Федеральное государственное бюджетное учреждение науки
Институт динамики систем и теории управления имени В.М. Матросова
Сибирского отделения Российской академии наук

На правах рукописи

Юрин Александр Юрьевич

Методы и программные средства создания интеллектуальных систем с декларативными базами знаний на основе модельных трансформаций

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

Диссертация на соискание ученой степени
доктора технических наук

Научный консультант:
ак. РАН,
И.В. Бычков

Иркутск – 2022

Оглавление

Перечень сокращений

БЗ – База знаний

ДС – деревья событий

ИИ – Искусственный интеллект

ИС – Интеллектуальная система

ЛПР – Лицо, принимающее решение

УМС – Уникальная механическая мистема

ЭПБ – Экспертиза промышленной безопасности

ЭС – Экспертная система

ЯПБЗ – Язык рпрограммирования баз знаний

ЯПЗ – Язык представления знаний

ATL – ATLAS Transformation Language (Язык трансформаций ATLAS)

CASE – Computer Aided Software/System Engineering (Автоматизированная разработка программного обеспечения/систем)

CIM – Computation Independent Model (Вычислительно-независимая модель

CLIPS – C Language Integrated Production System (Продукционная система, интегрируемая с языком C)

CRUD – Create, Read, Update, Delete (Создание, чтение, обновление, удаление)

CXL – Concept Mapping Extensible Language (Расширяемый язык отображения понятий)

DLL – Dynamic Link Library (Динамически связываемая библиотека)

EKB – External Knowledge Base (Внешняя база знаний)

EMF – Eclipse Modeling Framework (Среда моделирования Eclipse)

EUD – End-User Development (Разработка, ориентированная на конечного пользователя)

JESS – Java Expert Systems Shell (Оболочка экспертных систем на Java)

KBDS – Knowledge Base Development System (Система разработки баз знаний)

KMЗ – Kernel Meta Meta Model (Основная мете-мета модель)

M2C – Model-To-Code (Модель в код)

M2M – Model-To-Model (Модель в модель)

M2T – Model-To-Text (Модель в текст)

MDA – Model Driven Architecture (Архитектура, управляемая моделью)

MDD – Model Driven Development (Модельно-ориентированная разработка)

MDE – Model Driven Engineering (Модельно-ориентированная инженерия)

MIC – Model-Integrated Computing (Вычисления на основе интегрированных моделей)

MOF – Meta-Object Facility (Средство мета-описания объектов)

MTL – Model Transformation Language (Язык трансформаций моделей)

MVC – Model-View-Controller (Модель-Представление-Контроллер)

OCL – Object Constraint Language (Язык объектных ограничений)

ODA – Ontology Driven Architecture (Архитектура, управляемая онтологией)

OMG – Object Management Group (Консорциум (рабочая группа), занимающийся разработкой и продвижением объектно-ориентированных технологий и стандартов)

OWL – Web Ontology Language (Язык описания онтологий)

PDM – Platform Description Model (Модель описания платформы)

PHP – PHP: Hypertext Preprocessor (Препроцессор гипертекста для PHP)

PIM – Platform Independent Model (Платформо-независимая модель)

PKBD – Personal Knowledge Base Designer (Персональный редактор баз знаний)

PSM – Platform Specific Model (Платформо-зависимая модель)

QVT – Query / View / Transformation (Запрос / Представление / Преобразование)

RDF – Resource Description Framework (Среда описания ресурсов)

RVML – Rule Visual Modeling Language (Язык визуального моделирования правил)

SaaS – Software as a Service (Программное обеспечение как услуга)

SWRL – Semantic Web Rule Language (Язык правил для семантического веба)

TMRL – Transformation Model Representation Language (Язык представления моделей трансформации)

UML – Unified Modeling Language (унифицированный язык моделирования)

Введение

Актуальность. Одним из направлений повышения эффективности и надежности процессов обработки и передачи данных и знаний в вычислительных машинах, комплексах и компьютерных сетях является переход к передовым цифровым, интеллектуальным производственным технологиям, роботизированным системам, новым материалам и способам конструирования, создание систем обработки больших объемов данных (Big Data), машинного обучения и искусственного интеллекта (ИИ), что, в свою очередь, требует создания нового алгоритмического и программного обеспечения [349].

Современный уровень исследований в данных областях достаточно высок, однако проблема повышения эффективности и качества разработки программного обеспечения систем ИИ, включая базы знаний (БЗ), сохраняет свою актуальность, поскольку данный процесс остается трудоемким и требует привлечения и обеспечения взаимодействия специалистов различных специализаций и квалификаций, в том числе, профессиональных программистов. Для решения поставленной проблемы предлагаются новые языки и инструментальные программные средства, методы и технологии разработки. При этом перспективным является более полное включение в процесс создания систем ИИ конечных пользователей (end users) с передачей им отдельных функций, которые исторически реализовывались программистами. Подходы подобной направленности объединены в направление, известное как End-User Development (EUD) [7], основная идея которого – предоставить возможность конечному пользователю самому создавать и настраивать приложение. В рамках этого направления многообещающими способами спецификации предметной области и бизнес-логики приложений являются [35]: использование принципов визуального программирования, предметно/проблемно-ориентированных языков, а также модельных трансформаций с целью автоматизированной генерации программных кодов и спецификаций.

Значительный вклад в разработку и исследование моделей, методов и средств создания интеллектуальных систем, включая онтологии, БЗ и программные средства для их проектирования, внесли Абрамова Н.К., Аверкин А.Н., Баадер Ф., Берман А.Ф., Бычков И.В., Вагин В.Н., Ван Хармелен Ф., Варшавский П.Р.,

Васильев С.Н., Гаврилова Т.А., Голенков В.В., Грау Б., Грибова В.В., Грубер Т., Гуарино Н., Джарратано Дж., Джексон П., Еремеев А.П., Желтов С.Ю., Загорулько Ю.А., Клещев А.С., Колесников А.В., Кудрявцев Д.В., Люгер Г., Массель Л.В., МакГиннесс Д., Мешалкин В.П., Мотик Б., Николайчук О.А., Ноженкова Л.Ф., Норвиг П., Осипов Г.С., Осуга С., Патель-Шнайдер П., Попов Э.В., Поспелов Д.А., Райли Г., Рассел С., Рыбина Г.В., Стааб С., Смирнов Б.В., Смирнов С.В., Федоров М.В., Финн В.К., Фоминых И.Б., Хорошевский В.Ф., Хоррокс Я., Шрайбер Г., Штудер Р., Частиков А.П., Черняховская Л.Р. и др. В области автоматизации создания программных систем и их компонентов, разработки трансляторов, а также подходов трансформации моделей и программ можно отметить работы исследователей Агафонова В.Н., Ахо А., Баричели Б.Р., Гасевика Д., Горбунова-Посадова М.М., Гринфилда Дж., Д'Сильва А.Р., Ершова А.П., Клеппе А., Корандо Е., Крету Л.Г., Менса Т., Опарина Г.А., Сабельфельда В.К., Сендала С., Ульмана Дж., Фаулера М., Франкеля Д., Чарнецки К. и др.

Однако, существующие решения в области создания программного обеспечения систем ИИ и БЗ, в том числе, ориентированные на конечных пользователей, обладают рядом недостатков, в частности: акцентом на концептуализацию и формализацию знаний; слабой интеграцией, в том числе «по данным» с другим программным обеспечением, как в контексте использования разработанных ранее концептуальных моделей, так и создания отчуждаемых программных кодов и спецификаций.

Таким образом актуальна проблема, имеющая важное научно-техническое значение – разработка новых методов и средств создания программного обеспечения интеллектуальных систем, повышающего эффективность и надежность процессов обработки и передачи данных и знаний в вычислительных машинах, комплексах и компьютерных сетях за счет повторного использования и трансформации концептуальных моделей.

Цель исследования состоит в разработке методов, языков, алгоритмов и программных средств, повышающих эффективность создания интеллектуальных систем с декларативными базами знаний продукционного и прецедентного типа пользователями – специалистами в предметных областях (конечными пользователями - не специалистами в ИТ) за счет поддержки визуального

программирования, повторного использования концептуальных моделей и их трансформаций.

Задачи исследования:

- провести анализ языков, инструментальных средств и технологий создания интеллектуальных систем с декларативными базами знаний, в том числе, с использованием модельных трансформаций и ориентированных на конечных пользователей;
- разработать метод проектирования декларативных баз знаний интеллектуальных систем продукционного и прецедентного типа на основе модельных трансформаций;
- разработать методы проектирования программ трансформаций концептуальных моделей и программных компонентов-конверторов концептуальных моделей;
- разработать языки поддержки разработанных методов в части визуального программирования баз знаний и трансформаций концептуальных моделей.
- разработать алгоритмы и программные системы создания интеллектуальных систем с декларативными базами знаний на основе модельных трансформаций.
- провести апробацию и оценку эффективности разработки прикладных интеллектуальных систем с декларативными базами знаний на основе модельных трансформаций.

Объектом исследования являются интеллектуальные системы, модели, методы, алгоритмическое и программное обеспечение создания программных средств обработки знаний в вычислительных машинах, комплексах и компьютерных сетях.

Предметом исследования является модели, методы, алгоритмы и программное обеспечение создания интеллектуальных систем с декларативными базами знаний на основе модельных трансформаций.

Методы исследования. В работе использовались методы объектно-ориентированного и визуального программирования, трансформации моделей, построения трансляторов и предметно-ориентированных языков, а также методы и средства искусственного интеллекта и онтологического моделирования.

Научная новизна диссертации заключается в разработке теоретических и методологических основ решения проблемы повышения эффективности создания

интеллектуальных систем с продукционными и прецедентными базами знаний на основе трансформации концептуальных моделей разного уровня, при этом получены следующие результаты, выносимые на защиту:

1) создан оригинальный метод проектирования декларативных баз знаний интеллектуальных систем, который в отличие от известных реализаций модельно-ориентированного подхода обеспечивает использование новых моделей, языков и платформ, реализующих возможность непосредственного участия пользователей – специалистов в предметных областях (конечных пользователей - не специалистов в ИТ) на всех этапах процесса разработки;

2) разработан визуальный язык программирования продукционных баз знаний – RVML (Rule Visual Modeling Language), базирующийся на UML, характерной особенностью которого являются специализированные графические обозначения для представления элементов декларативных баз знаний и генерации программного кода;

3) предложен оригинальный текстовый декларативный язык программирования трансформаций концептуальных моделей – TMRL (Transformation Model Representation Language), который в отличие от известных описывает не только преобразуемые структуры и связи между ними, но и обеспечивает вызов внешних программных компонентов трансформаций;

4) разработаны методы проектирования программ трансформаций концептуальных моделей и программных компонентов-конверторов концептуальных моделей, отличающиеся от подобных использованием языка описания трансформаций моделей TMRL и реализацией принципов визуального программирования;

5) созданы алгоритмы и архитектура программных средств, обеспечивающие поддержку вышеупомянутых языков и методов, объединенные общей идеологией модельных трансформаций и формирующие новую технологическую платформу создания интеллектуальных систем с декларативными базами знаний продукционного и прецедентного типа.

Соответствие диссертации паспорту научной специальности. Тема и основные результаты диссертации соответствуют следующим областям

исследований паспорта специальности 05.13.11 – «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей»:

- Модели, методы и алгоритмы проектирования и анализа программ и программных систем, их эквивалентных преобразований, верификации и тестирования (Результаты 1, 4, 5).
- Языки программирования и системы программирования, семантика программ (Результаты 2, 3, 5).

Теоретическая значимость работы заключается в развитии методов и средств создания интеллектуальных систем на основе модельных трансформаций. Результаты диссертационного исследования использованы при выполнении государственных заданий и научных исследований ИДСТУ СО РАН, проектов РФФИ 15-37-20655 («Разработка моделей, методов и средств сервисно-ориентированной технологии синтеза баз знаний продукционных экспертных систем на основе трансформации концептуальных моделей»), 15-07-03088 («Разработка теории и принципов создания многоплатформенных продукционных экспертных систем на основе модификации модельно-управляемого подхода»), 19-07-00927 («Методы и инструментальные средства создания баз знаний на основе модельных трансформаций»), соглашения № 8770 от 04.10.2012 ФЦП «Научные и научно-педагогические кадры инновационной России» (рук.), грантов Президента России по проектам НШ-9508.2006.1, НШ–1676.2008.1, СП-2012.2012.5 («Разработка системы управления проблемно-ориентированными базами знаний для поддержки решения задач диагностирования и прогнозирования состояний опасных технических систем в нефтехимии»), работ по гранту Фонда содействия отечественной науке (рук.), проекта РНФ 22-21-00099 («Модели, методы и средства создания интегрируемых проблемно-ориентированных интеллектуальных помощников на основе модельно-ориентированного подхода»), гранта № 075-15-2020-787 Министерства науки и высшего образования РФ на выполнение крупного научного проекта по приоритетным направлениям научно-технологического развития (проект «Фундаментальные основы, методы и технологии цифрового мониторинга и прогнозирования экологической обстановки Байкальской природной территории»).

Практическая значимость результатов. Предложенные в рамках диссертационной работы модели, методы, алгоритмы и программное обеспечение позволяют снизить трудозатраты и сократить сроки разработки интеллектуальных систем и их баз знаний. Автором получено 14 свидетельств о регистрации программ для электронных вычислительных машин (ЭВМ), основными из которых являются: Personal Knowledge Base Designer (PKBD, рег.№№ 2016617733, 2012614093, 2007613714) – система разработки декларативных баз знаний и интеллектуальных систем и ее веб версия: Web PKBD; Knowledge Base Development System (KBDS, рег.№ 2019661803) – система создания программных компонентов трансформации концептуальных моделей; TreeEditorET/Extended Event Tree Editor (EETE, рег.№ 2012614092) – система визуального проектирования баз знаний на основе деревьев событий.

Практическая значимость результатов подтверждена полученными актами внедрения и справками использования программных систем АО «ИркутскНИИхиммаш», ООО «Смарт Технологии», ООО «ЦентраСиб», ИрНИТУ, МГТУ ГА.

Достоверность результатов проведенных исследований подтверждается обоснованным использованием методов модельных трансформаций, публикацией и индексацией полученных результатов в РИНЦ, WoS, Scopus, работоспособностью разработанного программного обеспечения, решением прикладных и тестовых задач.

Апробация результатов диссертации. Основные результаты диссертационного исследования докладывались автором на следующих научных мероприятиях: Национальная конференция по искусственному интеллекту (КИИ, 2008, 2010, 2014, 2016, 2018, 2020, 2021 гг.); Международная конференция «Знания – Онтологии – Теории» (ЗОНТ, 2015, 2017, 2021 гг.); Международная конференция «Системный анализ и информационные технологии» (САИТ, 2009, 2013, 2015, 2019 гг.); Международная научная конференция «Интеллектуальные информационные технологии в технике и на производстве» (IITI, Сочи, 2018 и 2021 гг.); Artificial intelligence and digital technologies in technical systems (Волгоград, 2020 и 2021 гг.); 3rd International Workshop on Information, Computation, and Control Systems for Distributed Environments (ICCS-DE, Иркутск,

2021); Scientific-practical Workshop Information Technologies: Algorithms, Models, Systems (ITAMS, Иркутск, 2018-2021 гг.); Байкальская Всероссийская конференция с международным участием «Информационные и математические технологии в науке и управлении» (ИМТ, Иркутск, 2009, 2011, 2013, 2014, 2016, 2019-2021 гг.); X Международная научно-практическая конференция «Актуальные проблемы и перспективы развития гражданской авиации» (Иркутск, 2021 г.); Международная конференция «Иванниковские чтения» (Нижний Новгород, 2021 г.); 4th Artificial Intelligence and Cloud Computing Conference (AICCC, Киото, Япония, 2021 г.); 24th International Database Engineering & Applications Symposium (IDEAS, Seoul, Korea, 2020 г.); International Conference «Information and Communication Technologies for Research and Industry» (ICIT, Saratov, 2019-2020 гг.); International Conference on Modelling and Development of Intelligent Systems (MDIS, Sibiu, Romania, 2019-2020 гг.); Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT, Yekaterinburg, 2020 г.); VIIth International Workshop «Critical Infrastructures: Contingency Management, Intelligent, Agent-Based, Cloud Computing and Cyber Security» (IWCI, Байкальск, 2020 г.); VII Всероссийская конференция «Безопасность и мониторинг природных и техногенных систем» (БиМПТС, Кемерово, 2020 г.); 13-я мультиконференция по проблемам управления (МТУиП, Санкт-Петербург, 2020 г.); 1st International Workshop on Advanced Information and Computation Technologies and Systems (Irkutsk, 2020 г.); International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON, Yekaterinburg, 2019 г.); International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO, Opatija, Croatia, 2018-2019 гг.); 3th International Conference «Computational Methods in Systems and Software» (CoMeSySo, 2019 г.); IV Всероссийская Поспеловская конференция с международным участием Гибридные и синергетические интеллектуальные системы (ГИСИС, Светлогорск, 2018 г.); Открытые семантические технологии проектирования интеллектуальных систем (ОСТИС, Минск, Белоруссия, 2016, 2018 гг.); Международная научно-практическая конференция «Программная инженерия: методы и технологии разработки информационно-вычислительных систем» (ПИИВС, Донецк, ДНР, 2016, 2018 гг.); Russian-Pacific Conference on Computer Technology and Applications (RPC,

Владивосток, 2017 г.); 12th International Forum on Knowledge Asset Dynamics(IFKAD, St. Petersburg, 2017 г.); XII Международная научно-практическая конференции Объектные системы (Ростов-на-Дону, 2016 г.); 8-й Всероссийская мультиконференция по проблемам управления (Дивноморское, 2015 г.); VI Всероссийская научно-техническая конференция с международным участием «Безопасность критичных инфраструктур и территорий» (Абзаково, 2014 г.); Российско-монгольская конференция молодых ученых по математическому моделированию, вычислительно-информационным технологиям и управлению (Ханх, Монголия, 2011, 2013, 2015 гг.); 3 Научная конференция «Автоматизация в промышленности» (Москва, 2009 г.); Международная научно-техническая конференция «Интеллектуальные системы» (AIS) и «Интеллектуальные САПР» (CAD) (Дивноморское, 2008 г.); 16 Международная конференция «Проблемы управления безопасностью сложных систем» (Москва, 2008 г.), а также семинарах ИДСТУ СО РАН.

Личный вклад автора. Все выносимые на защиту научные положения получены соискателем лично. В основных научных работах по теме диссертации, опубликованных в соавторстве, лично соискателем получены следующие результаты: метод и средства проектирования декларативных баз знаний интеллектуальных систем; визуальный язык программирования продукционных баз знаний. Результаты по разработке и реализации языка описания трансформаций концептуальных моделей, методам и средству проектирования программ трансформаций концептуальных моделей и программных компонентов-конверторов получены в неделимом соавторстве с Дородных Н.О. Вклад соискателя в ПО, зарегистрированное в соавторстве, состоит в формализации постановок задач, участии в разработке архитектуры программных комплексов и программной реализации.

Публикации. Результаты диссертационного исследования отражены в 110 научных работах. Основные публикации представлены в российских журналах, рекомендованных Высшей аттестационной комиссией для опубликования научных результатов диссертации, а также проиндексированых в международных базах цитирования Web of Science и Scopus, при этом 9 из них относятся к Q1 и Q2 по

рейтингу SJR. Получено 14 свидетельств о регистрации программ для электронных вычислительных машин (ЭВМ).

Структура диссертации. Диссертация состоит из введения, трех разделов и семи глав, заключения, библиографии из 374 наименований, списка принятых сокращений и 9 приложений. Общий объем основного текста работы – 264 страницы, включая 22 таблицы и 118 рисунков.

Во введении обосновывается актуальность диссертационного исследования, сформулированы его цель и основные задачи, показана научная и практическая значимость, а также новизна работы.

В первом разделе, включающем две главы, приведены результаты аналитического исследования в области разработки интеллектуальных систем и декларативных баз знаний, том числе с использованием модельных трансформаций. По результатам исследования обоснована актуальность разработки теоретических и методологических основ решения проблемы повышения эффективности создания ИС с декларативными БЗ на основе модельных трансформаций, ориентированных на конечных пользователей.

Во втором разделе приведено описание описание основных результатов диссертационного исследования. При этом в **главе три** дано описание разработанных языков для визуального программирования баз знаний (Rule Visual Modeling Language, RVML) и трансформаций концептуальных моделей (Transformation Model Representation Language, TMRL). **В главе четыре** представлены оригинальный метод проектирования декларативных баз знаний интеллектуальных систем, расширяющий принципы модельно-ориентированного подхода за счет новых моделей, языков и платформ в контексте инженерии знаний, а также программные системы, обеспечивающие его поддержку и ориентированные на конечных пользователей: Personal Knowledge Base Designer (PKBD), веб-версия PKBD (Web PKBD), TreeEditorET/Extended Event Tree Editor. **В главе пять** дано описание методов проектирования программ трансформаций концептуальных моделей и программных компонентов-конверторов концептуальных моделей, а также программной системы Knowledge Base Development System (KBDS), реализующей предлагаемые методы.

В разделе три, включающем две главы, приведены результаты апробации разработанного подхода для интеллектуализация и решения задач обеспечения надежности и безопасности оборудования в нефтехимии (**глава шесть**), в частности при разработке: прототипа продукционной ИС для выявления причин повреждений и разрушения элементов технических систем в нефтехимии; прецедентной ИС для подбора конструкционных материалов; программы для интеллектуального планировщика анализа отказов. В **главе семь** приведены результаты оценки эффективности применения предлагаемых языков, методов и средств по временному критерию в сравнении с классическим методом, который не предусматривает визуального программирования и синтез программного кода БЗ на основе трансформации концептуальных моделей, как косвенным способом на основе информации из литературных источников о трудоемкости разработки ИС, так и прямым – на тестовых примерах из учебного процесса ИрНИТУ. В **приложениях** приведены акты и справки о внедрении, а также примеры разработок баз знаний для решения задач в других областях: идентификации лицевых признаков эмоций; обнаружения нежелательных сообщений коммуникационного сервиса; поддержке технического персонала при поиске и устранении неисправностей системы электроснабжения воздушного судна; анализе и прогнозировании риска (опасности) лесного пожара на основе информации о классе пожароопасности лесов, метеоусловий и других факторов; создании программных компонентов трансформации концептуальных моделей в форме диаграмм классов UML, концепт-карт CmapTools (XTM) и ДС.

РАЗДЕЛ 1. Аналитический обзор: технологии, языки, средства создания интеллектуальных систем и модельные трансформации

Глава 1. Разработка интеллектуальных систем и баз знаний

1.1 Основные понятия и определения

Основные понятия и определения в области интеллектуальных систем достаточно полно рассмотрены в научной и учебной литературе [244, 262, 263, 313, 330, 331, 335, 336, 341].

В контексте данной диссертационной работы под знаниями понимаются закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области [244].

Знания могут быть классифицированы по различным основаниям:

- Поверхностные – знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области.
- Глубинные – абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов.

В базах знаний экспертных систем присутствуют оба вида знаний.

- Процедурные – знания, «растворенные» в алгоритмах, они управляют данными, для их изменения требуется изменять программы.
- Декларативные – знания в виде предложений, записанных на языках представления знаний, приближенных к естественному и понятных неспециалистам.

Использование декларативной формы представления знаний повышает технологичность создаваемых ИС, т.к. позволяет модифицировать БЗ без перекомпиляции или перепрограммирования всего приложения в целом. Знания в БЗ могут быть формализованы с использованием определенной модели и языка ее реализующего. В настоящее время все многообразие моделей представления знаний можно разбить на логические (формальные логические модели) и эвристические (семантические сети; фреймы; продукции или порождающие

правила). В свою очередь, ИС могут быть классифицированны по различным основаниям:

- по способу учёта временного признака: статические – решают задачи при неизменяемых в процессе решения данных и знаний. Статические системы осуществляют монотонное, непрерываемое решение задачи от ввода исходных данных до конечного результата; динамические – допускают такие изменения. Динамические системы допускают возможность пересмотра полученных данных и знаний;
- по способу формирования решений: аналитические – выбор решения из множества существующих альтернатив; синтетические – генерация неизвестных решений;
- по видам используемых знаний: с детерминированными знаниями; с неопределёнными знаниями (под неопределённостью знаний или данных понимается неполнота (отсутствие), недостоверность (неточность измерений), двусмысленность (многозначность), нечёткость (качественная оценка)).

В данной работе рассматривается проблематика и решение задачи повышения эффективности создания статических ИС аналитического типа с детерминированными знаниями.

В качестве целевых формализмов представления знаний используются продукции (порождающие правила) и прецеденты.

В основе продукционных БЗ лежит использование понятия «продукция» или «система продукций» [262, 263, 331, 356].

Что может быть формализовано в следующем виде:

$$(i)\text{: } Q\text{; } P\text{; } A \rightarrow B\text{; } N$$

где i – имя продукции; Q – сфера применения продукции; P – условие применимости ядра продукции; $A \rightarrow B$ – ядро продукции (Если … То …); N – постусловие продукции.

Достоинства: модульность правил, возможность эвристического управления поиском, возможности трассировки и трактовки, наличие специальных языков представления знаний и программ-оболочек.

Существует ряд языков и программных инструментальных средств,

обеспечивающих поддержку данного формализма, в частности, CLIPS, DROOLS и др. При этом существующие программные решения ориентированы как на программистов, так и на пользователей с низкими навыками в области программирования. Общим недостатком этих решений является невозможность использовать сторонние источники информации для автоматизированного формирования структур БЗ. Большинство же решений, ориентированных на непрограммирующих пользователей, не позволяют создавать отчуждаемый программный код для его последующей интеграции в сторонние системы.

Прецедентные БЗ основаны на использовании понятия «прецедент», под которым понимается специализированное представление фреймов с четко выраженными слотами описания проблемы и ее решения (а возможно, и последствия применения решения), которое используется при принятии решения «по аналогии» (в зарубежной литературе данное направление известно как case-based reasoning) [1, 133, 175]. Фрейм (от английского frame, что означает «каркас» или «рамка») был предложен Марвином Мински [319] в 70-е годы для обозначения структуры знаний для восприятия пространственных сцен. Фрейм – это абстрактный образ для представления некоего стереотипа восприятия. В большинстве случаев фрейм содержит набор слотов (свойств), каждый из которых содержит информацию о той или иной стороне описываемой ситуации.

Выделяют фреймы-прототипы и конкретные фреймы: фреймы-прототипы (обобщённые фреймы, образцы) хранят знания о предметной области; конкретные фреймы (фреймы экземпляры) пополняют эти единицы знания реальными данными.

Данные и знания во фреймовых структурах существуют одновременно.

Достоинства: наглядность представления информации, сочетание представления данных и знаний.

Применение фреймов в форме прецедентов дает возможность более эффективного использования накопленного опыта для решения различных задач без осуществления трудоемкого процесса извлечения глубинных знаний.

В общем случае прецедентные БЗ могут быть рассмотрены как базы данных с развитым механизмом формирования запросов. Однако, несмотря на то, что в большинстве случаев они содержать в своем составе базы данных, ключевым

отличием является использование меры близости – количественной оценки, используемой для оценки релевантных прецедентов при их поиске.

Основной принцип прецедентных систем – принятие решений «по аналогии», т.е. решение новой задачи (проблемы) путем повторного использования и адаптации (при необходимости) решений, которые были ранее получены при решении подобных задач.

Уточняя понятие «прецедент» отметим, что это структурированное представление накопленного опыта в виде данных и знаний, обеспечивающее его последующую автоматизированную обработку при помощи специализированных программных систем [1, 116, 133, 175, 240]. Основные характеристики прецедента:

• представляет особое знание, привязанное к контексту, что позволяет использовать знания на прикладном уровне;

• может принимать различную форму (вид): охватывая разные по продолжительности промежутки времени; связывая решения с описаниями проблем; результаты с ситуациями и т.д.;

• фиксирует только тот опыт, который может обучить (быть полезным), фиксируемые прецеденты потенциально могут помочь специалисту (ЛПР) достичь цели, облегчить ее формулирование в будущем или предупредить его о возможной неудаче или непредвиденной проблеме.

Структура прецедента зависит от задач, при решении которых необходимо обеспечить повторное использование опыта, однако, в большинстве случаев выделяют два основных компонента:

• *идентифицирующую (характеризующую) часть* – описывает опыт таким способом, который позволяет оценить возможность его повторного использования в определенной ситуации;

• *обучающую часть* – описывает урок (обучающее знание, решение) как часть единицы опыта, например, решение проблемы или его часть, доказательство (вывод) решения, альтернативные или неудавшиеся решения.

Формально прецедент может быть описан следующим образом:

$$M^{Task} = \{p_1, \ldots, p_M\},\ p_i \in Prop,\ Prop = \mathrm{U}_i p_i,\ i=[1,N] \qquad (1)$$

где M^{Task} – модель задачи; p_i – свойства задачи (значимые характеристики), $Prop$ – множество свойств.

Согласно (1) модель задачи определяется следующим образом:

$$M^{Task\_CBR}: Problem^{CBR} \rightarrow Decision^{CBR},$$

где M^{Task_CBR} – модель задачи в *терминах* прецедентного подхода; $Problem^{CBR}$ – описание задачи (проблемы), $Decision^{CBR}$ – решение проблемы, при этом:

$$Problem^{CBR} = <c^*, C>,\ C = \{c_1 \dots c_K\},\ c^* \notin C,$$

где c^* – это новый прецедент, C – база прецедентов.

$$Decision^{CBR} = \{d_1, \dots, d_R\},\ d_i = (c_i, s_i), c_i \in C,\ s \in [0;1]$$

где $Decision^{CBR}$ – решение проблемы в форме извлеченных прецедентов с оценками близости s_i.

Формализуем описание прецедента:

$$c_i = \{Prop_i^{Problem}, Prop_i^{Decision}\},$$

где $Prop_i^{Problem}$ – идентифицирующая (описательная) часть прецедента, $Prop_i^{Decision}$ – обучающая часть прецедента. Кроме того, каждая данная часть содержит свойства задачи и их сочетание сильно зависит от особенностей решаемой задачи:

$$Prop_i^{Problem} = \{p_1, \dots, p_m\},\ Prop_i^{Decision} = \{p_{m+1}, \dots, p_N\},$$

$$Prop_i^{Problem} \cup Prop_i^{Decision} = Prop,\ Prop_i^{Problem} \cap Prop_i^{Decision} = \varnothing.$$

Таким образом, при определении свойств задачи, формирующих модель прецедента, необходимо уточнить понятия предметной области, которые составляют эти части.

Существует ряд инструментальных средств, обеспечивающих поддержку данного формализма: CBR-Express, ReMaind, CASUEL, ReCall, Precedent, однако в контексте решаемой задачи они обладают слабой интеграционной способностью с различными системами визуального моделирования.

1.2 Технологии разработки интеллектуальных систем и баз знаний

Разработка интеллектуальных систем разных типов имеет свои особенности, однако можно выделить основные (универсальные) методологические этапы процесса их разработки [262, 263]:

1) Анализ и идентификация, включая решение задач: определение типа, характеристики задачи, состава участников процесса разработки; оценку требуемых и доступных ресурсов (временных, машинных, источников экспертных знаний, и др.); определение цели создания системы; выбор критериев оценки качества решений.

2) Концептуализация, включая решение задач: формулировки базовых концепции (понятий) и отношений между ними; построение концептуальной модели.

3) Формализация, включая решение задач: перевод концептуальной модели на некоторый формальный язык представления знаний; оценку полноты и степени достоверности (неопределенности) информации и других ограничений, накладываемых на логическую интерпретацию данных, таких как зависимость от времени, надежность и полнота различных источников информации.

4) Программная реализация, включая решение задач: определение структур данных; определение архитектуры; преобразование формализованных знаний в работающую программу.

5) Тестирование, включая проверку работы созданного варианта системы на большом числе репрезентативных задач для оценки его точности и полезности.

Для решения задач, возникающих на данных этапах, предлагается ряд методологий (подходов), ориентированных на конечных пользователей (end users), которые являются более перспективными в контексте решаемой задачи. Подобные подходы объединены в направление, известное как End-User Development (EUD) [7, 35, 142], включающее поднаправления: программирование для конечных пользователей (End-User Programming, EUP) и разработка (включая моделирование и проектирование) для конечных пользователей (End-User Software Engineering, EUSE). Основная идея направления – предоставить возможность конечному пользователю самостоятельно создавать и настраивать приложение, что позволит расширить круг потенциальных разработчиков за счет пользователей с разным уровнем навыков программирования и повысить интенсивность создания подобного рода систем и их компонентов для решения различных задач.

В настоящее время основными направлениями работ в данной области являются:

- создание инструментальных средств компонентной сборки, в том числе на основе облачных сервисов и их композиций;
- создание специализированных и проблемно/предметно-ориентированных языков для описания логики интеллектуальных систем;

- создание средств программирования логических правил, в том числе основанных на принципах интеллектуального интерфейса, шаблонах, мастерах, графических метафорах и модельных трансформациях;
- создание шаблонов проектирования и онтологий;
- создание методов, основанных на обработке естественного языка;
- создание методов, обеспечивающих использование различных источников информации (электронных таблиц, баз данных, концептуальных моделей) в контексте парадигмы больших данных;
- применение разработанных методов при решении практических задач.

Для целей диссертационного исследования перспективным является использование EUD подхода, основанного на модельных трансформациях и обеспечивающего поддержку принципов визуального программирования и предметно/проблемно-ориентированных языков [35].

Рассмотрим наиболее известные технологии, ориентированные на создание интеллектуальных систем и баз знаний и реализующие отдельные принципы EUD.

OSTIS (Open Semantic Technologies for Intelligent Systems) [74, 246, 247] – комплекс моделей, средств и методов, предназначенных для разработки интеллектуальных систем. OSTIS базируется на применении в качестве способа кодирования информации унифицированных семантических сетей с базовой теоретико-множественной интерпретацией их элементов. Такой способ назван SC-кодом (Semantic Code), а семантические сети, представленные в SC-коде, – SC-графами (SC-текстами или текстами SC-кода). Для визуализации текстов SC-кода используются такие внешние языки, как SCg (Semantic Code graphical), SCn (Semantic Code natural) и SCs (Semantic Code string). Модель какой-либо сущности, записанная средствами SC-кода, названа sc-моделью указанной сущности. Для поддержки технологии создана платформа IMS.OSTIS METASYSTEM.

Технология охватывает все этапы жизненно цикла интеллектуальных систем, позволяя решать задачи проектирования, генерации кода, разработки программного обеспечения. С точки зрения EUD данная технология реализует принципы компонентной сборки, визуального программирования, модельных трансформаций. При этом интеграционные возможности с другими системами

достаточно ограниченные, т.к. созданные интеллектуальные системы функционирую в рамках технологической платформы.

IACPaaS (Intelligence Application Control Platform as a Service) [76] – набор методов и средств создания интеллектуальных систем с декларативными базами знаний, основные принципы:

- двухуровневый подход к созданию компонентов, включающий: 1) построение структурной декларативной модели (онтологии), 2) интерпретацию этой модели;
- единый язык и редактор для описания моделей компонентов;
- автоматическая генерация редакторов для создания компонентов по моделям;
- реализация инструментария и прикладных систем в форме облачных сервисов.

Модели компонентов интеллектуальных сервисов сформированы на языке описания моделей и представлены в виде связанного размеченного корневого иерархического двоичного орграфа. Разметка определяет семантику правил формирования (создания и модификации) компонентов, накладывая ограничения на их структуру и содержание. Для поддержки технологии разработан инструментарий в форме облачной платформы IACPaaS (https://iacpaas.dvo.ru), реализующей трехуровневую архитектуру (три уровня компонентов): Toolkit Core, Basic Toolkit и Extensible Toolkit.

Технология представляет собой комплексное решение, обеспечивая, в том числе, решение задач проектирования, интерпретации спецификаций, разработки программного обеспечения. С точки зрения EUD данная технология реализует принципы компонентной сборки и модельных трансформаций. Созданные системы функционируют в рамках технологической платформы.

АТ-ТЕХНОЛОГИЯ [138, 139, 339-342] – совокупность методов и средств, реализующих задачно-ориентированную методологию компьютерного проектирования интегрированных экспертных систем (ИЭС), созданную Рыбиной Г.В.

Технология обеспечивает решение следующих задач: анализ системных требований пользователей на разработку ИЭС и построение модели архитектуры

ИЭС; прямое извлечение знаний из экспертов, естественноязыковых текстов и БД; структурирование полученных знаний в виде поля знаний и построение БЗ о ПрО; реализация функций традиционных (т.е. простых продукционных) ЭС; реализация гипертекстовой модели общения; реализация обучающих функций; реализация функций, обеспечивающих интеграцию средств представления и обработки знаний в ЭС с СУБД и ППП расчетного и графического характера; проектирование с использованием интеллектуального планировщика компонентов прикладной ИЭС (диалоговых форм, спецификаций процессов, обработчиков событий, и т.п.); программирование, конфигурирование и тестирование прототипов ИЭС; реализация сервисных функций.

Технология представляет собой комплексное решение, охватывающее все этапы создания интеллектуального программного обеспечения. При этом технология, хотя и поддерживает принципы компонентной сборки, но ориентирована в большей степени на пользователей с высокими навыками программирования.

CAKE (Computer Aided Knowledge Engineering Technique) [30] – подход для крупноблочного конструирования и визуализации сложных зависимостей между отдельными агентами (модулями или подсистемами). В настоящий момент технология не развивается, также не удалось обнаружить инструментарий, обеспечивающий ее поддержку. Согласно [30] она позволяет осуществлять крупноблочное моделирование и сборку программных систем с поддержкой визуального моделирования.

CommonKADS (Common Knowledge Acquisition and Documentation Structuring) [145, 146, 154] – технология структурирования и извлечения знаний, которые в дальнейшем могут быть использованы для разработки интеллектуальных систем, создавалась в рамках программы ESPRIT IT, в настоящий момент является стандартом де-факто в данной области.

Включает следующие компоненты: методологию для управления проектами, связанными с инженерией знаний; среду для управления знаниями; методологию для извлечения знаний.

CommonKADS содержит предопределенное множество моделей, ориентированных на определенные аспекты или части полного представления:

- *Организационную модель*, которая предлагает механизм для анализа организаций (компаний) и выявления возможных проблемы при создании систем, основанных на знаниях; определения потенциальной возможности их создания и оценки их влияния на организацию.
- *Модели задач,* которые создаются для бизнес-процесса в контексте анализа его структуры, входных и выходных данных, предварительных условий, критериев эффективности, а также необходимых ресурсов и компетенций.
- *Агентные модели*, которые используются для описания исполнителей задач. Агентом может быть человек, информационная система или любая другая сущность, способная выполнить задачу. Модель агента описывает характеристики агентов, в частности их компетенции, полномочия действовать и ограничения, а также коммуникационные каналы.
- *Модели знаний* используются для подробного описания типов и структур знаний, используемых при выполнении задач в независимой от программной реализации форме.
- *Модели коммуникаций* используются для описания коммуникативных транзакций между задействованными агентами в независимой от программной реализации форме.
- *Модель проектирования*. Перечисленные выше модели CommonKADS в совокупности составляют спецификацию требований к системе знаний в различных аспектах. Основываясь на этих требованиях, модель проектирования дает спецификацию технической системы с точки зрения архитектуры, платформы реализации, программных модулей, репрезентативных конструкций и вычислительных механизмов, необходимых для реализации функций, заложенных в моделях знаний и коммуникаций.

В качестве средств поддержки моделирования используются диаграммы UML: прецедентов, деятельностей, состояний и адаптированный вариант диаграмм классов, отличающиеся отсутствием моделирования функций (операций, методов). Для моделирования задач, агентов и коммуникаций используются специализированные текстовые нотации.

Технология в большей степени ориентирована на поддержку управления знаниями (knowledge management), содержит рекомендации и практики для их

идентификации и концептуализации. Имеются средства реализации, в частности, для SWI-Prolog. С точки зрения EUD поддерживаются модельные трансформации и визуальное программирование.

MOKA (Methodology and tools Oriented to Knowledge-based engineering Applications) [153] – методология и средства, использующие знания экспертов при создании интеллектуальных приложений (систем). Основные этапы методологии: идентификация задачи (определение необходимости и практической реализуемости приложения), доказательство (оценка целей и рисков создания приложения), захват (сбор и структуризация новых знаний), формализация (разработка моделей продуктов и процессов), упаковка знаний (разработка приложения), активация (представление, использование и сопровождение приложения).

Неформально, создаваемые модели состоят из ICARE (Illustrations (пояснения), Constraints (ограничения), Activities (деятельности), Rules (правила) и Entities (сущности)) форм, которые используются для декомпозиции и хранения элементов знаний. Впоследствии эти формы могут быть связаны для создания структурированной сети элементов знаний, которые вместе составляют представление проблемной области. Для визуализации моделей используется MML (Moka Modelling Language) - специализированная модификация UML. MML предназначен, в том числе, для классификации и структурирования элементов ICARE и формализованного описания моделей процессов и продуктов.

MOKA ориентирована на поддержку инженера по знаниям, а не конечного пользователя, поэтому отнесение данной технологии к EUD условно. Механизмы представления знаний и вспомогательные инструменты полностью не определены.

Сравнение рассмотренных методологий (подходов) приведено в таблице 1.2.1.

Таблица 1.2.1 Качественное сравнение EUD технологий для создания интеллектуальных систем.

Критерий / Технология	Цели технологии	Реализованные EUD направления	Поддержи-вающие инструменталь-ные средства	Целевые языки програм-мирования	Поддержи-ваемые языки моделиро-вания	Интегра-ционные возможнос ти
OSTIS	проектирование, генерация кода, разработка программного обеспечения	компонентная сборка, визуальное програм-мирование, модельные трансформации	IMS.OSTIS METASYSTEM	SC*	SC*	средние

IACPaaS	проектирование, интерпретация спецификаций, разработка программного обеспечения	компонентная сборка, модельные трансформации	облачная платформа IACPaaS	-	язык описания модели, иерархический бинарный граф	-
CAKE	крупноблочное моделирование, описание логики	визуальное програм-мирование, модельные трансформации	-	-	расширенные диаграммы пакетов UML	-
CommonKADS	крупноблочное моделирование, описание логики, концептуализация, формализация	визуальное програм-мирование, модельные трансформации	SWI-Prolog …	Prolog …	UML, деревья, CML**	средние
MOKA	крупноблочное моделирование, описание логики, концептуализация, формализация	визуальное програм-мирование, модельные транс-формации, мастера, шаблоны	-	-	MML***	-
Предлагаемая диссертантом технология	**Проектирование, генерация кода, интепретация спецификаций, разработка программного обеспечения**	**визуальное програм-мирование, модельные транс-формации, мастера**	**PKBD, WPKBD, KBDS, EETE**	**CLIPS, DROOLS, PHP**	**UML, RVML, концепт-карты, деревья, онтологии, таблицы**	**высокие**

*SC – Semantic Code, including: SCg (Semantic Code graphical), SCn (Semantic Code natural) and SCs (Semantic Code string)

**CML – The CommonKADS (textual) conceptual modelling language

***MML – MOKA Modelling Language, an adaptation of UML

На основе таблицы 1.2.1 можно определить следующие особенности рассмотренных технологий, ориентированных на создание ИС:

• акцент на концептуализацию и формализацию знаний, а также крупноблочное моделирование;

• ограниченный набор программных средств в открытом доступе и готовых к использованию без модификации (настройки) программистами;

• поддержка ограниченного набора языков программирования баз знаний;

• поддержка, преимущественно, общесистемных моделей (UML) при построении систем;

• слабая возможность интеграции, в том числе «по данным» с другим программным обеспечением и отсутствие возможности создания отчуждаемых программных модулей.

Выделенные недостатки обуславливают актуальность данной работы в части

разработки новой технологии создания ИС и БЗ, ориентированной на конечного пользователя, и обеспечивающей возможность использования концептуальных моделей, визуального программирования и автоматической кодогенерации.

1.3 Языки разработки декларативных баз знаний

В области ИИ разработано большое количество методов и средств обработки и представления знаний. При этом, наряду с семантическими технологиями, в том числе, онтологиями, при создании интеллектуальных систем и их БЗ продолжают активно применятся языки описания знаний, основанные на правилах, что обусловлено простотой данного формализма и наглядностью представления с его помощью предметных знаний, возможностью их модульного изменения, и трассируемостью получаемых решений.

В контексте обеспечения возможности применения формализма продукций конечными пользователями актуальна задача поддержки визуального программирования логических правил с использованием общесистемных и специализированных нотаций, а также их прямое отображение в код или спецификации (автоматическая кодогенерация БЗ и ИС на определенном ЯПЗ). Используя классификацию из [243] и ее модификацию [279] были определены основные группы способов/подходов автоматизации создания БЗ:

- текстовый, который обеспечивает прямое манипулирование языковыми конструкциями и ориентирован преимущественно на программистов; средства реализации – специализированные редакторы, например, Visual JESS [77] для JESS (Java Expert Systems Shell), поддерживающие механизмы контекстной помощи и цветового выделения основных языковых конструкций;
- графический, который обеспечивает манипулирование визуальными абстракциями, соответствующими элементам языковых конструкций, с последующим их прямым отображением в программные коды или спецификации. Именно этот подход позволяет вовлечь конечных пользователей в процесс разработки за счет поддержки визуального программирования.

Далее, в рамках графического подхода выделяются:

- предметно- и проблемно-ориентированные языки, ориентированные на частные задачи;

- универсальные семантические графовые структуры [82, 167] в форме концепт-карт и онтологий; при этом из-за отсутствия общепринятой содержательной трактовки отношений между элементами моделей при их преобразовании возникают сложности, которые ограничивают широкое их применение при разработке интеллектуальных систем и их БЗ: например, VisiRule [166] (Рис.1.3.1) предлагает цветовую индикацию узлов блок-схем (flowchart), а в [276] определяются правила именования концептов и их свойств, в [332] (Рис.1.3.2) используется определенная цветовая схема при обозначении дуг. Некоторые из языков данной группы, например, VIPR (VIsual Imperative Programming) [306] (Рис.1.3.3) содержат оригинальные графические элементы нераспространенные среди специалистов-предметников;

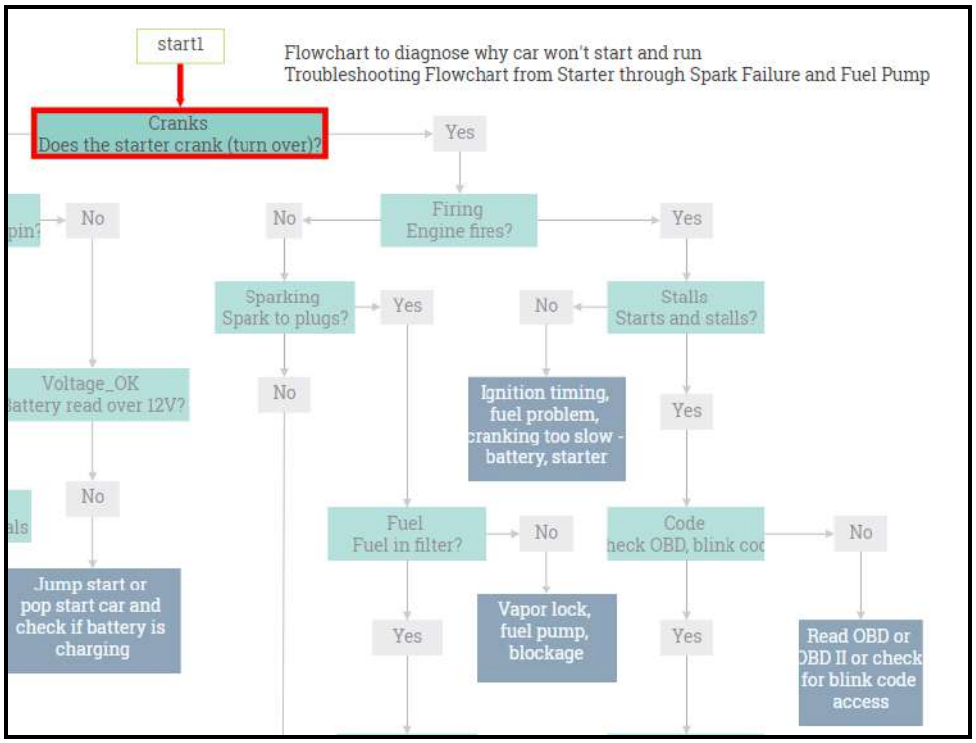


Рис. 1.3.1. Пример отображения продукций в VisiRule.

