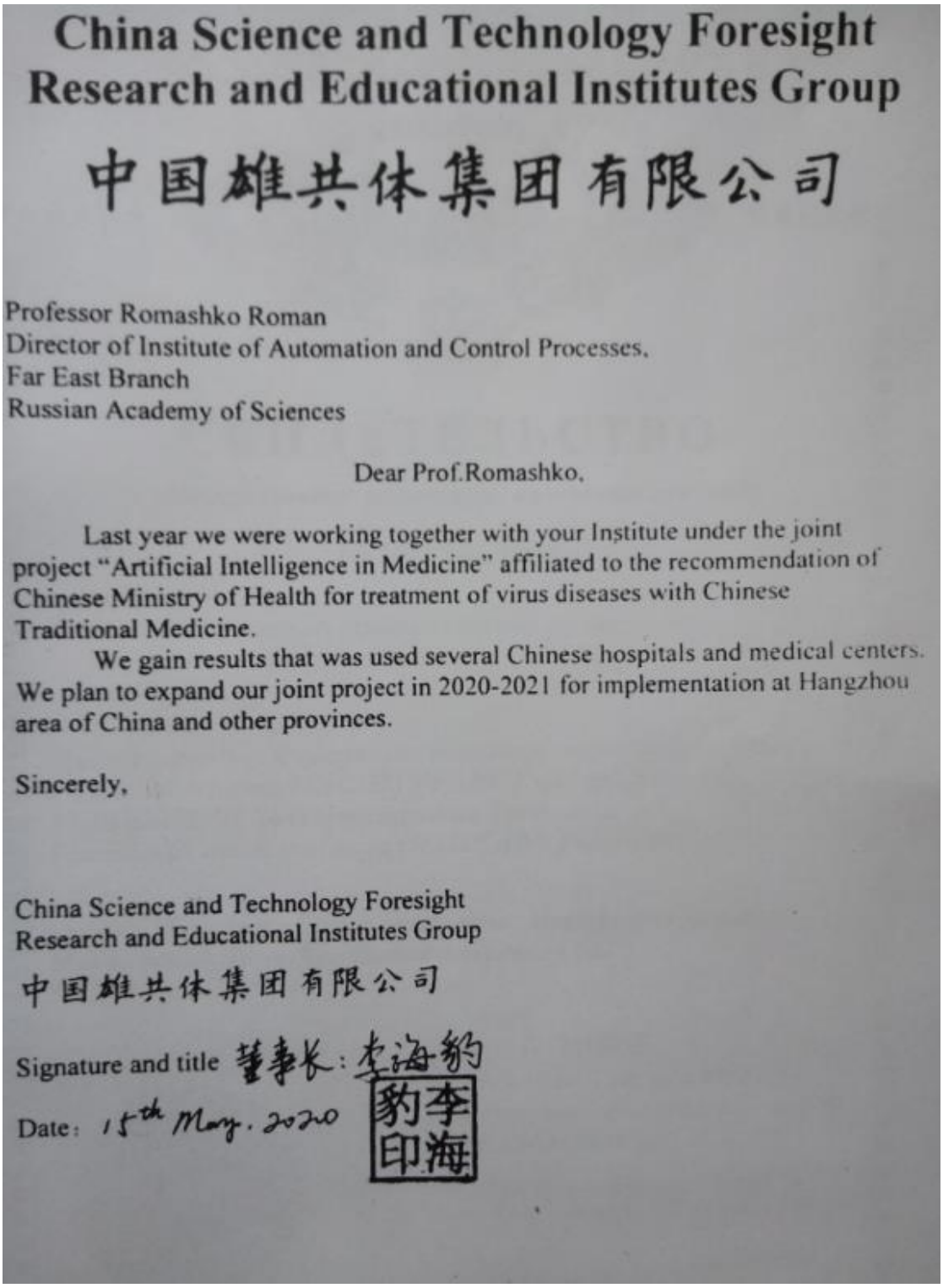
167. Sriram D., Maher. M. L. and Fenves, S. J. Knowledge-Based Expert Systems for Structural Design // Computers and Structures, January 1985. - P. 1-9.

168. Tudorache T., Noy N. F., Tu S.W., Musen M. A. Supporting collaborative ontology development in Protégé // 7th International Semantic Web Conference, Karlsruhe, Germany, Springer. 2008.

169. Wróblewska A. et al Application of semantic knowledge management system in selected areas of Polish public administration // Roczniki Kolegium Analiz Ekonomicznych, 2013, №29. P. 353-368.

170. Yourdon Ed. Modern Structured Analysis // Prentice-Hall, 1989. 672 p.

Приложение А



China Science and Technology Foresight
Research and Educational Institutes Group

中国雄共体集团有限公司

Professor Romashko Roman
Director of Institute of Automation and Control Processes,
Far East Branch
Russian Academy of Sciences

Dear Prof.Romashko,

Last year we were working together with your Institute under the joint project "Artificial Intelligence in Medicine" affiliated to the recommendation of Chinese Ministry of Health for treatment of virus diseases with Chinese Traditional Medicine.

We gain results that was used several Chinese hospitals and medical centers. We plan to expand our joint project in 2020-2021 for implementation at Hangzhou area of China and other provinces.

Sincerely,

China Science and Technology Foresight
Research and Educational Institutes Group
中国雄共体集团有限公司

Signature and title 董事长：李海豹

Date: 15th May. 2020

李海豹印

Акт внедрения в китайские медицинские учреждения облачного сервиса поддержки решений врача, являющегося результатом развития сервиса, созданного по представленной технологии.