- расширения/профили/специализации распространенных универсальных языков, ориентированные на представление логических и причинно-следственных отношений, например, UML-Based Rule Modeling Language (URML) [99, 100] (Рис.1.3.4) и Rule Visual Modeling Language (RVML) [184, 368] являются профилями Unified Modeling Language (UML);

- таблицы (табличные представления), предназначенные для их последующего преобразования в коды и спецификации, например, таблицы решений [214] или специализированные – eXtended Tabular Trees (XTT2) [115].

Рис. 1.3.2. Пример отображения продукций из [332].

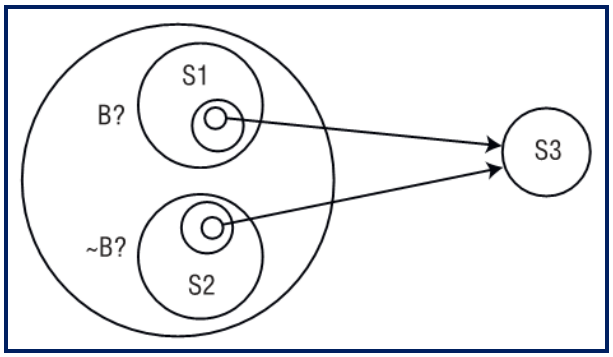


Рис. 1.3.3. Пример отображения продукций в нотации VIPR.

1.3.1 Текстовые

В рамках текстового подхода при создании баз знаний и экспертных систем широко используются CLIPS [356], JESS [83] и DROOLS [56]. Помимо данных языков, следует упомянуть языки разметки, такие как RuleML, SWRL.

RuleML (Rule Markup Language, а также Rule Modeling Language) [61, 137] представляет собой семейство унифицированных языков разметки для описания правил в Web. В основе использование языков схем (в соотвествии с Relax NG - REgular LAnguage for XML Next Generation [131]) для веб-документов и данных, первоначально разработанных для XML и позже перенесенных в другие форматы,

такие как JSON. RuleML связующее звено между RIF и общей логикой. Де-факто является отраслевым стандартом дополняемой спецификациями OMG (главным образом SBVR (Semantics Of Business Vocabulary And Business Rules) [143], PRR (Production Rule Representation) и API4KP (API for Knowledge Platforms) [124]) и составляющих основу спецификации OASIS (LegalRuleML). Является частью SWRL [132], позволяет расширять другие языки правил.

RuleML предназначен для единообразного представления и обмена основными видами веб-правил между различными логиками правил и платформами. Включает: Deliberation RuleML, Consumer RuleML and Reaction RuleML. Основан на Datalog/Prolog.

SWRL (Semantic Web Rule Language) [132] – язык описания правил и логики для Semantic Web, обеспечивает комбинацию OWL DL или OWL Lite с подмножеством RuleML. Правила SWRL имеют форму импликации между антецедентом и консеквентом.

R2ML (REWERSE I1 Rule Markup Language) – XML-подобный формат, ориентированный на обмен правилами между различными системами и инструментами и описание правил в онтологиях. R2ML интегрирует язык объектных ограничений (OCL), SWRL, RuleML.

IRL (ILOG Rule Language) – язык описания правил, а также поддержки их исполнения. Является подмножеством языка программирования Java.

CML (The commonKADS conceptual modelling language) [146, 154] – декларативный язык для описания концептуальных моделей.

1.3.2 Графические (нотации)

URML (UML-Based Rule Modeling Language) [99, 100] – расширение стандартного UML для моделирования правил. Расширение включает введение в метамодель UML понятия «правило». URML ориентирован на поддержку не только SWRL правил, включающих такие элементы как «условие» и «действие», но и элементы «постусловие» и «событие».

Пример URML правила (Рис.1.3.4): ЕСЛИ «машина в гараже» И «она не арендована» И «она не запланирована для техосмотра» ТО «она доступна».

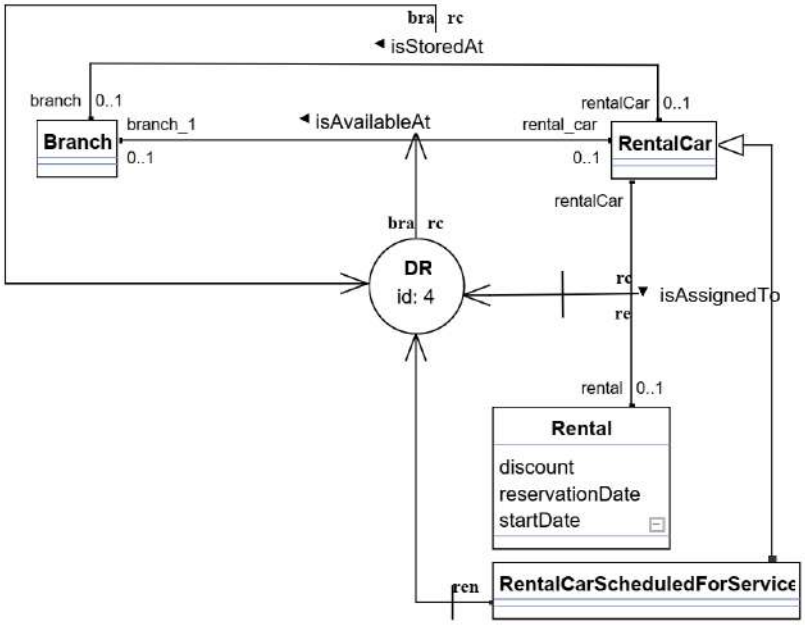


Рис. 1.3.4. Пример правила в нотации URML.

CAKE [30] (Computer Aided Knowledge Engineering Technique) –подход для конструирования и визуализации сложных зависимостей между отдельными элементами, в качестве которых могут выступать агенты или модули. Для визуализации моделируемых элементов используется расширение диаграмм сущность-связь (Рис.1.3.5).

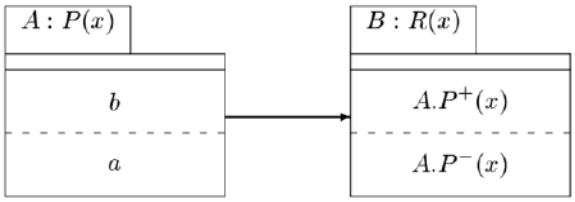


Рис. 1.3.5. Пример CAKE диаграммы [30]

CommonKADS (Common Knowledge Acquisition and Documentation Structuring) [145, 146, 154]. В рамках данной технологии используются свои оригинальные графические нотации. В частности, для моделирования знаний предметной области используется нотация подобная UML диаграммам классов, отличающаяся отсутствием моделирования функций (операций, методов) (Рис.1.3.6). Для моделирования задач, агентов и коммуникаций используются специализированные текстовые нотации.

MML (Moka Modelling Language) [153] – язык поддержки методологии MOKA, представляет собой специализированная модификацию UML, в частности, диаграмм классов.

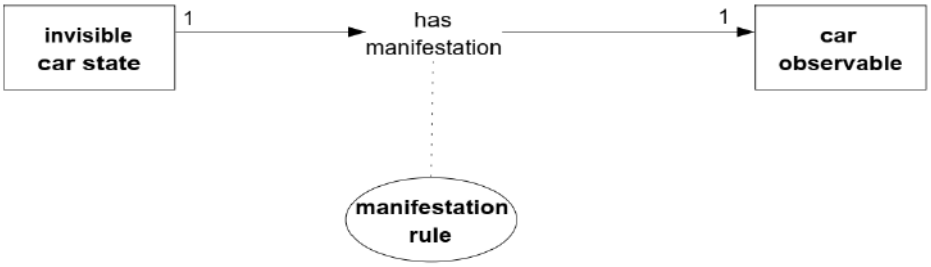


Рис. 1.3.6. Графическое изображение правила в одной из нотаций CommonKADS [154].

SCg (Semantic Code graphical) [74, 246, 247] – язык, используемый для визуализации текстов SC-кода в рамках проекта OSTIS (Рис. 1.3.7), используется для описания сложно-структурированных данных предметных областей. Представление информации на языке SC осуществляется при помощи ключевых узлов и дуг между ними. Язык содержит в своем составе множество подъязыков, направленных на решения определенных классов задач. Для представления знаний о предметной области, то есть логических соотношениях между понятиями предметной области используется язык SCL (Semantic Code Logic). В качестве подъязыка обработки SC-конструкций выступает SCP (Semantic Code Programming). Как и все языки семейства SC язык SCP относиться к разряду графовых языков. Средства языка SCQ (Semantic Code Questions) позволяют обрабатывать вопросы, задаваемые пользователем, при помощи однородных семантических сетей.

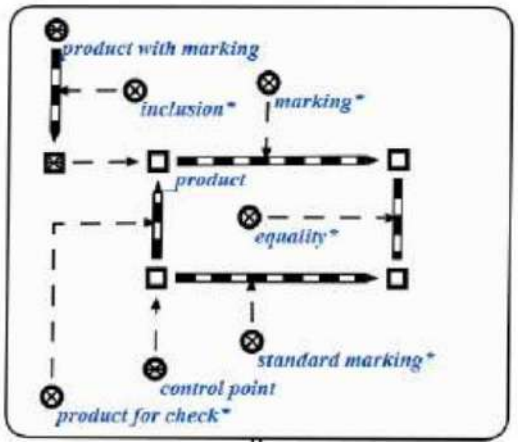


Рис. 1.3.7. Графическое изображение правила в нотации SCg [74].

RVML (Rule Visual Modeling Language) [368] – предлагаемая диссертантом нотация для моделирования логических правил типа «ЕСЛИ-ТО» в рамках

технологии PESoT (Prototyping Expert System based On Transformations), рассматривается как расширение UML, имеет поддержку в программных инструментальных средствах и примеры практического применения (Рис. 1.3.8).

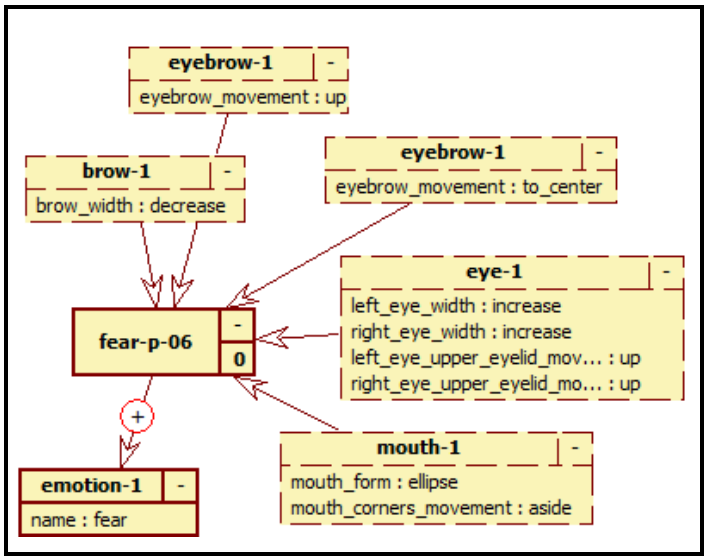


Рис. 1.3.8. Графическое изображение правила в нотации RVML [368].

В таблице 1.3.1 приведено сравнение упомянутых выше языков и нотаций.

Таблица 1.3.1. Сравнение основных языков построения продукционных баз знаний

Язык	Тип	Средство/ Среда поддержки	Назначение	Связанные языки, стандарты
RuleML	текстовый, разметки, декларативный	-	описание структур, обмен данными	RIF, SWRL
SWRL	текстовый, разметки, декларативный	Pellet	описание структур, логический вывод	OWL DL, RuleML
R2ML	текстовый, разметки, декларативный	-	описание структур, обмен данными	OCL, SWRL, OWL, RuleML
Prolog	текстовый, декларативный, графический	Visual Prolog	описание структур, логический вывод	
JESS	текстовый, декларативный, процедурный	Visual JESS (программы-мастера)	описание структур, логический вывод	Java
CLIPS	текстовый, декларативный, процедурный	ClipsWin	описание структур, логический вывод	C
Drools	текстовый, декларативный, процедурный	DroolsExpert	описание структур, логический вывод	Java
CML	текстовый, декларативный	-	описание структур	-
URML	графический	Модуль для Fujaba	описание структур	OCL
VIPR (VIsual Imperative Programming)	графический	-	описание структур	-
Flowchart	графический	VisiRule	описание структур	-

XTT2 (eXtended Tabular Trees 2)	графический	-	описание структур	-
CAKE	графический	-	описание зависимостей	UML
CommonKADS	графический	-	описание структур, процессов, состояний	UML
MML (Moka Modelling Language)	графический	-	описание структур	UML
SCg (Semantic Code graphical)	графический	IMS.OSTIS METASYSTEM	описание структур, логический вывод	SC (Semantic Code)
Предлагаемый диссертантом язык - RVML (Rule Visual Modeling Language)	**графический**	**PKBD, KBDS**	**описание структур и причинно-следственных зависимостей**	-

Рассмотренные текстовые языки создавались для программистов, и изначально не рассматривалась возможность их использования конечными пользователями с низкими навыками программирования. В дальнейшем стали появляться различные «надстройки» для этих языков в форме специализированных программных средств, реализующие такие EUD подходы, как шаблоны и диалоги мастера. В свою очередь графические языки своей целью ставили вовлечение непрограммирующих специалистов в процесс моделирования и проектирования, в том числе, программных систем. Однако, в своем большинстве они акцентируются на описании концептуальных моделей типа «сущность-связь» и не учитывают специфику создания интеллектуальных систем, в частности, не специфицируя особенности продукций и логических отношений в части операций со знаниями, коэффициентов уверенности, логических операций. В свою очередь, решения, которые учитывают эту специфику (URML, SCg), оперируют семантически сложными графическими элементами, даже при условии специализации UML.

Следует также отметить, что визуальное моделирование нечеткости и неопределенности довольно слабо представлено специализированными (проблемно- или предметно-ориентированными) языками. Примерами подобных языков являются нечеткие когнитивные карты [94], нечеткие ER-модели [205], нечеткие UML-модели [150] и др., но они не используются при разработке интеллектуальных систем и их БЗ. Таким образом, подтверждается актуальность данной работы в части разработки нового языка для моделирования декларативных БЗ ИС.

1.4 Программные инструментальные средства разработки баз знаний

Программные средства построения интеллектуальных систем и баз знаний можно классифицировать по следующим типам:

1) Среды разработки и специализированные редакторы, среди которых выделяются:

1.1) Ориентированные на программистов (WinCLIPS и др.), особенностью которых является: необходимость навыков программирования на определенном языке; поддержка основных формализмов представления знаний; низкая интеграционная способность с системами визуального моделирования; ограниченная поддержка таких компонентов интеллектуальных систем, как машина вывода, подсистема объяснения и др.

1.2) Ориентированные на непрограммирующих пользователей (ExSys, Visual Expert System Designer, Personal Knowledge Base Designer и др.), особенностью которых является: поддержка основных формализмов представления знаний; ориентация на непрограммирующих пользователей; использование принципов визуального моделирования и автоматической кодогенерации; ограниченная поддержка таких компонентов интеллектуальных систем, как машина вывода, подсистема объяснения и др.

2) «Оболочки» - инструментальные средства, полученные путем обобщения разработанных ранее интеллектуальных систем, среди которых выделяются:

2.1) Общего назначения (GURU и др.), особенностью которых является: полная поддержка всех компонентов интеллектуальных систем; предопределенный ограниченный набор стратегий вывода и формализмов представления знаний; необходимость навыков программирования на определенном языке.

2.2) Проблемно/Предметно-ориентированные (EMYCIN, ДИЭКС, СПРИНТ-РВ (ДИЭКС), E-INFOT и др.), особенностью которых является: ориентация на решение задач определенного класса (диагностика, прогнозирование, классификация и др.) или в определенной предметной области (авиация, нефтехимия и др.); заранее определенные структуры (знания) определенной проблемной /предметной области.

3) Средства онтологического и когнитивного моделирования, CASE-средства, среди которых выделяются:

3.1) Общего назначения (Protégé, FreeMind, Xebece, TheBrain, XMind, Rational Rose и др.), особенностью которых является: визуальное моделирование элементов и структур интеллектуальных систем; узкая специализация (не охватывают все этапы процесса построения); поддержка частичной кодогенерации.

3.2) Проблемно/Предметно-ориентированные (Tree Expert и др.), особенностью которых является: ориентация на определенные типы моделей; поддержка частичной кодогенерации.

В контексте диссертации были рассмотрены наиболее популярные программные инструментальные средства, реализующие принципы EUD и используемые при создании ИС и БЗ.

CLIPSWin [34]– оболочка для разработки продукционных баз знаний и экспертных систем на языке CLIPS, разработка которых осуществляется непосредственно написанием правил, фактов, шаблонов и т.д. программистом на языке CLIPS.

Обладает очень простым интерфейсом, ориентированным на профессиональных программистов. Мало подходит для применения специалистами-предметниками (Рис. 1.4.1). Позволяет создавать отчуждаемые базы знаний. Интеграция с системами концептуального моделирования и поддержка визуального программирования отсутствуют.

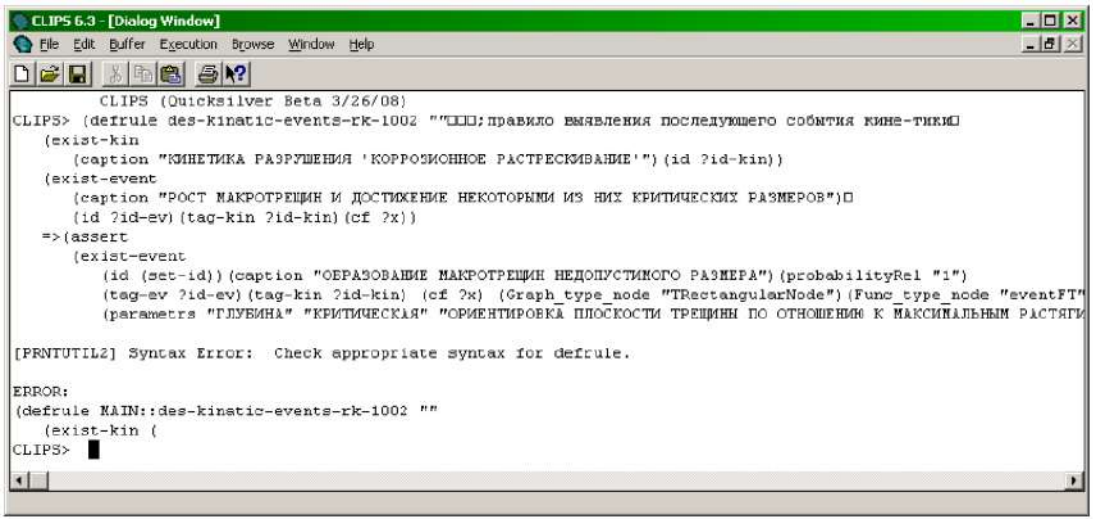


Рис.1.4.1. Пример интерфейса CLIPSWin

Visual JESS (Java Expert System Shell) [77] – средство поддержки программирования на Jess в форме генератора ЭС, реализующее с одной стороны доступ к конструкциям языка для программистов с цветовой подцветкой

синтаксиса, с другой стороны, с точки зрения методов EUD, использует метод диалогов-мастеров (Рис.1.4.2), которые при выполнении ряда операций позволяют абстрагироваться от конструкций языка программирования. Полученные программы могут быть использованы другими приложениями вне инструментальной системы. Отсутствует возможность интеграции с системами концептуального моделирования и CASE-средствами.

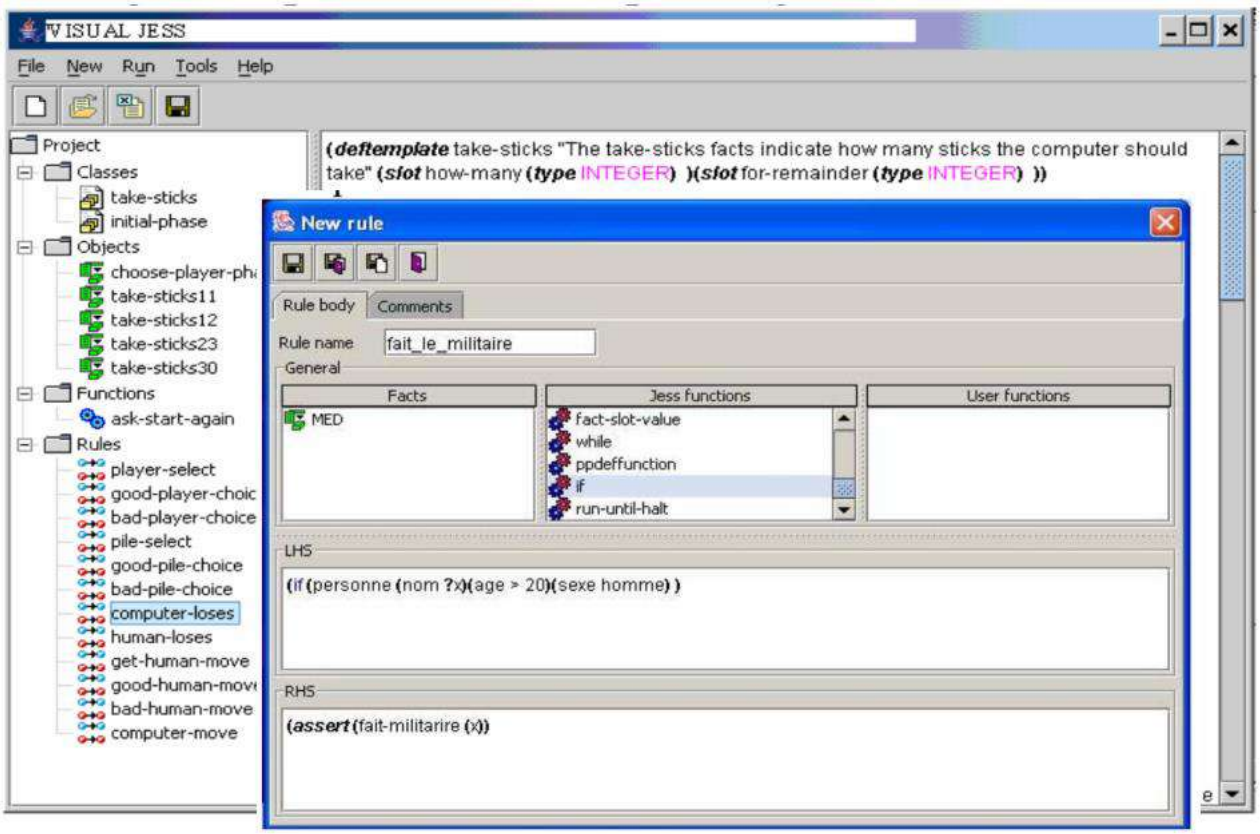


Рис.1.4.2. Пример интерфейса Visual JESS

Exsys (Expert System Development Tool) **Corvid** [66] – система разработки экспертных систем для непрограммирующих пользователей, ориентирована на язык представления знаний Exsys, базы знаний которого не отчуждаемы от программы оболочки. Используется механизм диалогов-мастеров, не интегрируется с системами концептуального моделирования (Рис. 1.4.3).

Visual Prolog [169] – среда разработки, ориентирвоанная на использование текстового языка программирования Prolog (Рис.1.4.4), включает следующие элементы: среду визуальной разработки (VDE - Visual Develop Environment), текстовый и графические редакторы, средства генерации кода, Prolog-компилятор, набор подключаемых файлов и библиотек, редактор связей, файлы, содержащие примеры и помощь.

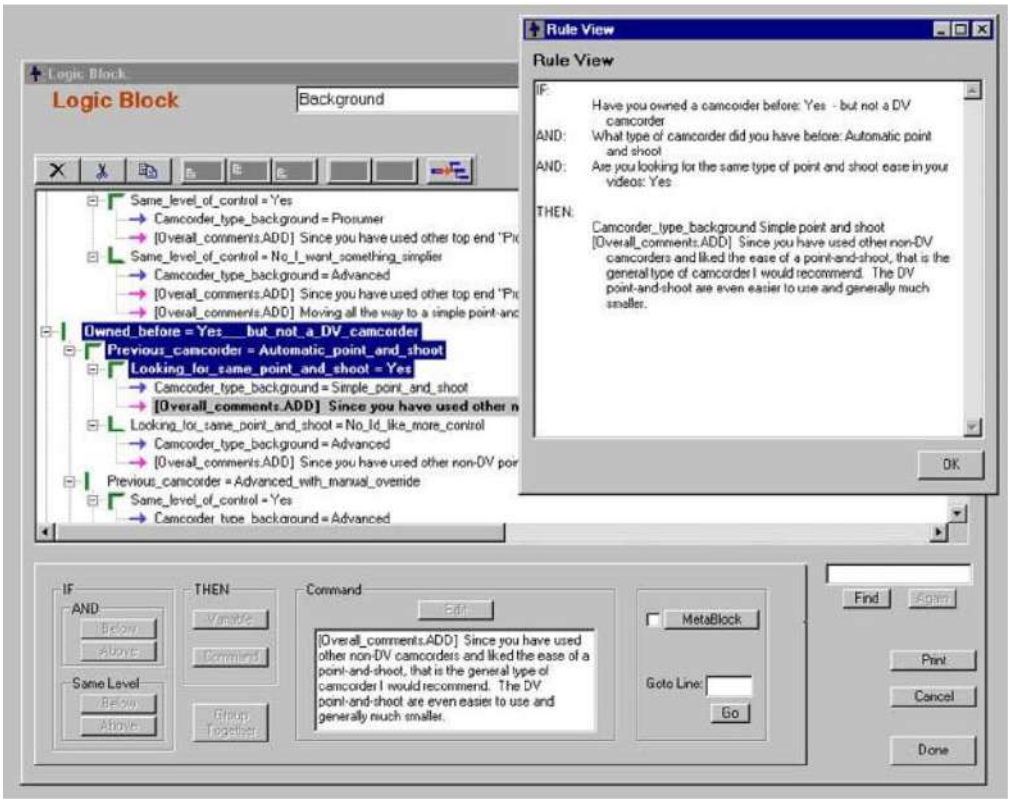


Рис.1.4.3. Пример интерфейса Exsys Corvid

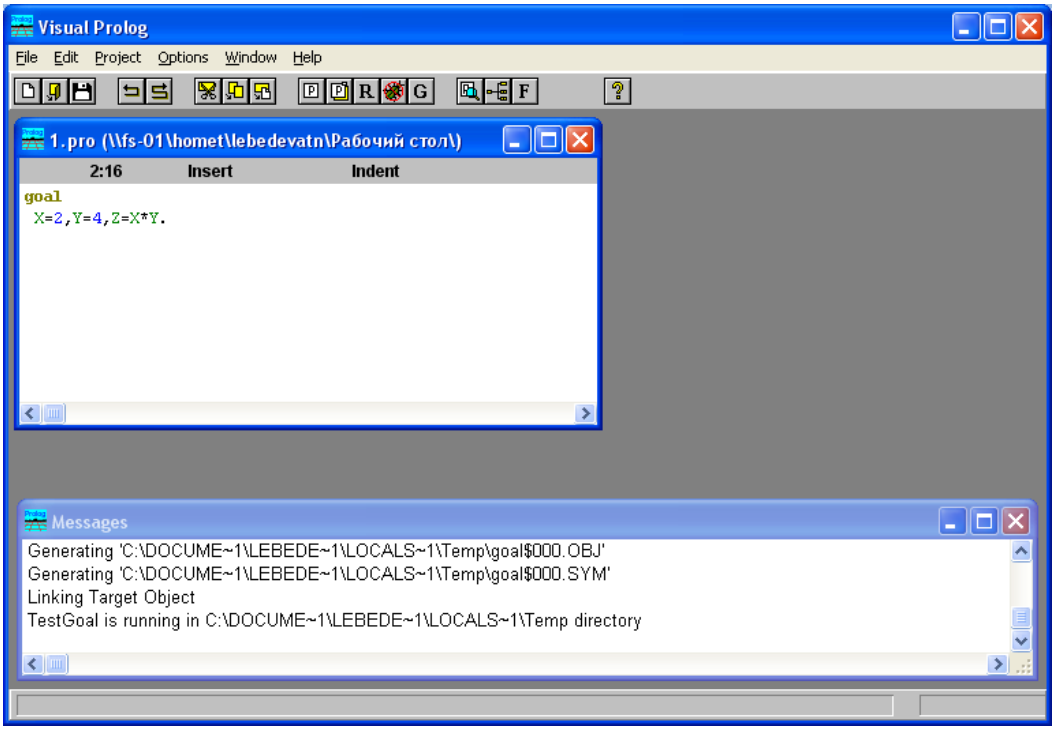


Рис. 1.4.4. Пример интерфейса Visual Prolog

Среда ориентирована на программистов. Полученные программы могут быть использованы другими приложениями вне инструментальной системы. Отсутствует возможность интеграции с системами концептуального моделирования и CASE-средствами.

Expert System Designer [65] – среда разработки, ориентированная на использование текстовых языков программирования CLIPS и JESS. Обеспечивает подцветку синтаксиса для языковых структур при редактировании программных кодов и использование диалогов-мастеров для создания основных элементов баз знаний. Среда в большей степени ориентирована на программистов. Полученные программы могут быть использованы другими приложениями вне инструментальной системы. Отсутствует возможность интеграции с системами концептуального моделирования и CASE-средствами. В настоящий момент система не поддерживается.

Expert System Creator [126] – среда разработки, ориентированная на непрограммирующих пользователей. Позволяет использовать визуальное представление логических правил в виде деревьев и таблиц решений. Позволяет синтегрировать полученные приложения в C++/Java приложения. Отсутствует возможность интеграции с системами концептуального моделирования и CASE-средствами.

ES-Builder Web [64] – веб-ориентированная среда разработки, ориентированная на непрограммирующих пользователей. Позволяет использовать визуальное представление логических правил в виде деревьев решений (Рис. 1.4.5). Создаваемые приложения не отчуждаются от среды разработки. Отсутствует возможность интеграции с системами концептуального моделирования и CASE-средствами.

Рис. 1.4.5. Пример интерфейса ES-Builder Web

VisiRule [166] – графический инструмент для проектирования, разработки и предоставления приложений для поддержки бизнес-правил и принятия решений на основе блок-схем (деревьев решений) (Рис. 1.4.6), представляющих логику принятия решений. Поддерживается генерация кода на Flex KSL. Возможность взаимодействия с функциями на Flex, Prolog, VB, C, C++. Отсутствует возможность интеграции с системами концептуального моделирования и CASE-средствами.

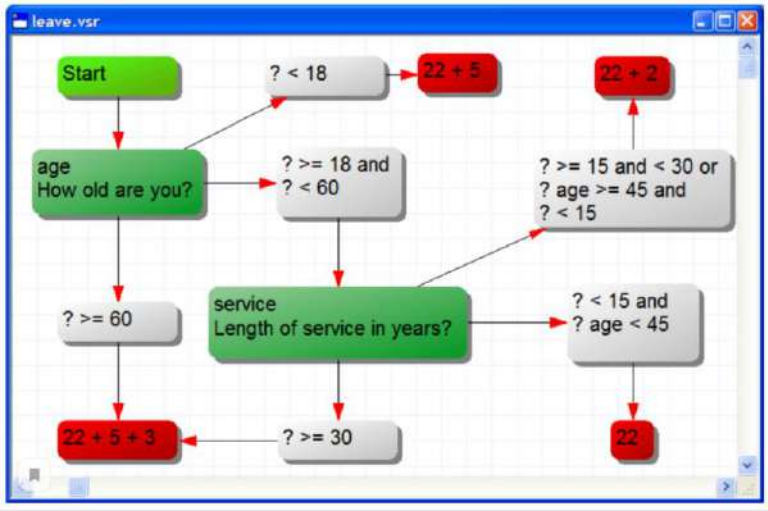


Рис. 1.4.6. Пример интерфейса VisiRule.

СИМПР-2015 [294, 295] – инструментальный комплекс, предназначенный для конструирования интеллектуальных систем поддержки принятия решений реального времени (ИСППР РВ) на основе продукционной модели представления знаний и специализированного табличного языка (Рис. 1.4.7).

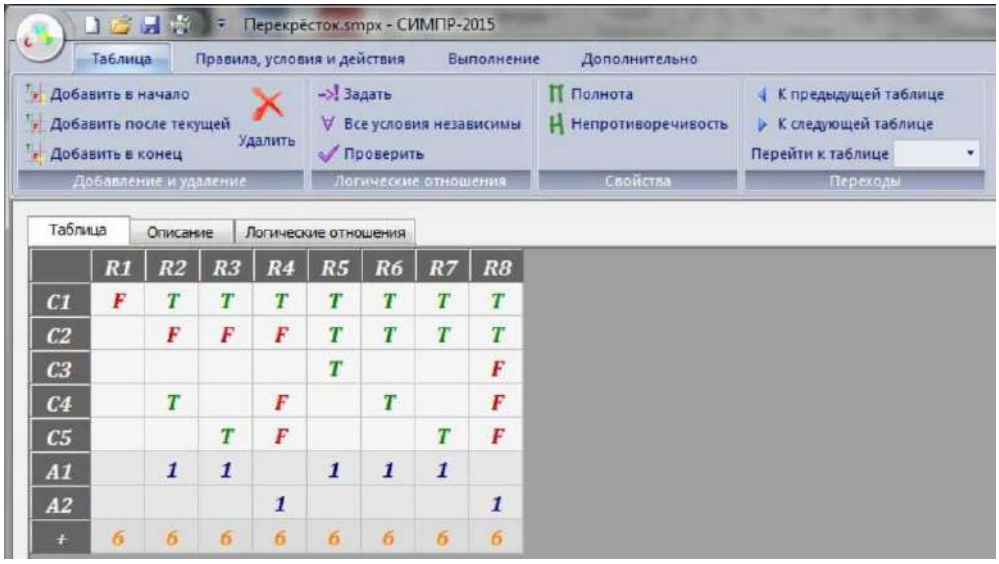


Рис. 1.4.7. Пример интерфейса СИМПР-2015

СИМПР-2015 разрабатывается на кафедре прикладной математики НИУ «МЭИ» под руководством Еремеева А.П., предназначен для создания ИСППР РВ для решения задач диагностики, мониторинга, планирования, прогнозирования. Создаваемые приложения не отчуждаются от среды разработки. Отсутствует возможность интеграции с системами концептуального моделирования и CASE-средствами.

ПРОБАЗ [302] - программный комплекс для создания баз знаний, экспертных систем и интеллектуальных систем поддержки принятия решений. Включающий следующие компоненты (приложения): конструктор базы знаний, эксперт базы знаний, интегратор базы знаний, Knowledge Base Query. Для описания правил используется текстовая форма, приближенная к естественному языку, а также таблицы решений (Рис. 1.4.8). Создаваемые приложения не отчуждаются от среды разработки, однако существует возможность взаимодействия с ними через API. Отсутствует возможность интеграции с системами концептуального моделирования и CASE-средствами.

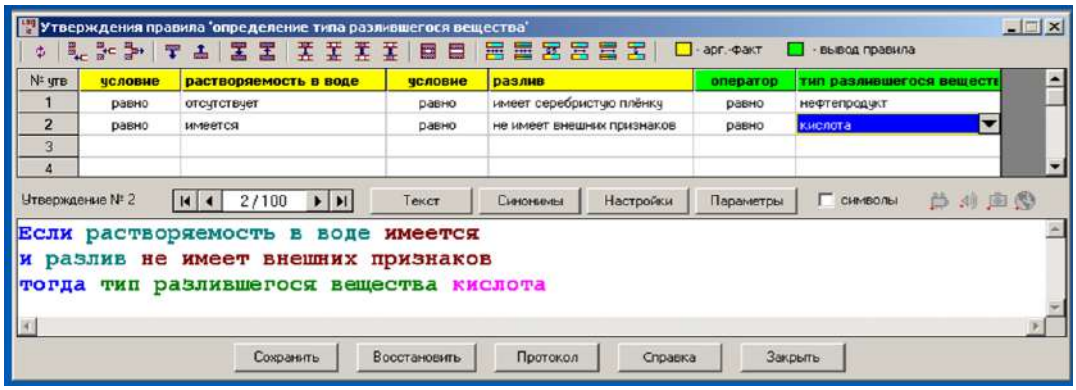


Рис. 1.4.8. Пример интерфейса ПРОБАЗ

В таблице 1.4.1 представлено сравнение рассмотренных систем.

Таблица 1.4.1. Сравнение рассмотренных программных инструментальных средств.

Программное средство / Критерий	Поддержка непрограммирующего пользователя	Поддержка визуального программирования	Генерация кодов	Проверка кодов	Генерация экспертных (интеллектуальных) систем	Интеграция с CASE-средствами	Использование других источников информации
ClipsWin	-	-	CLIPS	+	-	-	-
Visual JESS	+/-	-	JESS	+	+	-	-
Exsys Corvid	+	-	Exsys	+	+	+ XML	-
Visual Prolog	-	-	Prolog	+	+	-	-
Expert System Designer	+/-	+	CLIPS, JESS	+	+	-	-
Expert System	+	+ деревья и	CLIPS,	+	+	-	-

Creator		таблицы решений	JESS				
ES-Builder Web	+	+ деревья решений	-	+	+	-	-
VisiRule	+	+ диаграммы потоков (деревья решений)	Flex	+	-	-	-
СИМПР-2015	+	+ таблицы решений	-	-	+/-	-	-
ПРОБАЗ	+	+ таблицы решений	-	-	+/-	-	-
Разработанные автором средства: PKBD, WPKBD, KBDS, WPKBD	**+**	**+ RVML, таблицы решений**	**CLIPS, DROOLS, PHP**	**+**	**+**	**+**	**+ TabbyXL**

Среди рассмотренных программных инструментальных средств для создания ИС и БЗ только часть обеспечивает использование методов EUD и ориентированы на использование пользователями с низкими навыками программирования, при этом еще меньшее количество систем позволяют создавать отчуждаемые коды и интегрируются с внешними источниками информации в форме концептуальных моделей или таблиц.

Таким образом, подтверждается актуальность данного исследования в части разработки нового программного инструментария, реализующего принципы EUD и ориентированного на конечных (непрограммирующих) пользователей.

Выводы

Существующие технологии создания интеллектуальных систем обладают рядом недостатков: акцентом на концептуализацию, формализацию знаний и крупноблочное моделирование; поддержкой ограниченного набора языков программирования; поддержкой при проектировании преимущественно общесистемных моделей (например, UML); слабыми интеграционными возможностями, в том числе «по данным» с другим программным обеспечением и отсутствие возможности создания отчуждаемых программных модулей.

Выделенные недостатки обуславливают актуальность разработки новых методов и средств создания ИС с декларативными БЗ, ориентированной на конечных пользователей.

Анализ языков разработки декларативных БЗ показал необходимость создания новых средств визуального программирования баз знаний, учитывающих

особенности продукций и логических отношений и являющихся специализацией общесистемных нотаций, в частнности, UML.

В результате анализа программных инструментальных средств создания ИС и БЗ было установлено, что только часть из них обеспечивает использование методов EUD и ориентированы на конечных пользователей, при этом еще меньшее количество систем позволяют создавать отчуждаемые коды и интегрируются с внешними источниками информации в форме концептуальных моделей или таблиц. Таким образом, подтверждается актуальность разработки нового программного обеспечения, ориентированного на конечных пользователей и реализующего принципы модельно-ориентированного подхода в контексте инженерии знаний.

Глава 2. Модельно-управляемый подход и трансформации

В качестве основного, ориентированного на конечных пользователей (End-User Development, EUD) подхода для реализации новых методов и средств создания ИС и БЗ на основе модельных трансформаций, предлагается использовать модельно-управляемый подход [24, 38, 41, 43, 69, 72, 92, 109, 140, 144, 161, 254].

2.1 Основные понятия и определения

Модельно-управляемый подход (модельно-ориентированный подход) (Model Driven Engineering (MDE), Model Driven Development (MDD), Model Driven Software Development и т.д.) [24, 38, 41, 43, 69, 72, 92, 109, 140, 144, 161, 168] представляет собой подход к созданию программных систем, использующий в качестве артефактов процесса разработки различные концептуальные модели, которые используются для синтеза кодов, спецификаций и других моделей.

При этом под моделью понимается абстрактное представление программной системы, скрывающее информацию об определенных ее аспектах для упрощения восприятия пользователем и разработчиком. Модели визуализируются при помощи текстовых и графических языков, ориентированных на конечных пользователей.

Для описания основных элементов модели используется метамодель, как некая совокупность метаданных, определяющих язык описания моделей. Метамодель еще называю модель моделей [41]. В свою очередь для описания метамоделей используется мета-метамодель. Наиболее распространенными языками метамоделирования являются: MOF (Meta-Object Facility) [112], Ecore [60], KM3 (Kernel Meta Meta Model) [84] и др.

Некоторые исследователи включают MDE/MDD-подход в порождающее программирование (Generative Programming или Generative Software Development) [40]. Основные идеи MDE/MDD [24, 38, 41, 43, 69, 72, 92, 109, 140, 144, 161, 168]:

- модель – основной артефакт при создании программных систем;
- процесс создания программной системы – это цепочка модельных преобразований (трансформаций), где на каждом шаге происходит понижение абстрактности моделей, т.е. они дополняются деталями реализации.
- модельные преобразования (трансформации) описываются с использованием архитектуры (схемы метамоделирования),

включающей четыре разных уровня абстракции: M0-M3. При этом M0 – уровень объектов предметной области (реального мира); M1 – модели; M2 – метамодели; M3 – мета-метамодели.

MDE/MDD-подход можно считать дальнейшим развитием идеологии CASE-средств. Существует множество реализаций MDE/MDD, в том числе:

1) Архитектура, управляемая моделью (Model Driven Architecture, MDA) [69, 92, 103, 109, 161] – разновидность/реализация MDE/MDD от консорциума OMG и основанная на его основных стандартах, в частности: UML (Unified Modelling Language) – унифицированном языке моделирования; MOF (Meta Object Facility) – языке описания метамоделей; CWM (Common Warehouse Metamodel) – метамодели хранения данных; XMI (XML Metadata Interchange) – стандарте для обмена данными с использованием XML; Query/View/Transformation (QVT) – стандарте языков модельных трансформаций.

MDA рассматривает программную систему с трех точек зрения (default viewpoints) [69, 92, 103, 109, 161], соответствующим определенным моделям:

- вычислительно-независимой (Computation Independent Model, CIM) – описывающей требования к программной системе в части ее функций без учета структуры и бизнес-процессов, скрывая все технические детали реализации;
- платформо-независимой (Platform Independent Model, PIM) – описывающей функции и структуру без учета определенной технологической/программной платформы.
- платформо-зависимой (Platform Specific Model, PSM) – описывающей функции и структуру с учетом определенной технологической/программной платформы.
- модели платформы (Platform Model) – описывающей технические характеристики определенной технологической/программной платформы, например, в форме различных руководств.

2) Eclipse Modeling Framework (EMF) [59] – разновидность/реализация MDE/MDD, использующая программную платформу Eclipse, содержит средства создания метамоделей, их трансформаций и генерации спецификаций и кодов. Для описания метамоделей используется язык Ecore, а для их хранения – XMI.

3) Model-Integrated Computing (MIC) [108] – разновидность/реализация MDE/MDD основанная на использовании предметно-ориентированных языков моделирования (Domain-Specific Modeling Language, DSML) и оригинальной технологической платформы MIC.

В диссертационной работе в качестве основы предлагается использовать MDA подход, как наиболее документированный и стандартизованный из существующих разновидностей/реализаций MDE/MDD.

2.2 Модельные трансформации

Модельные трансформации, как область научных исследований, пересекаются с областью трансформаций программ [123, 297, 304, 305, 307] и считаются разновидностью последней. Считается, что в результате трансформации определенной модели может быть реализована трансформация программы, но только при условии, что сама программа основана на модели. Выделяют и отличия между этими направлениями: средства поддержки трансформации программ в большинстве случаев используют математически-ориентированные концепции, в частности: атрибутивные грамматики, переписывание (term rewriting), функциональное программирование, и др., в то время как при реализации модельных трансформаций преимущественно основаны на объектно-ориентированном подходе [40].

В модельных трансформациях в качестве основных объектов трансформаций (артефактов) рассматриваются разные схемы данных, программный код, характеристические модели (feature models) [87], спецификации интерфейсов, и др. В диссертационном исследовании основными артефактами являются концептуальные модели, представляющие собой совокупность элементов (понятий, их свойств и отношений), которые формируют смысловую структуру определенной предметной области. При этом учитываются классификации этих элементов по разным основаниям [206].

Концептуальные модели являются результатом моделирования предметной области, в том числе, при создании программного обеспечения, анализе бизнес-процессов и т.д. При их построении применяются различные языки и стандарты

(IDEF0, UML, BPMN и др.) как общего назначения, так и предметно- и проблемно-ориентированные (Domain-Specific Modeling Languages, DSL) [89, 159].

В рамках диссертации решается задача трансформации концептуальных моделей в контексте создания БЗ ИС, при этом делается допущение, что концептуальные модели описывают знания в явной форме, и требуется установить соответствие между этими знаниями и структурами БЗ, а затем – программными кодами и спецификациями.

В общем случае трансформация моделей – это процесс построения целевой (выходной) модели по исходной (входной) в соответствии с некоторым набором правил преобразования (трансформации). При этом под правилом преобразования подразумевается описание соответствия (отображения) конструкций на исходном (входном) и целевом (выходном) языках моделирования [43].

Основой для структурированного представления преобразования моделей в рамках MDA/MDE является архитектура или схема метамоделирования, описывающая иерархию моделей разной степени абстракции из четырех уровней от «M0» до «M3» [41, 69, 92, 103, 109, 161] (Рис. 2.2.1). Согласно данной схеме на уровне «M1», который является уровнем моделей, находится некоторая модель, подлежащая преобразованию (переводу) в другую модель при помощи специального оператора или программы трансформации. При этом уровень «M0» соответствует реальным данных, т.е. он отражает объекты и процессы реального мира (определенной предметной области). Этот уровень обычно опускают при визуализации схем метамоделирования. Все модели уровня «M1» строятся в соответствии с некоторыми метамоделями на уровне «M2», который является уровнем метамоделей, т.е. существуют специализированные формальные языки моделирования для их описания. Оператор или программа трансформации представляет собой совокупность правил преобразования, которые используют соответствующие элементы (понятия, свойства и отношения) метамоделей и формализуются с использованием специализированных языков описания трансформаций моделей (Model Transformation Language, MTL) [147], при этом все трансформации также соответствуют некоторой метамодели языка описания

трансформаций. Таким образом, трансформация моделей осуществляется на уровне метамоделей «M2».

В свою очередь, все метамодели уровня «M2» соответствуют некоторой мета-метамодели на уровне «M3» (уровень мета-метамодели), т.е. они также описываются с использованием некоторого языка метамоделирования. Стандарт MOF [112] является таким языком метамоделирования в рамках MDA-подхода [103], он выполняет функцию связующего звена между разными метамоделями, предоставляя основу для их формального описания. Т.е. если разные метамодели уровня «M2» соответствуют MOF и могут быть выражены средствами уровня «M3», то, следовательно, все основанные на них модели уровня «M1» могут сохраняться в общем репозитории и обрабатываться одними и теми же средствами поддержки модельных трансформаций.

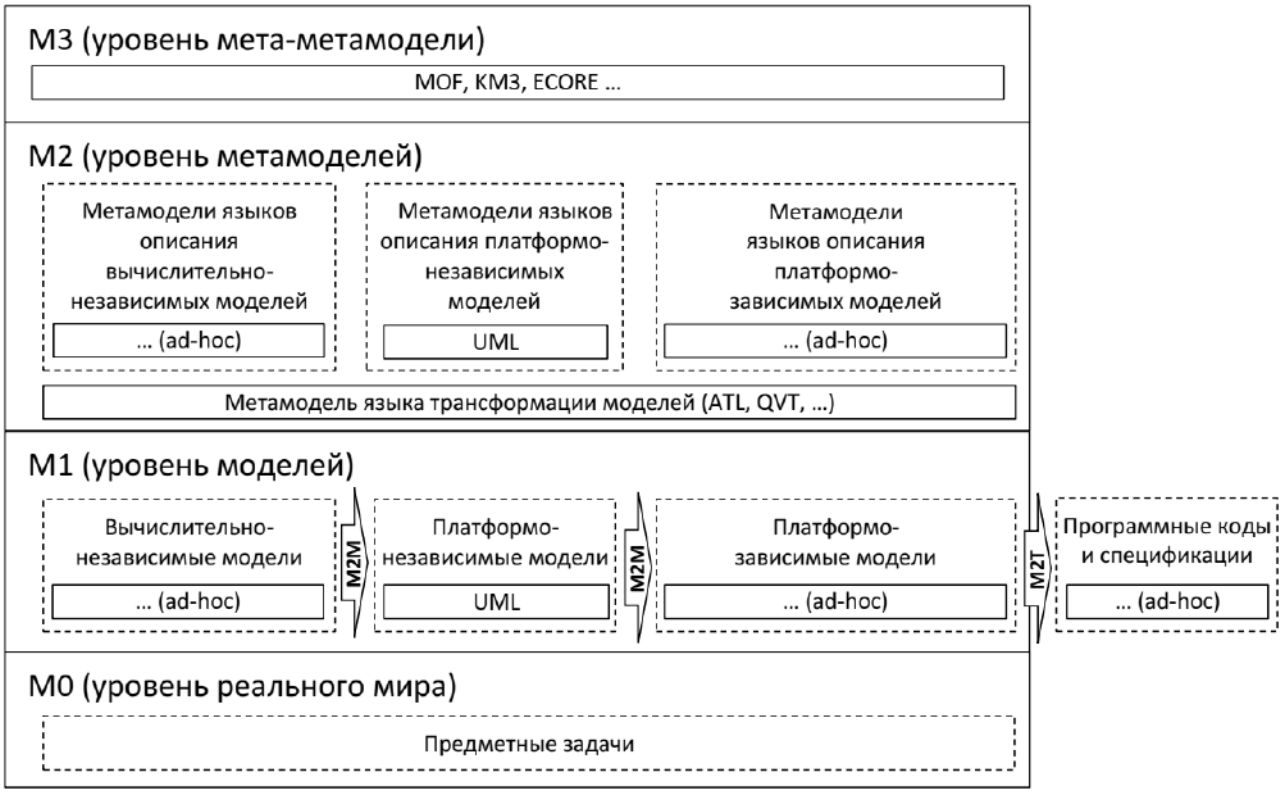


Рис. 2.2.1. Архитектура (схема) метамоделирования, описывающая трансформации моделей в рамках MDE/MDD/MDA-подхода на различных уровнях абстракции

Принято выделять следующие виды модельных трансформаций [40]:

1) Модель-Модель (Model-to-Model, M2M);

2) Модель-Текст (Model-to-Text, M2T или «pretty printing» – понятие из трансформации программ);

3) Текст-Модель (Text-to-Model, T2M).

Спецификации, исходный код, документация и т.д. также могут рассматриваться в рамках преобразования M2T в качестве целевых (выходных) текстовых артефактов, поэтому в литературе встречается его разновидность – Модель-Код (Model-to-Code, M2C),

Классификация модельных трансформаций по разным основаниям представлена в [107]. Наиболее популярные способы реализации трансформаций M2M и M2T подробно описаны в [40]. При этом в рамках каждого способа создаются специализированные языки для описания трансформаций, большая часть которых разрабатывается, совершенствуется и применяется в образовательной и исследовательской средах.

На настоящий момент наиболее популярными языками описания модельных трансформаций являются следующие (Таблица 2.2.1):

QVT (Query/View/Transformation) [128] – одна из разработок консорциума OMG, представляющая собой совокурность трех специализированных языков описания модельных трансформаций: декларативных Core и Relations, а также императивного Operational. Языки спецификации QVT применяются в MDA [103] для описания трансформаций M2M.

ATL (ATLAS Transformation Language) [84] – язык описания модельных трансформаций, использующий спецификацию QVT и язык описания объектных ограничений OCL (Object Constraint Language) [120], является универсальным и мощным средством, с помощью которого может быть выражено преобразование любой исходной модели в указанную целевую модель. Преобразование производится на уровне метамоделей «M2». Кроме того ATL представляет собой гибридный язык, так как интегрирует декларативный и императивный способы описания соответствий, что позволяет производить сложные трансформации.

VIATRA2 (VIsual Automated model TRAnsformations) [164] – язык преобразования для управления графовыми моделями, основанный на правилах и паттернах.

Таблица 2.2.1. Сравнение основных языков описания модельных трансформаций

Язык	Инициатива	Тип	Средство/ Среда поддержки	Ориентация на непрограммирую щего пользователя	Связанные языки, стандарты
QVT (Core, Relations, Operational)	MDA	декл., импер.	Eclipse	-	MOF, UML, XMI, OCL
ATL	EMF	декл., импер.	Eclipse	-	Ecore, MOF, QVT, UML, XMI, OCL
VIATRA2	GT (графовые трансформац ии)	декл. (GT), импер. (ASM)	VIATRA2 framework	-	VPM, VTML, VTCL, ASM
GReAT	GR (переписыван ие графов)	декл. + импер. (визуальные схемы)	Generic Modeling Environment (GME)	-/+ (визуальное програмиирование правил в виде схем)	Ecore, UML
Henshin	EMF	декл., импер.	Eclipse (GMF)	-	Ecore, MOF, SCM, Java
Epsilon Transformation Language (ETL)	EMF	импер.	Eclipse (Epsilon)	-	EOL, EGL, EVL, ECL, EML, EPL, Flock
XSLT	W3C	декл.	Altova, FreeFormatter, Online Toolz, Visual Studio	-	XML, HTML
Предлагаемый диссертантом язык - TMRL	**MDA**	**декл., импер.**	**KBDS**	**+ (визуальное программирован ие правил)**	**XML**

GReAT (Graph REwriting And Transformation) [6] – специализированный язык описания модельных трансформаций, использующий принципы подхода тройных трансформаций графа, при этом правила перезаписывания графа применяются к исходной (входной) модели с целью формирования результирующей (выходной).

Henshin [5] – еще один язык трансформации моделей, основанный на переписывании графов и использующий правила на основе шаблона, которые могут быть структурированы во вложенные единицы преобразования с четко определенной семантикой.

Epsilon [62] – семейство языков и инструментальных средств для реализации модельных трансформаций, проверки корректности моделей, генерации кода и спецификаций, миграций и рефакторинга. Основой Epsilon является императивный модельно-ориентированный язык Epsilon Object Language (EOL), который сочетает особенности процедурного стиля JavaScript и языка объектных ограничений OCL.

Epsilon Transformation Language (ETL) – еще один язык семейства Epsilon, представляет собой язык описания модельных трансформаций на основе правил, поддерживающий возможности: преобразования нескольких разных входных моделей в выходные, наследование правил трансформации, построение «ленивых» и «жадных» правил преобразования моделей, а также строить запросы к обрабатываемым моделям и изменять их.

XSLT (eXtensible Stylesheet Language Transformations) [178] – спецификация консорциума W3C, представляющая собой специализированный язык трансформации XML-документов. Таблицы стилей XSLT включают набор шаблонов, при применении которых к преобразуемому XML-документу, содержащему исходное дерево узлов, формируется новое выходное дерево, которое может быть представлено в форме HTML или XML-документа, а также текстового файла с произвольной структурой.

Следует также отметить, что программные средства поддержки рассмотренных языков в своем преобладающем большинстве не обеспечивают визуальное программирование правил трансформации моделей, которые создаются в специальных текстовых редакторах, ориентированных на программистов. Также в качестве существенного ограничения (недостатка) почти всех языков описания модельных трансформаций можно выделить жесткую привязку к определенному программному инструментарию, в частности, к платформе Eclipse [59], где поддержка языков реализована в виде модулей/плагинов расширения и EMF. Совокупность выделенных факторов затрудняет практическое использование рассмотренных языков и программных средств конечными пользователями (аналитиками, предметниками, инженерами по знаниям и т.д.), в частности, при разработке БЗ ИС и ЭС на основе модельных трансформаций.

Таким образом актуальна разработка нового языка и программного средства его поддержки, обеспечивающих создание программ трансформаций концептуальных моделей, важными свойствами которых являлось бы наличие человекочитаемого синтаксиса и реализация принципов визуального программрования с ориентацией на конечных пользователей.

2.3 Применение модельных трансформаций в разработке интеллектуальных систем

Можно выделить две основные группы работ направленных на решение задачи разработки БЗ и ИС на основе модельных трансформаций.

Первая группа рассматривает онтологии, как особый вид БЗ и использует для их получения различные концептуальные модели. Так в работе [91] проведено детальное сравнение и соотнесение конструкций диаграмм классов UML и онтологий OWL. В работах [110, 111] предлагается метамодельный управляемый подход к трансформации моделей для передачи (обмена) правилами между онтологиями OWL/SWRL и моделями UML/OCL. В качестве промежуточного представления знаний в виде правил используется язык – REWERSE Rule Markup Language (R2ML), а для описания правил трансформации использован язык трансформации моделей ATL. В работах [203, 204] исследователи представляют подход трансформации диаграмм классов UML в онтологии OWL 2 с использованием языка QVT-R. В [11] предложен подход преобразования диаграмм классов UML в онтологии OWL на основе графовых трансформаций и с использованием программного средства AToM3. В работе [122] описан подход TwoUse, позволяющий совместить моделирование UML и BPMN с семантическими технологиями OWL и SWRL. В рамках данного подхода разработан новый язык – SPARQLAS, как расширение языка SPARQL, позволяющий указывать специальные выражения ограничений и запросов. В работе [43] представлена метамодель ODM (Ontology Definition Metamodel) для описания онтологий OWL в рамках концепции четырехуровневой архитектуры MDA – MOF. ODM позволяет более широко использовать известную нотацию UML в онтологической инженерии, в частности, устанавливать отображения между элементами UML и OWL. Данная концепция ODM, а вместе с принципами MDD/MDA-подхода, также были использованы в работе [26]. В данном исследовании представлен подход для визуального моделирования онтологий OWL DL и OWL Full на основе преобразования концептуальных моделей UML Profile. Подход реализован в форме двух программных инструментов (плагинов) – Visual Ontology Modeler (VOM) и Integrated Ontology Development Toolkit (IODT) в составе платформы EMF. В работах [71, 114] описаны подходы к преобразованию

UML-моделей в OWL-онтологии с использованием XSLT. Другими успешными примерами решения задачи автоматизированной разработки БЗ, прежде всего в форме онтологий путем трансформации UML-моделей являются [37, 67, 68, 105, 180, 260, 320, 358].

Как отмечалось ранее, при разработке БЗ, в частности, онтологий, активно используются концепт-карты. Так, в [151] предложен метод интеграции концептуальных карт с онтологиями OWL, который представляет собой набор формальных преобразований, реализованных в средстве – MACOSOFT и применяемых к концептуальным картам в формате XML, а также семантического анализа отношений, связывающих понятия. В работах [25, 152] представлены подходы к трансформации концепт-карт CmapTools в онтологию предметной области в формате OWL.

Во всех рассмотренных работах делается акцент на конкретных языках и нотациях моделирования, при помощи которых описываются исходные концептуальные модели. Построение правил соответствий между элементами исходной концептуальной моделью и целевой БЗ осуществляется на уровне их концептов и отношений, т.е. на абстрактном уровне (abstract syntax). При этом, как правило, опускается информация о конкретном текстовом синтаксисе (concrete syntax) представления данных моделей.

Вторая группа ориентирована на анализ и преобразование XML-спецификации, рассматривая последние как универсальный и наиболее распространенный способ интеграции программных систем и обеспечения обмена информацией между приложениями. Большинство концептуальных моделей предметных областей сериализуются в виде XML-документов, например, для обмена UML-моделями используется XMI (XML Metadata Interchange) [176].

Таким образом, задача преобразования исходных концептуальных моделей в целевые БЗ может быть рассмотрена только с учетом особенностей конкретного текстового синтаксиса данных моделей. В [23] предлагают подход и программное средство XML2OWL для преобразования XML-моделей данных в онтологии OWL. Трансформация осуществляется на уровне XML-схемы модели данных, с сопоставлением элементов XSD (XML Schema Definitions) и OWL. Для реализации трансформаций используется XSLT. В работах [134, 135] представлен подход для

отображения XML-схем в терминологическую часть онтологий OWL (уровень иерархии понятий – TBox) и автоматического перевода элементов XML-документа в конкретные онтологические объекты OWL (уровень утверждений – ABox). Подход реализован на Java в виде фреймворка JXML2OWL для автоматического преобразования различных источников данных в формате XML в семантическую модель, определенную на языке OWL. В [118] предлагают предметно-ориентированный язык – XMLMaster, предназначенный для преобразования XML-документов в произвольные онтологии в формате OWL. Язык основан на Манчестерском синтаксисе OWL и его расширении с помощью языка запросов XPath для работы с XML-документами. В [10] представлен набор шаблонов (паттернов) для прямого автоматического преобразования XML-схем в онтологии OWL 2 RL, позволяя тем самым интегрировать множество данных, представленных в формате XML в семантическую сеть. При этом подход предусматривает нормализацию, фильтрацию и трансформацию XML-данных. Подход реализован на Java в виде программного средства – Janus.

В Таблице 2.3.1 приведена краткая сравнительная характеристика всех рассмотренных работ в области создания БЗ на основе трансформации различных концептуальных моделей, включая работы по преобразованию XML-данных и интеграции БЗ.

Таблица 2.3.1. Сравнение работ, описывающих разработку баз знаний на основе трансформации концептуальных моделей

Работа	Исходные концептуальные модели	Целевые БЗ	Используемые методологии (стандарты)	Использованные языки и подходы к трансформации	Реализация
[110, 111]	UML/OCL и R2ML	R2ML и OWL DL/SWRL	MDA/MOF, EMF/ODM (промежуточное представление R2ML)	ATL	Плагин для платформы EMF
[203, 204]	UML	OWL 2	MDA/MOF	QVT-R	Ad-hoc
[11]	UML	OWL DL	MDE/MDA	Трансформация графов	AToM3
[122]	UML/OCL и BPMN	OWL DL/ SWRL	Ad-hoc + EMF (MOF/Ecore)	SPARQLAS	TwoUse Toolkit
[43]	UML	OWL DL	ODM	Ad-hoc	-
[26]	UML Profile	OWL DL и OWL Full	MDA/ODM	QVT	Плагины VOM и IODT для EMF
[114]	UML	OWL	Ad-hoc	XSLT	Ad-hoc
[260, 320]	UML	OWL	Ad-hoc	Ad-hoc	Ad-hoc
[358]	UML	OWL DL/ SWRL	Ad-hoc	Ad-hoc	Ad-hoc

[358]	UML	OWL DL/ SWRL	Ad-hoc	Ad-hoc	Ad-hoc
[180]	UML	OWL DL	Ad-hoc	Ad-hoc (алгоритм U2OTrans) и XSLT	UML2OWL
[67, 68]	UML	DAML+OIL	Ad-hoc	Ad-hoc	Модуль CAWICOMS
[105]	UML	OWL	Ad-hoc	Ad-hoc	Ad-hoc
[37]	UML (XMI)	RDF	MDA	Ad-hoc на основе API JMI	Ad-hoc
[132]	SBVR	OWL и SWRL	Ad-hoc	Ad-hoc на основе набора шаблонов	Ad-hoc
[3]	ORM	OWL 2	Ad-hoc (2 этапа отображения)	XSLT	NORMA и ORM2OWL
[163]	MeSH и WordNet	RDF(S) и OWL	Ad-hoc	XSLT и Prolog	Ad-hoc
[113]	ER	OWL Lite	Ad-hoc	Ad-hoc	SFSU ER Design Tools
[344]	УФО-модели	RDF	Ad-hoc	Ad-hoc	Ad-hoc
[151]	Концепт-карты XML (на основе WordNet)	OWL Lite (DL частично)	Ad-hoc	Ad-hoc	Ad-hoc
[25]	CmapTools	OWL DL	Ad-hoc	Prolog	Translator Module
[152]	CmapTools	OWL	Ad-hoc	Ad-hoc	Ad-hoc
[23]	XML-схема и объекты	OWL DL	Ad-hoc	XSLT	XML2OWL
[134, 135]	XML-схема	OWL DL	Ad-hoc	XPath	JXML2OWL
[118]	XML-схема и объекты	Manchester OWL syntax	Ad-hoc	XPath	XMLMaster
[10]	XML-схема	OWL 2 RL	Ad-hoc	Ad-hoc	Janus
[157]	DTD и XML-объекты	OWL DL	Ad-hoc	XSLT	DTD2OWL
[4]	XML-схема	OWL DL	Ad-hoc	Ad-hoc	Ad-hoc
[181]	XML-схема (XSG)	OWL DL	Ad-hoc (цепочка средств)	Ad-hoc	Trang, XSOM, JUNG и Jena
[8]	RDF	CLIPS/ COOL	Ad-hoc	Ad-hoc и XSLT	R-DEVICE
[104]	OWL DL	CLIPS/ COOL	Ad-hoc	Ad-hoc	CLIPS-OWL
[63]	OWL DL	JESS	Ad-hoc	Ad-hoc	Плагин JessTab для Protégé
[106]	OWL DL	JESS	Ad-hoc	Ad-hoc и XSLT	OWL2JESS
[172]	SWRL	JESS	Ad-hoc (4-х этапный подход)	Ad-hoc	Плагин SWRLTab для Protégé и средства SweetRules
[96]	OWL	Prolog	Ad-hoc (+ доп. преобразования)	Ad-hoc	SweetProlog

На основе сравнительного анализа данных работ можно выделить следующие их недостатки:

1) Узкая специализация в части поддержки форматов (языков моделирования) концептуальных моделей (как правило, это один язык, например,

UML). Вследствие чего отсутствует гибкость при разработке БЗ, т.к. пользователь при необходимости не сможет трансформировать концептуальные модели других форматов. Исключением здесь могут быть работы [10, 23, 118, 134, 135, 157, 181], т.к. они направлены на трансформацию различных концептуальных моделей, конкретный синтаксис которых представлен в виде XML.

2) Узкая специализация в части поддержки форматов ЯПЗ. В большинстве случаев исследователи используют онтологии в качестве целевой модели представления знаний и, как правило, язык OWL для их описания. При этом данный формат поддерживается не полностью. Например, поддержка только конкретного диалекта (так в ряде работ используется устаревшая спецификация OWL Lite) или определенной версии OWL (1 или 2).

3) Жесткое задание соответствий между элементами исходной концептуальной модели и БЗ на целевом ЯПЗ при реализации. Вследствие чего отсутствует возможность изменить интерпретацию отображений элементов без внесения изменений в программный код соответствующего программного средства.

4) Достаточно сложная реализация, что, в свою очередь, обуславливает разнообразие программного обеспечения, используемого исследователями в рамках каждого отдельного преобразования (иногда на каждом этапе преобразования используется отдельное специализированное средство).

5) Использование разных методологий (подходов) к решению поставленной задачи: от так называемых «ad-hoc решений», пригодных для конкретного описываемого случая (конкретной задачи), до различных общих стандартов в области программной инженерии и модельно-ориентированного подхода, в том числе разных инициатив MDE/MDD.

6) Высокие квалификационные требования к пользователям. Как правило, существующие подходы ориентированы на программистов и инженеров по знаниям, а не на экспертов предметной области.

Таким образом перспективна разработка технологии разработки БЗ, ориентированной на конечного пользователя, поддерживающей возможность

использования широкого набора концептуальных моделей и целевых ЯПЗ, что может быть обеспечено за счет использования принципов стандартизированного модельно-ориентированного подхода, в том числе, модельных трансформаций.

Помимо разработки БЗ различного типа на основе модельных трансформаций, также в научной литературе встречается описание применения EUD подходов, в большей или меньшей степени использующих принципы MDA/MDE для разработки экспертных (интеллектуальных) систем. Анализ этих работ показал, что их можно разделить на две группы:

1. В **первую группу** входят работы, в которых явно не указано использование методологии и принципов MDE, однако они фактически используют концептуальные модели для описания предметной области и некоторые преобразования (трансформации) этих моделей для интерпретации или генерации программных кодов. В этом случае специализированные языки преобразования не используются, и основные результаты представлены либо в виде конкретных приложений, либо в форме проблемно-ориентированных оболочек.

В частности, в [58] представляет метод автоматического создания веб-ориентированных экспертных систем на основе XML-описаний знаний предметной области. В качесте решаемой задачи рассмотрены правила выбора и прохождения университетских курсов. В работе используется концепция генератора, а метод ориентирован на программиста. В [117] описывается инструментарий для разработки веб-ориентированных экспертных систем на основе технологию Semantic Web, которая позволяет инженерам по знаниям и экспертам в предметной области определять знания без глубоких знаний о языках программирования и искусственном интеллекте (ИИ). Факты знаний могут быть аннотированы с использованием семантических понятий и отношений, найденных в онтологии WordNet, и интерпретированы в инструментарии. В [149] используется онтология для моделирования знаний предметной области и правил принятия решений. Описанная система обеспечивает интеграцию Protege в качестве базы предметных знаний и Java Expert System Shell (JESS) в качестве операционной базы знаний в

единую экспертную систему. Прикладная задача – определение корпоративного финансового рейтинга. В этой работе реализована концепция генератора, поэтому основными результатами являются программные коды JESS и Java. Специальные языки модельных трансформаций не используются. В [136] описан инструмент разработки ЭС для экспертов, не связанных с ИИ. Предлагаемый инструмент позволяет разрабатывать ЭС на основе моделей представления знаний. Модели описываются в виде деревьев и интерпретируются в инструменте. Специальные языки трансформаций моделей не используются. В [160] описан подход к построению ЭС на основе UML, который использует расширение CLIPS, называемое VCLIPS_UML. Расширение разработано на Java и позволяет автоматически генерировать соответствующие сценарии в соответствии с языком CLIPS. Этот подход ориентирован на непрограммирующих пользователей. В [86] представлено инструментальное средство для построения продукционных ЭС для задач диагностики - Diagnosis Domain Tool for Rule-based Expert System (DDTRES). Эта система разработана и реализована с использованием языка программирования Visual Prolog, представляет собой проблемно-ориентированную оболочку, структура модели предметной области уже определена и заполнена с использованием методов интеллектуального анализа данных.

2. **Вторая группа** содержит работы, в которых явно используются принципы MDE в контексте инициатив EMF или MDA. Например, в [31] представляют MDE для разработки веб-приложений, использующих продукции. Предлагаемый подход использует онтологию для описания предметной области и специальный язык Conceptual Modeling Language (CML) вместо UML для описания онтологии (как вычислительно-независимых (CIM) и платформо-независимых (PIM) моделей) в качестве формализма моделирования правил. Реализация выполнена в рамках проекта моделирования Eclipse (с использованием платформы моделирования Eclipse); соответственно, преобразования моделей (модель-модель и модель-код) описываются с помощью языка преобразования ATLAS (ATL). CIM и PIM не разделены, и выбор возможного формализма представления знаний не предлагается. Онтология и правила преобразуются в JESS.

В [32] описан подход к созданию продукционных систем на основе концепции EMF. В частности, мета-метамодель Ecore используется для описания vетамодели правил, которая определяет концептуальную модель представления знаний эксперта предметной области в форме правил JESS, но отсутствует четкое описание вычислительно-независимой (Computation-Independent Model, CIM) и платформо-независимой (Platform-Independent Model, PIM) моделей, а также платформо-зависимой модели (PSM). В целом реализован принцип генератора для платформы JESS (ориентированной на программистов). В [29] предлагается внедрение MDA для платформы PRISMA. Этот инструмент позволяет генерировать диагностические ESS (в виде архитектурных моделей PRISMA) на основе концептуальных моделей, описывающих различные аспекты программного обеспечения: модель функций, дерево решений, концептуальная модель предметной области, концептуальная модель предметной области и т.д. В этом случае первые две модели рассматриваются как CIM, а остальные - как PIM. Для реализации модельных трансформаций не используются специализированные языки. Основными результатами являются сгенерированные программные коды для C# и .NET.

Сравнительный анализ примеров применения модельных трансформаций при создании интеллектуальных систем (Таблица 2.3.2) показал отсутствие единой унифицированной технологии разработки программных систем, сложность интеграции полученных решений и их ориентацию на программирующих пользователей.

Таблица 2.3.2. Сравнение рассмотренных работ, использующих модельные трансформации для создания интеллектуальных систем (сокращения: ПО – предметная область; МПО – модель предметной области)

Критерий/Работа	[58]	[117]	[149]	[136]	[160]	[86]	[31]	[32]	[29]	**Разрабатываемая диссертантом технология**
Концепция реализации: Г - генератор И - интерпретатор	Г	И	Г	И	Г (для БЗ), И	И	Г	Г	Г	**Г (для БЗ), И**
Используемые концептуальные модели	XML	Онтология WordNet XML	Онтология (OWL, Protege)	Деревья	UML	-	Онтология	Модели правил	Концептуальные модели	**Онтология (OWL), UML, концепт-карты, деревья, таблицы, диаграммы Исикавы**
CIM	-	-	-	-	-	-	Концептуальные модели на Conceptual Modeling Language (CML)	-	Модель свойств, деревья решений	**OWL, UML, концепт-карты, деревья, таблицы, диаграммы Исикавы**
PIM	-	-	-	-	-	-		-	МПО, Прикладные МПО	**UML RVML**
PSM	-	-	-	-	-	-	Java, JSF Web, JESS	-	PRISMA	**RVML**
Платформа	HTML, Perl, Prolog	Собственная	JESS Java	Собственная	CLIPS	Prolog	JESS, Java	JESS, Java	PRISMA C# .NET	**CLIPS, DRL, PKBD, PHP**
Методология (подход)	Собственный	Собственный	Собственный	Собственный	Собственный	Собственный	MDD, 2 трансформации	MDD, 1 трансформация	MDD, много трансформаций	**MDD + Инженерия знаний 3 трансформации**
Инициатива	-	-	-	-	-	-	EMF	EMF	MDA	**MDA**
Универсальность	Нет. ПО: Университетские курсы	Да. Оболочка	Нет. ПО: Финасовый рейтинг корпораций	Да. Оболочка	Да. Оболочка	Нет. Диагностическая оболочка	Да	Да	Нет. ПО: Диагностика	**Да. Оболочка**
Используемые специальные языки и стандарты трансформаций	- Частное решение	- Частное решение	- Частное решение	- Частное решение	- Частное решение	- Частное решение	+ ATL	- ECore	- Частное решение	+ **Частное решение, TMRL**
Ориентация на непрограммистов	-	+	-	+	+	+	-	-	+	+

Выводы

При наличии специализированных методов и программных средств, повышающих эффективность разработки ИС с декларативными БЗ продукционного и прецедентного типа, перспективным является использование EUD (End-User Development) подходов, ориентированных на конечных пользователей.

Среди EUD подходов наиболее актуальными являются: модельно-ориентированный (управляемый) подход, в частности, его стандартизированная разновидность – MDA (Model-Driven Architeture), а также использование шаблонов и диалогов-мастеров.

Высокие квалификационные требования к пользователю (специалисту) при разработке правил (сценария) трансформаций на существующих языках трансформаций обуславливает актуальность создания нового языка описания модельных трансформаций концептуальных моделей и программных систем его поддержки, ориентированных на конечных пользователей.

Существующие решения в области применения модельных трансформаций для создания интеллектуальных систем не объединены единой унифицированной технологией разработки и не ориентированы на программирующих пользователей.

Данные выводы в совокупности с выводами по главе 1 подтверждают актуальность разработки новых методов и средств создания программного обеспечения ИС с декларативными БЗ на основе модельных трансформаций, ориентированных на пользователей с низкими навыками программирования. включая:

- оригинальные языки визуального программирования декларативных БЗ и описания трансформаций концептуалтных моделей, ориентированные на конечных пользователей;
- методы создания интеллектуальных систем и декларативных баз знаний, использующие разработанные языки;
- программные средства для поддержки новых языков и методов.

Проведённый анализ позволяет сделать вывод об актуальности диссертационного исследования по разработке методов и средств создания ИС с декларативными БЗ на основе модельных трансформаций, включая:

- оригинальные специализированные языки визуального программирования декларативных баз знаний и описания трансформаций концептуалтных моделей, ориентированные на конечных пользователей;
- новые методы создания интеллектуальных систем и декларативных баз знаний, использующие разработанные языки;
- комплекс взаимосвязанных оригинальных программных средств для поддержки языков и методов (Рис.2.1), включающий:
  - Personal Knowledge Base Designer (PKBD) [184, 185, 195, 202, 257, 258, 269, 283, 367] – систему программирования, реализующую предлагаемый метод создания ИС и БЗ. Основное назначение: интеграция с системами концептуального моделирования; создание БЗ; генерация кодов и спецификаций, интерпретация (прогон) и верификация БЗ; создание проблемно-ориентированных редакторов;
  - Web PKBD – веб-версию PKBD, реализующую возможность создания БЗ; генерации кодов и спецификаций, интерпретацию и верификацию БЗ;
  - инструментальное программное средство – Knowledge Base Development System (KBDS) [281, 283, 366], реализующее предлагаемый метод автоматизации процесса создания программных компонентов-конверторов и моделей трансформаций. Основное назначение: создание БЗ на основе трансформации концептуальных моделей, генерация их кодов и создание компонентов трансформации моделей;
  - TreeEditorET/Extended Event Tree Editor [224, 287] – системы визуального программирования БЗ на основе деревьев событий.

Созданные программные средства объединены общей идеологией модельных трансформаций и формируют технологическую платформу создания интеллектуальных систем. Интеграция программных средств реализуется путем использования единого формата межпрограммного обмена данными и знаниями,

основанного на XML – EKB (External Knowledge Base); а также взаимодействия через REST API. Далее элементы технологии рассмотрены подробнее.

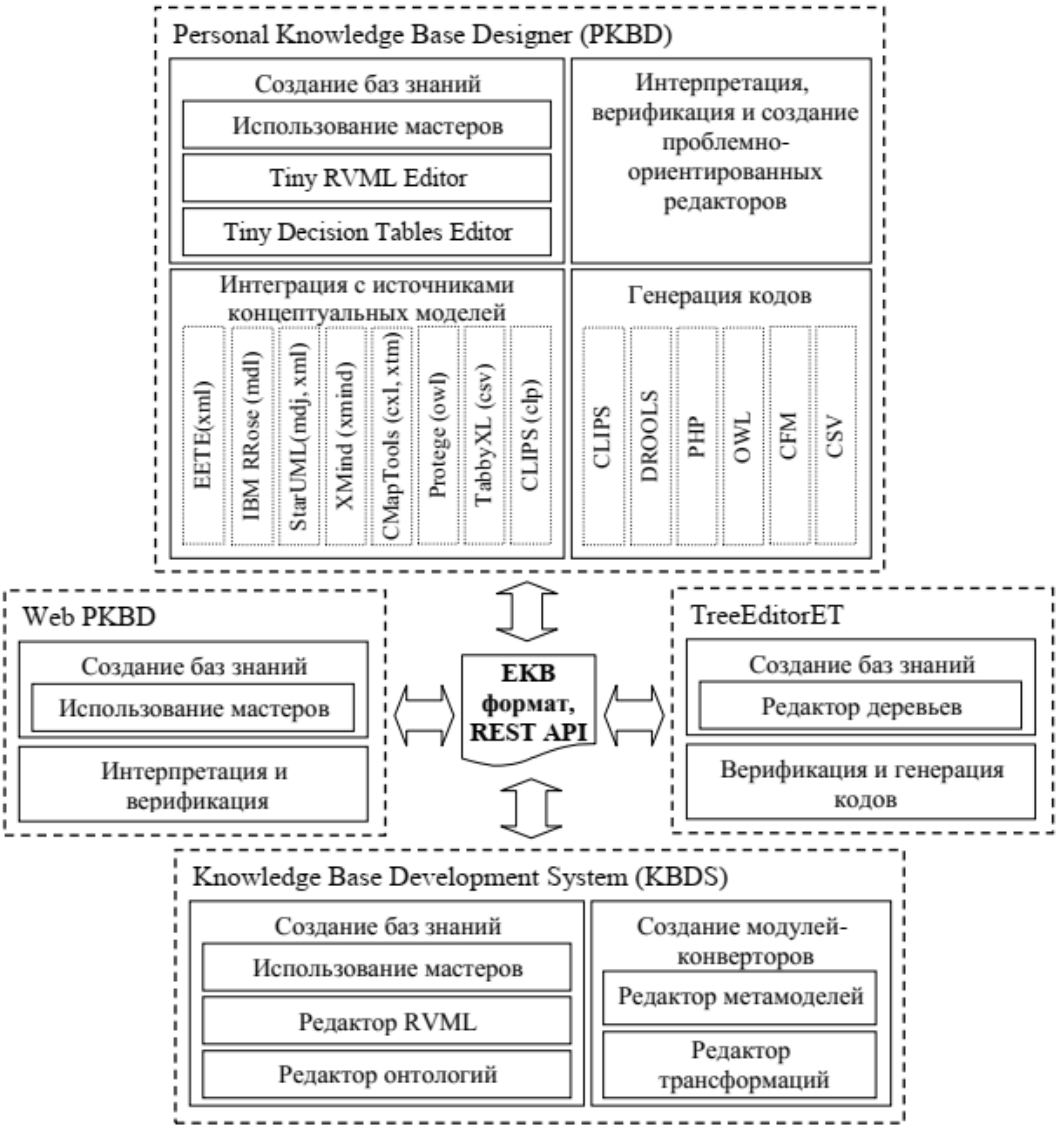


Рис. 2.1. Структура комплекса программных средств

Глава 3. Новые языки программирования

В рамках диссертационного исследования были разработаны следующие оригинальные языки:

- язык визуального программирования декларативных баз знаний продукционного типа – Rule Visual Modeling Language (RVML);
- язык программирования трансформаций концептуальных моделей – Transformation Model Representation Language (TMRL).

3.1 Язык визуального программирования декларативных баз знаний - Rule Visual Modeling Language

Для визуального программирования декларативных баз знаний и генрации программных кодов был разработан специальный язык Rule Visual Modeling Language (RVML) [52, 184, 185, 198, 279, 368]. Язык ориентирован на использование формализма продукций (логических правил типа «ЕСЛИ-ТО») с возможностью кодогенерации. В качестве основы для разработки языка был использован UML, в частности, диаграммы классов. Поэтому RVML может рассматриваться как его профиль расширения с использованием модифицированных базовых элементов «класс» и «ассоциация». RVML позволяет представить логические правила в обобщенном виде, абстрагируясь от особенностей конкретных языков программирования. В то же время он содержит средства для определения приоритетов правил и значений слотов «по умолчанию».

RVML имеет расширение, называемое FuzzyRVML [49, 52, 285, 286, 288], поддерживающее использование лингвистических (нечетких) переменных и коэффициентов уверенности для учета нечеткости и неопределенности в рассуждениях. Значение лингвистической переменной определяется с помощью, так называемых нечетких множеств [57]. Нечеткое множество определяется через базовую шкалу (набор базовых значений) и функцию принадлежности $\mu(x)$, которая определяет, как каждая точка в диапазоне соотносится со значением принадлежности (или степенью принадлежности) в интервале [0, 1]. Таким образом, функция принадлежности определяет субъективную степень уверенности эксперта в том, что конкретное значение базовой шкалы соответствует определенному нечеткому множеству. Существует два способа задания функции принадлежности: табличный и аналитический. Определены следующие типы описания функции принадлежности для аналитического метода: треугольная; трапециевидная; S-образная; Z-образная; U-образная и др.

3.1.1 Формальное описание RVML

Представим в РБНФ описание основных элементов RVML:

```
     RVML = {Fact}, {Template}, {Rule}, {Relationship}, {FuzzyVar},
{GeneralizedRule}.
     Rule = (Condition {Condition}), Core, (Action {Action}).
     Condition = ConditionElement {ConditionElement}.
     Action = ActionElement {ActionElement}.
     ConditionElement = ConditionOperator, Fact {Fact} |
ConditionOperator, ConditionElement.
     ActionElement = ActionOperator, (Fact {Fact}).
     ActionOperator = "Add" | "Delete" | "Modify" | "Stop".
     ConditionOperator = "AND" | "OR" | "NOT".
     Fact = Name, CertaintyFactor, Slot {Slot}.
     Slot = Name, Constraint, Value, [DataType], [Term].
     Core = Name, CertaintyFactor, Priority.
     Name = String.
     Value = Number | String | Set.
     String = Symbol {Symbol}.
     Symbol = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
"J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U"
| "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d" | "e" | "f" |
"g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r"
| "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z" | "А" | "а" | "Б" |
"б" | "В" | "в" | "Г" | "г" | "Д" | "д" | "Е" | "е" | "Ё" | "ё" | "Ж"
| "ж" | "З" | "з" | "И" | "и" | "Й" | "й" | "К" | "к" | "Л" | "л" |
"М" | "м" | "Н" | "н" | "О" | "о" | "П" | "п" | "Р" | "р" | "С" | "с"
| "Т" | "т" | "У" | "у" | "Ф" | "ф" | "Х" | "х" | "Ц" | "ц" | "Ч" |
"ч" | "Ш" | "ш" | "Щ" | "щ" | "Ъ" | "ъ" | "Ы" | "ы" | "Ь" | "ь" | "Э"
| "э" | "Ю" | "ю" | "Я" | "я"| "0" | "1" | "2" | "3" | "4" | "5" | "6"
| "7" | "8" | "9" .
     Set = Value ";" {Set}.
     Constraint = ">" | "<" | "=" | ">=" | "<=" | "<>".
     CertaintyFactor = [0,1].
     Priority = [1,100].
     Template = Name, Slot {Slot}, FuzzyVar {FuzzyVar}.
     DataType = "String" | "Number" | "Fuzzy".
     Relationship = Element, Connection, Element.
     Connection = Kind, Name.
     Kind = "Association" | "Dependenses".
     Element = Condition | Action | Template | Core | FuzzyVar | Term
| Fact.
     Term = Name, FunctionType, TermValue {TermValue}.
     TermValue = Value, Probability.
     Probability = [0,1].
     FuzzyVar = Name, Term {Term}, Domain, Units, FunctionType.

     FunctionType = Tabular | Analytical.
     Tabular = "T".
     Analytical = "A".
     Domain = "[" Number {Number} "," Number {Number} "]".
     Number = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" |
"9".
     Units = String.
     GeneralizedRule = (Template {Template}), Core, (Template
{Template}).
```

3.1.2 Метамодель RVML/FuzzyRVML

Разработанный язык является средством описания логических правил декларативных баз знаний в рамках предлагаемого подхода. В соответствии с основными положениями MDA/MDE были разработаны метамодели RVML/FuzzyRVML (Рис. 3.1.1), учитывающая результаты формализации и описывающая язык с точки зрения объектно-ориентировнного подхода.

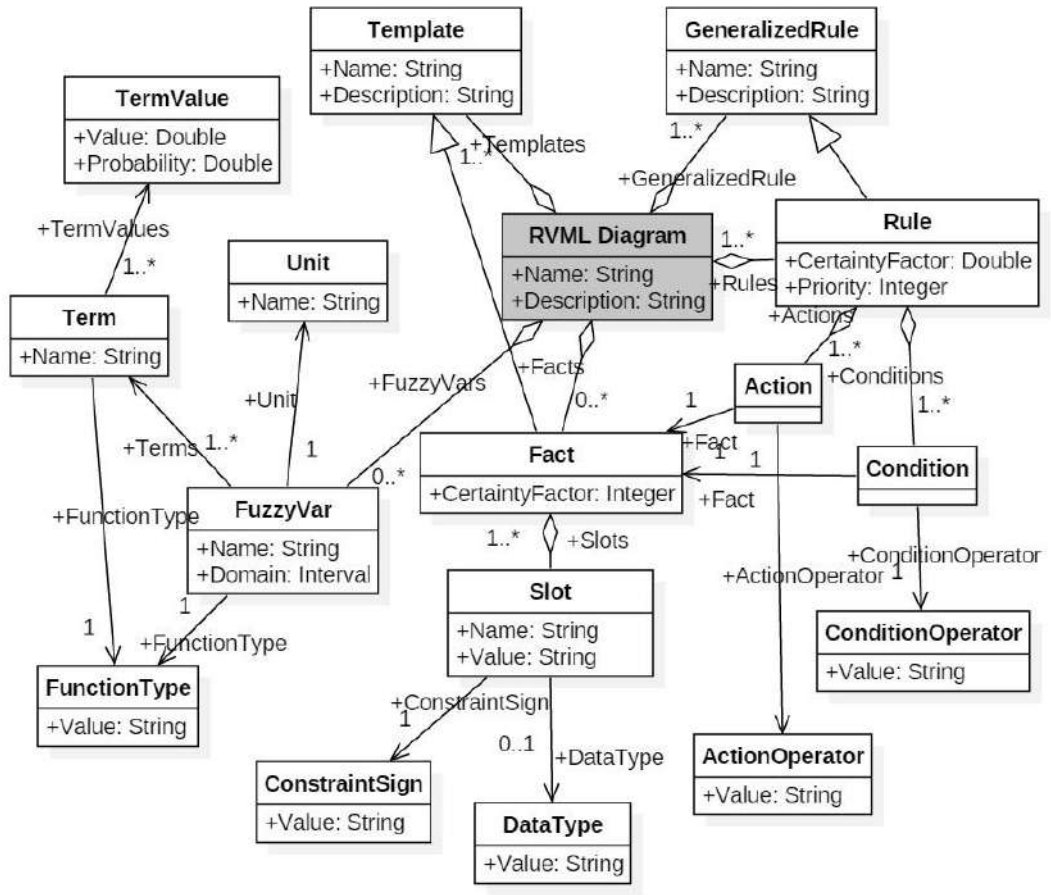


Рис. 3.1.1. Метамодель RVML/ FuzzyRVML.

3.1.3 Графические элементы RVML/FuzzyRVML

RVML включает 8 графических элементов (Рис. 3.1.2 и 3.1.3), при этом 5 из них являются основными или базовыми, 3 используются в расширении FuzzyRVML.

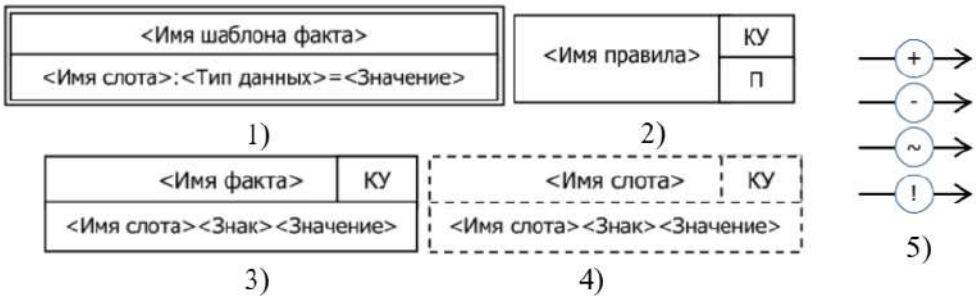


Рис. 3.1.2. Основные элементы RVML: 1) шаблон факта; 2) узловой элемент правила; 3) факт; 4) условие; 5) связи элементов с указанием действий.

Основные элементы:

1. Шаблон факта (Рис. 3.1.2, 1) используется для описания шаблонов фактов и частей обобщенных правил (шаблонов правил), таких как условие и действие. Включает наименование и описание слотов. При этом каждый слот содержит

собственное наименование, тип данных и значение «по умолчанию».

2. Узловой элемент (или ядро) правила (Рис. 3.1.2, 2) является связующим звеном при представлении правила как композиции множества фактов (как условий и действий). Узловой элемент позволяет представить наименование правила, коэффициент уверенности (КУ) и приоритет (П).

3. Факт, как элемент действия правила (Рис. 3.1.2, 3), используется для представления фактов, над которыми производятся манипуляции в рабочей памяти. Данный элемент позволяет представить наименование факта и его описание через определенные слоты, но и коэффициент уверенности (КУ).

4. Факт, как элемент условия правила (Рис. 3.1.2, 4), используется для представления фактов, активирующих правила в рабочей памяти. Для обозначения управляющего характера этих фактов, по аналогии с диаграммами управляющих потоков, используется пунктирное изображение всех линий графического элемента. Данный элемент позволяет представить наименование факта и его описание через определенные слоты, но и коэффициент уверенности (КУ).

5. Соединение между элементами (Рис. 3.1.2, 5), представлено как ассоциация, содержащая узел для отображения операции с фактами в рабочей памяти: «+» – добавление; «-» – удаление; «~» – изменение; «!» – остановка логического вывода.

Элементы расширения для обозначения их нечеткого или недоопределенного характера отображаются точечными линиями:

1. Нечеткая или лингвистическая переменная (Рис. 3.1.3, 1) содержит информацию о наименовании переменной, диапазоне возможных значений, единицах измерения, типе функции принадлежности, термах.

2. Терм (Рис. 3.1.3, 2), определяющий значение определенной нечеткой переменной согласно заданной функции принадлежности. Содержит наименование, тип функции принадлежности, возможные значения с указанием их вероятности.

3. Соединение между элементами (Рис. 3.1.3, 3), представлено как зависимость, фиксирующая связь между нечеткой переменной и ее темами (Рис. 3.1.3, 4), а также шаблонами и нечеткими переменными (Рис. 3.1.4, 1), фактами и термами (Рис. 3.1.4, 2).

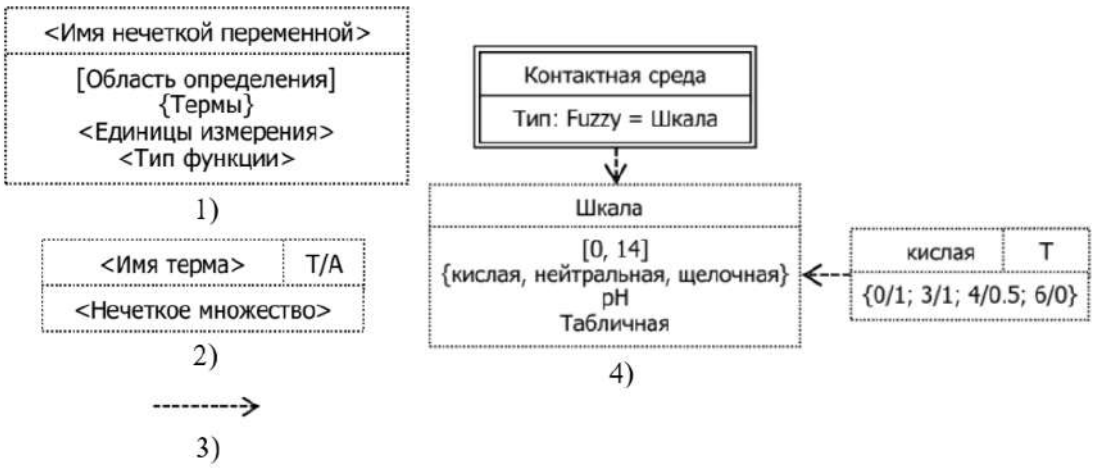


Рис. 3.1.3. Основные элементы FuzzyRVML: 1) лингвистическая (нечеткая) переменная; 2) терм; 3) связь типа «зависимость»; 4) совместное использование элементов RVML и FuzzyRVML.

Основные элементы и элементы расширения могут использоваться совместно, в частности, на Рис. 3.1.3, 4 приведены примеры описания шаблона факта с лингвистической (нечеткой) переменной, а также описания факта с нечетким термом.

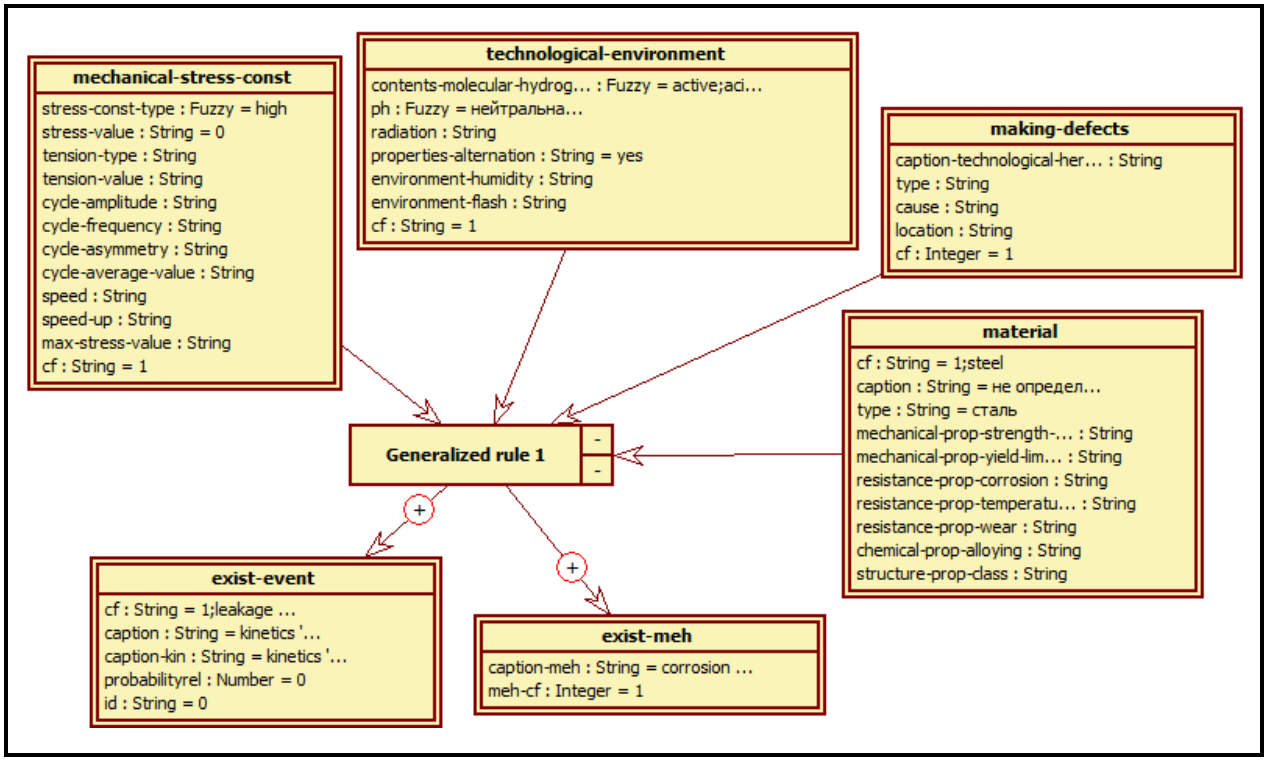


Рис. 3.1.4. Пример обобщённого правила (шаблона правила) в нотации RVML

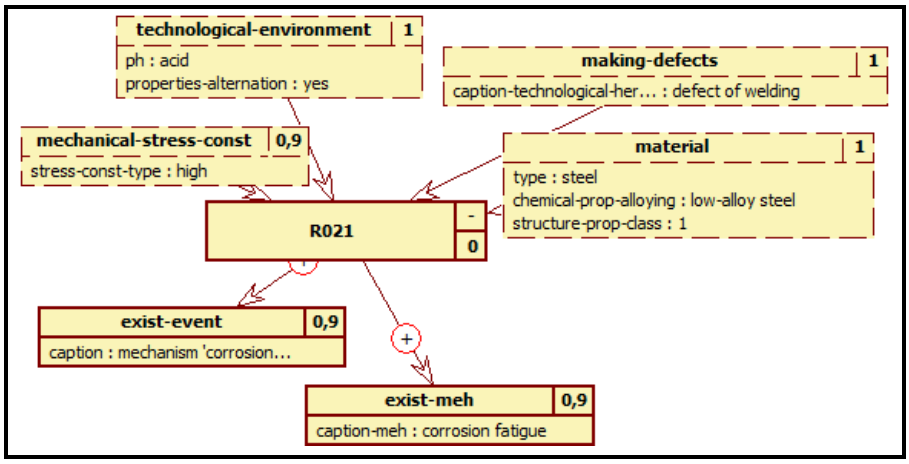


Рис. 3.1.5. Пример правила (экземпляра шаблона правила) в нотации RVML

3.1.4 Преобразования элементов RVML/FuzzyRVML в код

Одним из назначений RVML является преодоление семантического разрыва между визуальными моделями и программными кодами, в связи с этим обеспечено однозначное соответствие между графическими элементами RVML/FuzzyRVML и языковыми конструкциями языков программирования правил для интеллектуальных систем, в частности, CLIPS/FuzzyCLIPS и Drools. Данное соответствие в дальнейшем используется для реализации автоматизированной кодогенерации. В таблице 3.1.1 приведены примеры соответствий.

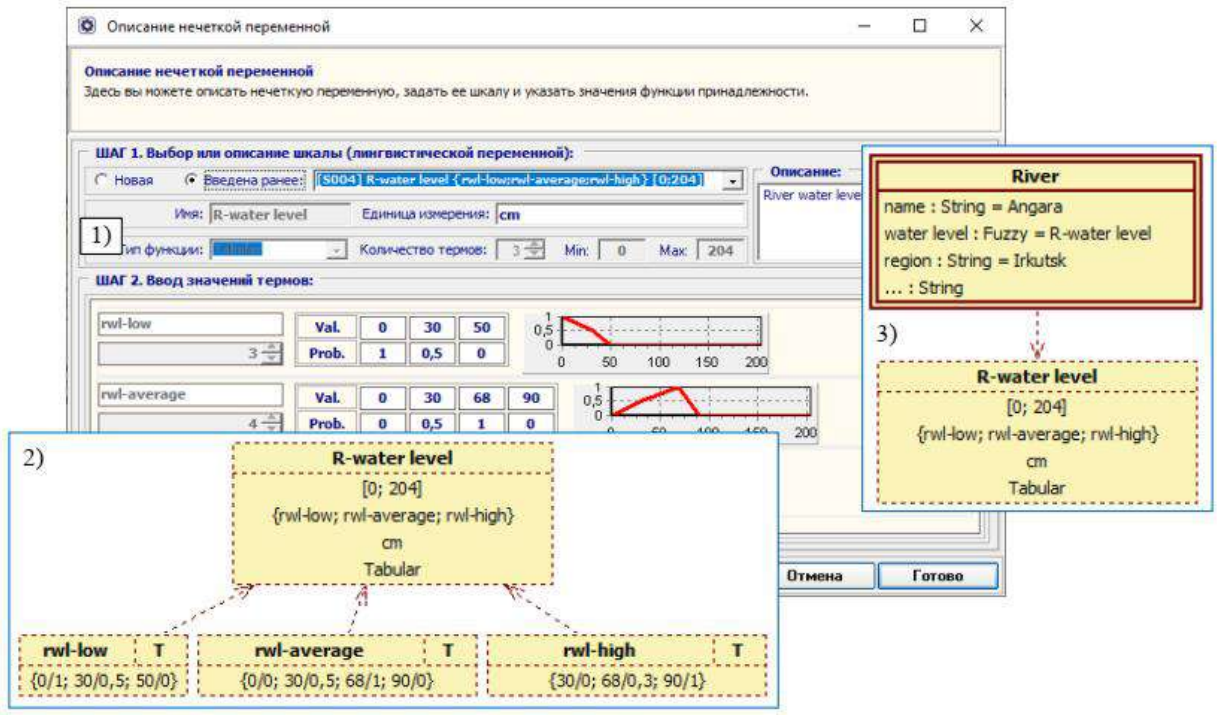
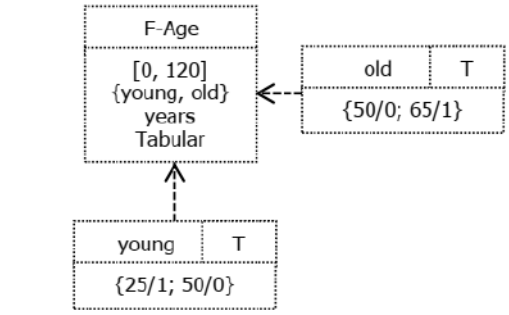
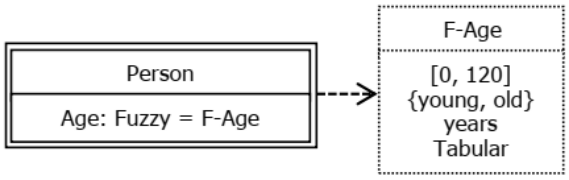


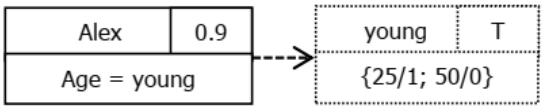
Рис. 3.1.6. Примеры форм пользовательского интерфейса разработанной системы программирования: 1) форма описания лингвистической (нечеткой) переменной; 2) представление лингвистической (нечеткой) переменной и ее термов в FuzzyRVML; 3) интегрированное представление элементов RVML и FuzzyRVML при описании шаблона фактов с лингвистической (нечеткой) переменной.

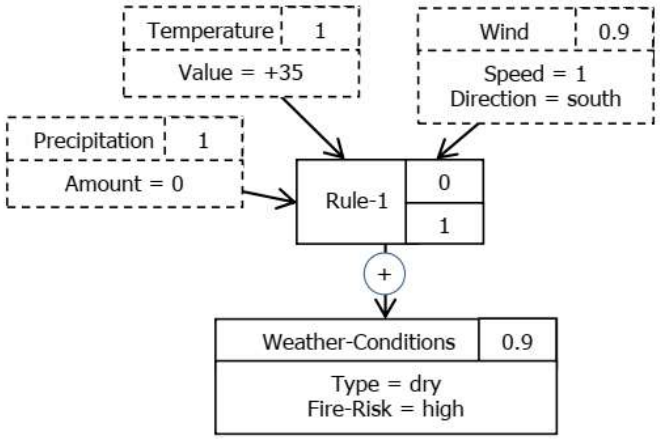
Таблица 3.1.1. Примеры соответствий для элементов RVML/FuzzyRVML и CLIPS/FuzzyCLIPS

Примеры элементов RVML/FuzzyRVML	Соответствующие элементы CLIPS/FuzzyCLIPS

	`(deftemplate F-AGE`
	` 0 120`
	` (`
	` (YOUNG (25 1) (50 0))`
	` (OLD (50 0) (65 1))`
	` )`
	`)`

	`(deftemplate Person`
	` (slot age (default "F-AGE"))`
	`)`

	`(Alex`
	` (age "YOUNG")`
	` ) CF 0.9`

	`(defrule Rule-1 "Description"`
	` (Temperature ;Temperature`
	` (Value "+35")`
	` (cf "1")`
	` )`
	` (Precipitation ;Precipitation`
	` (Amount 0)`
	` (cf "1")`
	` )`
	` (Wind ;Wind`
	` (Speed 1)`
	` (Direction "SOUTH")`
	` (cf "0.9")`
	` )`
	`=>`
	`(assert`
	` (Weather-conditions ;`
	` (Type "DRY")`
	` (Fire-Risk "HIGH")`
	` (cf "0.9")`
	` ))`
	`)`

Коэффициент уверенности в правилах	`(defrule <RuleName>`
	` (declare (CF`
	`<CertaintyFactorValue>))`
	` …`

Примеры преобразований RVML в код приведены в главах 6 и 7.

3.1.5 Преимущества RVML/FuzzyRVML и сравнение с аналогами

Текущая спецификация RVML 1.2 ориентирована на описания простых правил, поэтому не поддерживаются следующие элементы языков программирования баз знаний: процедуры и функции, переменные, вычисляемые

значения. В дальнейшем планируется ее дополнение поддержкой языка объектных ограничений – Object Constrained Language (OCL) [120].

По сравнению с аналогами, в частности, URML и UML, RVML/FuzzyRVML обладает следующими преимуществами:

- Использует специализированные графические элементы для всех компонентов правил, а не один типовой элемент, отличающийся стереотипом (как в UML).
- Обеспечивает однозначную визуальную индикацию действий, продуцируемых правилами: добавления, удаления, изменения фактов, остановка логического вывода.
- Может рассматриваться как профиль расширения UML, использующий терминологию диаграмм классов («класс», «ассоциация», «зависимость») и ориентированный на моделирование логических правил.
- Обеспечивает абстрагирование от определенных языков представления/программирования знаний: логические правила представляются в обобщенном виде.
- Может использоваться для моделирования неполноты и нечеткости и содержит специализированные элементы: тип данных (Fyzzy); лингвистическая (нечеткая) переменная (FuzzyVar) и набор нечетких термов как возможные значения лингвистической переменной; коэффициент уверенности (Certainty Factor).
- Может использоваться для синтеза программных кодов (поддерживает прямое отображение графических элементов в код) на CLIPS, FuzzyCLIPS, Drools и др.

Сравнение RVML/FuzzyRVML с другими языками разработки декларативных баз знаний приведено в Таблице 1.3.1 (см. 1.3).

3.2 Язык программирования трансформаций концептуальных моделей - Transformation Model Representation Language

Для программирования (описания) трансформаций концептуальных моделей разработан текстовый декларативный язык – Transformation Model Representation Language (TMRL) [46, 239, 268, 271, 272]. Грамматика TMRL принадлежат к классу контекстно-свободных грамматик (КС-грамматик – LL(1)) [213, 316]. Программы на TMRL удовлетворяют требованиям полноты, понятности и точности [207, 208], т.е. объекты модели хорошо формализованы, содержат необходимую информацию, а спецификации понятны (читабельны) и компактны.

3.2.1 Формальное описание TMRL

TMRL включает 15 служебных лексем (элементов), при этом: 7 ликсем используются для описания исходной и целевой метамодели; 7 – описывают трансформации; 1 – обеспечивает вызов внешних программных компонентов трансформации.

Формальное описание TMRL с использованием (РБНФ):

```
Модель трансформации на TMRL = "Transformation Model" , Заголовок, "{" , Модель трансформации , "}" , Оператор вызова.
Модель трансформации = Исходная метамодель, Целевая метамодель, Оператор трансформации.
Исходная метамодель = "Source Meta-Model" , Заголовок, "{" , Тело метамодели, "}".
Целевая метамодель = "Target Meta-Model" , Заголовок, "{" , Тело метамодели, "}".
Оператор трансформации = "Transformation" , Заголовок оператора трансформации , "{" , Тело трансформации, "}".
Заголовок = Буква , { Буква | Цифра | Символ }.
Буква = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z" | "А" | "а" | "Б" | "б" | "В" | "в" | "Г" | "г" | "Д" | "д" | "Е" | "е" | "Ё" | "ё" | "Ж" | "ж" | "З" | "з" | "И" | "и" | "Й" | "й" | "К" | "к" | "Л" | "л" | "М" | "м" | "Н" | "н" | "О" | "о" | "П" | "п" | "Р" | "р" | "С" | "с" | "Т" | "т" | "У" | "у" | "Ф" | "ф" | "Х" | "х" | "Ц" | "ц" | "Ч" | "ч" | "Ш" | "ш" | "Щ" | "щ" | "Ъ" | "ъ" | "Ы" | "ы" | "Ь" | "ь" | "Э" | "э" | "Ю" | "ю" | "Я" | "я".
Цифра = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
Символ = "-" | ".".

Тело метамодели = Элементы , Связи.
Элементы = "Elements" , "[" , { Элемент } , "]".
Элемент = Наименование , [ Атрибуты ] , ",".
Атрибуты = "attributes" , "(" , { Атрибут } , ")".
Атрибут = Наименование , ",".
Наименование = Буква , { Буква | Цифра | Символ }.
Связи = "Relationships" , "[" , { Связь } , "]".
Связь = Ассоциация | Связь по идентификатору.
Ассоциация = Левый элемент ассоциации , "is associated with" , Правый элемент ассоциации , ",".
Левый элемент ассоциации = Наименование.
Правый элемент ассоциации = Наименование.
Связь по идентификатору = Левый элемент связи по идентификатору, "is" , Правый элемент связи по идентификатору , ",".
Левый элемент связи по идентификатору = Наименование элемента , "(" , Наименование атрибута , ")".
Правый элемент связи по идентификатору = Наименование элемента , "(" , Наименование атрибута , ")".
Наименование элемента = Наименование.
Наименование атрибута = Наименование.
Заголовок оператора трансформации = Название исходной метамодели , "to" , Название целевой метамодели.
```

Название исходной метамодели = Заголовок.
Название целевой метамодели = Заголовок.
Тело трансформации = { Правило }.
Правило = Заголовок правила , "[" , Тело правила , "]".
Заголовок правила = **"Rule"** , Исходные элементы , **"to"** , Целевые элементы , **"priority"** , Цифра.
Исходные элементы = Единичный исходный элемент | Множество исходных элементов.
Единичный исходный элемент = Наименование.
Множество исходных элементов = **"("** , Исходный элемент , { Дополнительный исходный элемент } , **")"**.
Дополнительный исходный элемент = "," , Исходный элемент.
Исходный элемент = Наименование.
Целевые элементы = Единичный целевой элемент | Множество целевых элементов.
Единичный целевой элемент = Наименование.
Множество целевых элементов = **"("** , Целевой элемент , { Дополнительный целевой элемент } , **")"**.
Дополнительный целевой элемент = "," , Целевой элемент.
Целевой элемент = Наименование.

Тело правила = { Выражение определения } , { Условное выражение }.
Выражение определения = Левый элемент выражения , **"is"** , Правый элемент выражения.
Левый элемент выражения = Целевой элемент , **"("** , Атрибут целевого элемента , **")"**.
Атрибут целевого элемента = Наименование.
Правый элемент выражения = Исходный элемент , **"("** , Атрибут исходного элемента , **")"** , [Дополнительный правый элемент выражения].
Атрибут исходного элемента = Наименование.
Дополнительный правый элемент выражения = { **"or"** Исходный элемент , **"("** , Атрибут исходного элемента , **")"** }.
Условное выражение = Левый элемент выражения , **"is"** , Значение выражения , [Условие] , [Альтернативное значение выражения .

Условие = **"["** , Описание условия , **"]"**
Альтернативное значение выражения = **"or"** , Значение выражения , **"["** , Условие , **"]"**.
Описание условия = **"if"** , **"("** , Правый элемент выражения , **"is"** , Значение выражения **")"** , [Дополнительное условие].
Дополнительное условие = **"and"** , **"("** , Правый элемент выражения , **"is"** , Значение выражения **")"**.
Значение выражения = "`" , { Буква | Цифра | Символы } , "´".
Символы = **"@"** | **"#"** | **"$"** | **"%"** | **"^"** | **"&"** | **"*"** | **"/"** | **"+"** | **"-"** | **"."** | **"?"** | **"!"** | **","** | **";"** | **":"** | "'" | **"["** | **"]"** | **"{"** | **"}"** | **"="** | **"_"** | " " | **"\"**.
Оператор вызова = **"Call"** , Название программного компонента трансформации , "," , Путь к концептуальной модели , ["," , Путь сохранения базы знаний].
Название программного компонента трансформации = { Буква | Цифра | Символы }.
Путь к концептуальной модели = { Буква | Цифра | Символы }.
Путь сохранения базы знаний = { Буква | Цифра | Символы }.
Нечеткое множество = Вид описания [Вид функции принадлежности (Параметры функции принадлежности)].
Вид описания = Табличное | Аналитическое

```
Вид функции принадлежности = Треугольная | Трапециевидная | S-
образная сплайн-функция | Z-образная сплайн-функция | Линейная S-
образная функция | Кинейная Z-образная функция | П-образная.
```

3.2.2 Метамодель TMRL

На Рис. 3.2.1 приведена метамодель TMRL, определяющая основные его элементы.

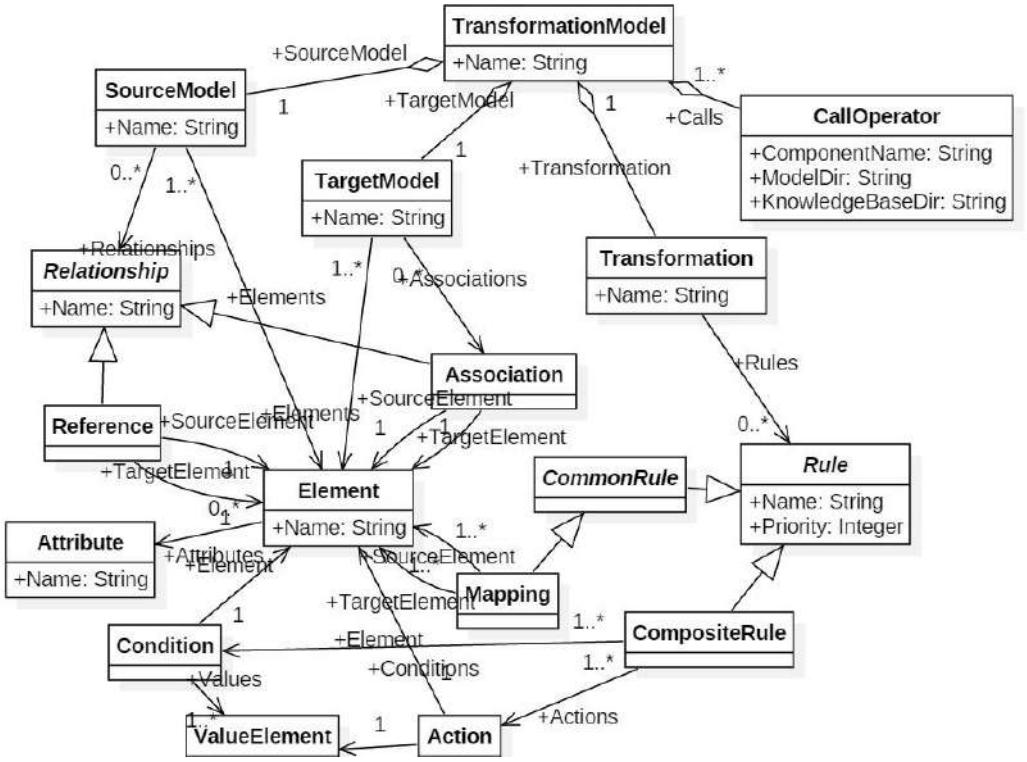


Рис. 3.2.1. Метамодель TMRL

3.2.3 Структура программы TMRL

Программа на TMRL включает четыре основных раздела [239]:

1) описание структуры исходной модели (основные понятия и отношения): «**`Source Meta-Model`**»;
2) описание структуры целевой метамодели (основные понятия и отношения): «**`Target Meta-Model`**»;
3) описание правил трансформации: «**`Transformation`**»;
4) вызов внешних модулей: «**`Call`**».

Рассмотрим разделы подробнее на примере программы для трансформации UML диаграмм классов:

```
Source Meta-Model UML-diagram-class {
  Elements [
    Model,
    Class attributes (xmi.id, name),
    …
  ]
  Relationships [
    Model is associated with Namespace.ownedElement,
    Namespace.ownedElement is associated with Class,
    DataType(xmi.id) is Attribute(type),
    …
  ]
}
```

В разделе описания структуры исходной модели существуют подразделы:

- Элементов (понятий) модели («**Elements**»), в примере – это элементы «Model» и «Class», при этом элемент «Class» обладает свойствами «xmi.id» и «name»;
- Отношений между элементами (связи) («**Relationships**»), в том числе по идентификаторам, например связь атрибута с типом данных («DataType(xmi.id) **is** Attribute(type)»).

Структура раздела описания структуры целевой модели аналогична:

```
Target Meta-Model Ontology {
  Elements [
    ExtendedOntology attributes (id, name),
    Class attributes (id, name),
    …
  ]
  Relationships [
    Ontology is associated with Class,
    …
  ]
}
```

В разделе описания трансформаций приведены правила преобразования (соответствия) элементов моделей с указанием приоритета или очередности выполнения («**priority**»).

```
Transformation UML-diagram-class to Ontology {
    Rule (Class, ModelElement.name) to Class priority 2 [
      Class(name) is Class(name) or ModelElement.name
      Class(id) is Class(xmi.id)
    ]
  …
}
```

Структура программы рассмотрена на примере отображения классов UML в классы онтологии. При описании правил возможно определение оператора условного выбора («**if**») и логических операторов «И» и «ИЛИ» («**and**», «**or**»).

Использование оператора «**Call**» позволяет воспользоваться

функциональностью одного из разработанных ранее модулей:

```
Call <название программного компонента трансформации> ,
  "<путь к концептуальным моделям>" ,
  "<путь сохранения баз знаний>"
```

Примеры программ на TMRL приведены в 7.5.

3.2.3 Преимущества TMRL и сравнение с аналогами

Наиболее близкими к TMRL являются языки модельных трансформаций (Model Transformation Language, MTL), в частности, ATL [85], что позволяет осуществить наглядное сравнение их синтаксиса на примере правила трансформации элемента «`Class`» из диаграммы классов UML в онтологию OWL.

TMRL правило преобразования:

```
Rule Class to Class priority 1 [
  Class(ID) is Class(name)
]
```

ATL правило преобразования [110]:

```
rule UMLClass2OWLClass {
  from
    c : UML!uml::Class (
      c.oclIsTypeOf(UML!uml::Class) and
        not thisModule.sequenceOfUnionClass.includes(c)
    )
  to
    oc : OWL!OWLClass (uriRef <- u),
    u : OWL!URIReference (fragmentIdentifier <- l,uri <- uri),
    l : OWL!LocalName (name <- c.name),
    uri : OWL!UniformResourceIdentifier (name <- c.name)
}
```

Основные отличия TMRL от других языков трансформации моделей общего назначения (универсальных) и его особенности заключаются в следующем:

- Лаконичность и простота использования, достигаемая за счет ограниченного набора конструкций.
- Самодостаточность (полнота) - TMRL не является расширением других языков и не требует применения других языков, например, OCL для создания программ.
- Наличие человекочитаемого синтаксиса, который обеспечивает возможность редактирования TMRL программ.
- Возможность взаимодействия с другими TMRL программами и программными модулями.

Сравнение TMRL с другими языками описания трансформаций приведено в

Таблице 2.2.1 (см. 2.2).

Выводы

Разработан новый специализированный язык визуального программирования RVML (Rule Visula Modeling Language), основанный на UML и расширяющий его выразительные способности в контексте создания логических правил (продукций). Особенностью RVML является абстрагирование от определенных языков представления/программирования знаний, однозначная визуальная индикация компонентов правил и продуцируемых действий, поддержка моделирования неполноты и нечеткости, прямое отображение графических элементов в код.

Создан новый декларативный текстовый язык программирования трансформаций концептуальных моделей - TMRL (Transformation Model Representation Language), включающий конструкции для описания не только преобразуемых структур и связей между ними, но и механизма взаимодействия с другими программными компонентами трансформаций. Особенностью TMRL является его ориентированность (специализация) на описание обработки концептуальных моделей, сериализуемых в XML-подобных форматах; самодостаточность (полнота); наличие человекочитаемого синтаксиса, который обеспечивает возможность редактирования TMRL программ.

Разработанные языки использованы в рамках оригинальных методов проектирования декларативных баз знаний на основе трансформаций концептуальных моделей.

Глава 4. Метод и программные средства проектирования декларативных баз знаний интеллектуальных систем

В главе представлены оригинальные модели, метод и программные средства для создания программного обеспечения систем ИИ с декларативными базами знаний.

4.1 Метод проектирования декларативных баз знаний интеллектуальных систем

В рамках диссертационного исследования разработанны модели и метод проектирования декларативных баз знаний и интеллектуальных систем [27, 53, 184, 185, 192, 256, 257] реализующие следующие EUD подходы: визуальное программирование и модельно-ориентированный подход (MDA/MDE) от OMG.

Согласно MDA/MDE [24, 38, 41, 43, 69, 71, 92, 109, 140, 144, 161, 168], разрабатываемое программное обеспечение, включая описание основных понятий, взаимосвязей и методов их обработки, представляется в виде набора информационных, в частном случае, концептуальных моделей, определяющих его состав, структуру и поведение. При этом процесс разработки программного обеспечения представляет собой последовательный переход от моделей с большей абстракцией, к моделям с меньшей абстракцией и программным кодам и спецификациям БЗ и ИС.

В настоящее время MDA/MDE не использует задекларированную в спецификациях подхода вычислительно-независимую модель, аналог технического задания. В связи с этим актуальным является развитие данного подхода, заключающееся как в использовании в процессе разработки приложений вычислительно-независимой модели, ее спецификации и определения, так и расширении области применения MDA-подхода на область разработки интеллектуальных систем (систем, основанных на знаниях), в частности, ИС и БЗ продукционного и прецедентного типа.

4.1.1 Формальное описание метода

Приведем формальное описание MDA/MDD [24, 38, 41, 43, 69, 71, 92, 109, 140, 144, 161, 168] в следующем виде:

$$MDA/MDD = \left\langle \begin{matrix} MOF, UML, CIM, PIM, PSM, PDM, \\ F_{CIM\text{-to-}PIM}, F_{PIM-to-PSM}, F_{PSM-to-CODE} \end{matrix} \right\rangle,$$

где *MOF (Meta Object Facility*) – абстрактный язык для описания моделей (язык описания мета-моделей); *UML (Unified Modelling Language)* – унифицированный язык моделирования, как основной язык для описания всех моделей; *CIM* (Computation Independent Model) – вычислительно-независимая модель – модель, скрывающая любые детали реализации, описывает только требования к системе и ее окружению; *PIM* (Platform Independent Model) – платформо-независимая модель – модель, скрывающая детали реализации системы, зависимые от платформы, и содержащая элементы, не изменяющиеся при взаимодействии системы с любой платформой; *PSM* (Platform Specific Model) – платформо-зависимая модель – модель системы с учетом деталей реализации и процессов, зависимых от конкретной платформы; *PDM* (Platform Description Model) – модель платформы – набор технических характеристик и описаний технологий и интерфейсов, составляющих программную (технологическую) платформу; $F_{CIM\text{-}to\text{-}PIM}: CIM \rightarrow PIM$, $F_{PIM\text{-}to\text{-}PSM}: PIM \rightarrow PSM$, $F_{PSM\text{-}to\text{-}CODE}: PSM \rightarrow CODE$ – правила преобразования моделей.

Тогда предлагаемый метод в контексте решения задачи разработки БЗ и ИС на основе модельных трансформаций будет иметь следующий вид [27, 53, 184, 185, 192, 257]:

$$MDA/MDD^{ES} = \left\langle \begin{matrix} MOF, L^{ES}, CIM^{ES}, PIM^{ES}, PSM^{ES}, PDM^{ES}, \\ F^{ES}_{CIM-to-PIM}, F^{ES}_{PIM-to-PSM}, F^{ES}_{PSM-to-CODE} \end{matrix} \right\rangle,$$

где *ES* – индекс прделагаемого метода; L^{ES} – набор языков и формализмов, используемых для моделирования, в рамках решаемой задачи:

$$L^{ES} = \{UML, CM, DT, ET, CT, RVML\}$$

где *UML* – унифицированный язык моделирования (стандартное средство подхода); *CM* – формализм концепт карт и карт памяти; *DT* – формализм представления таблиц решений; *ET* – формализм представления деревьев событий; *CT* – формализм представления канонических таблиц [47, 50] (специализированной

формы представления таблиц, предназначенной для унификации их представления для дальнейшей автоматизированной обработки); *RVML* – Rule Visual Modeling Language [52, 184, 185, 198, 279, 368], как специализация UML для моделирования логических правил;

CIM^{ES} – вычислительно-независимая модель в контексте решаемой задачи, представляемая посредством L^{ES};

PIM^{ES} – платформо-независимая модель в контексте решаемой задачи, описывает архитектурные элементы ИС и логические правила (структуру БЗ) в нотации RVML;

PSM^{ES} – платформо-зависимая модель в контексте решаемой задачи, учитывает особенности целевых платформ, строится с использованием RVML;

PDM^{ES} – набор моделей описания платформ, в рамках решаемой задачи:

$$PDM^{ES} = \{CLIPS, DROOLS, PHP, PKBD\}$$

где *CLIPS* и *DROOLS* – специализированные языки программирования ИС и БЗ; *PHP* – язык программирования общего назначения; *PKBD* (Personal Knowledge Base Designer) [27, 184, 195, 202, 269] – авторская инструментальная платформа, поддерживающая предлагаемый метод.

$F^{ES}_{CIM-to-PIM}, F^{ES}_{PIM-to-PSM}, F^{ES}_{PSM-to-CODE}$ – правила трансформаций реализованных либо императивно, с использованием языка разработки платформы PKBD, либо декларативно, с использованием TMRL [46, 239, 268, 271, 272].

С точки зрения архитектуры (схемы) метамоделирования (см. 2.2.2) предлагаемый метод может быть представлен следующим образом (Рис.4.1.1), где серым цветом отмечены блоки, расширяющие базовую схему.

Рассмотрим основные этапы предлагаемого подхода, которые в дальнейшем сформируют методику. В качестве примера будут рассмотрены БЗ и ИС продукционного типа, полагая, что процесс разработки БЗ и ИС прецедентного типа аналогичен.

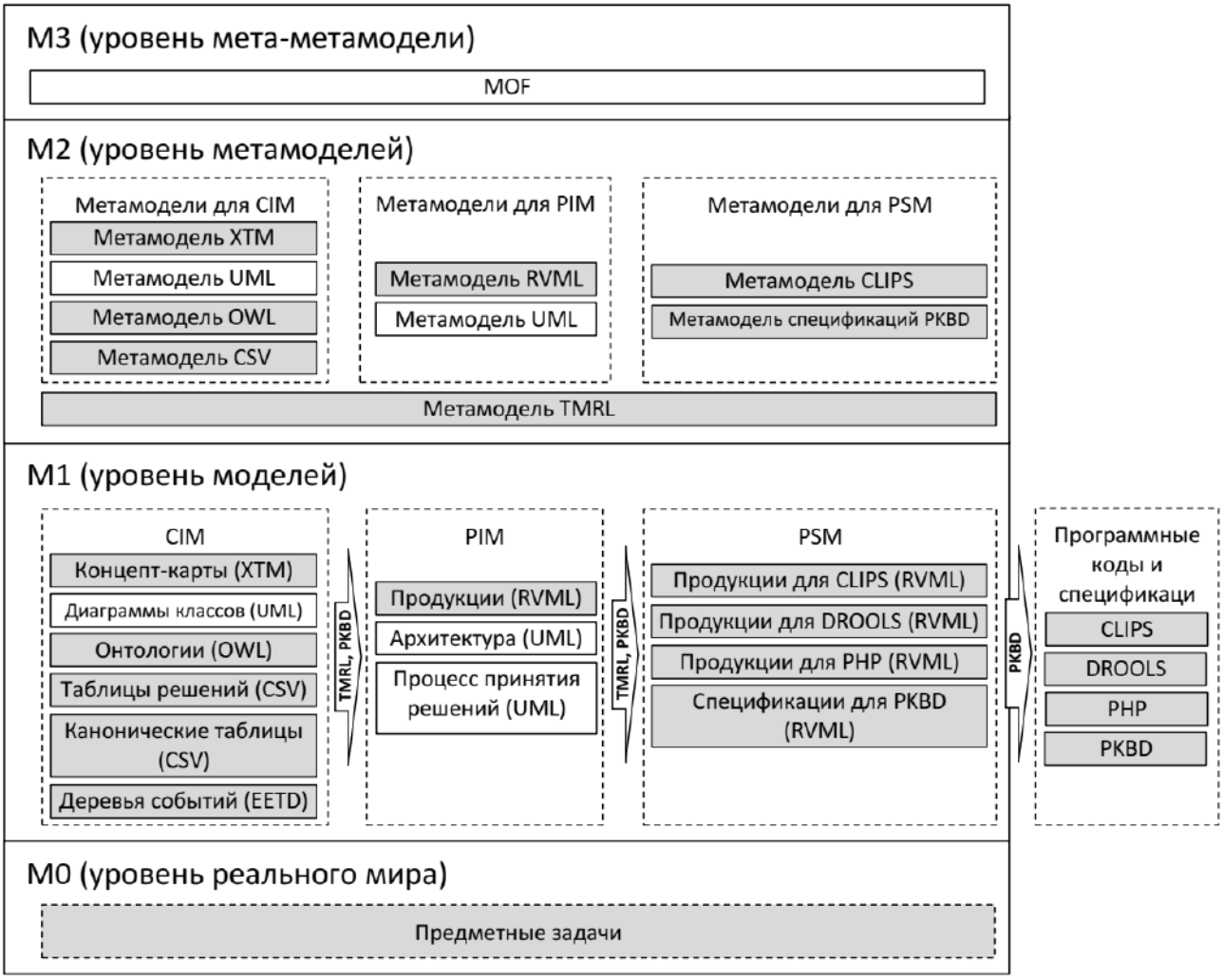


Рис. 4.1.1. Архитектура (схема) метамоделирования, описывающая трансформации моделей в рамках предлагаемого метода на различных уровнях абстракции

4.1.2 Основные этапы метода

Процесс разработки представлен последовательностью этапов (Рис. 4.1.2), обеспечивающих создание и трансформацию концептуальных моделей.

1) Построение модели предметной области является первым этапом. Выходная информация этого этапа:

- модель предметной области;
- концептуальная модель ИС.

Полученные модели рассматриваются как вычислительно-независимые и могут быть представлены в виде OWL-онтологии, UML-моделей (в частности, в виде UML-диаграмм классов) или карт памяти (Рис. 4.1.3) [184].

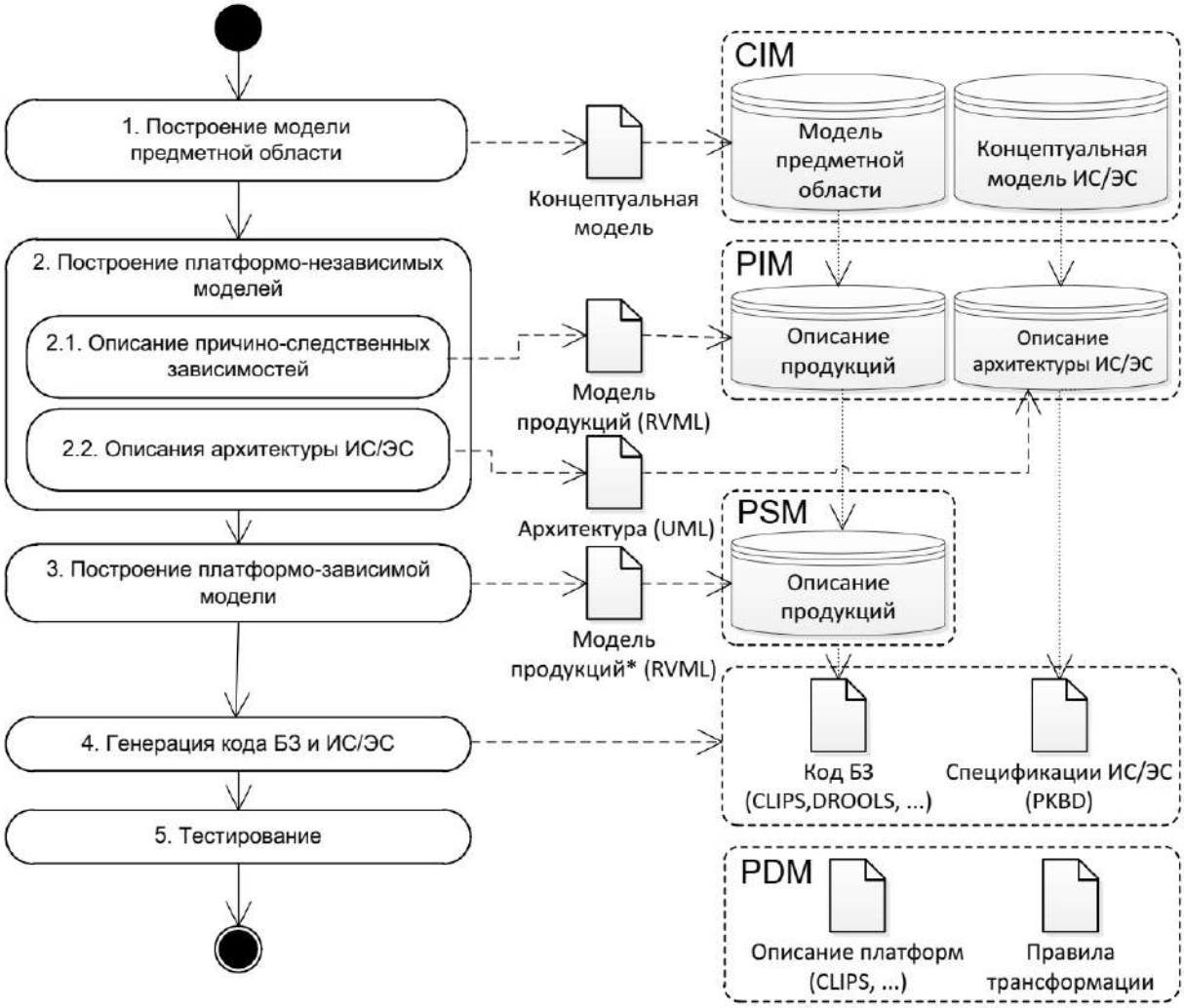
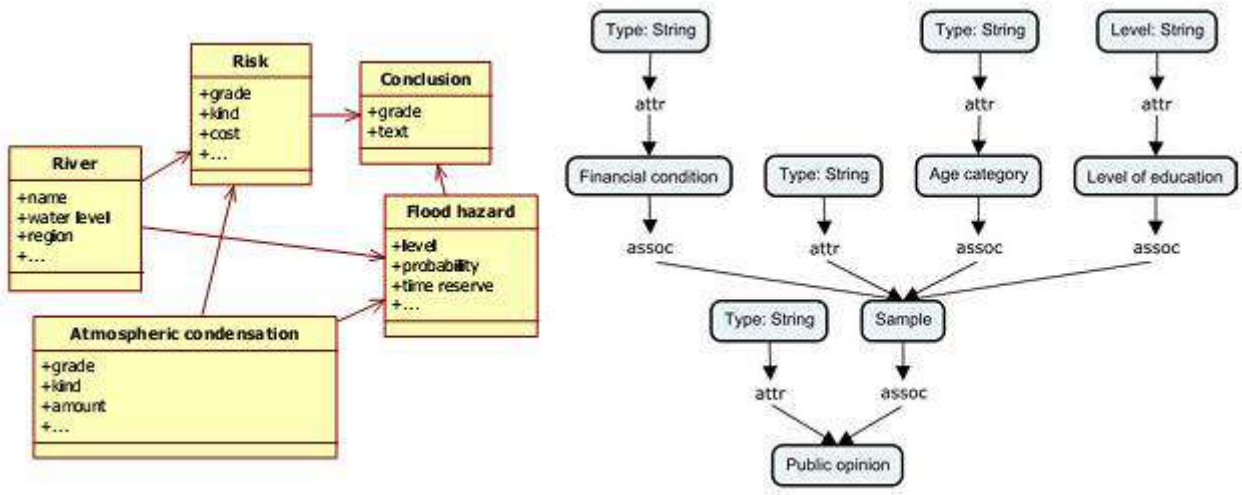


Рис. 4.1.2. Основные этапы и модели предлагаемого метода



а) б)

Рис. 4.1.3. Примеры фрагментов моделей предметных областей, используемых в учебном процессе для создания прототипов баз знаний: а) для прогнозирования речных паводков в виде диаграмм классов (StarUML), б) для прогнозирования общественного мнения в форме концепт-карты (IHMC CmapTools).

Способы формирования этих моделей в рамках метода не конкретизируется. Соответствующие понятия могут быть выявлены как при извлечении знаний из экспертов, так и при анализе существующих онтологий и моделей данных. Эффективность данного этапа может быть повышена путем повторного использования существующих концептуальных моделей, созданных с использованием различных онтологических и концептуальных (когнитивных) редакторов или CASE-средств (Protégé, OntoStudio, CmapTools, FreeMind, TheBrain, Xmind, IBM Rational Rose Enterprise, StarUML, и т.д.) [266, 270, 274-277].

В случае создания продукционных БЗ и ИС важно, чтобы полученная на данном этапе предметная модель была семантически значимой, то есть описывала какие-либо причинно-следственные связи в конкретной предметной области, чтобы на ее основе можно было создать БЗ. Таким образом при использовании онтологий в качестве средства формализации вычислительно-независимых моделей (их унифицированного представления), в дополнение к отношениям «является частью» и «является», вводится отношения «зависит от» – связь обеспечивает описание причинно-следственных связей. В случае диаграмм классов UML типы отношений определяются с помощью механизма стереотипов.

Концептуальная модель продукционной ИС формируется на уровне требований к составу ее основных модулей. Поскольку тип создаваемых систем определен заранее – это продукционные ИС, то для формирования этой модели используется соответствующий шаблон, который определяет основные архитектурные элементы и включает соответствующие понятия) (Рис. 4.1.4) [184, 185, 257].

Основными архитектурными элементами ИС/ЭС являются: «форма ввода»; «форма вывода» и др., которые являются производными от «графической формы пользовательского интерфейса»; «механизм вывода», который является производным от «обработчик»; «база знаний».

Большинство программных систем, которые поддерживают MDA подход (например, Bold for Delphi), не реализуют этот этап и предлагают начинать разработку программного обеспечения, начиная со следующего этапа. В этом случае концептуальная модель предметной области (даже представленная в виде онтологии) рассматривается как платформо-независимая модель, описывающая

основные понятия и бизнес-логику (что приемлемо для баз данных). В случае разработки интеллектуальных систем этот этап является необходимым и соответствует этапу концептуализации знаний и качественно отличает предложенную специализацию MDA/MDD.

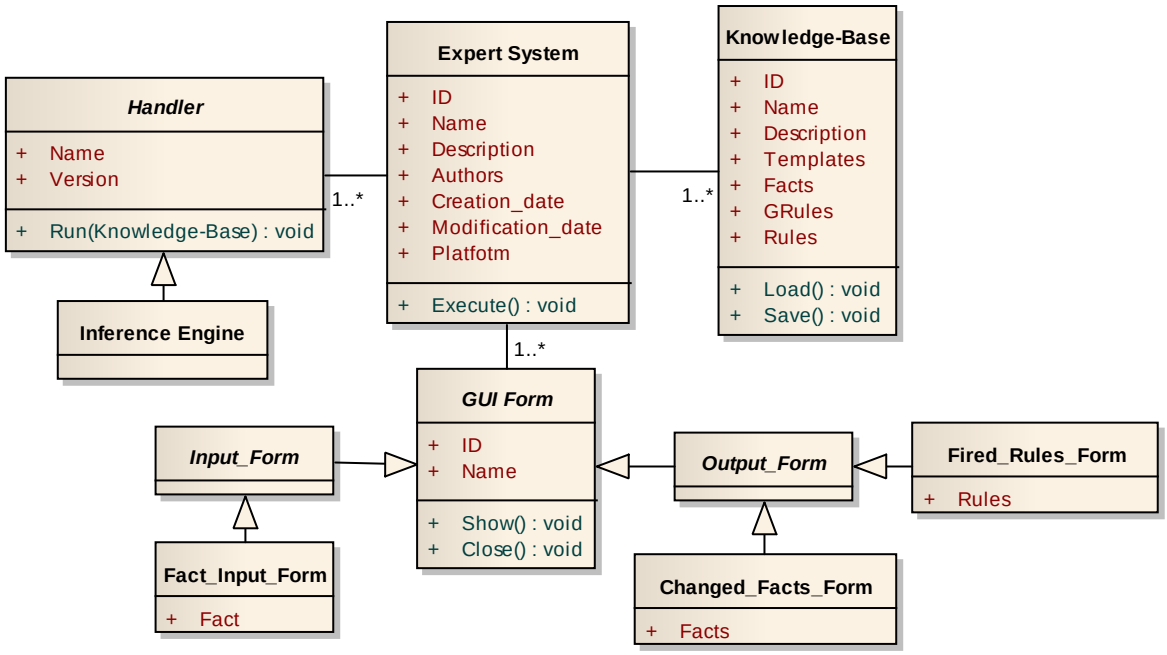


Рис. 4.1.4. Концептуальная модель архитектуры ИС/ЭС

2) Построение платформо-независимых моделей является вторым этапом. Выходная информация этого этапа:

- описание правил;
- подробное описание архитектуры ИС.

Модели данного этапа являются результатом трансформации вычислительно-независимой модели, в процессе преобразования которой понятия преобразуются в шаблоны фактов и элементы правил, такие как условия и действия, а причинно-следственные связи преобразуются в логические правила. RVML используется для визуализации и последующего уточнения элементов полученной модели (Рис. 4.1.5), т.е. на данном этапе реализуется оператор $F_{CIM-to-PIM}^{ES}$.

Проектирование ИС/ЭС включает в себя проектирование структуры ИС и пользовательского интерфейса. Таким образом, используются элементы подхода, известного как «Ontology Driven Architecture» (ODA, раздел в MDA) [43]. Данный подход предназначен для разработки теории и инструментов построения

программного обеспечения на основе онтологических преобразований. Следовательно, после создания базы правил пользователю предлагается выбрать одно «начальное», которое используется для формирования списка форм ввода. Таким образом, добавляются экземпляры листовых классов лист («Inference Engine», «Fact_Input_Form», «Changed_Facts_Form» and «Fired_Rules_Form»).

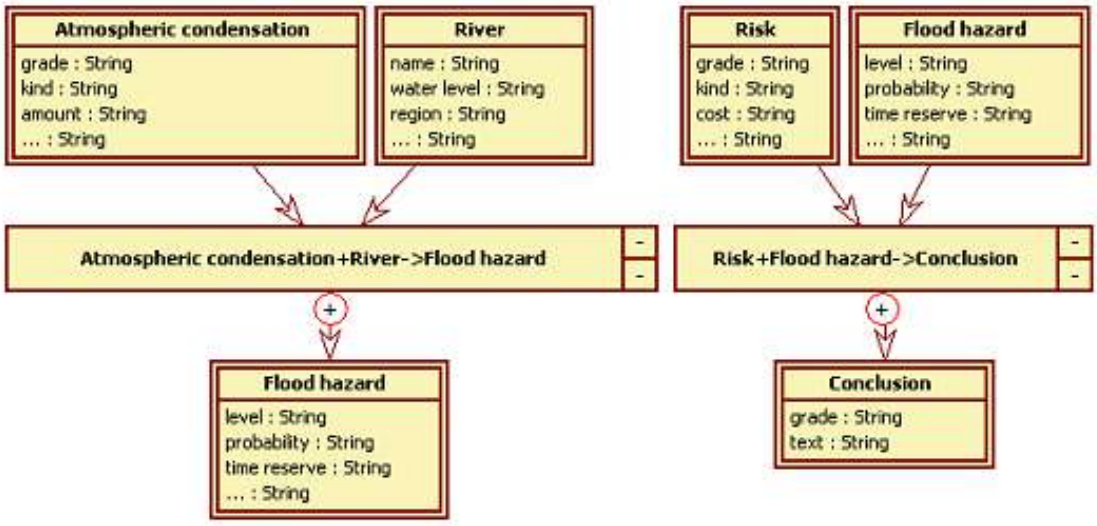


Рис. 4.1.5. Примеры шаблонов логических правил (RVML), полученных в результате преобразования модели прогнозирования речных паводков (Рис. 4.1.3, а)

3) Третьим этапом является построение платформо-зависимых моделей. Количество этих моделей определяется количеством платформ, для которых создается ИС. Платформо-зависимые является результатом автоматических преобразований платформо-независимой модели с помощью специальных инструментов с последующей модификацией конечным пользователем. На данном этапе реализуется оператор $F^{ES}_{PIM-to-PSM}$.

В нашем случае конечный пользователь должен уточнить RVML модели правил с учетом особенностей определенного языка представления/программирования знаний (например, CLIPS), такие как: приоритеты правил, значения слотов «по умолчанию» и коэффициент уверенности (определенности) (Рис. 4.1.6).

4) Четвертым этапом является генерация программных кодов и спецификаций БЗ и ИС. На этом этапе выполняется интерпретация диаграмм, описывающих архитектуру программного обеспечения и БЗ, т.е. на реализуется оператор $F^{ES}_{PSM-to-CODE}$.

В результате выполнения данного этапа пользователь получает:

- код БЗ для определенного языка программирования (например, CLIPS) (Рис. 4.1.7, а);
- спецификации ИС для интерпретатора (Таблицы 4.1.1-4.1.3, Рис. 4.1.7, б).

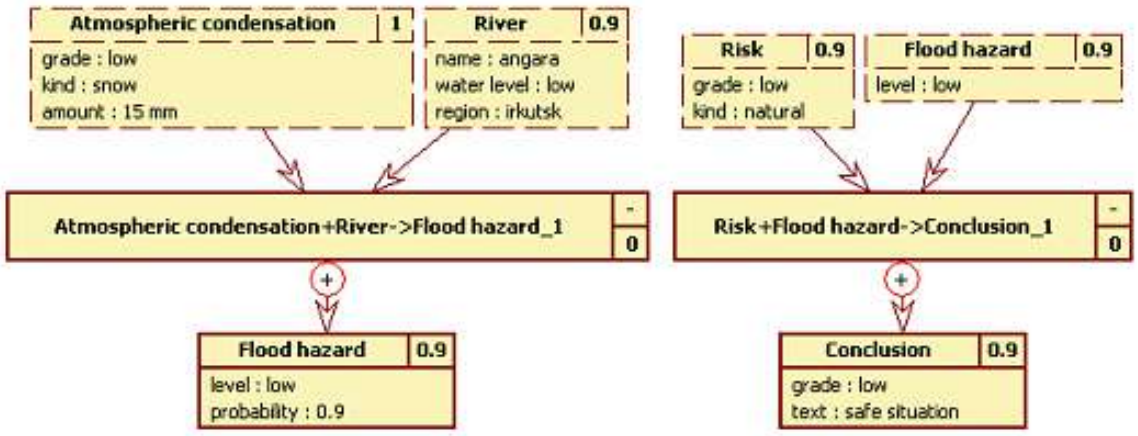


Рис. 4.1.6. Примеры конкретных логических правил (RVML) с коэффициентами уверенности

В примере модель платформы описывает синтаксис CLIPS и требования к спецификациям для интерпретатора PKBD. В этом случае спецификации PKBD и код CLIPS создаются синтаксически корректно (в соответствии со спецификациями платформ) и не зависят от семантического содержания моделей.

Таблица 4.1.1. Структура типизированного файла для описания шаблона факта

Элемент файла	Описание	Пример
[**Metadata**]	Заголовок файла	**[Metadata]**
;description	Описание шаблона	;Наблюдаемые повреждения
tempale_name=<template name>	Имя шаблона, поле используется для синтезирования графического пользовательского интерфеса и программных кодов.	tempale_name=Damage
edited_by_user=<No|Yes>	Инструкция для генератора пользовательского интерфейса, определяющая возможность создания форм редактирования для данного шаблона.	edited_by_user=No
[**Fields**]	Заголовок раздела описания слотов шаблона.	**[Fields]**
<slot_name>=<datatype>	Описание слота, где <slot_name> – внутреннее имя, используемое для генерации графичсекого пользовательского интерфейса и кодов на CLIPS; <datatype> = string | integer | float | <имя_переменной_из_списка значений>, где <имя_переменной_из_списка	dam-type=string dam-orientation=val3:1

Элемент файла	Описание	Пример
	значений> = <значение>; ...; <значение> определяется в разделе «Values».	
[**Captions**]	Заголовок раздела для наименований слотов шаблона, которые отображаются на пользовательских формах и используются для мультиязыковой поддержки.	**[Captions]**
<slot_name> = <name>	Описание слота.	form=Exist damage dam-type=Type of the damage
[**Values**]	Заголовок раздела описания возможных значений слотов.	[**Values**]
<value_name> = <value_1>,…,<value_N>	Описание возможных значений слотов.	Val3:1=LONGITUDINAL, CROSS-SECTION

Таблица 4.1.2. Структура типизированного файла для описания шаблона правила

Элемент файла	Описание	Пример
[Generalized rules]	Заголовок файла	**[Generalized rules]**
#<Name of the process>	Наименование деградационного процесса, который описывается правилом. Данное имя отображается на формах пользователя.	#Corrosion cracking
##<Name of the process modification>	Наименование модификации процесса или подпроцесса. Данное имя отображается на формах пользователя.	##1
<rule_name> = <title> : <template names from the rule conditions> : <template names from the rule actions>	<rule_name> – имя правила, используемое для синтеза элементов пользовательского интерфейса и кодов CLIPS, <title> – имя правила, отображаемое на формах пользователя, <template names from the rule conditions> = <template_name>, …, <template_name>.	fail-mechanism-fail-ky =Rule_for defenition_ of_the_failure_ mechanism: exist-meh-des,exist-des:exist-meh-fail

Таблица 4.1.3. Структура XML файла формата EKB

Элемент файла	Описание
<KnowledgeBase/>	Описание БЗ, содержит основные вложенные элементы: <Templates/> – шаблоны фактов; <Facts/> – факты; <Rules/> – правила; <GRules/> – шаблоны правил.
<Template/>	Описание шаблона, содержит вложенные элементы: <ID/> – идентификатор; <Name/> – имя для форм пользователя; <ShortName/> – имя для генератора форм пользователя и кода CLIPS; <Description/> – описание; <Slots/> – слоты.
<Slot/>	Описание слота, содержит вложенные элементы: <Name/> – имя для форм пользователя; <ShortName/> – имя для генератора форм пользователя и кода CLIPS; <Description/> – описание; < Value/> – значение, в зависимости от принадлежности слота шаблону или факту, это может быть либо значение «по умолчанию», либо фактическое значение; < DataType/> – тип данных; < Constraint/> – ограничение на значение.

Элемент файла	Описание
<Fact/>	Описание факта анлгчично описанию шаблона.
<GRule/>	Описание шаблона правила, содержит вложенные элементы: <ID/> – идентификатор; <Name/> – имя для форм пользователя; <ShortName/> – имя для генератора форм пользователя и кода CLIPS; <Description/> – описание; < Salience/> – важность правила или его приоритет; < Conditions/> – описание условий; < Actions/> – описание действий.
<Rule/>	Описание правила аналогично описанию шаблона правила.

```
(defrule Atmospheric-condensation+River->Flood-hazard-1
"Description of the rule: Atmospheric condensation+River->Flood hazard 1"
(Atmospheric-condensation
(grade "LOW")
(kind "SNOW")
(amount "15 MM")
(cf "1")
)
(River ;River
(name "ANGARA")
(water-level "LOW")
(region "IRKUTSK")
(cf "0.9")
)
=>
(assert
(Flood-hazard ;Flood hazard
(level "LOW")
(probability "0.9")
(cf "0.9")
))
)
```

а)

```
<Structure><KnowledgeBase><ID>6348265468</ID><Name>Knowledge base 6348265468</Name><ShortName></ShortName><Kind>0</Kind><Description></Description><Vars/><Templates><Template><ID>T001</ID><Name>Atmospheric condensation</Name><ShortName>Atmospheric-condensation</ShortName><Description></Description><PackageName></PackageName><RootPackageName></RootPackageName><DrawParams></DrawParams><Slots><Slot><Name>grade</Name><ShortName>grade</ShortName><Description></Description><Value>low</Value><DataType>String</DataType><Constraint></Constraint></Slot><Slot><Name>kind</Name><ShortName>kind</ShortName><Description></Description><Value>snow</Value><DataType>String</DataType><Constraint></Constraint></Slot><Slot><Name>amount</Name><ShortName>amount</ShortName><Description></Description><Value>15 mm</Value><DataType>String</DataType><Constraint></Constraint></Slot> …
```

б)

Рис. 4.1.7. Примеры кода CLIPS (а) и XML-подобных спецификаций для PKBD (б), созданных для платформо-независимых и зависимых моделей (Рис. 4.1.5, 4.1.6).

5) Пятым этапом является тестирование, связанное, в том числе с проверкой корректности полученных структур знаний.

Поскольку генерация кода полностью автоматизирована и синтаксически корректна, конечный пользователь может проверить только семантическую корректность разработанных моделей, «выполнив» их для разных значений исходных фактов.

Критерием тестирования является правильность логического вывода и корректность его результатов.

Следует отметить, что конечные пользователи (например, эксперты предметной области или аналитики) активно участвуют в разработке

вычислительно-независимой и платформо-независимой моделей и лишь частично платформо-зависимой. Все преобразования моделей и генерация программных кодов осуществляются с помощью специализированного программного обеспечения, включающего модель платформы.

Описанная последовательность этапов почти полностью совпадает с классическим MDA/MDE подходом, но содержание этапов переопределяется на основе особенностей процесса проектирования БЗ и ИС. В случае создания прецедентных ИС и БЗ общие принципы подхода и основные этапы остаются без изменений [280]. Далее подробнее рассмотрим описанные модели.

4.1.3 Модели метода

Описание моделей и их преобразования важны для MDA/MDE. Представим модели в теоретико-множественной форме.

Вычислительно-независимая модель (CIM) может быть представлена в виде онтологий (Рис. 4.1.8) и описана следующим образом:

$$CIM^{ES} = \left\langle Ont^{D}, Ont^{RB\_ES} \right\rangle,$$

где Ont^{D} – онтология предметной области (например, надежности технических систем); Ont^{RB_ES} – онтология продукционных ЭС, включающая описание основных архитектурных элементов, необходимых для реализации предложенного подхода.

Онтология предметной области Ont^{D} включает понятия (т.е. классы и экземпляры классов) и отношения между ними.

Следующие выражения представляют основные понятия Ont^{D} в расширенной нотации Бакуса-Наура (РНБН):

```
<ONT_D> = <Class> {<Class>}, <Relationship> {<Relationship>}
<Class> = <Class name>, (<Property> {<Property>}).
<Property> = <Property name>, <Property variable name>,
<Property data type>, <Property value>, <Property constraint>, <Units of
measurement>.
<Property data type> = <Class name> | <Set> | ∅.
<Property value> = <Object> | <Set> | ∅.
<Property constraint> = <Class name> | <Set>.
<Set> = <Discrete set> | <Dense set>.
<Dense set> = <Set element>, <Set>.
<Set element> = <ID>.
<Dense set> = <Interval> | <Interval part> | <Segment>.
We define the concept of the object in relation to the class as follows:
<Object> = <Class name>, <ID>, (<Object property> {<Object property>}).
<Object property> = <Property>, <Property value>.
Next, we define the concept of a 'relationships':
```

```
      <Relationship> = <Left side concept>, <Relation>, <Right side concept>.
      <Relation> = <Relation type>, <Relation name>.
      <Relation type> = <is-a> | <is-part-of> | <depends-on>.
      <Left concept> = <Class> | <Object>.
      <Right side concept> = <Class> | <Object>|<Property>.
      <Concept relation>:<is-a> = <Left side concept>:<Class>,
<Relation>:<Relation type>:<is-a>, <Right side concept>:<Class>.
      <Concept relation>:<is-part-of> = <Left side concept>:<Class>,
<Relation>:<Relation type>:<is-part-of>, <Right side concept>:<Property>.
      <Concept relation>:<depends-on> = <Left side concept>:(<Class> |
<Object>), <Relation>:<Relation type>:<depends-on>, <Right side
concept>:(<Class> | <Object>).
```

Онтология продукционных ЭС представляет собой следующее выражение:

$$Ont^{RB\_ES} = \langle KB, Int, Rsng, GUI \rangle,$$

где *KB* – информация о БЗ; *Int* – интерпретатор (машина вывода); *Rsng* – описание стратегии принятия решений в виде цепочки рассуждений, которая связывает все сформированные ранее понятия и отношения в динамическую систему поля знаний; *GUI* – графический пользовательский интерфейс ЭС.

Следующие выражения определяют основные понятия Ont^{RB_ES}:

```
      <ONT_RB_ES> = <Expert system>
      <Expert system> = <Knowledge base>, <Interpreter>, <Decision-making
strategy>>, <GUI-Form set>
      <Knowledge base> = <Knowledge base name>, (<Rule_ONT> {<Rule_ONT>})
      <Rule_ONT> = <Condition>, <Action>.
      <Condition> = <Condition element>.
      <Action> = {<Action element>}.
      <Condition element> = (<Condition operator>, <Constant>{<Constant>}) |
(<Condition operator>, <Condition element>).
      <Constant> = <Class>:<Class with constraints>, <Certainty factor (CF)>.
      <Property> = <Property name>, <Property constant>, <Priority (P)>.
      <Certainty factor> = [0, 1].
      <Importance> = [1, 100].
      <Action element> = <Action operator>, <Fact>{<Fact>}.
      <Action operator> = Add | Delete | Modify.
    <Condition operator> = AND | OR | NOT.
    <Fact> = <Object>.
    <Interpreter> = Clips | Jess | … .
    <Decision-making strategy> = (<Rule ID>, <Rule>), {<Rule ID>, <Rule>}
    <GUI-Form set> = (<Form ID>, <Form>), {<Form ID>, <Form>}
    <Form> = <Form name>, <Form type>, (<Form property> {<Form property>})
    <Form type> = <Input form>|<Output form>
    <Output form> = <Results form>|<Explanation form>
    <Form property> = <Object>
```

Семантика онтологий зависит от проблемы и предметной области.

Платформо-независимая модель описывается, в свою очередь, двумя моделями (Рис. 4.1.9) и может быть представлена следующим образом:

$$PIM^{ES} = \langle UML^{RB\_KB}, UML^{RB\_ES} \rangle,$$

где UML^{RB_KB} – модель базы знаний; UML^{RB_ES} – модель архитектуры ЭС.

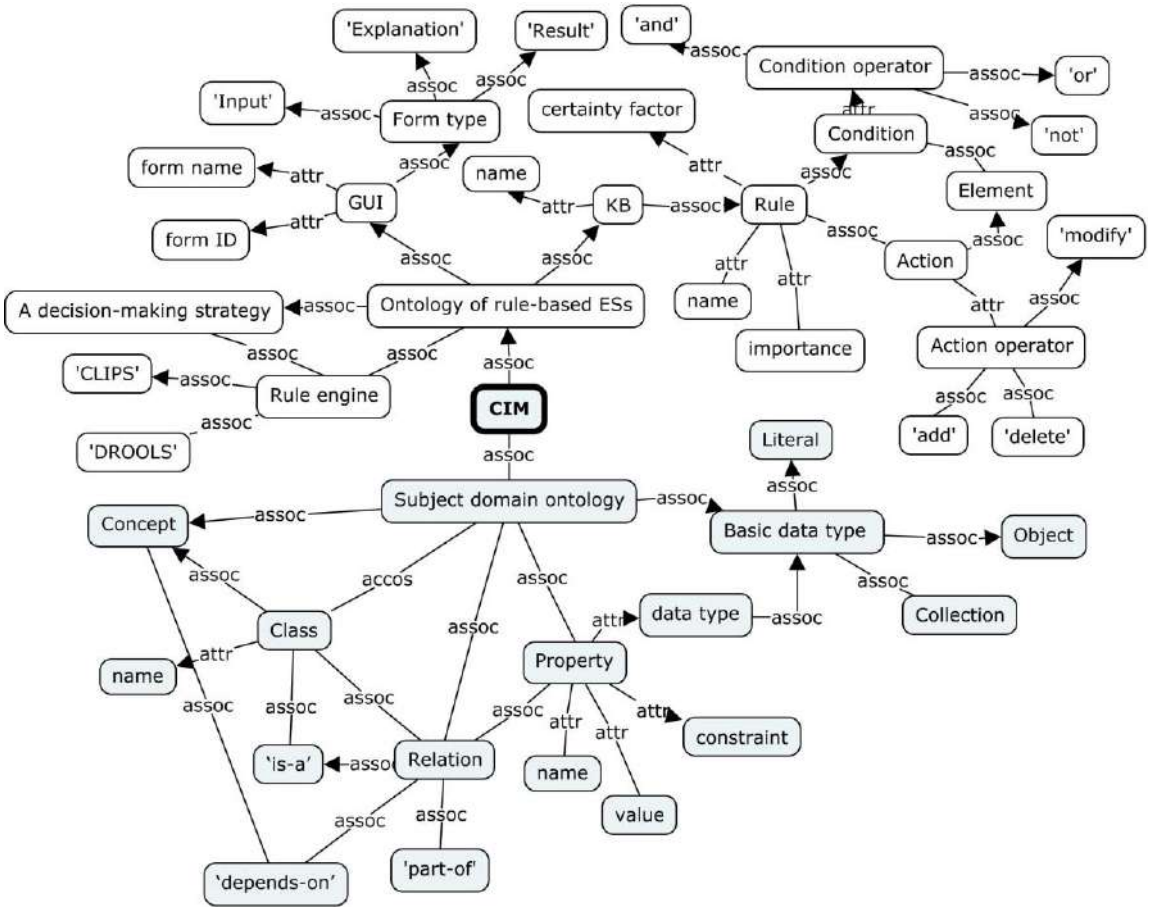


Рис. 4.1.8. Структура вычислительно-независимой модели для прототипирования продукционных интеллектуальных систем и баз знаний

Диаграммы классов UML с дополнительными классами (например, «Input From Class», «Output From Class»), которые расширяют стандартные классы «Border Class», «Entity» и «Control» используются для представления и моделирования архитектуры ЭС.

Следующие выражения определяют основные понятия UML^{RB_ES}:

```
<UML_RB_ES> = <Class>+
<Class> = <Border Class> | <Entity> | <Control>
<Border Class> = <Input Form Class> | <Output Form Class>
<Input Form Class> = Facts input form
<Output Form Class> = Results form | Explanation form
<Control> = CLIPS interpreter.
```

Нотация RVML используется для платформо-независимого моделирования логических правил. Эта нотация позволяет описывать причинно-следственные связи и абстрагироваться от особенностей языков программирования на основе правил.

Кроме того, уточнение некоторых элементов нотации (таких как приоритет и коэффициент уверенности) обеспечивает возможность создания платформо-зависимой модели, в частности для CLIPS.

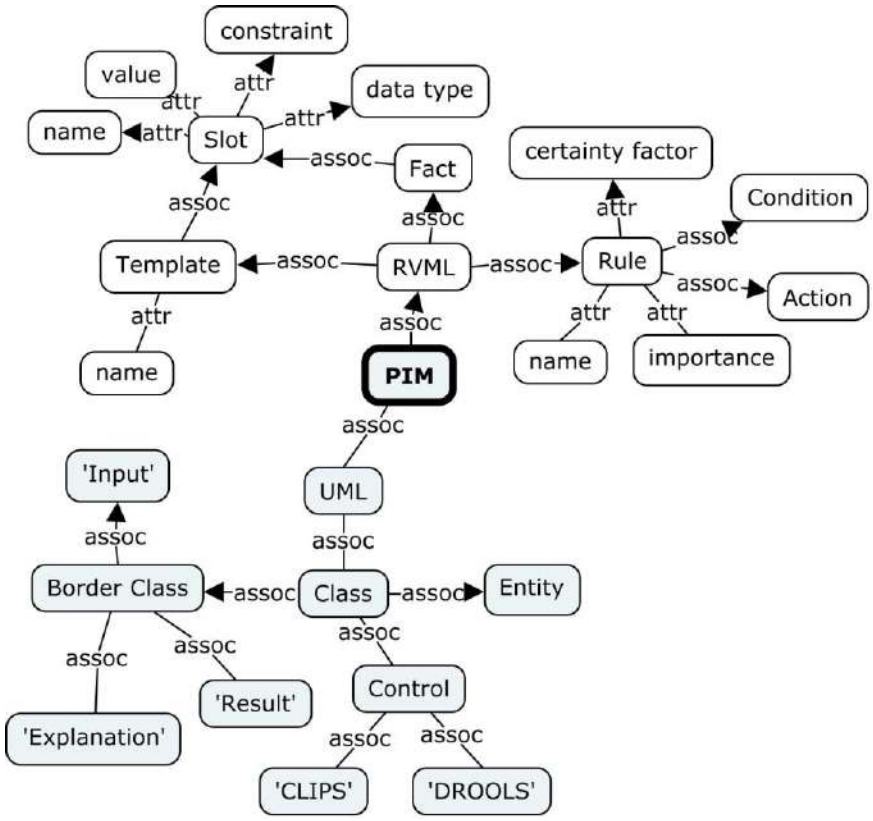


Рис. 4.1.9. Структура платформо-независимой модели для прототипирования продукционных интеллектуальных систем

В качестве модели описания платформы выбран CLIPS. Следующие выражения представляют основные элементы CLIPS в РФБН:

```
      <CLIPS> = <template>*, <fact>*, <rule>*.
      <template> = (deftemplate <template-name>     [<optional-comments>]
[<slot-definition>*]).
      <slot-definition> = <simple-slot-definition> | <composite-slot-
definition>.
      <simple-slot-definition> = (slot <slot-name> <template-attributes>).
      <template-attributes> = <attribute-default-value> | <restriction-
attribute>
      <restriction-attribute> = <type-attribute>
      <type-attribute> = (type <type-specification>)
      <type-specification> = float | integer | symbol | string | external-
address | fact-address | instance-name | instance-address
      <fact> = (deffacts <facts-list-name> [<optional-comments>] [<fact>*])
      <rule> = (defrule <rule-name> <comment>]  [<rule-property- definition>]
                  <antecedent>   ; rule LHS
                  =>
                  <consequent>   ; rule RHS)
```

Более подробное описание представлено в работе [356].

4.1.4 Трансформации моделей

Для обеспечения выполнения рассмотренной последовательности этапов и формирования представленных моделей необходимо реализовать последовательность экзогенных горизонтальных трансформаций:

- M2M-трансформацию для $F^{ES}_{CIM-to-PIM} : CIM \rightarrow PIM$;

- M2M-трансформацию для $F^{ES}_{PIM-to-PSM} : PIM \to PSM$;
- M2C-трансформацию для $F^{ES}_{PSM-to-CODE} : PSM \to CODE$.

Ниже приведен фрагмент правил преобразования (трансформации) в РФБН:

```
<Transformation> = <Transformation rule> {<Transformation rule>}.
<Transformation rule> = Rule <name> {<Source model element>,
<Result>}.
<Source model element> = <ONT_D element> | <ONT_RB_ES element> |
<UML_RB_ES element> | <UML_RB_KB element>.
<Result> = <UML_RB_ES element>|<UML_RB_KB element>|<CLIPS
element>|<GUI element>.
```

Где <ONT_D element>, <ONT_RB_ES element>, <UML_RB_ES element>, <UML_RB_KB element>, <CLIPS element>, <GUI element> элементы онтологии предметной области, онтологии продукционных ЭС, UML-моделей продукционных ЭС, правил в форме RVML, CLIPS-моделей и моделей графического пользовательского интерфейса, соответственно.

Таким образом, используются следующие шаблоны правил преобразования:

$T^{ES}_{CIM-to-PIM}$ = {**Rule** ONT_D-TO-UML_RB_KB; **Rule** ONT_RB_ES-TO-UML_RB_ES},

$T^{ES}_{PIM-to-PSM}$ = {**Rule** UML_RB_KB-TO-UML_RB_KB*}, $T^{ES}_{PSM-to-CODE}$ = {**Rule** UML_RB_KB*-TO-CLIPS; **Rule** UML_RB_ES-TO-GUI}:

```
Rule ONT_D-TO-UML_RB_KB {<ONT_D element>, <UML_RB_KB element>}.
Rule ONT_RB_ES-TO-UML_RB_ES {<ONT_RB_ES element>, <UML_RB_ES
element>}.
Rule UML_RB_KB*-TO-CLIPS {<UML_RB_KB* element>, <CLIPS element>}.
Rule UML_RB_ES-TO-GUI {<UML_RB_ES element>, <GUI element>}.
```

Ниже приведены правила преобразования, которые мы используем для:

- ONT_D-TO-UML_RB_KB:

```
Rule Class-Template {<Class>, <Template_UML_RB_KB>}.
Rule Object-Fact {<Object>, <Fact_UML_RB_KB>}.
Rule is-a-Slot {<Relation>:<Relation type>:<is-a>, <Slot>}.
Rule Class-Property-Slot {<Relation>:<Relation type>:<is-part-
of>, <Slot>}.
Rule Cause-Rule {<Relation>:<Relation type>:<depends-on>,
<Rule_UML_RB_KB>}.
Rule Property-Slot {<Property>, <Slot>}.
Rule Value {<Property value>, <Slot value>}.
```

- ONT_RB_ES-TO-UML_RB_ES:

```
Rule Form-UML {<Form>, <Border Class>}.
Rule Knowledge_base-UML_RB_ES {<Knowledge base>, <Entity>}.
Rule Interpreter-UML_RB_ES {<Interpreter>, <Control>}.
```

- UML_RB_KB*-TO-CLIPS:

```
Rule Template-CLIPS {<Template_UML_RB_KB*>, <template>}.
Rule Fact-CLIPS {<Fact_UML_RB_KB*>, <fact>}.
Rule Rule-CLIPS {<Rule_UML_RB_KB*>, <rule>}.
Rule Slot-CLIPS {<Slot_UML_RB_KB*>, <slot-value>}.
Rule Value-CLIPS {<Slot_value_UML_RB_KB*>}.
```

- UML_RB_ES-TO-GUI:

```
Rule Border_Class-GUI {<Border Class>, <GUI Border Class>}.
Rule Entity-GUI {<Entity>, <GUI Entity>}.
Rule Entity-GUI {<Control>, <GUI Control>}.
```

Где `<GUI Border Class>`, `<GUI Entity>`, `<GUI Control>` элементы пользовательского интерфейса для классов с соответствующими стереотипами.

Элементы моделей представлены в таблице (Табл. 4.1.4).

Таблица 4.1.4. Фрагмент таблицы отображения элементов моделей (вычислительно-независимой модели в платформо-независимую и платформо-независимой в CLIPS)

Элементы онтологии (вычислительно-независимая модель - Ont^{D})	**Элементы БЗ (платформо-независимая модель – UML^{RB_KB})**	**Элементы CLIPS кода**
Project (name, description)	Knowledge base (name, description)	-
Class (name, description)	Template (name, description)	deftemplate
Object (name, description)	Fact (name, description)	deffacts
Method	-	-
Property	Slot (description, value)	slot
Property value	Slot value	default
Property type	Slot type	type
Relationship	Rule (nodal element)	defrule

Правила преобразования моделей могут быть реализованы в виде спецификаций на императивном языке программирования, либо в декларативной форме с использованием TMRL. Реализованные спецификации отвечают требованиям полноты, формальности и гибкости [147, 168]. Данные спецификации содержат всю необходимую (в рамках предлагаемого подхода) информацию для решения поставленной задачи, все объекты модели хорошо формализованы, при этом спецификации достаточно компактны и понятны (читабельны).

4.1.5 Преимущества нового метода проектирования баз знаний

- Использование EUD подходов, в частности: визуального программирования и модельно-ориентированной разработки в контексте создания ИС и БЗ.
- Вовлечение в процесс создания программных систем вычислительно-независимой модели.

- Использование оригинальных языков RVML и TMRL для описания элементов баз знаний и моделей трансформаций.
- Использование концепт-карт, онтологий, деревьев событий, канонических таблиц и таблиц решений при описании вычислительно-независимых моделей.

Сравнение предлагаемого метода и включающей его технологии с другими технологиями разработки интеллектуальных систем и баз знаний приведено в Таблице 1.2.1 (см. 1.2).

4.2 Программные средства проектирования декларативных баз знаний интеллектуальных систем

Для реализации предлагаемого метода создано оригинальное программное обеспечение: Personal Knowledge Base Designer (PKBD, рег.№№ 2016617733, 2012614093, 2007613714) – система разработки декларативных баз знаний и интеллектуальных систем и ее веб версия: Web PKBD; TreeEditorET/Extended Event Tree Editor (EETE, рег.№ 2012614092) – система визуального проектирования баз знаний на основе деревьев событий

4.2.1 Personal Knowledge Base Designer (PKBD)

Personal Knowledge Base Designer (PKBD) [184, 185, 195, 202, 257, 258, 269, 283, 367] представляет собой систему проектирования и прототипирования деларативных ИС и ЭС.

В качестве основных функций PKBD выделены следующие:

- создание элементов продукционных БЗ (шаблонов фактов и правил, а также фактов и правил) непрограммирующим пользователем, благодаря использованию набора подпрограмм-мастеров, предварительно подготовленных шаблонов фактов и правил, а также обобщенной модели продукций, которая позволяет абстрагироваться от особенностей их описания в разных ЯПБЗ;
- использование авторской нотации RVML (Rule Visual Modeling Language) [368] для визуального представления логических правил (продукций);
- интеграция с системами концептуального моделирования (IBM Rational Rose, StarUML, CmapTools, XMind, EETD [287] и др.) в части импорта моделей;

- интеграция с платформой TabbyXL [148, 197] в части импорта канонических таблиц;
- интеграция с CLIPS [356], в части синтеза отчуждаемого программного кода БЗ, а также его тестирования, путем включения в состав модулей программной системы машины вывода CLIPS;
- интеграция с программной системой KBDS [281, 283, 366] в части доступа к разработанным компонентам-конверторам и БЗ, используя ее как облачный веб-сервис;
- генерация программных кодов на CLIPS, DROOLS и PHP, а также CFM спецификаций;
- интепретация CFM спецификаций и создание прототипов интеллектуальных систем;
- возможность функционирования в режиме «проблемно-ориентированный редактор», используя предварительно разработанные описания шаблонов фактов и правил [230] и ограничивая возможность их изменения;
- формирование специализированных отчетов.

Для реализации функций разработана архитектура (Рис. 4.2.1) [363], включающая следующие основные модули:

- управления базами знаний – обеспечивает загрузку и сохранение БЗ в формате EKB – XML-подобный формат программной системы для хранения знаний, а также внутреннее представление продукционной модели знаний, которое не зависит от определенного ЯПБЗ, и манипулирование (создание, удаление, редактирование) элементами этой модели;
- управления модулями поддержки ЯПЗ – обеспечивает подключение и отключение модулей ЯПБЗ, а также доступ к их функциям;
- интеграции с источниками концептуальных моделей – обеспечивает загрузку информации о понятиях и отношениях из файлов различных форматов: деревьев событий EETE (формат xml), диаграмм классов IBM Rational Rose (формат mdl), диаграмм классов StarUML (форматы mdj и xml), концепт-карт и диаграмм Исикавы XMind (формат xmind), концепт-карт CMapTools (форматы cxl и xtm), онтологий Protégé (формат owl), канонических таблиц TabbyXL (формат csv), баз знаний CLIPS (формат clp);

- управления машинами вывода – обеспечивает использование машин вывода (в виде динамических библиотек) для тестирования БЗ, включая объяснение полученных результатов;
- графический пользовательский интерфейс – обеспечивает доступ к перечисленным функциям, а также функционирование системы в режиме «проблемно-ориентированный редактор».

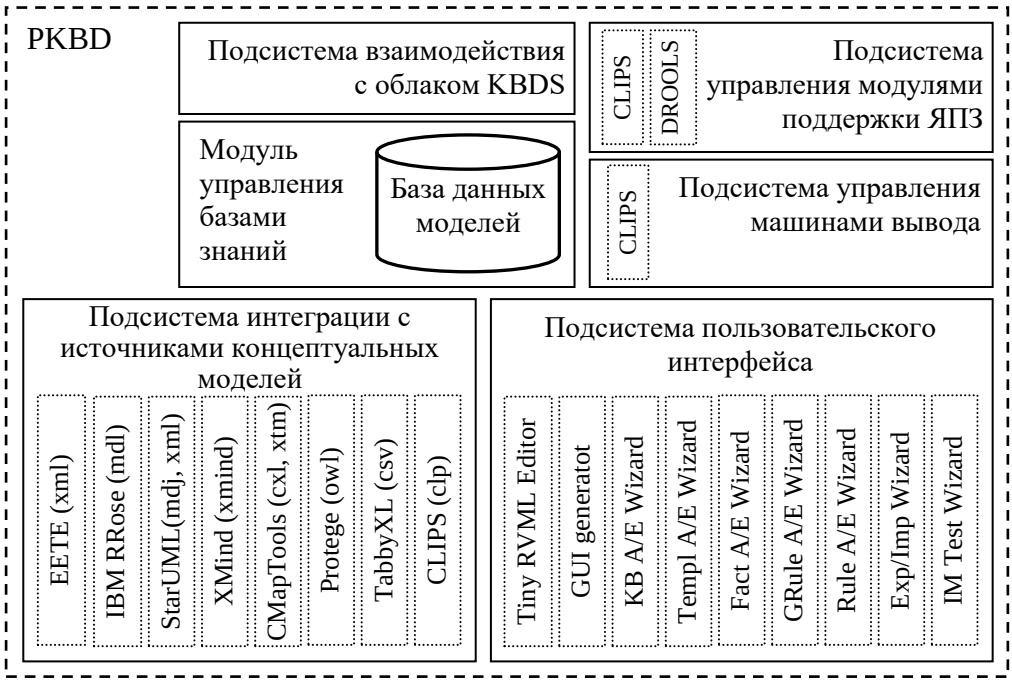


Рис. 4.2.1. Архитектура PKBD

Важным элементом данной программной системы является использование обобщенной модели продукций (рассмотрена выше) для внутреннего хранения и представления знаний, включающей понятия: база знаний, шаблон, факт, слот, правило, условие, действие и т.д. Разработанная модель позволяет абстрагироваться от особенностей описания продукций в разных языках представления знаний и хранить знания в собственном независимом формате.

Пример графического интерфейс пользователя PKBD представлен на Рис. 4.2.2-4.2.9.

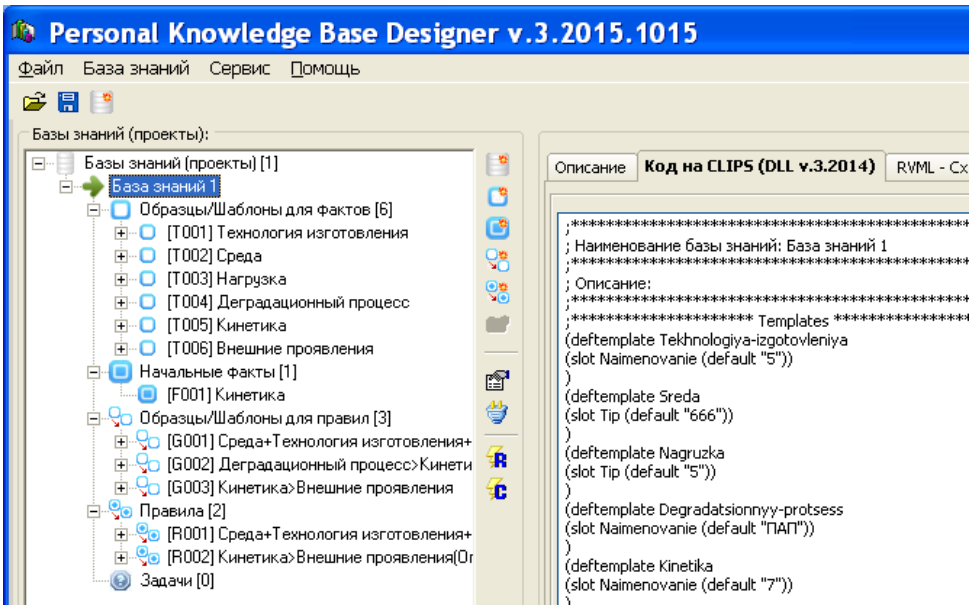


Рис. 4.2.2. Пользовательский интерфейс PKBD: дерево проектов

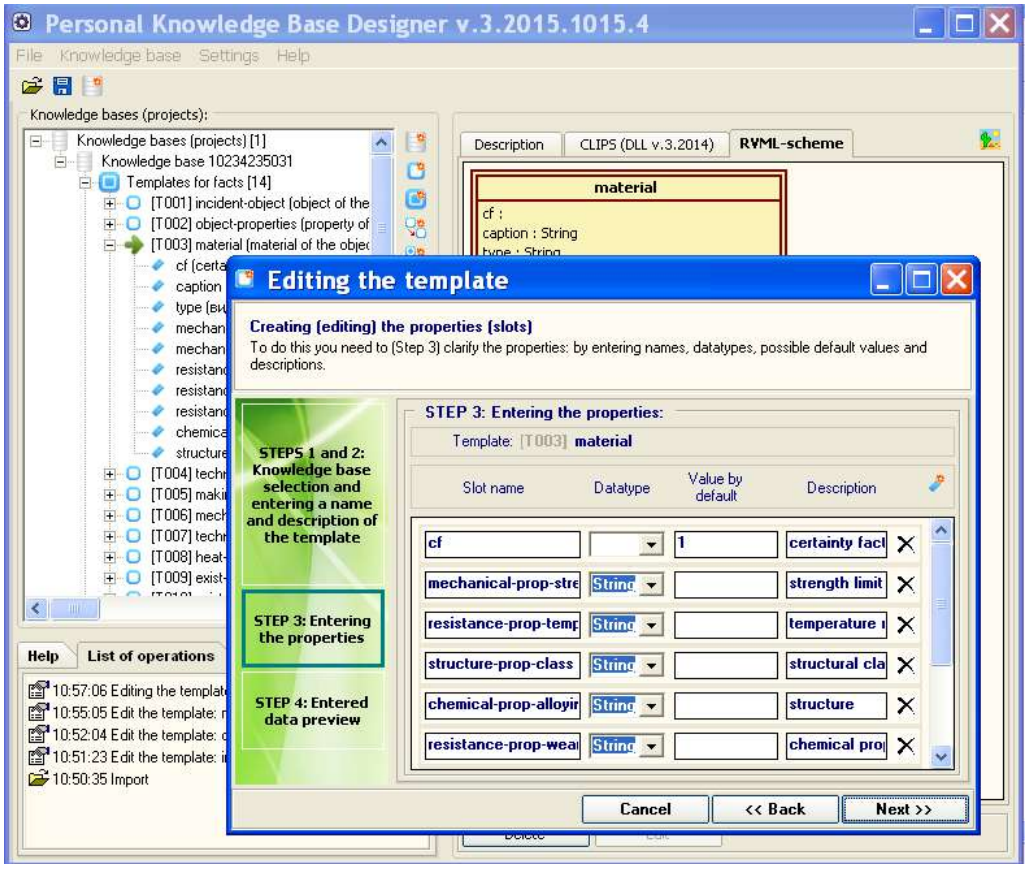


Рис. 4.2.3. Пользовательский интерфейс PKBD: добавление шаблонов фактов

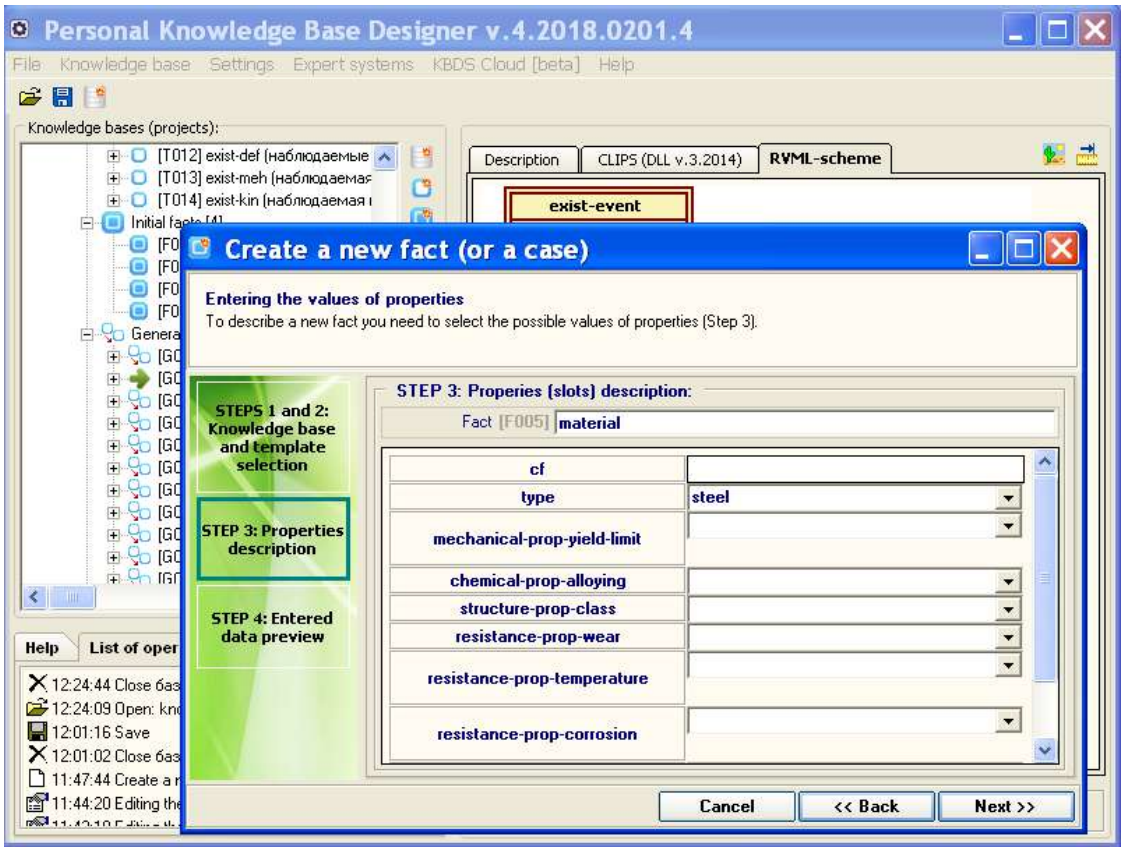


Рис. 4.2.4. Пользовательский интерфейс PKBD: добавление начальных фактов

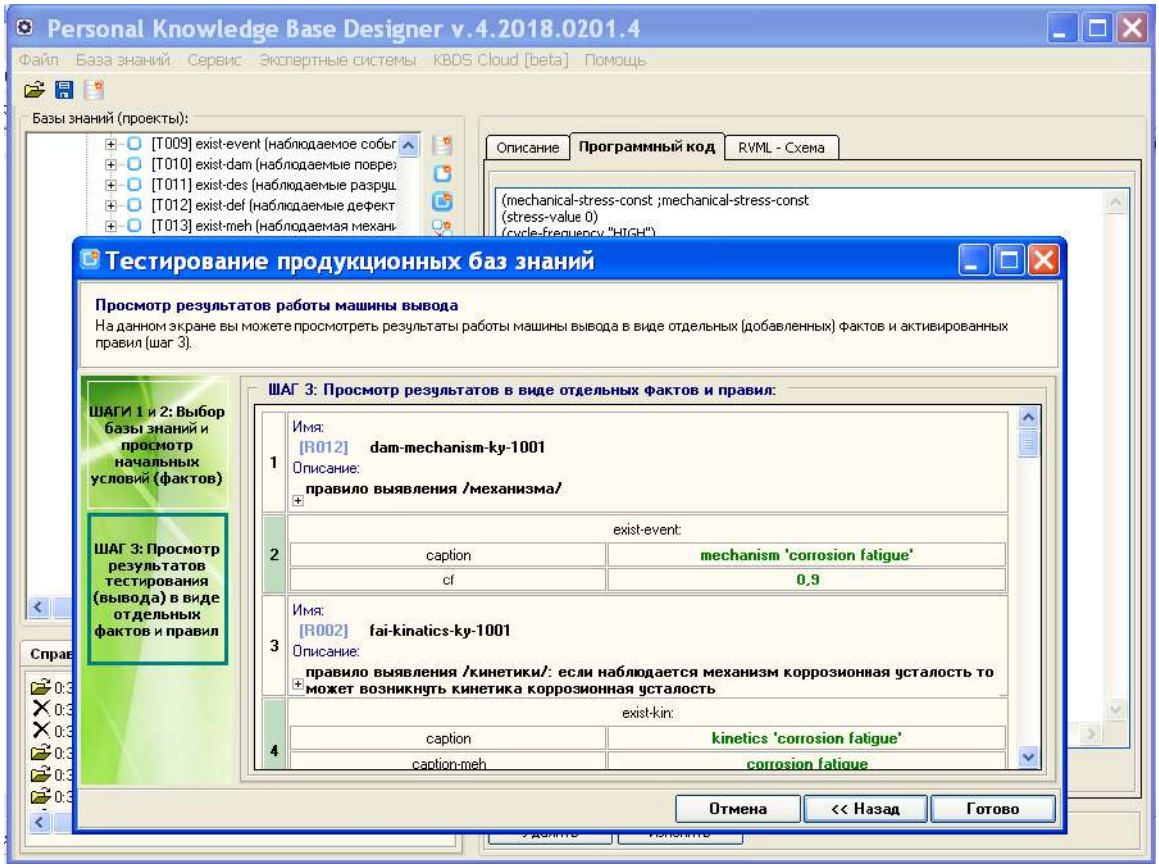


Рис. 4.2.5. Пользовательский интерфейс PKBD: просмотр активированных правил

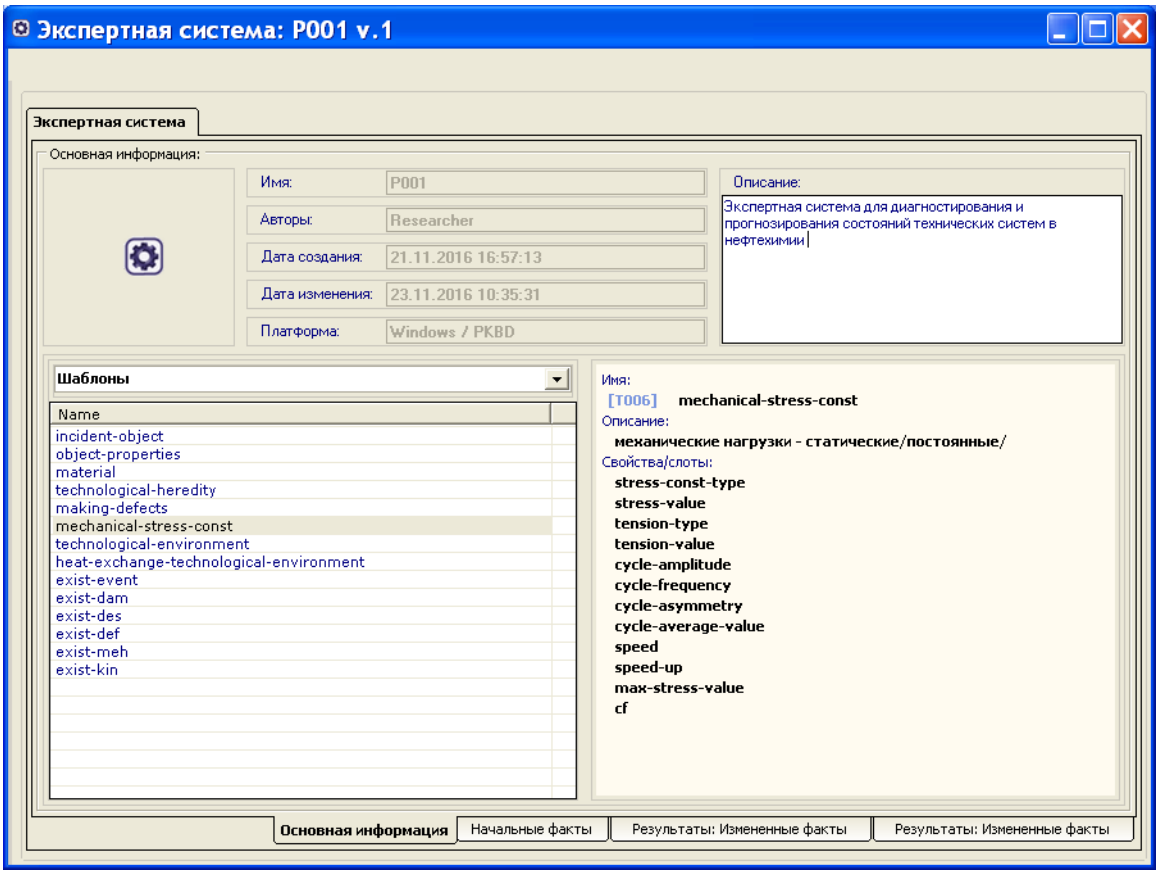


Рис. 4.2.6. Экранная форма созданной экспертной системы: просмотр общей информации

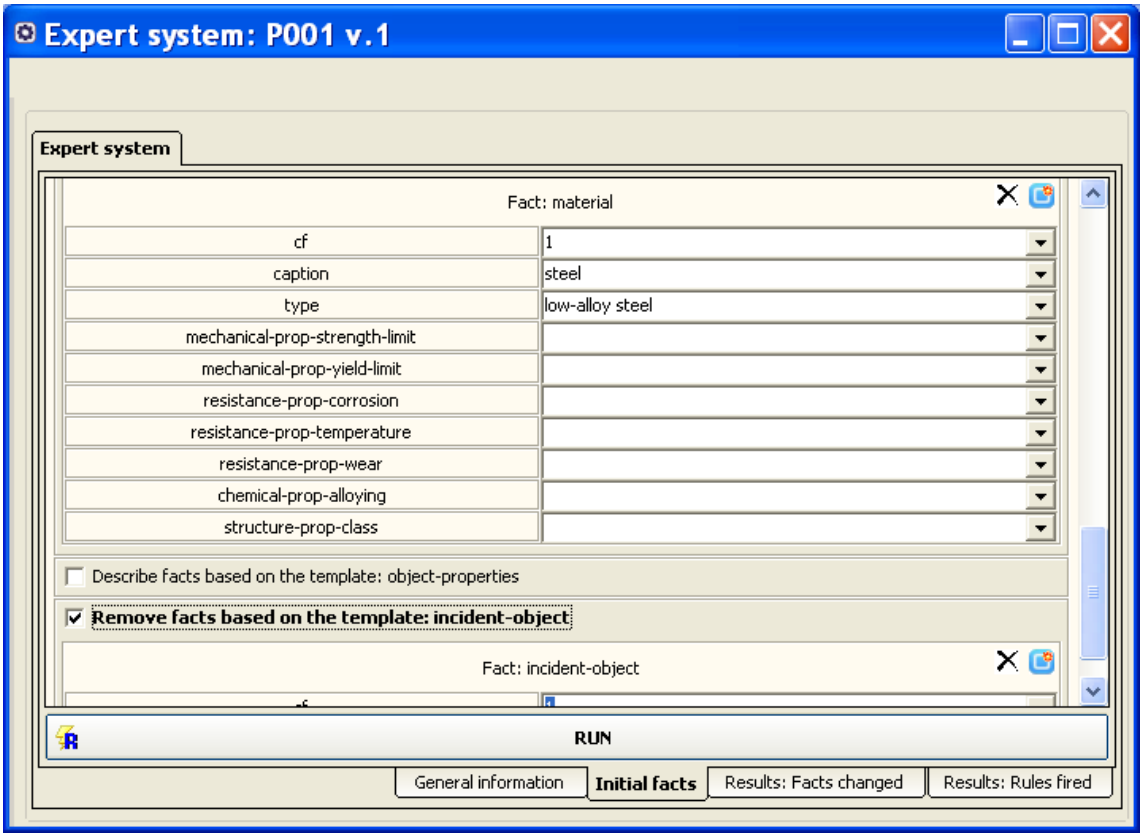


Рис. 4.2.7. Экранная форма созданной экспертной системы: просмотр начальных фактов

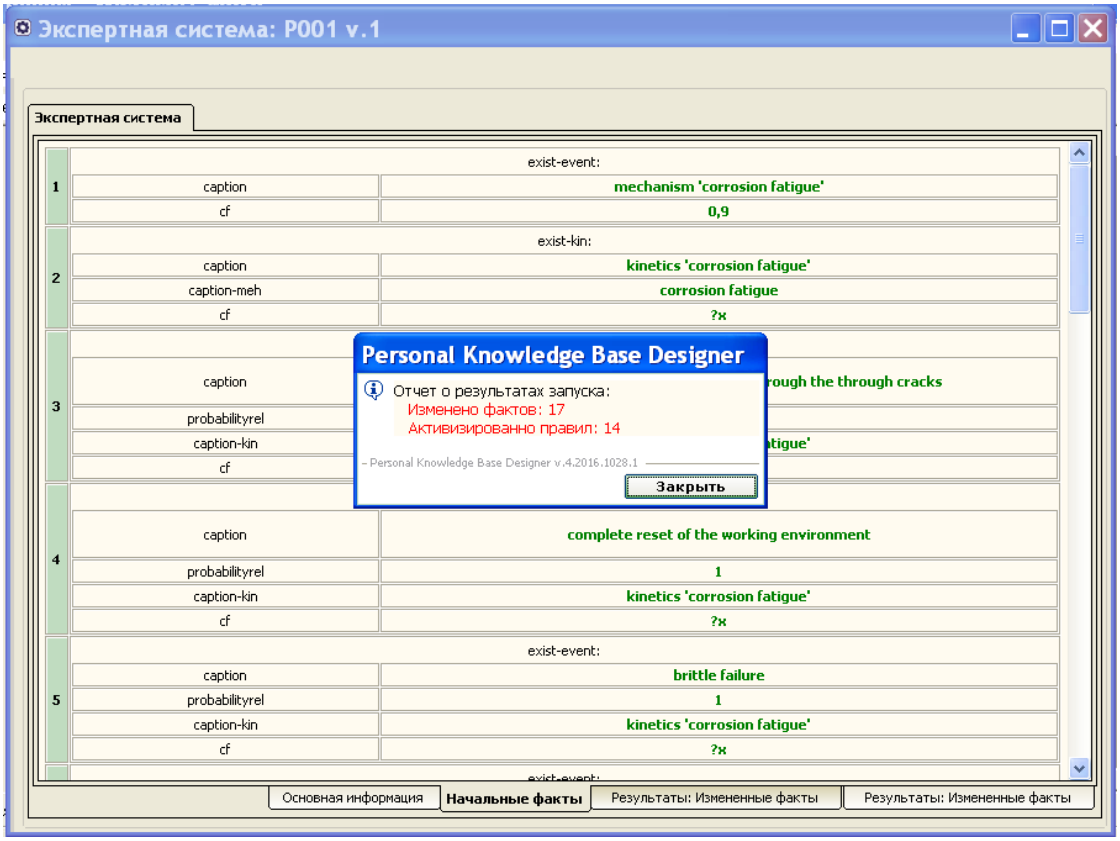


Рис. 4.2.8. Экранная форма экспертной системы: результаты логического вывода

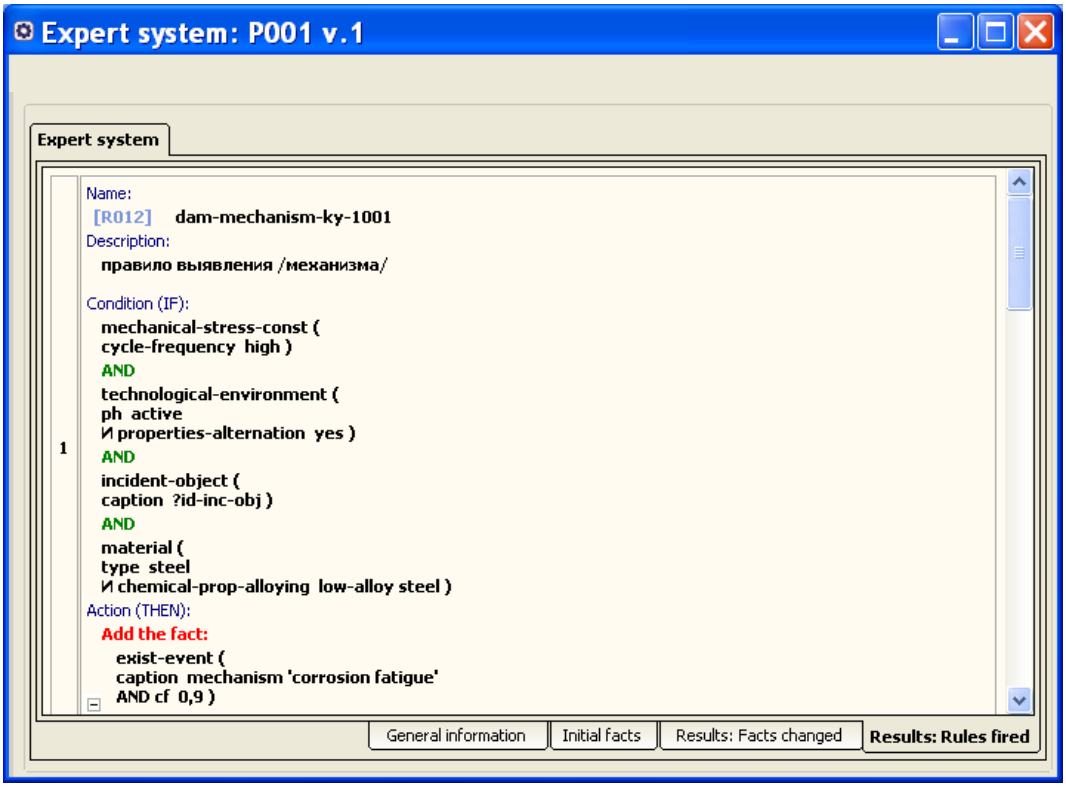


Рис. 4.2.9. Экранная форма экспертной системы: просмотр активированных правил

Использование спецификаций и их интерпретация обеспечивает функционирования PKBD в режиме не только универсального [255, 363], но и проблемно-ориентированного редактора [230].

Для поддержки данного режима на ранних этапах реализации технологии использовался типизированный удобочитаемый формат спецификаций – CFM (Content Formal Model), представляющих собой платформо-независимые модели, который в дальнейшем был преобразован в XML-подобную спецификацию EKB.

Назначением проблемно-ориентированного редактора является поддержка процесса создания баз знаний продукционного типа непрограммирующим пользователем с учетом особенностей предметных задач. При этом его основной набор функций совпадает с набором функций программной системы. Тестирование редактора осуществлялось на примере задачи оценки технического состояния и остаточного ресурса нефтехимического оборудования [228].

В данном режиме PKBD предусмотрен механизм ролей пользователей, обеспечивающий управление доступом к функциям системы. В частности, выделены две основные роли:

1) «Инженер по знаниям (администратор)» – обеспечивает настройку редактора, которая требует глубоких знаний проблемной области и включает доступ к функциям изменения шаблонов.

2) «Эксперт (специалист-предметник)» – обеспечивает ввод правил, без возможности внесения изменений в шаблоны.

Описание формата CFM приведено в 4.1. CFM описания обрабатываются интерпретатором PKBD и позволяют динамически формировать элементы интерфейса. Данные файлы создаются администратором (инженером по знаниям) однократно перед передачей редактора конечному пользователю (эксперту), но могут быть изменены (откоррктированы) в процессе эксплуатации.

Методика создания ИС и БЗ на основе модельных трансформаций [184, 185, 195, 202, 257, 258, 269, 283, 367] конкретизирует описанный ранее метод в части применения определенного инструментария и схематически представлена на Рис. 4.2.10. При этом содержание этапов может качественно переопределяться в зависимости от особенностей решаемой задачи (предметной области), однако их назначение в части получаемого результата, принципиально не изменяется. Рассмотрим этапы подробнее.

1) Анализ предметной области и выделение основных сущностей и отношений выполняется с использованием комплекса программных средств PKBD

и KBDS. В зависимости от формы представления исходных данных: «0. Информация предметной области» могут быть использованы те или иные модули трансформации: диаграмм классов UML, концепт-карт, канонических таблиц TabbyXL, деревьев событий или таблиц решений при этом результат: «1. Модель предметной области» представляется в форме набора основных понятий и отношений, формирующие вычислительно-независимую модель.

2) Формализация понятий и отношений выполняется с помощью реализованных в PKBD и KBDS подходов EUD: визуального программирования с поддержкой RVML и программ-мастеров. Основной результат данного уточнения – это «2. База знаний», представляющая собой платформо-независимую модель, обеспечивающую формализацию модели предметной области (вычислительно-независимой модели), в частном случае, в форме RVML-схем или таблиц решений.

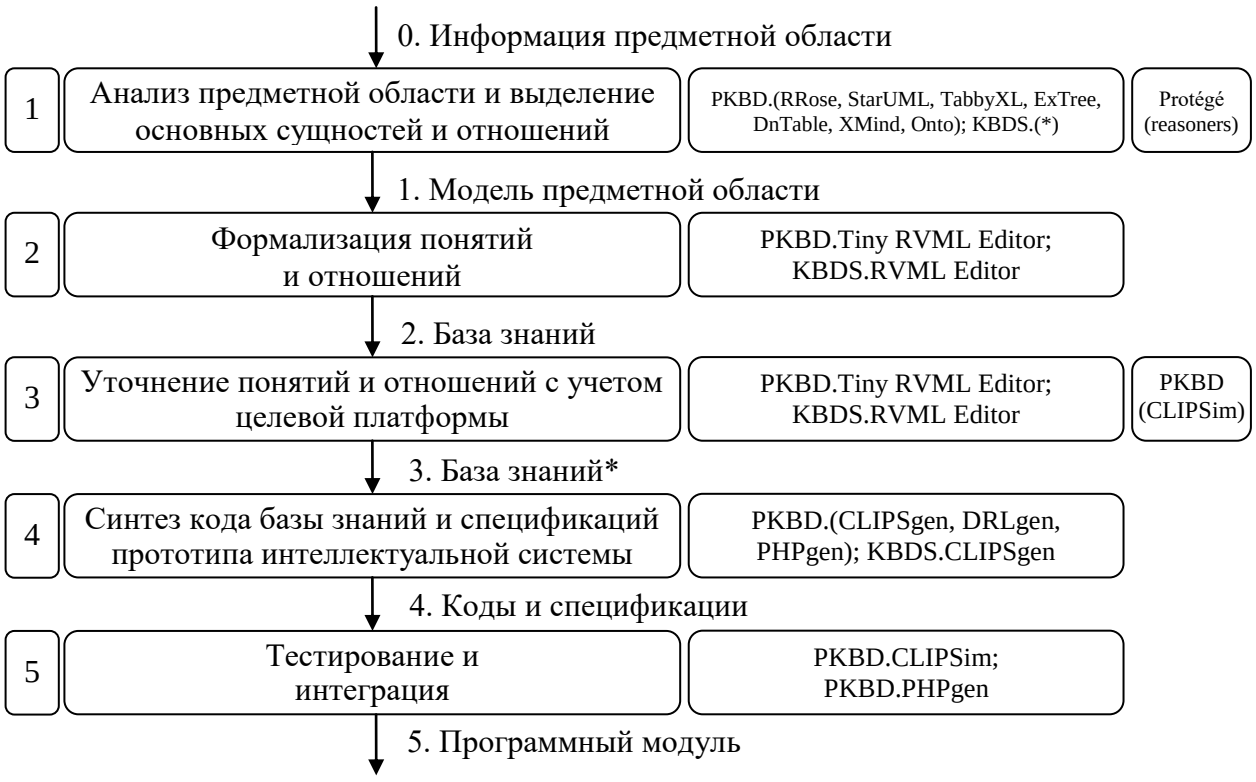


Рис. 4.2.10. Схема методики создания ИС и БЗ на основе модельных трансформаций с указанием основных этапов и программных средств

3) Уточнение понятий и отношений с учетом целевой платформы включает, в том числе, указание типов данных, значений «по умолчанию», приоритетов и пр. Основной результат этапа «3. База знаний*» представляется в форме уточненных RVML-схем, представляющих собой платформо-зависимые модели.

4) Синтез кода базы знаний и спецификаций прототипа интеллектуальной системы выполняется с помощью реализованных в PKBD и KBDS генераторов программных кодов под CLIPS, DROOLS, PHP. Основной результат этапа «4. Коды и спецификации».

5) Тестирование и интеграция осуществляется средствами PKBD за счет использования встроенной машины вывода, обеспечивающей «прогонку» полученных БЗ на предмет выявления логических ошибок. Интеграция осуществляется вне средств разработки и в некоторых случаях не требуется (если исполнение ЭС осуществляется в интерпретаторе PKBD), поэтому данный этап может быть пропущен, а тестирование совмещено с предыдущим этапом 4.

В методике предложена реализация методов верификации для двух форм представления знаний: онтологий и систем продукций. Для онтологий рекомендуется использование машин вывода (reasoners) Protégé на этапе 1, для этого полученные модели предметной области должны быть выгружены из среды разработки в формат OWL и затем загружены во внешний инструментарий (Protégé). Для систем продукций – интегрированной в инструментарий машины вывода в форме динамической библиотеки на этапе 3.

Достаточно подробно применение данной методики и различные варианты исполнения ее этапов рассмотрено в разделе 3 и на сайте www.knowledge-core.ru [93].

Особенностями разработанной программной системы являются:

- Использование принципов визуального программирования в контексте разработки БЗ.
- Использование преобразований (трансформаций) концептуальной модели.
- Абстрагирование от конкретных языков программирования БЗ (например, CLIPS, JESS, Drools, SWRL и т.д.) путем использования унифицированного внутреннего представления в виде продукционной модели.
- Поддержка генерации отчуждаемых кодов БЗ и спецификаций ЭС.
- Ориентация на конечных пользователей с низкими навыками программирования (например, экспертов предметной области, аналитиков).

4.2.2 Web PKBD

Web PKBD (WPKBD) – программная реализация PKBD под веб (Рис.4.2.11), включает следующие основные компоненты:

- подсистему синтеза веб-интерфейса – обеспечивает создание форма ввода и просмотра: шаблонов фактов, фактов и правил (для заполнения базы знаний), результатов логического вывода;
- модуль вывода на основе правил CLIPS (C language Integrated Production System) – обеспечивает поиск решения на основе сформированной спецификации базы знаний (используется существующая DLL);
- подсистему визуализации элементов базы знаний в виде графических примитивов языка RVML (Rule Visual Modeling Language);
- подсистему формирования отчетов.

Важной особенностью WPKBD является поддержка формата EKB.

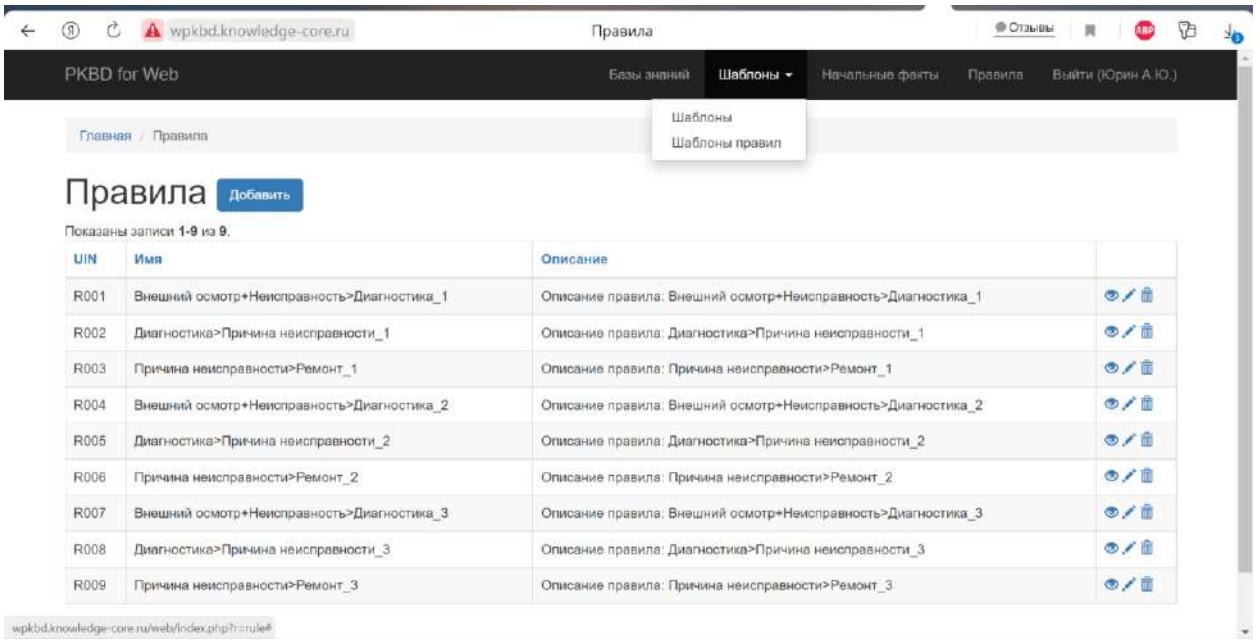


Рис. 4.2.11. Пример интерфейса Web PKBD

4.2.3 TreeEditorET/Extended Event Tree Editor (EETE)

TreeEditorET [224] – настольная программная система, ориентированная на визуальное программирование БЗ с использованием формализма деревьев событий. Основные функции системы:

- построение и отображение деревьев событий, как путем манипулирования графическими элементами, так и текстовых спецификаций;

- экспорт/Импорт результатов моделирования, как форме графических файлов, так и CLIPS и EKB файлов;
- использование механизма шаблонов при построении деревьев.

Архитектура системы включает (Рис. 4.2.12): модуль автоматизированного построения деревьев событий; графический редактор деревьев; библиотеку вычислительных модулей, реализующую операции над деревьями (минимальное сечение, вычислении вероятности); модуль генерации отчета; хранилище шаблонов; генератор кода CLIPS.

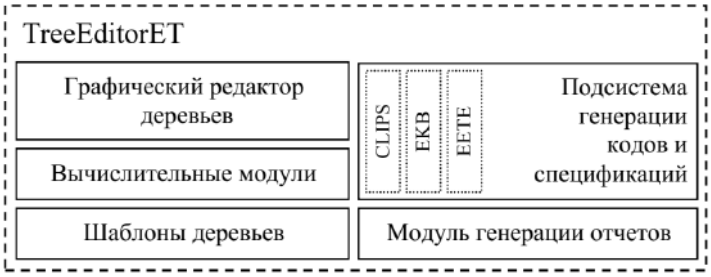


Рис. 4.2.12. Архитектура TreeEditorET

Пример интерфейса редактора приведен на Рис.4.2.13.

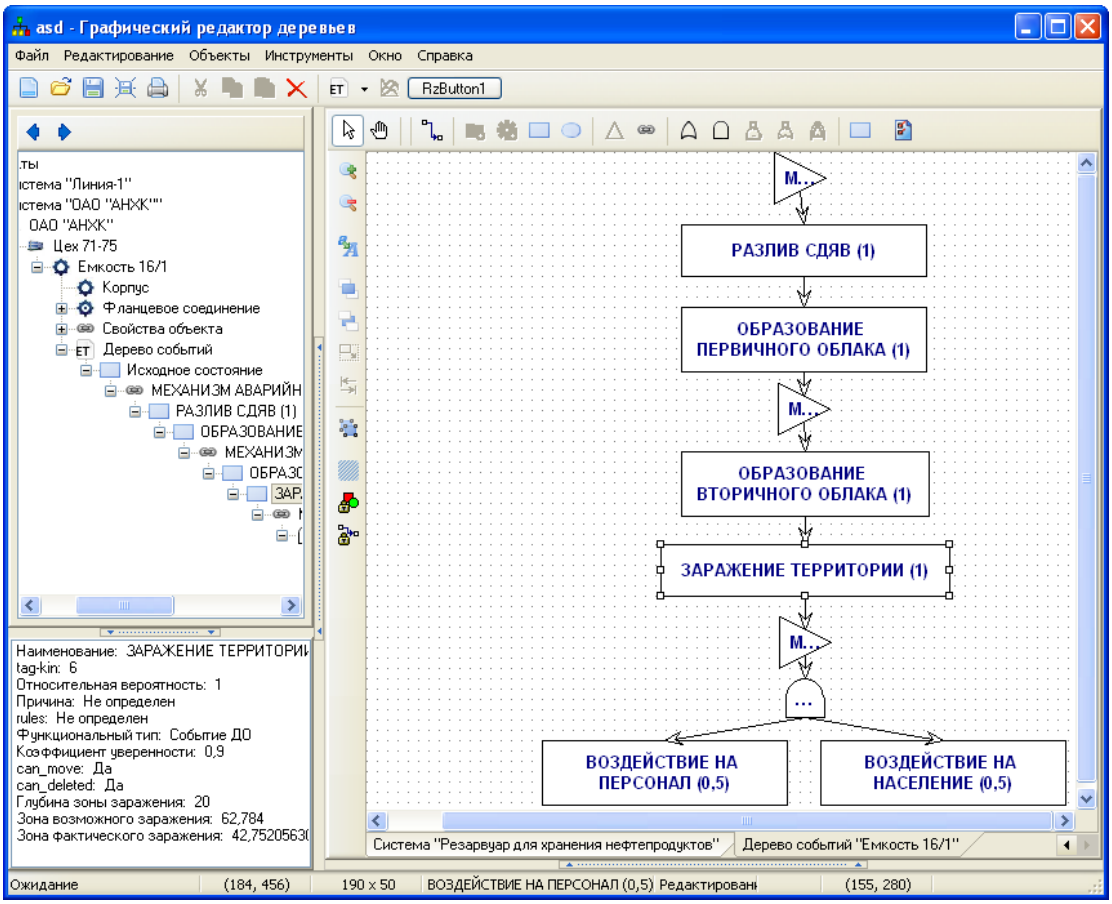


Рис. 4.2.13. Пример интерфейса TreeEditorET

Особенностью программного средства является интеграция с другими программными средствами программного комплекса, а также поддержка расширенного формализма деревьев событий [225].

Дальнейшим развитием TreeEditorET является веб-средство Extended Event Tree Editor (EETE) [287], важными функциями которого являются: импорт информации из OWL онтологий и поддержка специализированного элемента деревьев – механизма (Рис.4.2.14).

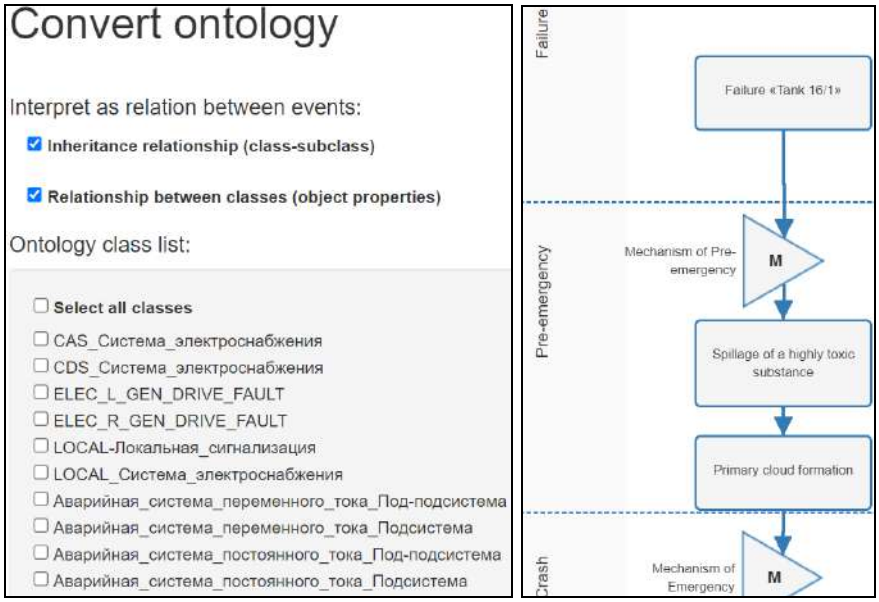


Рис. 4.2.14. Пример интерфейса EETE: Импорт OWL онтологий
и поддержка специальных графических элементов при визуальном моделировании

Выводы

Разработан новый метод создания интеллектуальных систем и декларативных баз знаний продукционного и прецедентного типа на основе модельных трансформаций, который в отличие от известных реализаций модельно-ориентированного подхода обеспечивает использование новых моделей, языков и платформ, ориентированных на более полное вовлечение конечных пользователей в процесс разработки.

Особенностью метода, определяющим его новизну, является использование EUD подходов, в частности: визуального программирования и модельно-ориентированной разработки в контексте создания ИС и БЗ, оригинальных языков RVML и TMRL для описания элементов баз знаний и моделей трансформаций, а

также концепт-карт, онтологий, деревьев событий, канонических таблиц и таблиц решений при описании вычислительно-независимых моделей.

Метод реализован в форме оригинальных программных средств Personal Knowledge Base Designer (PKBD)/Web PKBD, а также TreeEditorET/Extended Event Tree Editor.

Применение метода и программных средств позволит использовать разработанные ранее концептуальные модели, привлечь к процессу разработки непрограммирующих пользователей и сократить время создания БЗ, в том числе, за счет автоматической кодогенерации.

Глава 5. Методы и программное средство проектирования программ трансформаций концептуальных моделей и программных компонентов-конверторов

В главе рассмотрены оригинальные модели, методы и программное средство Knowledge Base Development System (KBDS) [281, 283, 366] программирования трансформаций концептуальных моделей [46, 239] и создания программных компонентов-конверторов для реализации трансформаций концептуальных моделей [46, 239].

5.1 Метод проектирования трансформаций концептуальных моделей

Решая задачу создания средств описания трансформаций концептуальных моделей конечными пользователями, был разработан новый метод, основанный на применении разработанного ранее декларативного языка описании модельных трансформаций TMRL [239]. При этом в контексте данной работы, под модельной трансформацией [39, 107, 147] понимается генерация целевой модели из исходной, согласно определенному набору правил преобразования (трансформаций). Потенциально, в процессе трансформации могут преобразовываться множество исходных и/или целевых моделей с целью их объединения (слияния или дополнения) в одну или наоборот – получения нескольких моделей на основе единственной. При этом первый вид преобразований может быть использован для получения одной общей платформо-независимой модели (Platform Independent Model, PIM) в результате преобразования нескольких вычислительно-независимых (Computation-Independent Model, CIM), а второй – для получения из одной PIM нескольких (по количеству целевых/выходных платформ) платформо-зависимых (Platform Specific Model, PSM). Основными целевыми платформами в рамках данной работы являются OWL онтологии и продукции на CLIPS.

5.1.1 Формальное описание трансформаций

Используя [107, 147, 283] формально определим понятие модельной трансформации:

$$M_T = \langle MM_{IN}, MM_{OUT}, T \rangle, \qquad (5.1.1)$$

где MM_{IN} – метамодель исходной (входной) концептуальной модели; MM_{OUT} – метамодель целевой (выходной) модели БЗ; T – оператор преобразования моделей (модельной трансформации), при этом:

$$MM_{IN} \in \{MM_{CM}, MM_{PR}, MM_{ONT}\}, \quad (5.1.2)$$

$$MM_{OUT} \in \{MM_{PR} MM_{ONT}, MM_{CLIPS}, MM_{OWL}\},$$

где MM_{CM} – метамодель концептуальной модели CM_{XML}, сериализованной в XML-подобном формате; MM_{PR} – метамодель обобщенной модели продукций M_{PR}; MM_{ONT} – метамодель обобщенной модели онтологии M_{ONT}; MM_{CLIPS} – метамодель языка CLIPS; MM_{OWL} – метамодель языка OWL.

Из (5.1.2) следует, что помимо метамоделей MM_{CM}, MM_{CLIPS} и MM_{OWL}, применяемых для описания преобразования концептуальных моделей в БЗ форматов CLIPS и OWL, в понятие модели трансформации также могут входить MM_{PR} и MM_{ONT} метамодели, которые используются для описания обобщенных моделей продукций M_{PR} и онтологии M_{ONT}, соответственно. Цель введения данных элементов – обеспечить унифицированное представление и хранение извлеченных из концептуальных моделей знаний.

Выделенные метамодели MM_{PR}, MM_{ONT}, MM_{CLIPS}, MM_{OWL} в дальнейшем будут реализованы и интегрированы в программное средство.

Согласно (5.1.1) уточним описание MM_{CM}: $MM_{CM} = \langle E_{CM}, R_{CM} \rangle$,

где E_{CM} – множество элементов трансформируемой концептуальной модели, формирующих метамодель; R_{CM} – множество отношений (связей) между элементами метамодели, при этом:

$E_{CM} = \{e_1^{cm}, ..., e_n^{cm}\}, e_i^{cm} = \langle name_{i,1}, p_{i,2}, ..., p_{i,k} \rangle, i \in \overline{1,n}$, где $name_{i,1}$ – наименование i элемента; $p_{i,2}, ..., p_{i,k}$ – свойства i элемента.

MM_{CM} включает определение отношений следующих типов:

$$R_{CM} = \langle R_{AS}^{cm}, R_{ID}^{cm}, R_{part-of}^{cm} \rangle,$$

где R_{AS}^{cm} – ассоциация, как бинарное отношение между элементами метамодели; R_{ID}^{cm} – связь по идентификатору, как бинарное отношение между элементами метамодели на основе определенного идентификатора (текстового или числового

значения); $R^{cm}_{part-of}$ – связь «часть-целое» (композиция), как n-арное отношение между n элементами метамодели, при этом:

$$R^{cm}_{AS} = \left\{r^{as}_1,...,r^{as}_m\right\}, r^{as}_j = \left\langle e^{cm}_{j,1}, e^{cm}_{j,2}\right\rangle, j \in \overline{1,m},$$ где $e^{cm}_{j,1}$ – левая часть ассоциативной связи, $e^{cm}_{j,2}$ – правая часть ассоциативной связи.

$R^{cm}_{ID} = \left\{r^{id}_1,...,r^{id}_h\right\}, r^{id}_l = \left\langle r^{id}_{lhs_l}, r^{id}_{rhs_l}\right\rangle, l \in \overline{1,h}$, где $r^{id}_{lhs_l}$ – левая часть связи; $r^{id}_{rhs_l}$ – правая часть связи, при этом $r^{id}_{lhs_l} = e^{cm}_i(p_i), i \in \overline{1,n}$ и $r^{id}_{rhs_l} = e^{cm}_j(p_j), j \in \overline{1,n}$, где $e^{cm}_i(p_i)$ и $e^{cm}_j(p_j)$ – элементы метамодели, участвующие в связи по идентификатору, т.е. значение свойств p_j и p_i равны.

Связь «часть-целое» предполагает, что i элемент метамодели e^{cm}_i включает составные части, соединяемые между собой отношением R^{cm}_{AS} или R^{cm}_{ID}:

$$R^{cm}_{part-of} = \left\{r^{part-of}_1,...,r^{part-of}_g\right\}, r^{part-of}_f = \left\langle (r^{part-of}_{lhs_f}, r^{part-of}_{rhs^1_f}),...,(r^{part-of}_{lhs_f}, r^{part-of}_{rhs^u_f})\right\rangle, f \in \overline{1,g}$$

В качестве общей основы метамоделирования для представления и хранения метамоделей метода создано концептуальное пространство моделирования (Conceptual Modeling Space), представляющее собой мета-метамодель (Рис. 5.1.1).

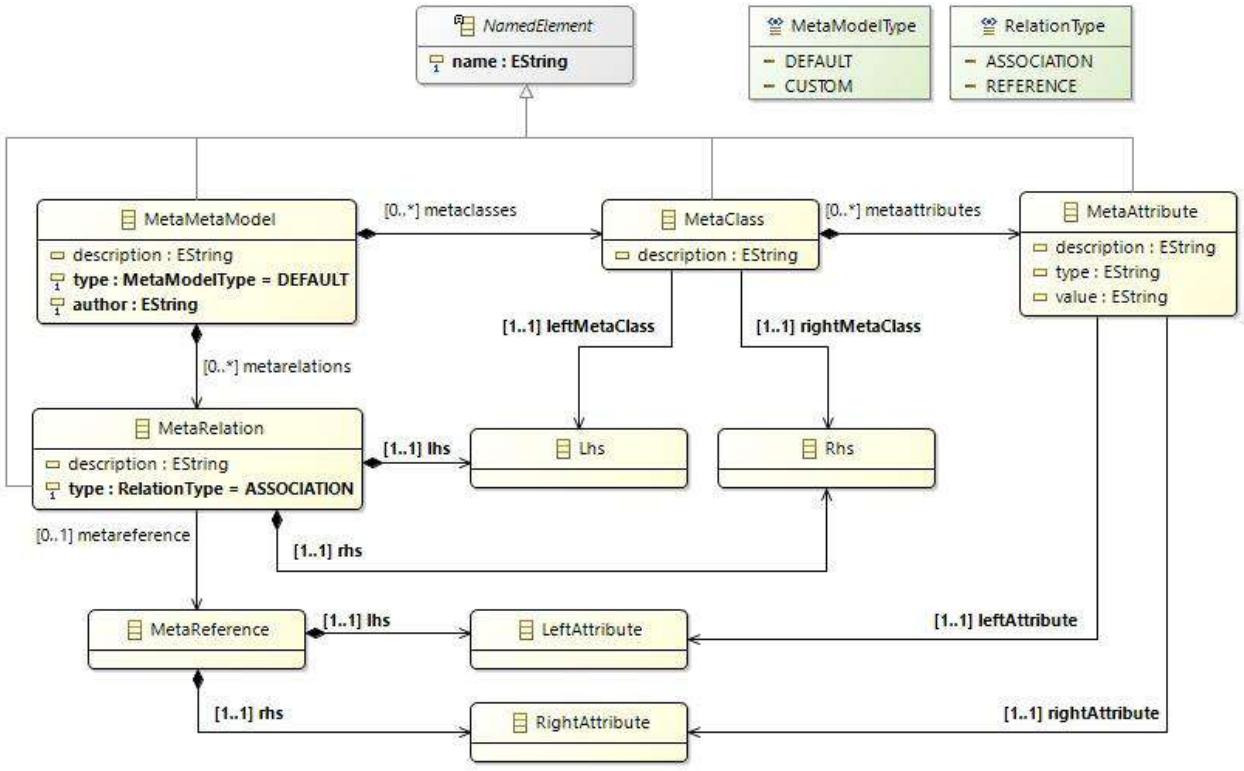


Рис. 5.1.1. Мета-метамодель метода

Согласно принципам метамоделирования модельно-ориентированного подхода, если две разные метамодели используют единую мета-метамодель, т.е. выражаются посредством ее элементов, то для хранения и обработки основанных на них моделей могут использоваться общие репозитории и средства реализации модельных трансформаций.

Уточним оператор преобразования T из (5.1.1):

$$T : CM \to KB, \tag{5.1.4}$$

где CM – преобразуемая концептуальная модель; KB – результирующая БЗ.

Согласно [75] для уточнения (5.1.4) конкретизированы виды синтаксиса:

1) Абстрактный синтаксис (abstract syntax), определяющий основные понятия и структуру выражений языка, т.е. характеризующий в абстрактной форме виды элементы и правила их сочетания. Абстрактный синтаксис определяет семантику.

2) Конкретный синтаксис (concrete syntax), определяющий форму отображения элементов при их использовании, т.е. конкретный синтаксис является графической или текстовой нотацией.

3) Служебный синтаксис или синтаксис сериализации (serialization syntax), используемый для хранения и обмена языковыми выражениями между различными программными системами.

Определим (5.1.4) на абстрактном, конкретном и служебном уровнях:

$$T_{AS} : MM_{CM} \to MM_{KB},\ T_{CS} : CM_{XML} \to Code_{KRL}, \tag{5.1.5}$$

где T_{AS} – оператор трансформации концептуальной модели в БЗ на абстрактном уровне; MM_{CM} – метамодель трансформируемой концептуальной модели; MM_{KB} – метамодель результирующей (выходной) БЗ; T_{CS} – оператор трансформации концептуальной модели в БЗ на конкретном (служебном) уровне; CM_{XML} – трансформируемая концептуальная модель в XML-формате; $Code_{KRL}$ – код БЗ на ЯПЗ, $KRL = \{CLIPS, OWL\}$.

На основе (5.1.3) и (5.1.4) конкретизируем операторы преобразования:

$$T \in \left\{T_{CM-KB}, T_{CM-UM}, T_{UM-KB}\right\}, \tag{5.1.6}$$

где $T_{CM–KB}$ – оператор преобразования концептуальной модели в код БЗ; $T_{CM–UM}$ – оператор преобразования концептуальной модели во внутренний

унифицированный формат (модель) представления знаний в форме онтологии или продукций; T_{UM-KB} – оператор преобразования онтологии или продукций в код БЗ, при этом $KB = \{CLIPS, OWL\}$.

Используя (5.1.5) детализируем (5.1.6):

$$T_{CM-KB} : MM_{CM} \rightarrow MM_{KB},$$

$$T_{CM-UM} : MM_{CM} \rightarrow MM_{UM},\ T_{UM-KB} : MM_{UM} \rightarrow MM_{KB},$$

где $MM_{KB} \in \{MM_{CLIPS}, MM_{OWL}\}$, $MM_{UM} \in \{MM_{PR}, MM_{ONT}\}$.

Согласно (5.1.5) преобразование CM_{XML} в $Code_{KRL}$ происходит на основе определения соответствий между элементами преобразуемой (MM_{CM}) и результирующей (MM_{KB}) метамоделей.

Выделим следующие основные типы соответствий:

1) Эквивалентность представляет собой однозначное соответствие, при котором каждому элементу MM_{CM} соответствует единственный элемент MM_{KB}.

2) Синонимия представляет собой неоднозначное соответствие, при котором несколько элементов MM_{CM} могут соответствовать только одному элементу MM_{KB}.

3) Омонимия представляет собой неразличимое соответствие, при котором один элемент MM_{CM} может одновременно соответствовать нескольким элементам MM_{KB}.

Особые (крайние) случаи соответствий:

1) Избыточность, означающая отсутствие в целевой метамодели MM_{KB} соответствий для отдельных элементов MM_{CM}.

2) Дефицит выразительной способности, означающий отсутствие соответствий для отдельных элементов MM_{KB} в исходной метамодели MM_{CM}.

Опишем процесс построения правил трансформации (преобразования) для рассмотренных выше типов соответствий. При этом с помощью оператора R_T обозначим совокупность правил соответствия:

$$R_T = (r_1, r_2 .. r_n) : MM_{CM} \rightarrow MM_{KB},$$

где r_i – правило трансформации одного из следующих типов:

$$r_i \in \{r_i^s, r_i^c\}, i \in \overline{1,n},$$

где r_i^s – простое бинарное; r_i^c – составное (сложное) n-арное.

$$r_i^s = \left\langle e_i^{in}, e_i^{out}, p_i \right\rangle, i \in \overline{1,n},$$

где e_i^{in} – элемент исходной метамодели MM_{CM} (левая часть правила); e_i^{out} –элемент целевой метамодели MM_{KB} (правая часть правила); p_i – приоритет правила, $p_i \in \overline{1,k}$.

$$r_i^c = \left\langle \left\{ e_{i,1}^{in}, \ldots, e_{i,j}^{in} \right\}, e_i^{out}, p_i \right\rangle, i \in \overline{1,n},$$

где $\left\{ e_{i,1}^{in}, \ldots, e_{i,j}^{in} \right\}$ – множество элементов MM_{CM}; e_i^{out} – элемент MM_{KB}; p_i – приоритет правила, $p_i \in \overline{1,k}$.

Наличие r_i^c определяется связями «часть-целое» ($R_{part-of}^{cm}$), которые могут быть описаны в MM_{CM}.

Рассмотренные типы правил трансформации применимы ко всем выделенным соответствиям.

Предлагаемый метод основан на концепции модельных трансформаций, которая является одним из основных аспектов в концепции модельно-управляемого подхода (MDD/MDA) [24, 38, 41, 69, 71, 92, 109, 140, 144, 161, 168]. Общая схема трансформации концептуальных моделей в БЗ, основанная на схеме трансформации моделей и реализуемая методом, приведена на Рисунке. 5.1.2.

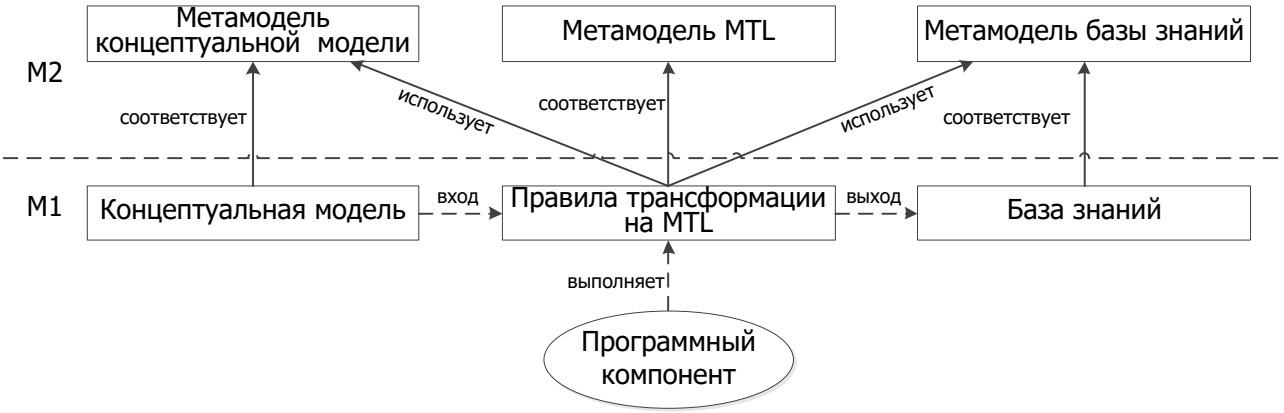


Рис. 5.1.2. Общая схема трансформации концептуальных моделей реализуемая методом

Приведенная схема (Рис. 5.1.2) определяет горизонтальную экзогенную трансформацию [107], под которой понимается манипулирование моделями одного уровня иерархии схемы метамоделирования [112], но представленными с помощью разных языков моделирования (т.е. с использованием разных концептуальных моделей и ЯПЗ). Данная схема соответствует четырехуровневой архитектуре

метамоделирования (иерархии моделей) MOF [112] и отображает два основных уровня: моделей «M1» и метамоделей «M2», остальные уровни (мета-метамодели «M3» и данных «M0») скрыты для акцентирования восприятия на ключевых аспектах метода.

На уровне «M1» размещены преобразуемые концептуальные модели и модели целевой БЗ, для представления которых могут использоваться ЯПЗ CLIPS или OWL. В реализуемых модельных трансформациях, начальными и конечными артефактами является текст: в первом случае – это текст XML-документа концептуальной модели, во втором – программный код БЗ на CLIPS или OWL. Таким образом, сперва требуется трансформировать текст исходного XML-документа, соответствующего концептуальной модели, в полноценную модель, т.е. по сути – выполнить перевод конкретной спецификации в абстрактную, который литературе [107] известен как T2M-трансформация (Text-to-Model). При этом полученная модель должна соответствовать некоторой метамодели на уровне «M2». Для преобразования полученной концептуальной модели в модель БЗ необходимо выполнить уже M2M-трансформацию (Model-to-Model). Правила этой трансформации описываются с использованием языка модельных трансформаций TMRL и соответствуют его метамодели. После получения целевой модели БЗ, она преобразовывается в код на ЯПЗ CLIPS или OWL, при этом производится преобразование абстрактной спецификации в конкретную спецификацию, т.е. M2T-трансформация (Model-to-Text). Трансформации T2M и M2T осуществляются в автоматическом режиме.

5.1.2 Модели и метамодели метода

Для поддержки трансформаций концептуальных моделей в рамках предлагаемого метода были разработаны метамодели онтологии, продукций, OWL и CLIPS.

5.1.2.1 Обобщенная модель (метамодель) онтологии

Для унифицированного представления и хранения знаний, извлеченных из концептуальных моделей, предлагается использовать специальную модель онтологии M_{ONT}. Данная модель позволяет абстрагироваться от особенностей

описания знаний в различных ЯПЗ, используемых при реализации БЗ (например, CLIPS, JESS, и др.), и хранить знания в собственном независимом формате.

Предлагаемая модель является высокоуровневой абстракцией представления знаний в форме онтологий и содержит только общие конструкции, поддерживаемые наиболее распространенными онтологическими языками.

Используя представление из [282, 327], формализуем описание метамодели онтологии,:

$$MM_{ONT} = \langle DT, CN, PN, Obj, R \rangle,$$

где DT – перечень базовых типов данных, при этом DT = {литерал, объект, коллекция}; CN – множество имен классов; PN – множество имен свойств классов; Obj – набор понятий предметной области; R – конечное множество отношений (связей) между понятиями предметной области, при этом:

$R \in \{R^{subclass-of}, R^{is-a}, R^{p}, R^{c}\}$, где $R^{subclass-of}$ – отношение наследования между классами CN; R^{is-a} – связь между классами CN и их экземплярами (объектами) Obj; R^{p} – связь между классами и свойствами, $pn\, R^{p} \langle cn, cn\_dt \rangle$, где $pn \in PN$, $cn \in CN$, $cn\_dt \in CN \cup DT$, последнее означает, что свойство с именем pn_i характеризует класс cn_j, а значениями свойства могут быть элементы типа другого класса из CN или основного множества типов DT; R^{c} – отношения между классами, при этом $R^{c} = \{r_1^c ... r_k^c\}$, $r_j^C = \langle name_j, cn_{LHS_j}, cn_{RHS_j} \rangle$, $j \in \overline{1,k}$, где $name_j$ – имя отношения; cn_{LHS_j} – левый класс связи (ссылка на класс); cn_{RHS_j} – правый класс связи (ссылка на класс).

Метамодель MM_{ONT}, определяющая основные понятия онтологии, представлена на рисунке 5.1.3.

5.1.2.2 Обобщенная модель (метамодель) продукций

Для поддержки возможности работы со знаниями в виде продукций (логических правил) разработана специальная модель продукций M_{PR}, которая является высокоуровневой абстракцией в контексте метода описания трансформаций. Данная модель содержит только общие конструкции, поддерживаемые большинством языков программирования продукционных

(логических) правил. Для ее визуального представления использована авторская графическая нотация – Rule Visual Modeling Language (RVML) [52, 184, 185, 198, 279, 368], рассмотренная ранее.

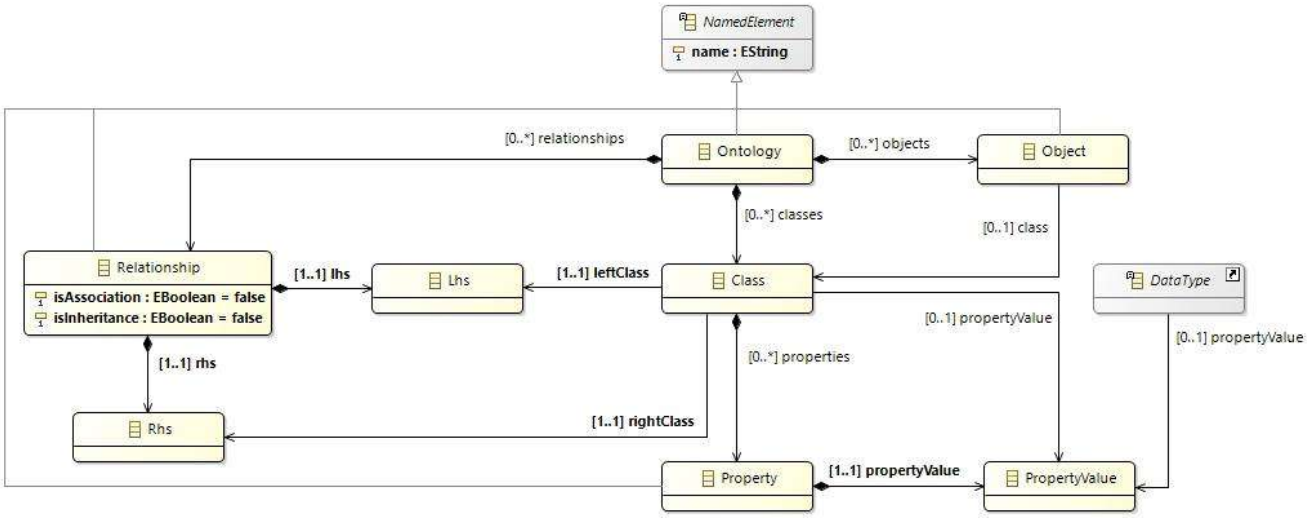


Рис. 5.1.3. Метамодель онтологии, реализуемая методом

Метамодель MM_{PR}, определяющая основные концепты, из которых состоит M_{PR}, представлена на Рисунке 5.1.4.

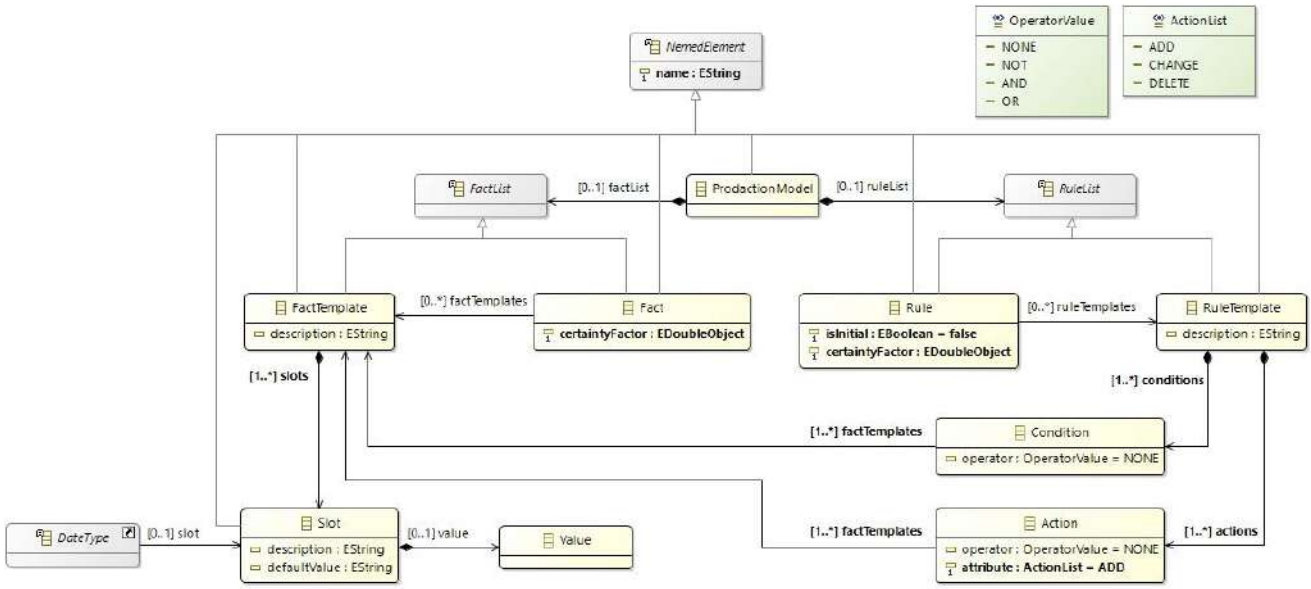


Рис. 5.1.4. Метамодель продукций, реализуемая методом

5.1.2.3 Метамодель OWL

По результатам аналитического обзора был сделан вывод о перспективности использования в качестве целевого ЯПЗ для описания онтологий OWL2 DL [121] с поддержкой синтаксиса RDF/XML [130].

Метамодель OWL2 (abstract syntax), определяющая его основные концепты (семантику) представлена на Рисунках 5.1.5 и 5.1.6.

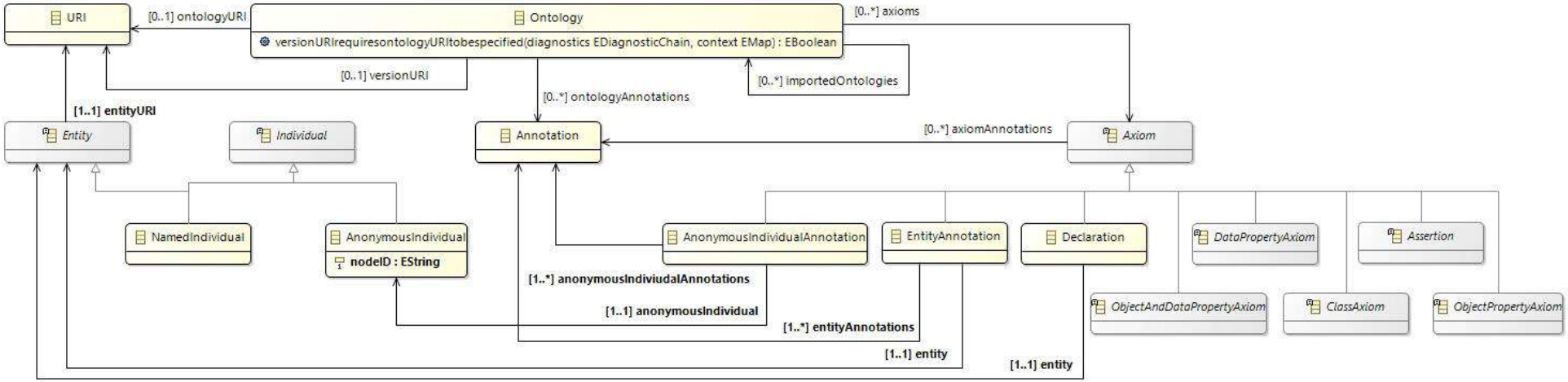


Рис. 5.1.5. Метамодель OWL2 DL (основные концепты онтологии)

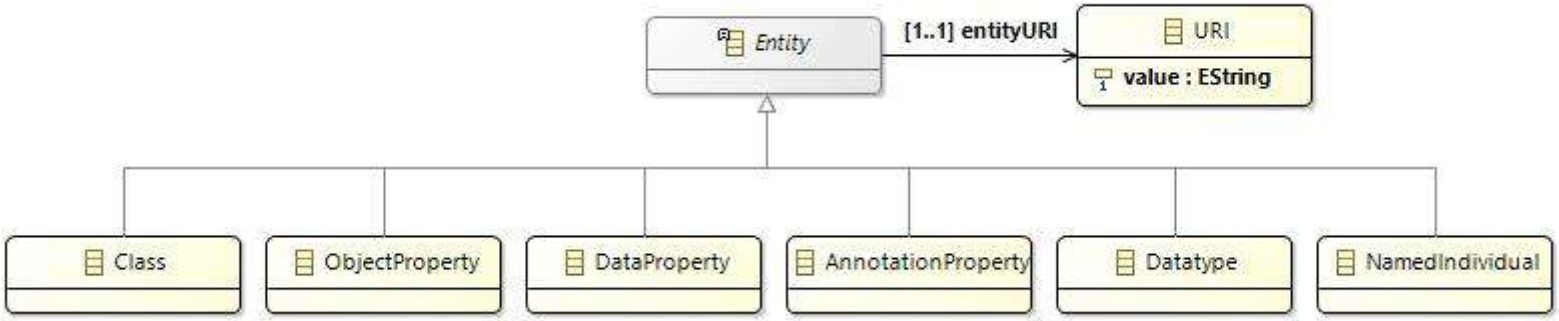


Рис. 5.1.6. Метамодель OWL2 DL (основные сущности)

Метамодель OWL2, соответствующая техническому пространству моделирования MOF (MOF Technical Space) [42] и представленная в формате Ecore [60], определена в [121]. Данная метамодель также представлена в формате XSD [170] и определена в [121].

5.1.2.4 Метамодель CLIPS

В дополнение к OWL2 в качестве второго целевого ЯПЗ для описания продукций (логических правил) был использован CLIPS [33].

Метамодель CLIPS (abstract syntax), определяющая основные концепты (семантику), из которых состоит данный язык, представлен на Рисунках 5.1.7-5.1.9. Следует отметить, что в данной работе не используется язык COOL, в дальнейшем рассматривается возможность его применения, но на текущий момент метамодель содержит только основные определения и конструкции CLIPS.

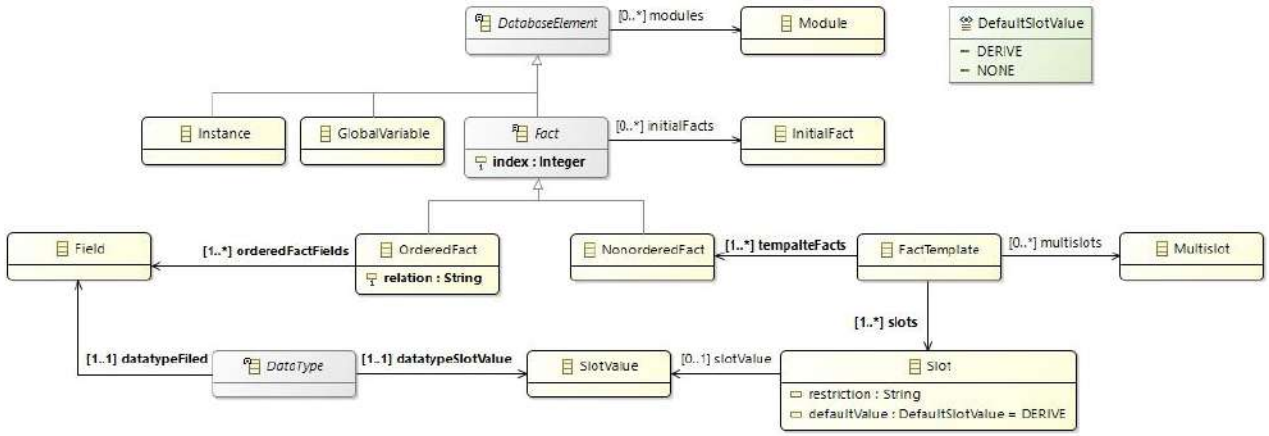


Рис. 5.1.7. Метамодель CLIPS (данные)

5.1.3 Основные этапы метода

Процесс описания трансформаций в рамках применения предлагаемого метода представлен в виде следующей последовательности этапов:

1) Анализ структур исходной (MM_{CM}) и целевой (MM_{KB}) метамоделей с целью формирования наборов основных их элементов.
2) Формирование частичной модели трансформации, состоящей только из полученных наборов элементов исходной (MM_{CM}) и целевой (MM_{KB}) метамоделей.

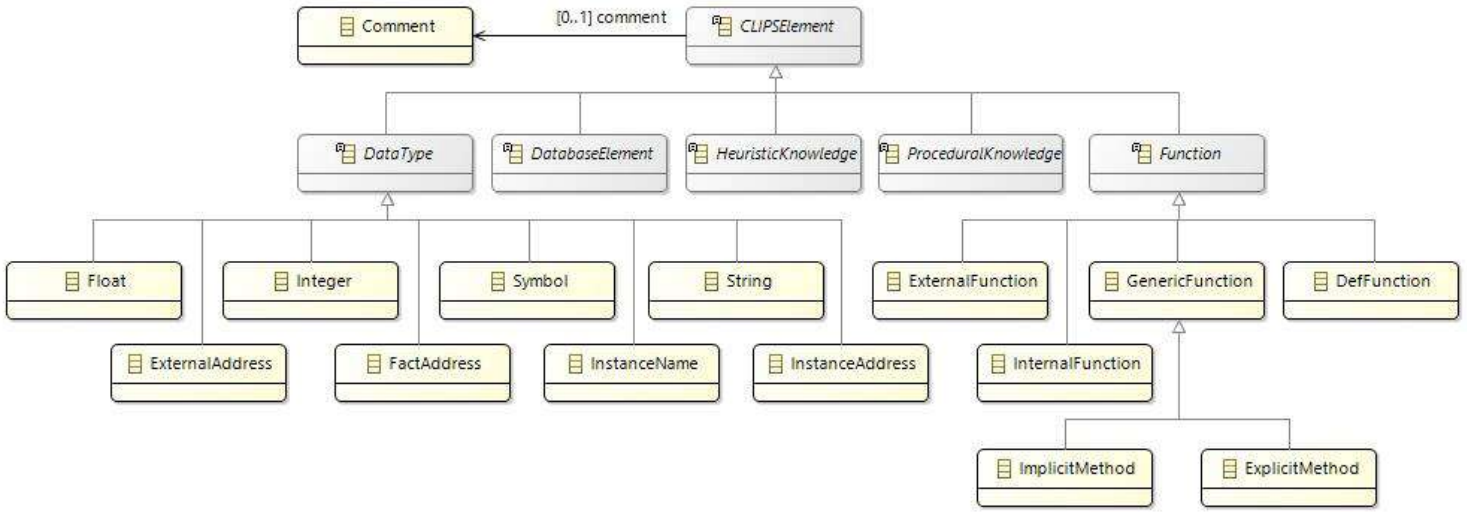


Рис. 5.1.8. Метамодель CLIPS (иерархия основных элементов)

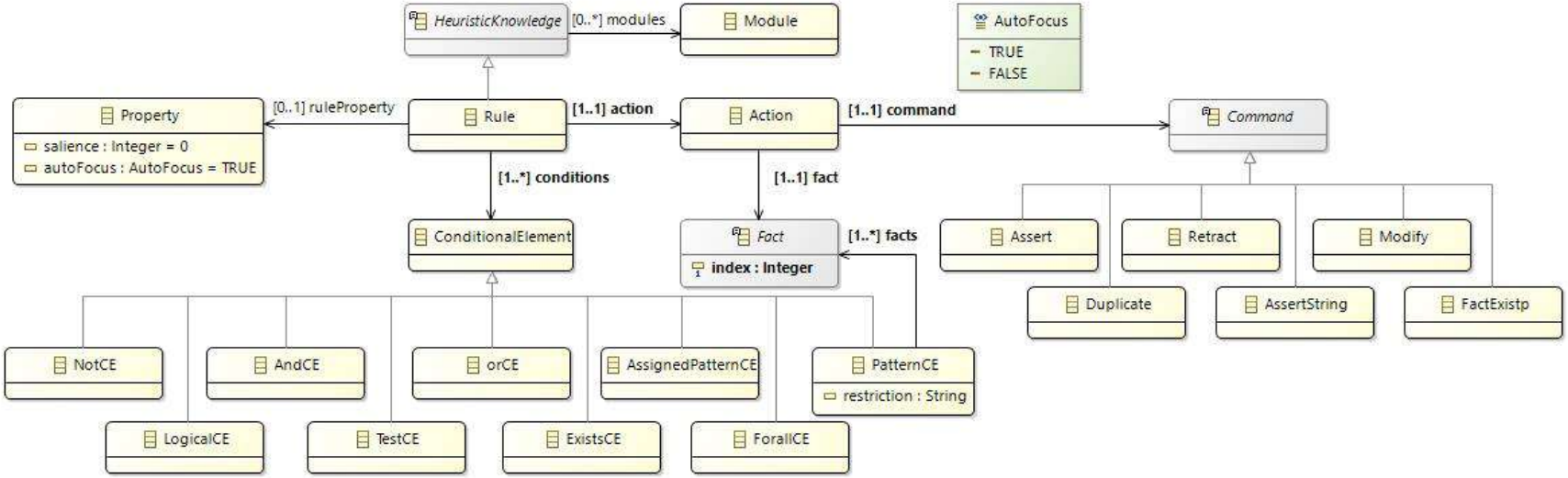


Рис. 5.1.9. Метамодель CLIPS

3) Формирование полной модели трансформации, путем определения правил трансформации *T*, описывающий соответствия между элементами исходной (MM_{CM}) и целевой (MM_{KB}) метамоделей.
4) Генерация кода трансформации на TMRL.
5) Уточнение TMRL кода с целью описания сложных соответствий.

Необходимо отметить, что при автоматической генрации метамоделей и кодов, сформированная программа трансформации может быть неполной или некорректной. Основные причины нарушения полноты и корректности:

1) Ограниченность методов парсинга (анализа) XML-схем при проведении процедуры обратной инженерии, которые не позволяют выявить абсолютно все связи элементов из-за особенностей реализации механизма их определения. В частности, связь элементов может устанавливаться путем указания прямых ссылок на уникальные идентификаторы элементов:

```
<DataType id="номер типа данных"… />
<Attribute type="номер типа данных" …> … </UML:Attribute>
```

Подобная связь между элементами не будет отображена в XML-схеме, следствием чего является неполнота модели трансформации при автоматическом построении метамодели. Приведенная особенность является одним из характерных недостатков использования в качестве исходных входных метамоделей XML-схем в контексте применения процедуры обратной инженерии. Выявленная проблема решается определением данного типа связи вручную конечным пользователем на этапах формирования полной модели трансформации и уточнения TMRL кода .

2) Использование вычисляемых (определяемых) текстовых констант при формировании значений элементов целевой метамодели MM_{KB}. При этом константы зависят от значений элементов исходной метамодели MM_{CM} и не всегда могут быть определены в процессе пошагового парсинга (анализа).

С учетом определенных факторов, можно выделить следующие типы правил преобразования:

- простые или биективные – задают однозначные соответствия между элементами исходных и целевых метамоделей (данные правила могут определяться на этапе 3);

- сложные или составные – задают значения (константы) элементов целевой метамодели, в зависимости от значений соответствующих элементов исходной метамодели.

Таким образом, простые правила трансформации (преобразования) элементов исходной и целевой метамоделей, определенные на этапе 3 могут не охватывать все варианты соответствий, т.е. не формировать полную модель трансформации. Для формирования полной модели на этапе 5 происходит уточнение трансформаций, т.е. определяются недостающие сложные правила преобразования.

В целом, для повышения эффективности метода при описании правил преобразования использованы принципы визуального программирования, а при формировании метамоделей – автоматизированного парсинга (анализа) XML-схем.

5.1.4 Проверка корректности трансформаций концептуальных моделей

В процесс формирования правил преобразования обеспечивается их проверка (верификация). Основная цель данного процесса – выявить ошибки, вызывающие получение синтаксически или семантически некорректных метамоделей и программ трансформации. В частности, проверка направлена на выявление:

1) Некорректных метамоделей с точки зрения их синтаксиса, ошибки в которых обусловлены процессом кодирования, наиболее распространенная из них – это опечатки.

2) Некорректных метамоделей с точки зрения семантики, ошибки в которых обусловлены заданием неверных связей между элементами или их отсутствием.

3) Некорректных метамоделей с точки зрения их неполноты, обусловленной отсутствием правил преобразования, устанавливающих соответствия для отдельных элементов исходной или целевой метамодели.

4) Некорректных соответствий с точки зрения синтаксиса, ошибки в которых обусловлены процессом кодирования правил преобразования, наиболее распространенная из них – это опечатки.

5) Некорректных соответствий с точки зрения семантики, ошибки в которых обусловлены заданием неверных соответствий в правилах преобразования.

Для устранения подобных ошибок используется два подхода: первый основан на их предупреждении, второй – на проверке корректности (тестировании, верификации) полученных моделей и БЗ, как результатов преобразований.

Первый подход реализован средствами автоматизированного контроля спецификаций создаваемых моделей и правил преобразования в разработанном программном средстве. При построении метамоделей обнаруживаются следующие возможные ошибки:

- наличие несвязанных («висящих») элементов;
- наличие «пустых» элементов без свойств;
- дублирование (повтор) связей одних и тех же элементов;
- наличие «петли» – кольцевой связи одного типа у пары элементов;
- наличие рекурсивных связей.

При формировании правил преобразования проверяются следующие ограничения:

- наличие как минимум одного правила;
- использование в правилах только элементов входной и выходной метамоделей;
- указание соответствий между свойствами элементов входной и выходной метамоделей возможно только после указания соответствий между самими элементами;
- указание соответствий между свойствами элементов и самими элементами не допускаются;
- допускается наличие «висячих» (несвязанных) элементов в ситуации избыточности или дефицита выразительной способности входной и выходной метамоделей;
- наличие у всех правил приоритетов, которые определяют очередность исполнения правил при интерпретации;
- использование натуральных чисел при задании приоритетов;

- уникальность значений приоритетов;
- отсутвие пропусков в значении приоритетов правил, формирующих очередность их исполнения.

При этом затруднительно проверить коорректность очередности исполнения правил с точки зрения структурных и содержательных особенностей входной и выходной метамоделей без привлечения человека-эксперта.

Второй подход реализует проверку корректности путем тестирования (верификации) полученных БЗ и обеспечивает обнаружение логических ошибок в представлении знаний и структурах вывода. В ряде случаев, рассматривая верификацию ИС по аналогии с верификацией традиционных программ, ее трактуют как доказательство правильности БЗ. В настоящее время кроме нахождения обычных ошибок и опечаток в коде БЗ, обеспечивается обнаружение различного рода аномалий, которые формируют две основные группы [339] (Рис. 5.1.10):

Рис. 5.1.10. Классификация аномалий [339]

1) Нарушения согласованности (consistency) БЗ ИС (например, противоречивость, наличие циклов, избыточность, наличие пересечений и т.д.). Примерами статических аномалий нарушения согласованности в продукционных БЗ ИС являются: противоречивые, циклические, избыточные и пересекающиеся правила.

2) Нарушения целостности (completeness) БЗ ИС (например, неполнота, отсутствие ссылок, некорректность и т.д.). Примерами статических аномалий нарушения целостности в продукционных БЗ ИС являются: отсутствие правил, приводящих к значениям целевых атрибутов; неиспользованные входные значения; наличие атрибутов, на которые нет ссылок; неверные значения атрибутов.

В данной работе упомянутые методы реализованы путем использования интерпретаторов и валидаторов целевых моделей и кодов БЗ, в частности:

- Системы Personal Knowledge Base Designer (PKBD) [184, 195, 202, 269] с динамической библиотекой машины вывода, реализующей алгоритм RETE; используется для проверки корректности сгенерированных продукционных БЗ в формате CLIPS.
- Системы Protégé, включающей машины вывода (reasoners) Pellet, FaCT++, HermiT; используется для проверка корректности сгенерированных онтологических БЗ в формате OWL.

В дальнейшем для решения задачи верификации будут использованы системы CHECK, КЛАСС и ДИФКЛАСС.

5.1.5 Особенности метода проектирования трансформаций концептуальных моделей

- Использование оригинального языка описания трансформаций TMRL, обладающего простым человекочитаемым синтаксисом, обеспечивающего взаимодействие с разработанными ранее программами и ориентированного на описание трансформаций концептуальных моделей, сериализованных в XML-подобных форматах.
- Использование онтологической модели и обобщенной модели продукций для внутреннего унифицированного представления преобразуемых и целевых форматов.

- Использование оригинальных метамоделей, описывающих ЯПЗ OWL и CLIPS.

5.2 Метод создания программных компонентов-конверторов концептуальных моделей

Для реализации возможности трансформации концептуальных моделей с использованием TMRL программ разработан метод создания специализированных программных компонентов-конверторов. При этом под компонентами понимаются элементы структуры программной системы, определенным образом выделенные среди окружения, решающие некоторые подзадачи в рамках общих задач системы и взаимодействующие с окружением через определенный интерфейс [249, 250, 265].

В данной работе программный компонент-конвертор – это элемент программного средства, обеспечивающий автоматизированное создание БЗ на путем трансформации концептуальных моделей, сериализованных в виде XML-документов. При этом CLIPS и OWL являются целевыми ЯПЗ.

Ключевые элементы разработанного метода:

1) Модель типового программного компонента-конвертора, представляющего собой каркас (шаблон) со слотами [249, 250], настраиваемый на определенные ЯПЗ благодаря использованию программ трансформаций на TMRL.

2) Принцип создания (порождения) новых компонентов путем копирования типового и его дальнейшей настройки.

5.2.1 Формальное описание метода

Используя определение программного компонента [249, 250, 265], представим модель типового комопнента-конвертора следующим образом:

$$M_{TPC} = \langle M_T, A_{IN}, CG_{OUT}, I \rangle, \qquad (5.2.1)$$

где M_T – модель трансформации (или слот программы трансформации на TMRL); A_{IN} – анализатор (парсер) входных моделей (слот для анализатора); CG_{OUT} – генератор выходных моделей (слот для генератора); I – программный интерфейс взаимодействия, обеспечивающий доступ к анализатору и генератору, в том числе, внешним системам.

Уточним $A_{IN} \in \{A_{IN}^{CM}, A_{IN}^{ONT}\}$, где A_{IN}^{CM} – анализатор входных моделей, сериализованных в XML-формате; A_{IN}^{ONT} – анализатор входной модели в форме модели онтологии [266], данная модель используется как унифицированное представление, предназначенное для промежуточного хранения знаний из концептуальных моделей.

$CG_{OUT} \in \{CG_{OUT}^{ONT}, CG_{OUT}^{CLIPS-KB}, CG_{OUT}^{OWL-KB}\}$, где CG_{OUT}^{ONT} – генератор выходной модели онтологии; $CG_{OUT}^{CLIPS-KB}$ – генератор кода БЗ на ЯПЗ CLIPS; CG_{OUT}^{OWL-KB} – генератор кода БЗ на ЯПЗ OWL.

$I = \{i_1,...,i_n\}, i_j = \langle name_j, command_j \rangle, j \in \overline{1,n}$, где $name_j$ – наименование j метода взаимодействия; $command_j$ – управляющая команда. Данный интерфейс взаимодействия позволяет внешним программным средствам взаимодействовать с компонентом-конвертором с помощью REST запросов с целью создания БЗ на определенном ЯПЗ путем импорта концептуальной модели.

5.2.2 Ограничения метода

Особенностью разработанного метода создания программных компонентов-конверторов, определяющей его новизну, является использование: языка TMRL для определения M_T, оригинальной модели типового компонента, а также реализация принципов визуального программирования. При этом трансформация моделей определяется на абстрактном уровне (abstract syntax), понятном разработчику программного компонента. В связи с этим, необходимо определить ограничения на входные и выходные метамодели, а также некоторые допущения:

1) Для представления метамодели исходной концептуальной модели MM_{CM} может быть использован XML Schema Definition (XSD) формат [170]. В настоящий момент для автоматического построения XML-схем созданы ряд паттернов (шаблонов) проектирования, в том числе: Russian Doll, Salami Slice, Venetian Blind, Garden of Eden, которые имеют количественные и качественные отличая по количеству и наименованию используемых элементов и типов. В данной работе использованы паттерны Venetian Blind и Garden of Eden [90].

2) Для получения метамодели исходной концептуальной модели MM_{CM}

может быть использована процедуры обратной инженерии (reverse engineering) [107], основанная на парсинге (анализе) XML-схем и выделении элементов, свойств и отношений.

3) Допускается семантическая некорректность полученной метамодели исходной концептуальной модели MM_{CM}. Допускаемая некорректность может быть обусловлена семантическими ошибками в анализируемых XML-документах, вызванными использованием для именования тегов понятий уровня модели (M1), а не метамодели (M2).

4) Существуют объективные ограничения на получение информации о связях элементов метамодели исходной концептуальной модели MM_{CM} на основе анализа XML-схем, а именно: связь «по идентификаторам» R_{ID}^{cm}, определенная внутренней индексацией элементов; связь «часть-целое» $R_{part-of}^{cm}$, определенная переносом части семантики элемента в дочерние элементы.

5) Существуют объективные ограничения на однозначное отображение элементов метамоделей, обусловленные дефицитом выразительной способности.

6) В качестве целевых метамоделей «по умолчанию» определены метамодели онтологии и продукций.

7) В качестве генераторов кодов «по умолчанию» определены генераторы ЯПЗ CLIPS и OWL.

Данные ограничения призваны снизить сложность процесса создания программных компонентов-конверторов конечными пользователями, включая программирование трансформаций, за счет автоматизации отдельных этапов данного процесса.

5.2.3 Основные этапы метода

Основными этапами метода создания программных компонентов-конверторов, использующего определенную выше модель типового программного компонента, программы на TMRL и принципы визуальное программирования, являются следующие:

1) Разработка метамодели для исходной концептуальной модели как путем автоматизированного анализа XML-схем (XSD-файлов), описывающих структуру входных XML-документов, так и путем их прямого

определения конечным пользователем (ручной режим).

2) Выбор целевой метамодели БЗ из множества доступных: обобщенной модели онтологии или продукций. Выбор целевой метамодели БЗ определяет назначение блоков генераторов CG_{OUT} для дальнешего синтеза кодов OWL или CLIPS, соответственно.
3) Описание программы трансформации M_T (см.5.1).
4) Создание программного компонента-конвертора путем специализации типового программного компонента на основе сформированной модели трансформации и выбранных блоках анализатора (A_{IN}) и генератора (CG_{OUT}).

Разработанный в соответствии с данным методом программный компонент-конвертор обеспечивает генерацию кода БЗ на целевом ЯПЗ CLIPS или OWL путем трансформации исходных концептуальных моделей согласно программе на TMRL.

5.2.4 Особенности метода

- Использование оригинальной модели типового программного компонента-конвертора, представляющего собой каркас (шаблон) со слотами, настраиваемый на определенный тип входной концептуальной модели и выходной тип БЗ благодаря использованию декларативной программы трансформации на TMRL.
- Реализация принципа создания программных компонентов-конверторов путем копирования и специализации типового программного компонента.

5.3. Программное средство автоматизации создания программных компонентов-конверторов и моделей трансформаций - Knowledge Base Development System

Для поддержки разработанных методов описания моделей трансформации и создания программных компонентов-конверторов создано оригинальное программное средство – Knowledge Base Development System (KBDS) (рег.№ 2019661803) [281, 283].

Выделим основные функции KBDS:

- создание программных компонентов-конверторов для трансформации концептуальных моделей, путем копирования и специализации

типового (шаблонного) программного компонента, включая формирование правил соответствия между элементами входной и выходной метамоделей с реализацией принципов визуального программирования и прямого манипулирования конструкциями TMRL;

- хранение знаний концептуальных моделей и БЗ с использованием специальных унифицированных моделей онтологии и продукций;
- описание преобразований (трансформаций) концептуальных моделей, включая: автоматизированный парсинг (анализ) входных концептуальных моделей с целью выделения ключевых элементов (понятий, свойств и отношений); визуальное отображение и редактирование понятий, свойств и отношений предметной области в виде графа (модель онтологии); визуальное программирование соответствий между элементами входных и выходных метамоделей; генерация и редактирование декларативных программ на TMRL;
- создание БЗ на основе трансформаций концептуальных моделей, включая: визуальное программирование логических правил с использованием RVML; генерацию кода БЗ на CLIPS;
- обеспечение доступа внешних программных средств к компонентам-конверторам и БЗ KBDS с целью создания онтологических и продукционных моделей на основе анализа концептуальных моделей, генерации кода БЗ, а также манипулирования элементами онтологической и продукционной моделью (создание, редактирование и удаление).

Для реализации основных функций была предложена архитектура (Рис.5.3.1) [281, 283, 366] программного средства, включающая следующие модули, подразделяемые на три группы:

1) Информационные, которые обеспечивают манипулирование, представление и хранение информации, используемой подсистемами как в процессе выполнения служебных, так и пользовательских функций.

Рис. 5.3.1. Концептуальная архитектура KBDS

2) Системные, которые представляют собой набор всех предлагаемых конечным пользователям подсистем, обеспечивающих как базовое взаимодействие с программным средством, так и предоставляющие программный интерфейс взаимодействия с внешними системами, а также предоставляющие инструментарий для создания прикладных программных компонентов-конверторов на основе типового.

3) Прикладные, которые представляют собой совокупность разработанных пользователями программных компонентов-конверторов (не встроенных), обеспечивающих трансформацию концептуальных моделей и синтез БЗ. Программные компоненты-конверторы создаются путем копирования и специализации типового.

С точки зрения клиент-серверной архитектуры была выделена клиентская часть программного средства, включающая:

- графический редактор продукций (логических правил), обеспечивающий поддержку авторского языка RVML [368];
- графический редактор онтологий, обеспечивающий визуальное отображение и редактирование моделей предметной области в виде графа;
- графический редактор метамоделей, предназначенный для визуального отображения и редактирования основных элементов входных и

выходных метамоделей;

- графический редактор трансформаций, обеспечивающий визуальное отображение и редактирование правил преобразования, описывающих соответствия элементов входных и выходных метамоделей.

Состав серверной части программного средства:

- модуль администрирования, обеспечивающий взаимодействие конечных пользователей с системой, включая контроль прав доступа, сбор статистической информации, мониторинг работоспособности и др.;
- подсистема управления базами знаний, обеспечивающая работу с проектами конечных пользователей;
- подсистема управления мета-уровнем представления знаний, реализующая поддержку внутреннего представления знаний в виде онтологической и продукционной моделей, с целью абстрагирования от особенностей описания знаний в различных ЯПЗ БЗ (CLIPS, Drools, OWL, SWRL и др.), а также возможность их хранения в собственном независимом формате;
- подсистема разработки программных компонентов-конверторов, предоставляющая возможность их создания и учета, а также генерации TMRL кода и их спецификаций;
- программные компоненты-конверторы, обеспечивающие преобразование концептуальных моделей во внутреннее представление (продукции или онтология – по выбору конечного пользователя) и генерацию кода БЗ на CLIPS или OWL.

Примеры интерфейса отдельных подсистем и модулей KBDS представлены на Рис. 5.3.2-5.3.5.

Поддержка взаимодействия KDBS со внешними программными системами реализована посредством программного интерфейса – API (Application Programming Interface), на основе архитектурного стиля REST (Representational State Transfer) и GET или POST запросов по протоколу HTTP.

Ниже приведены основные методы интерфейса взаимодействия (подробное описание приведено в Приложении Б):

- getAllModulesList – получение списка всех программных компонентов-

конверторов KBDS;

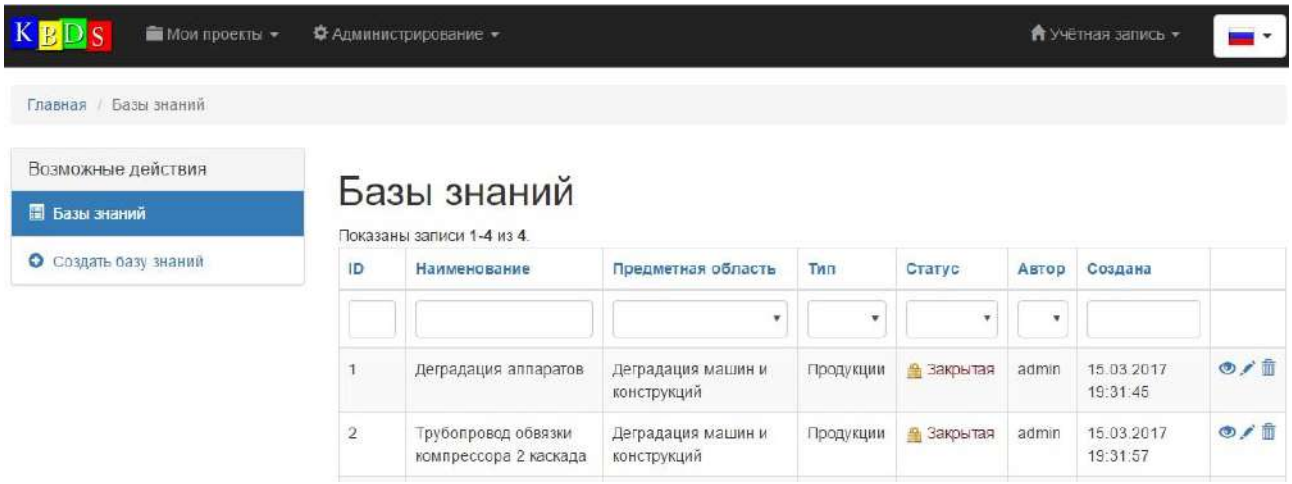


Рис. 5.3.2. Интерфейс KBDS: Администрирование проектов БЗ

Рис. 5.3.3. Интерфейс KBDS: Редактор модели онтологии

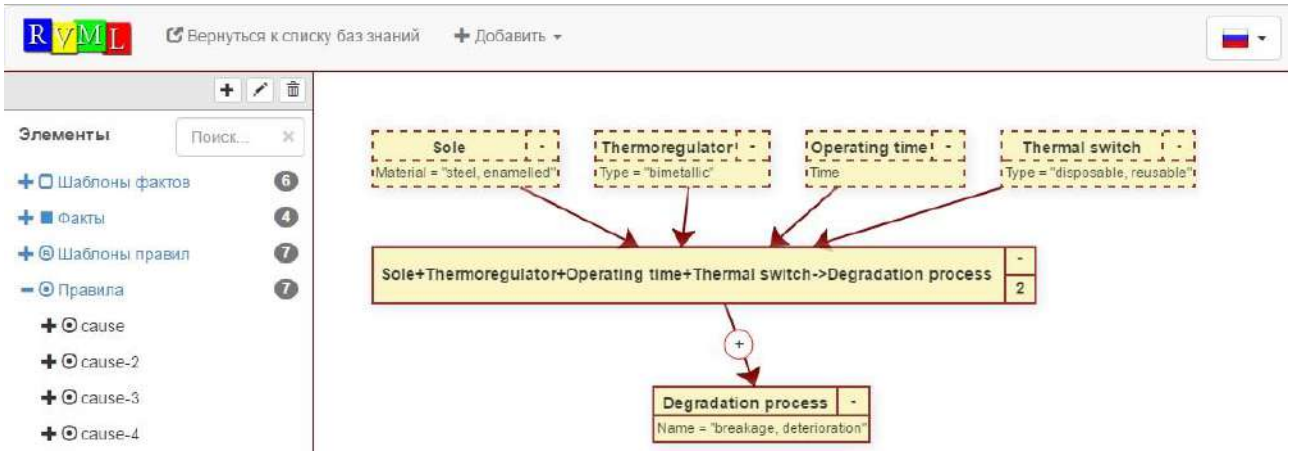


Рис. 5.3.4. Интерфейс KBDS: Редактор RVML

- getAllKnowledgeBasesList – получение списка всех БЗ KBDS;
- getKnowledgeBasesList – получение списка БЗ KBDS с определенным типом и статусом;

- importConceptualModel – импорт концептуальной модели и создание на ее основе БЗ в KBDS;
- exportKnowledgeBase – экспорт кода БЗ из KBDS.

Далее опишем ключевые этапы методики описания программ трансформаций с использованием KBDS и TMRL [46, 239], при этом первые два являются подготовительными:

1) Создание концептуальной модели средствами внешних программ.

2) Сериализация созданной модели в XML-подобном формате средствами внешних программ, т.к. XML являются наиболее распространенным способом внешней интеграции программных систем и обеспечения обмена информацией между ними и именно его поддержка обеспечена в KBDS.

3) Парсинг (анализ) сериализованного файла входной концептуальной модели с помощью модуля KBDS, с целью извлечения основных элементов (понятий, свойств и связей) метамодели.

4) Формирование метамодели на основе информации об извлеченных элементах в специальном графическом редакторе метамоделей KBDS, также она может быть выбрана из списка сформированных ранее моделей.

5) Выбор целевой (выходной) метамодели, по умолчанию в KBDS предлагается использование только двух выходных моделей: онтологической и продукций, для которых затем можно будет получить программный код OWL и CLIPS, соответственно.

6) Определение соответствий между элементами исходной и целевой метамоделей в графическом редакторе трансформаций KBDS. При этом каждое соответствие представляет собой правило трансформации (преобразования), очередность исполнения которого устанавливается их приоритеты.

7) Генерация TMRL кода, соответствующего определенным трансформациям.

8) Уточнение и доопределение трансформаций путем описания сложных зависимостей на TMRL в специальном текстовом редакторе KBDS.

Подробное применение данной методики при описании различных трансформации рассмотрено в Приложении И.

Методика создания компонентов-конверторов трансформации

концептуальных моделей [46, 239] в KBDS состоит из следующих основных этапов:

1) Определени типа и статуса компонента-конвертора (типы и статусы описаны в Приложении Б), определяющих набор доступных метамоделей для входных и выходных форматов.

2) Выбор или формирование выходной и выходной метамоделей.

3) Разработка программы трансформации на TMRL.

4) Создание программного компонента-конвертора из соответствующих готовых блоков анализатора и генератора (исходя из выбранного ранее типа программного компонента).

Особенностями разработанного программного средства являются:

- Расширяемость в части создания новых программных компонентов-конверторов, обеспечивающих преобразование различных концептуальных моделей, сериализованных в XML-подобных форматах, и генерацию кода БЗ на ЯПЗ CLIPS или OWL.
- Наличие программного интерфейса (API) для взаимодействия со внешними программными системами и предоставления доступа к программным компонентам-конверторам KBDS в части импорта концептуальных моделей и получения кода БЗ на их основе.

Выводы

Разработаны новые методы проектирования программ трансформаций концептуальных моделей и программных компонентов-конверторов концептуальных моделей, отличающиеся от известных использованием оригинального языка описания моделей трансформаций – TMRL.

Методы реализованы в составе оригинального программного средства – Knowledge Base Development System (KBDS), ориентированного на непрограммирующих пользователей и включающей средства для визуального программирования метамоделей, онтологий и трансформаций, а также генерации программных кодов на CLIPS или OWL.

Применение методов и программного средства позволит создавать компоненты-конверторы для использования разработанных ранее концептуальных

моделей при автоматическом формировании структур БЗ и кодогенерации, что снизит трудоемкость процесса проектирования ИС. При этом программы трансформации концептуальных моделей могут создаваться конечными пользователями.

РАЗДЕЛ 3. Применение разработанных языков, методов и программных средств

Применение и оценка эффективности применения разработанных языков, методов и средств осуществлено при решении ряда практических и учебных задач, в частности:

1) Обеспечения надежности и безопасности оборудования в нефтехимии для (АО ИркутскНИИХимМаш), в том числе:

- разработка БЗ для прогнозирования развития деградационных процессов в нефтехимии [15, 16, 184, 185, 228, 234] для ИАС «Экспертиза ПБ» [309] с использованием деревьев событий и концепт-карт, описывающих онтологию [45, 259];
- прототипирование продукционных и прецедентных ИС на основе модельных трансформаций, в том числе для задач выявления причин повреждений и разрушения элементов технических систем [16, 184, 185, 257] и подбора конструкционного материала на этапе проектирования изделия [18, 229];
- применение модельных трансформаций в интеллектуальном планировщике ИС «INFOT-3» [12, 13, 223] для анализа отказов;
- применение модельных трансформаций для создания проблемно-ориентированного редактора баз знаний [186, 230];
- применение модельных трансформаций для создания систем идентификации технических состояний конструкций с помощью программы-оболочки «E-INFOT» [116, 325];
- применение модельных трансформаций при прототипировании онтологий для задач экспертизы промышленной безопасности [48, 54, 194, 197].

2) Разработка базы знаний модуля идентификации лицевых признаков (ООО «Смарт Технологии») [193, 201, 202].

3) Разработка базы знаний модуля обнаружения нежелательных сообщений (ООО «ЦентраСиб») [193, 201, 202, 371].

4) Прототипирование баз знаний для системы поддержки технического персонала при поиске и устранении неисправностей системы электроснабжения воздушного судна [53, 95, 199] (МГТУ ГА).

5) Прототипирование баз знаний сервиса для анализа и прогнозирования риска (опасности) лесного пожара на основе информации о классе пожароопасности лесов, метеоусловий и других факторов [200, 202].

6) Создании программных компонентов для трансформации концептуальных моделей, в том числе:

- диаграмм классов UML [198, 276, 267, 366], разработанных в CASE-средстве IBM Rational Rose и представленных с использованием стандарта XMI;
- концепт-карт (карт знаний/интеллект-карт) [277], разработанных в редакторе IHMC CmapTools и представленных с использованием стандарта XTM;
- деревьев событий (ДС) [274], разработанных в графическом редакторе TreeEditorET [224] и сохраненных в XML-подобном формате.

Глава 6. Интеллектуализация решения задач обеспечения надежности и безопасности оборудования в нефтехимии

В главе описаны примеры применения ящыков, методов и средств для решения задач в области техногенной безопасности, часть которых решалось во взаимодействии с сотрудниками ОА «ИркутскНИИХимМаш»:

- разработка БЗ прогнозирования развития деградационных процессов в нефтехимии для ИАС «Экспертиза ПБ» (рег.№ 2016610757) [228, 234, 309] с использованием деревьев событий и отказов [17, 274], концепт-карт и онтологий [275-277];
- прототипирование продукционных и прецедентных ИС на основе модельных трансформаций [16, 18, 184, 185, 257, 229];
- применение модельных трансформаций в интеллектуальном планировщике процесса анализа отказа в ИС «INFOT-3» (рег.№ 2007613715) [12, 13, 218, 223];
- разработка систем идентификации технических состояний конструкций с помощью программы-оболочки «E-INFOT» (рег.№ 2005611217) [116, 325];
- разработка онтологии для задач экспертизы промышленной безопасности на основе полуавтоматического анализа электронных таблиц из отчетов по ЭПБ [47, 48, 54, 196, 197, 289].

По результатам применения технологии к решению данных задач получен акт об использовании результатов научных исследований (см. Приложение А).

6.1 База знаний ИАС «Экспертиза ПБ» для прогнозирования развития деградационных процессов в нефтехимии

Раздел посвящен описаню применения разработанных в диссертации методов и средств при создании БЗ для информационно-аналитической системы «Экспертиза промышленной безопасности» (ИАС «Экспертиза ПБ») [228, 234, 309, 369].

6.1.1 Экспертиза промышленной безопасности

Проблема оценки и повышения безопасности промышленных объектов с течением времени не теряет свою актуальность, что обусловлено высокими темпами старения (деградации) оборудования во многих отраслях промышленности, которые превышают темпы его замены и модернизации, как по субъективным, так и объективным причинам. Особенно это относится к длительно эксплуатирующемуся нефтехимическому, нефтеперерабатывающему и химическому оборудованию, представляющему потенциальную угрозу для населения и окружающей среды. По этой причине необходимо периодическое и полное (стопроцентное) обследование такого оборудования с целью выявления потенциально опасных зон и принятия соответствующих решений для предотвращения катастрофических отказов, аварий и чрезвычайных ситуаций. Качество решения задач обследования определяет величину потерь при авариях, стоимость проведения диагностирования, а также периодических и восстановительных ремонтов.

Значительное повышение надежности и безопасности нефтехимического оборудования возможно обеспечить путем создания и активного использования методов и средств искусственного интеллекта, в частности ИС. ИС (ЭС) могут применяться для интерпретации технических условий и параметров эксплуатации, обоснования программы технического диагностирования, интерпретации результатов диагностирования, которые, в свою очередь, обеспечат более точную оценку технического состояния, определение причин его изменения, объем и содержание восстановительного ремонта, что в целом состоваляет сущность

процедуры экспертизы промышленной безопасности (ЭПБ).

В частности, процедура ЭПБ включает следующие основные этапы:

- планирование работ для ЭПБ;
- анализ технической документации;
- формирование карты исходных данных;
- разработка программы ЭПБ;
- техническая диагностика и анализ (в том числе и интерпретация) результатов;
- вычисление остаточного ресурса;
- принятие решений по ремонту.

Анализ этапов процедуры ЭПБ показал, что успешное выполнение некоторых из них, в частности: «разработка программы ЭПБ», «анализ (в том числе и интерпретация) результатов», «принятие решений по ремонту» связано с обработкой больших объемов плохо формализованной данных. При этом эффективность обработки может быть увеличена путем применения ИС, которые обеспечат поддержку специалиста при решении задач:

- интерпретации условий и параметров функционирования;
- обоснования программы технической диагностики;
- интерпретации результатов диагностики.

ИС, ориентированные на решение указанных задач, могут использовать формализм продукций (логических правил), выразительная способность которого позволяет описать причинно-следственный комплекс (ПСК) изменения технического состояния узлов и элементов обследуемого оборудования.

ПСК [236] отражает взаимосвязь условий эксплуатации, внешних воздействующих факторов, ошибок и несовершенств проектирования, изготовления и эксплуатации, и обусловленных этим деградационных процессов. ПСК формируется исходя из особенностей структуры объекта и возможной последовательности переходов (изменений) его состояний в процессе функционирования и может быть представлен логической моделью. В контексте предметной области определены следующие классы возможных технических состояний деталей и конструкций: исходная дефектность, поврежденность,

разрушение, отказ. Каждое из состояний характеризуется множеством параметров со значениями. Изменение технического состояния вызвано развитием деградационных процессов, возникающих в объекте исследования при наличии определенных внешних и внутренних воздействующих факторов, и представлено в виде последовательности стадий развития этих процессов.

Представление ПСК в форме БЗ для диагностической или прогностической ИС накладывает на нее требования полноты и непротиворечивости. Одним из способов выполнения которых является использование при построении БЗ структурированной процедуры экспертного опроса. Именно эта процедура обеспечивает получение от эксперта всей необходимой информации для формирования БЗ на терминологическом и аксиоматическом уровнях. При этом именно набор предметных концептуальных моделей, сформированных с применением разных нотаций является результатом процедур извлечения (получения) и структурирования экспертных знаний.

В рамках задач диссертационного исследования для создания БЗ были использованы: диаграммы Исикавы и средство для их построения XMind, деревья событий и средство для их построения TreeEditorET (рег.№ 2012614092) [224], диаграммы классов UML и средства StarUML и IBM Rational Rose, а также концепт-карты и средство для их построения IHMC CmapTools.

Отметим, что выбор разных видов концептуальных моделей был обусловлен не только исследовательским характером выполняемых работ, но и многоаспектностью объекта исследования. В частности, практика показала, что деревья событий позволили наглядно моделировать динамику систем, диаграммы Исикавы – причинно-следственные отношения, концепт-карты – построить детальные модели отдельных понятий, а диаграммы классов UML – отразить структурный аспект предметной области.

Процесс создания БЗ в целом соответствует методу, рассмотренному в 4.1, первым этапом которого является анализ предметной области (Рис. 6.1.1). Небольшие отличия касаются заключительных этапов, поскольку цель заключалась разработка БЗ, а не ИС полностью.

6.1.2 Использование деревьев событий для проектирования баз знаний

По результатам анализа предметной области была создана концептуальная модель в форме деревьев событий (ДС). ДС представляют собой формализм моделирования последовательностей событий с четким определением начального (корневого или исходного) события и отсутствием циклов. ДС используются для детализации и анализа различных вариантов развития аварий и других нежелательных и опасных состояний: повреждений, разрушений, отказов, аварийных ситуаций и чрезвычайных ситуаций [222, 233].

Схема разработки БЗ на основе анализа концептуальных моделей (в том числе и ДС) для ИАС «Экспертиза ПБ» представлена на Рис. 6.1.1.

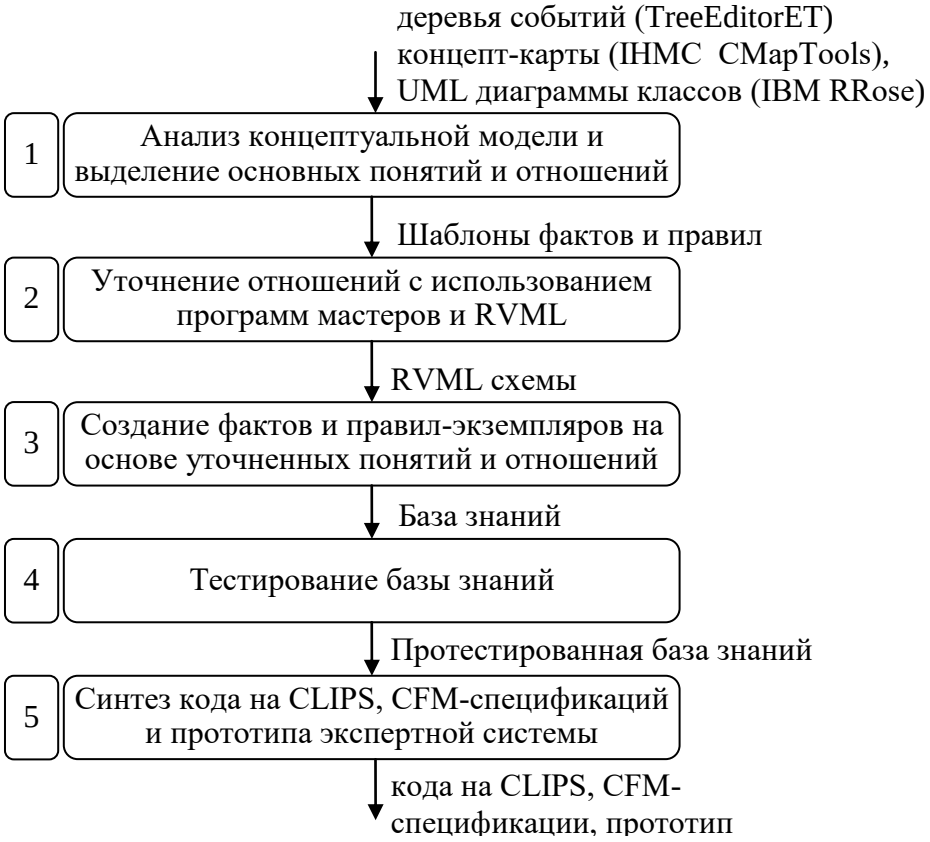


Рис. 6.1.1. Схема разработки баз знаний на основе анализа концептуальных моделей для ИАС «Экспертиза ПБ»

Для построения ДС в рамках диссертационного исследования было разработано и применено программное средство TreeEditorET [224], основной частью которого является редактор построения деревьев событий Рис. 6.1.2.

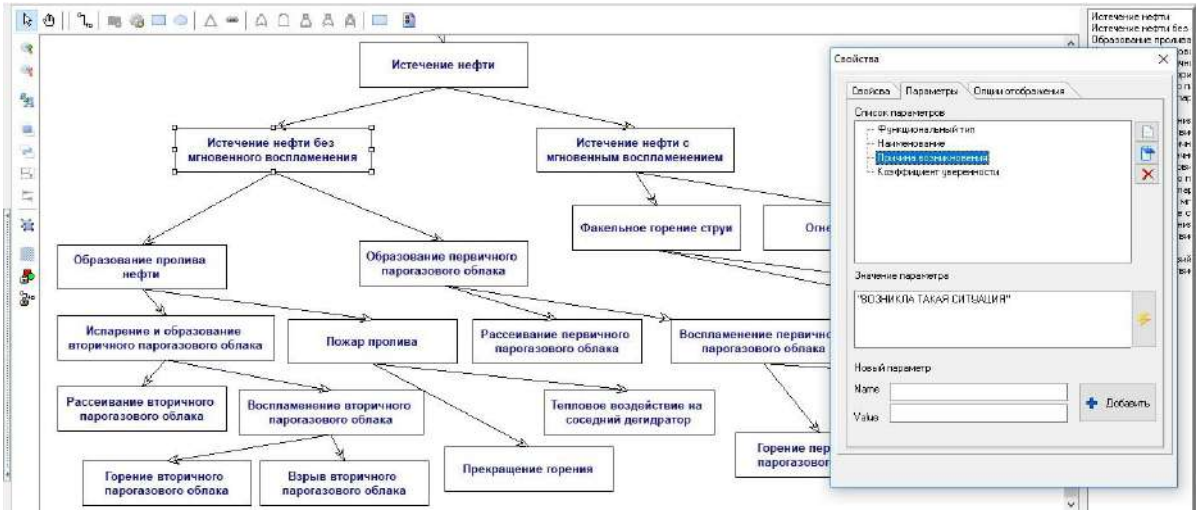


Рис. 6.1.2. Пример интерфейса графического редактора ДС TreeEditorET

Разработанное средство обеспечивало возможность сохранения построенных ДС в XML-подобном формате для обеспечения их дальнейшей трансформации. Пример XML-кода, соответствующего ДС на Рис. 6.1.2, представлен ниже:

```
<EventTree caption="Дерево событий"
  Graph_type_node="TRectangularNode"
  Func_type_node="system_EventTree"
  can_deleted="true" can_move="false" SGmode="tmEventTree"
  SGCaption="Дерево событий" Name="8">
  <InitialEvent caption="Истечение нефти"
    Func_type_node="eventET"
    id="1" can_deleted="true" can_move="true"
    Graph_type_node="TRectangularNode" Name="9"/>
  <Event Func_type_node="eventET"
    Graph_type_node="TRectangularNode" can_move="true"
    can_deleted="true" id="3" caption="Истечение нефти
    без мгновенного воспламенения" tag-ev="0" Name="10">
    <Parameter id="p1" caption="probability" value="0,95"/>
    <Parameter id="p2" caption="existance" value=
      "ДА"/>
    <Parameter id="p3" caption="cf" value="0,9"/>
 </Event>
 <CauseEffectRelation Graph_type_relation="TDirectedArrow"
  id="40" Name="32">
  <Cause id="41" event="1" Name="33"/>
  <Effect id="42" event="3" Name="34"/>
 </CauseEffectRelation>
```

Фрагменты одного из построенных ДС, описывающие динамику развития аварии «Истечение нефти» приведены на Рис. 6.1.3 и 6.1.4.

После согласования построенных деревьев с экспертом было осуществлено их преобразование с помощью модулей PKBD и KBDS, основная цель которого состояла в установлении соответствия между элементами ДС и структурами продукционной БЗ с дальнейшим их отображением в виде диаграмм RVML.

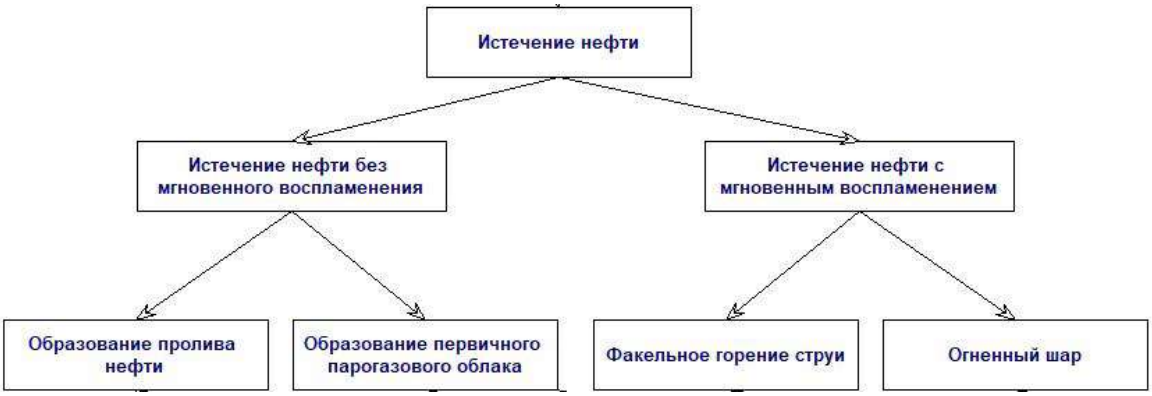


Рис. 6.1.3. Фрагмент ДС развития аварии «Истечение нефти» (1)

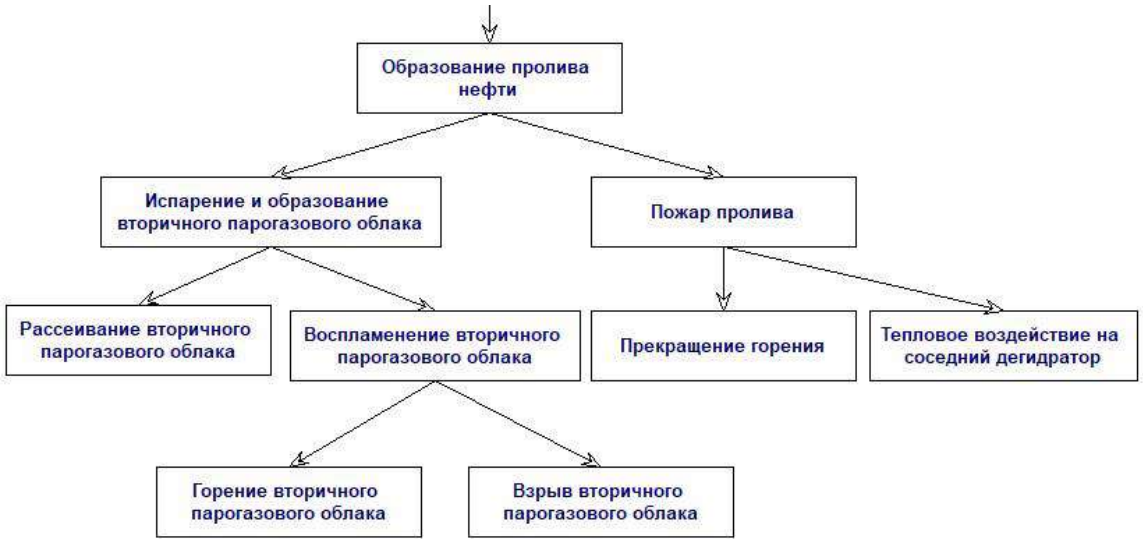


Рис. 6.1.4. Фрагмент ДС развития аварии «Истечение нефти» (2)

В результате выполнения преобразования были получены наборы шаблонов фактов и правил (Рис. 6.1.5-6.1.9).

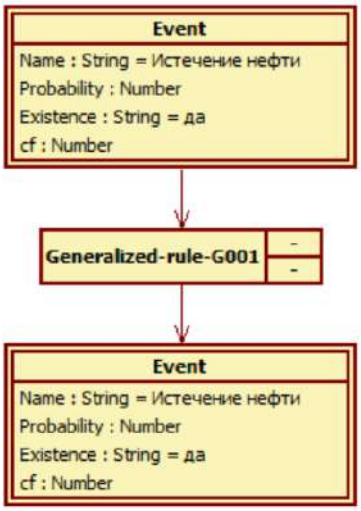


Рис. 6.1.5. Диаграмма RVML, отображающая шаблон правила

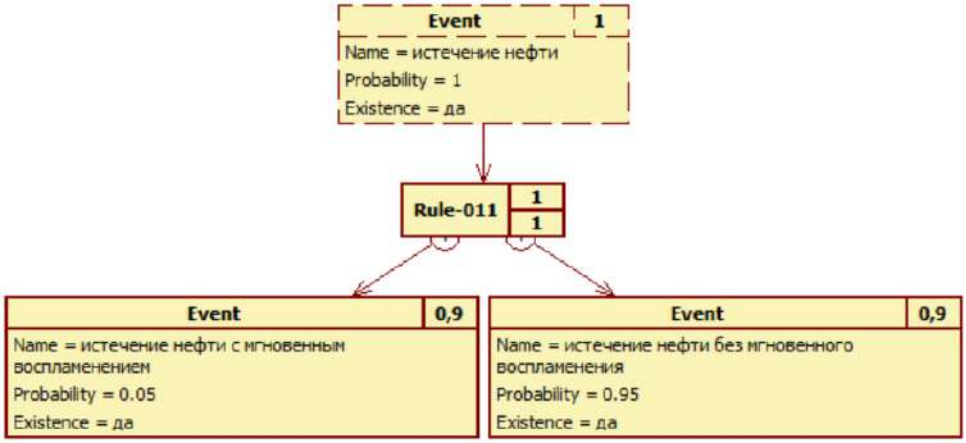


Рис. 6.1.6. Диаграмма RVML, отображающая правило, соответствующее шаблону на Рис.6.1.5

Рис. 6.1.7. Диаграмма RVML, отображающее правило развития аварии «Истечение нефти» (1)

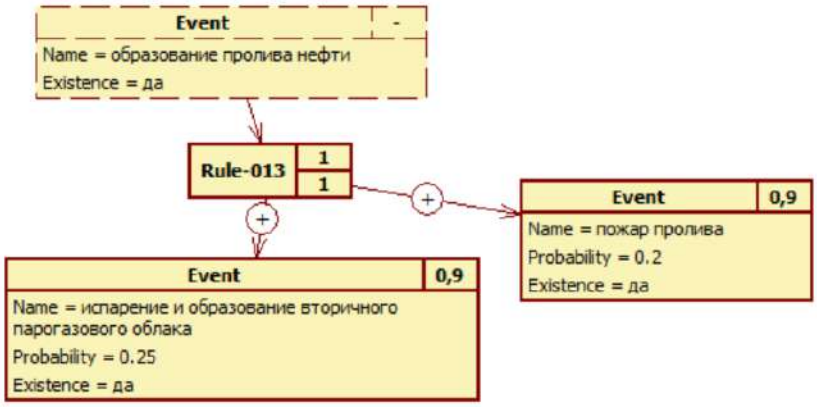


Рис. 6.1.8. Диаграмма RVML, отображающее правило развития аварии «Истечение нефти» (2)

После уточнения построенных диаграмм и тестовых «прогонов» (отладки) в PKBD был сгенерирован программный код БЗ на CLIPS, который затем был перенесен в ИАС «Экспертиза ПБ».

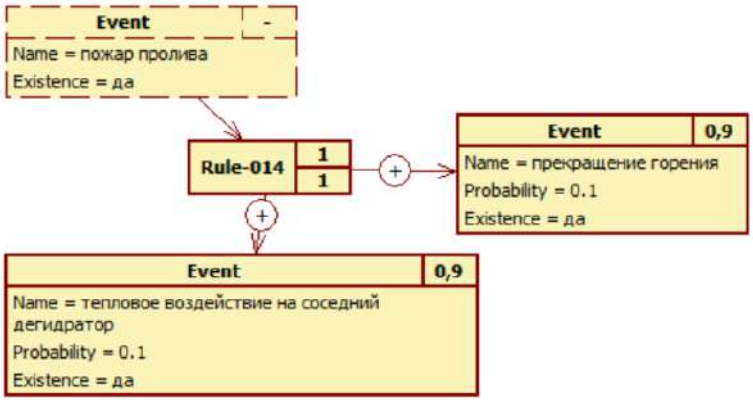


Рис. 6.1.9. Диаграмма RVML, отображающее правило развития аварии «Истечение нефти» (3)

Фрагмент сгенерированного кода на CLIPS:

```
;********************* Templates ****************************
(deftemplate Event ;Description of the template T001
 (slot Name)
 (slot Probability)
 (slot Existence (default "ДА"))
 (slot cf)
)
…
;************************ Rules ***************************
(defrule Rule-011 "Description of the rule: Generalized-rule-G001 1"
(declare (salience 1))
 (Event ;Event
  (Name "ИСТЕЧЕНИЕ НЕФТИ")
  (Probability 1)
  (Existence "ДА")
 )
=>
(assert
 (Event ;Event
  (Name "ИСТЕЧЕНИЕ НЕФТИ С МГНОВЕННЫМ ВОСПЛАМЕНЕНИЕМ")
  (Probability 0.05)
  (Existence "ДА")
 ))
(assert
 (Event ;Event
  (Name "ИСТЕЧЕНИЕ НЕФТИ БЕЗ МГНОВЕННОГО ВОСПЛАМЕНЕНИЯ")
  (Probability 0.95)
  (Existence "ДА")
  ))
 )
…
```

Помимо моделирования динамики развития аварий, используя выразительные способности ДС, исследовался причнно-следственный комплекс (ПСК) [236] деградационных процессов, включающий такие понятия как механизм, кинетика и их параметры. Только на основе корректно и своевременно определенных элементов ПСК возможно обосновать решения по предотвращению и снижению последствий развития таких процессов [228, 236].

В рамках исследования были выделены следующие типы деградационных процессов [228]:

- коррозионное растрескивание;
- коррозионная усталость;
- механическая усталость;
- водородное охрупчивание;
- радиационное охрупчивание;
- водородное растрескивание;
- изнашивание.

Для более полного (детального) моделирования ПСК была использована расширенная модель ДС [225], включающая следующие новые элементы: структурные блоки для визуализации стадий развития исследуемых процессов на различных уровнях (субмикроуровень, микроуровень, мезоуровень, макроуровень); отдельные графические элементы для отображения понятий «механизм» и «кинетика», представляющие собой узлы с особой семантикой, при этом каждый из специализированных узлов может быть дополнительно описан набором сойств со значениями. Фрагмент обобщенной (без конкретизации определенного деградационного процесса) расширенной модели ДС представлен на Рис. 6.1.10.

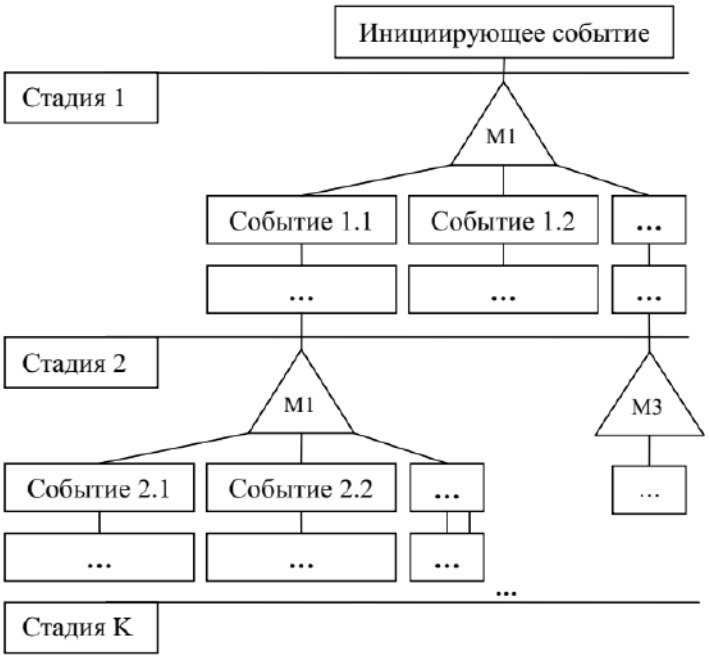


Рис. 6.1.10. Фрагмент обобщенной расширенной модели ДС

Для отображения понятия «механизм» в нотации расширенных ДС используется узловой элемент в форме треугольника. С содержательной точки зрения «механизм» представляет собой комплекс воздействующих факторов и свойств объекта, обуславливающих зарождение определенного деградационного процесса. Данный комплекс может быть формализован с помощью продукций, имеющих следующую структуру:

ЕСЛИ (Свойство объекта$_1$ **И … И** Свойство объекта$_n$) **И**

(Воздействующий фактор$_1$ **И … И** Воздействующий фактор$_m$)

ТО Механизм *j*-стадии развития *i*-исследуемого процесса **И**

(Событие$_1$ ° Событие$_k$)

где ° – некоторая логическая операция, $\circ \in \{\wedge, \vee, \oplus\}$.

Метамодель расширенной нотации ДС, определяющая в абстрактной форме основные элементы ДС, и более подробно описание XML-формата для хранения разработанных с помощью TreeEditorET ДС приведено в [225].

Проиллюстрируем процесс создания БЗ на основе трансформации ДС с последующей генерацией кода на CLIPS. В качестве исходной концептуальной модели использован фрагмент ДС (Рис. 6.1.11) для стадии «повреждение» деградационного процесса «коррозионная усталость» исследуемой элемента «фланцевое соединение (Ду 6)» в трубопроводе обвязки компрессора.

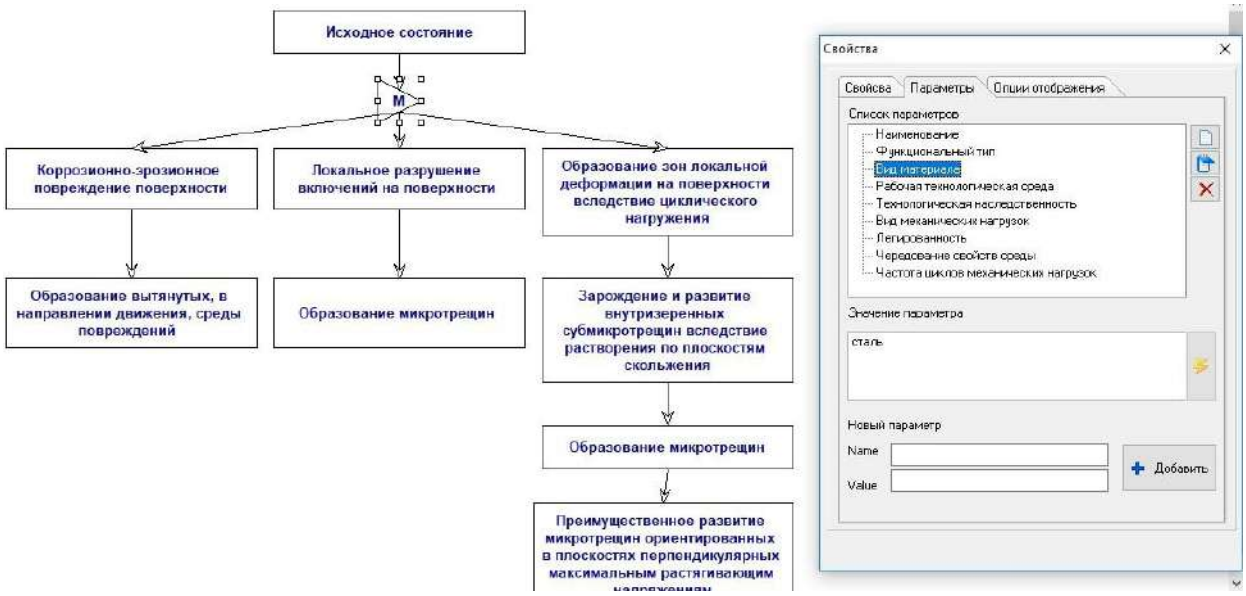


Рис. 6.1.11. Фрагмент ДС в расширенной нотации с описанием развития деградационного процесса «коррозионная усталость»

Графическому представлению ДС на Рис. 6.1.11 соответствует следующий фрагмент XML-кода, описывающий механизм деградационного процесса:

```
      <Mechanism id="MEC-1" name="Механизм повреждения" event="IE-1">
        <Operator id="OPR-1" name="AND">
          <InfluencingFactor id="IF-1" name="Вид материала"
value="сталь"/>
          <InfluencingFactor id="IF-2" name="Легированность"
value="низколегированна сталь"/>
          <InfluencingFactor id="IF-3" name="Технологическая наследственность"
value="дефекты изготовления ИЛИ повреждаемость поверхности вследствие
воздействия агрессивной среды"/>
          <InfluencingFactor id="IF-4" name="Рабочая технологическая среда"
value="активная"/>
          <InfluencingFactor id="IF-5" name="Вид механических нагрузок"
value="переменные"/>
          <InfluencingFactor id="IF-6" name="Чередование свойств среды"
value="да"/>
          <InfluencingFactor id="IF-7" name="Частота циклов механических
нагрузок" value="высокая (> 60 цикл/мин)"/>
        </Operator>
        <ProcessMechanism id="PM-1" name="Коррозионная усталость"/>
      </Mechanism>
```

По результатам анализа и трансформации ДС (см. Рис. 6.1.11) построена продукционная БЗ, элементы которой представлены на Рис. 6.1.12-6.1.14 в виде диаграмм RVML для дальнейшего уточнения конечным пользователем.

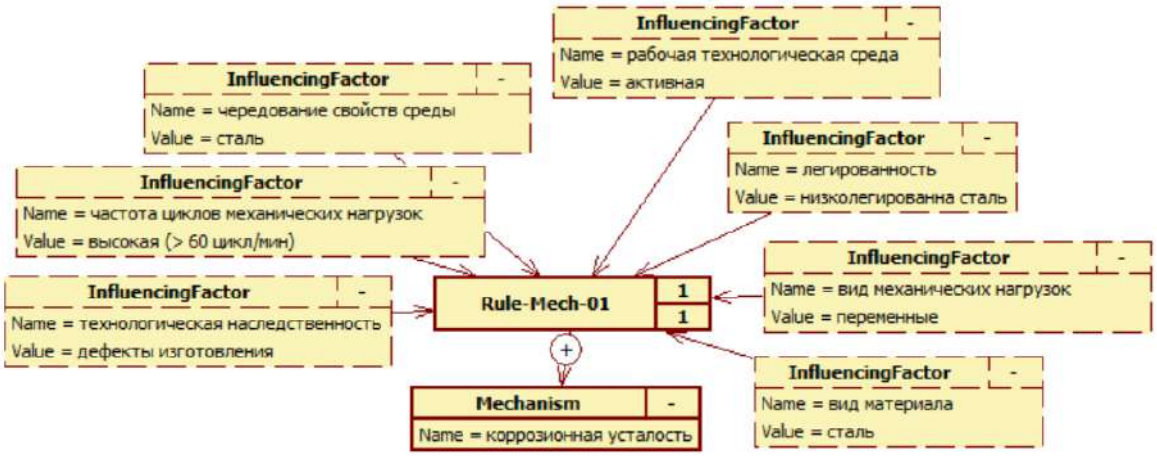


Рис. 6.1.12. Логическое правило в нотации RVML: факторы, обусловившие механизм

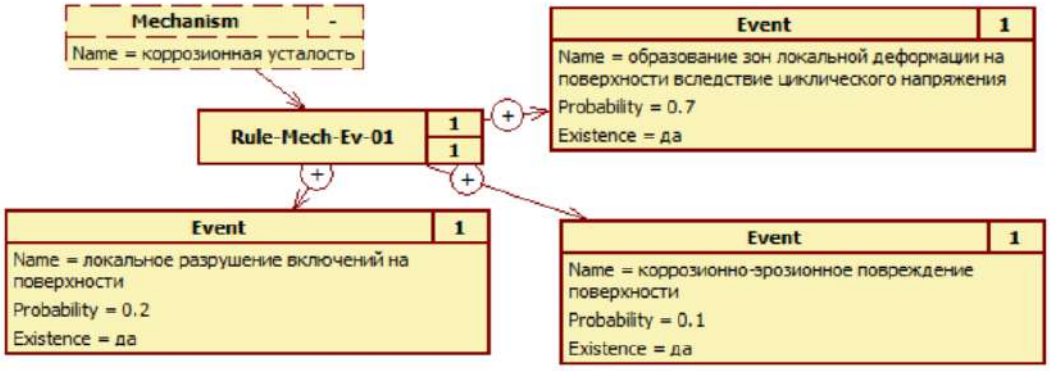


Рис. 6.1.13. Логическое правило в нотации RVML: события, обусловленные механизмом

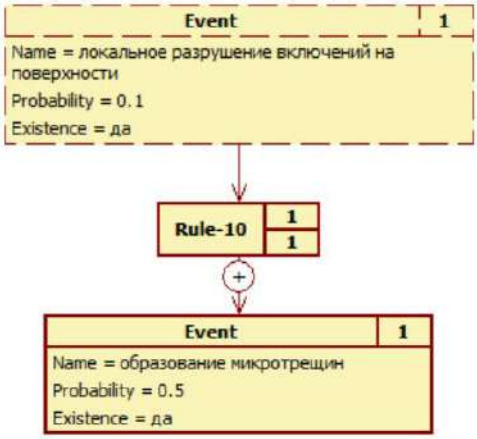


Рис. 6.1.14. Логическое правило в нотации RVML: последовательность событий

После уточнения структуры и содержания БЗ был сгенерирован код на CLIPS (фрагмент для Рис. 6.1.12):

```
(defrule Rule-Mech-01 "Description of the rule: Generalized-rule-G002 1"
(declare (salience 1))
 (InfluencingFactor ;InfluencingFactor
  (Name "ЧАСТОТА ЦИКЛОВ МЕХАНИЧЕСКИХ НАГРУЗОК")
  (Value "ВЫСОКАЯ (> 60 ЦИКЛ/МИН)")
 )
 (InfluencingFactor ;InfluencingFactor
  (Name "ЧЕРЕДОВАНИЕ СВОЙСТВ СРЕДЫ")
  (Value "СТАЛЬ")
 )
 (InfluencingFactor ;InfluencingFactor
  (Name "ВИД МЕХАНИЧЕСКИХ НАГРУЗОК")
  (Value "ПЕРЕМЕННЫЕ")
 )
 (InfluencingFactor ;InfluencingFactor
  (Name "РАБОЧАЯ ТЕХНОЛОГИЧЕСКАЯ СРЕДА")
  (Value "АКТИВНАЯ")
 )
 (InfluencingFactor ;InfluencingFactor
  (Name "ТЕХНОЛОГИЧЕСКАЯ НАСЛЕДСТВЕННОСТЬ")
  (Value "ДЕФЕКТЫ ИЗГОТОВЛЕНИЯ")
 )
 (InfluencingFactor ;InfluencingFactor
  (Name "ЛЕГИРОВАННОСТЬ")
  (Value "НИЗКОЛЕГИРОВАННА СТАЛЬ")
 )
 (InfluencingFactor ;InfluencingFactor
  (Name "ВИД МАТЕРИАЛА")
  (Value "СТАЛЬ")
 )
=>
 (assert
  (Mechanism ;Mechanism
   (Name "КОРРОЗИОННАЯ УСТАЛОСТЬ")
  ))
)
```

В результате трансформации рассмотренного на Рис. 6.1.11 ДС была получена БЗ включающая: 3 шаблона для построения фактов, 3 шаблона правил, 4

начальных (initial) факта и 8 конкретных правил. Код БЗ на CLIPS полученный для данной концептуальной модели и для других моделей, описывающих деградационные процессы (механической усталости, изнашивания, эрозии и т.д.) в дальнейшем был синтегрирован в ИАС «Экспертиза ПБ».

6.1.3 Использование концепт-карт и онтологий для проектирования баз знаний

Наряду с ДС при создании БЗ использовались концепт-карты, как средство визуальное представления отдельных онтологических сегментов [51, 55, 259]. Основное их назаначение – детальное описание структуры наиболее важных предметных понятий, к которым были отнесены такие понятия как: «технический объект» (Рис. 6.1.13) и «деградационный процесс» (ДП) (6.1.14) [236].

Рис. 6.1.13. Фрагмент концепт-карты «технический объект»

При этом под деградационными процессами понимаются объективные физико-химические процессы, вызванные технологической наследственностью и эксплуатационными, структурными и производственными нарушениями, обусловившими повреждения и разрушение материалов и элементов технических систем и аппаратов, повлекшие, в свою очередь, их отказы [236]. Для описания факторов, обусловивших зарождение и развитие ДП введены понятия «механизм» и «кинетика». Механизм ДП – это набор свойств исследуемого объекта и воздействующих на него факторов. Кинетика ДП – это множество микро- и/или макроскопических явлений (событий), возникающих в результате накопления

элементарных актов движения. Детальное представление кинетики включает описание этих события, их параметров и функциональных отношений (при возможности) для определения значений параметров в определенный момент времени. В области исследования технической и техногенной безопасности ДП называются опасными процессами, что обусловлено характером их последствий.

Последовательность основных этапов при разработке БЗ на основе трансформации концепт-карт соответствует предлагаемому методу (см. 4.1).

На первом этапе было осуществлено построение вычислительно-независимой модели в форме онтологии предметной области, при этом использовалась концептуальная модель динамики технических состояний из [223], согласно которой выделяются такие стадии развития ДП механических и технических систем как повреждение, разрушение и отказ. Онтология представлялась в виде концепт-карт с помощью программного средства IHMC CmapTools и описывала различные ДП (коррозионная усталось, водородное охрупчивание и др.).

На Рис. 6.1.14 и 6.1.15 приведены фрагменты разработанных концепт-карт для механизма и кинетики ДП «водородное охрупчивание» на стадии повреждения. Для описания иерархии понятий использовано отношение типа «имеет подкласс», а отношения типа «имеет свойство» и «имеет» – для индикации явных связей между понятием и его свойством. Остальные связи было решено интерпретировать как причинно-следственные.

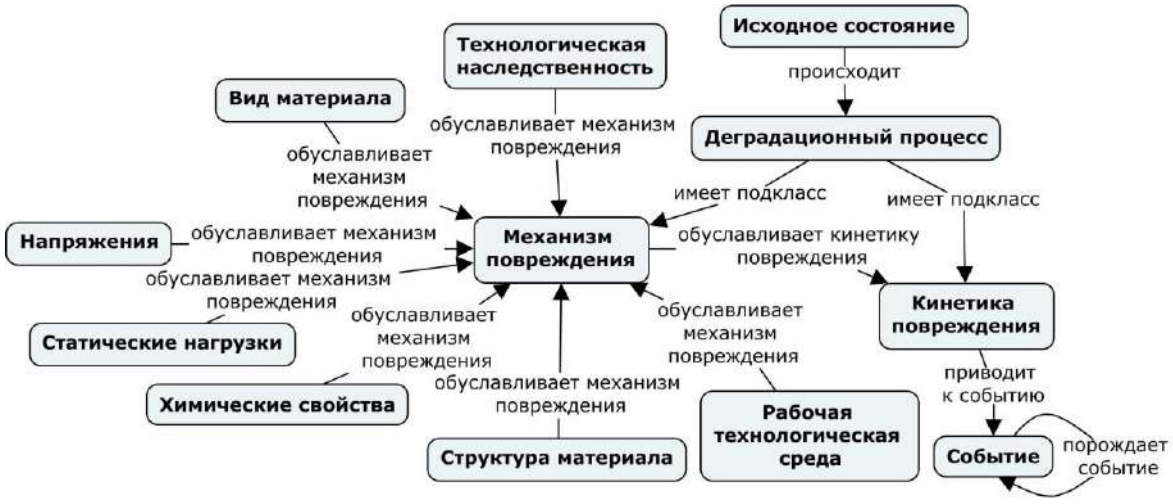


Рис. 6.1.14. Фрагмент концепт-карты «Деградационный процесс»

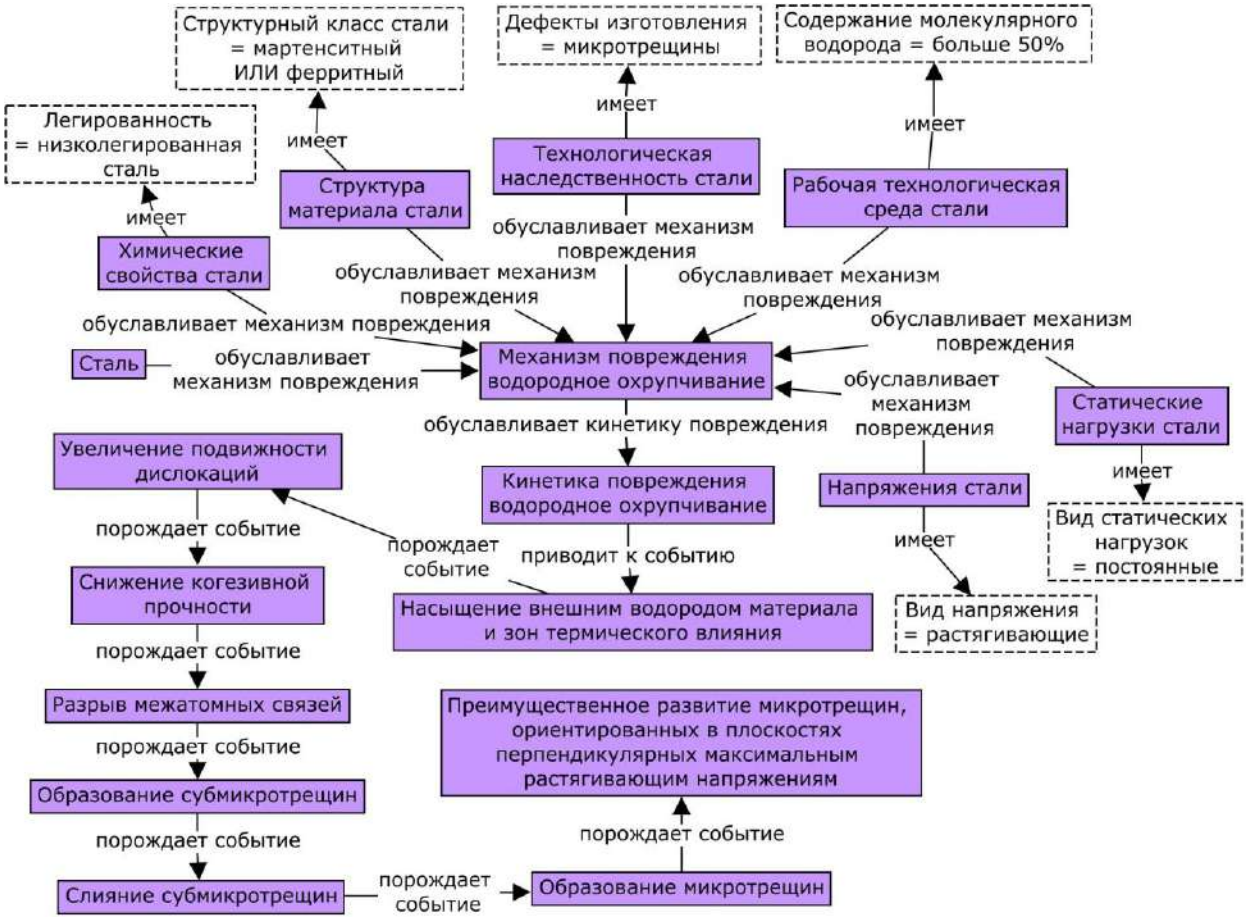


Рис. 6.1.15. Фрагмент концепт-карты ДП «водородное охрупчивание»

Рис. 6.1.16. Фрагмент онтологической модели в визуальном редакторе онтологий KBDS

Построенные концепт-карты были сериализованы и сохранены в XML-подобном формате IHMC CmapTools. Трансформация файлов с моделями произведена с помощью программного средства KBDS, в результате чего понятия,

их свойства и отношения предметной области были отображены в обобщенную модель онтологии. Корректировка и уточнение онтологической модели осуществлялась в визуальном редактора онтологических моделей KBDS (Рис. 6.1.16).

В дальнейшем на основе онтологической модели был сгенерирован код на OWL. Фрагмент полученного OWL-кода (описание классов, объектных свойств и свойств значений) соответствующего Рис. 6.1.14-6.1.16 приведено ниже:

```
...
<owl:Class rdf:about="ВоздействующийФактор" />
<owl:Class rdf:about="МеханическиеНагрузки">
  <rdfs:subClassOf rdf:resource="ВоздействующийФактор" />
</owl:Class>
<owl:Class rdf:about="ДеградационныйПроцесс" />
<owl:Class rdf:about="МеханизмПовреждения">
  <rdfs:subClassOf rdf:resource="ДеградационныйПроцесс" />
</owl:Class>
...
<owl:ObjectProperty rdf:about="Происходит">
  <rdfs:domain rdf:resource="#ИсходноеСостояние" />
  <rdfs:range rdf:resource="#ДеградационныйПроцесс" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="ОбуславливаетМеханизмПовреждения">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="ВидМатериала" />
        <rdf:Description rdf:about="ХимическиеСвойства" />
        <rdf:Description rdf:about="СтруктураМатериала" />
        <rdf:Description rdf:about="ТехнологическаяНаследственность" />
        <rdf:Description rdf:about="РабочаяТехнологическаяСреда" />
        <rdf:Description rdf:about="СтатическиеНагрузки" />
        <rdf:Description rdf:about="Напряжения" />
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#МеханизмПовреждения" />
</owl:ObjectProperty>
...
<owl:DatatypeProperty rdf:about="СодержаниеМолекулярноговодорода">
  <rdfs:domain rdf:resource="#РабочаяТехнологическаяСреда" />
  <rdfs:range rdf:resource="xsd:string" />
</owl:DatatypeProperty>
...
<МеханизмПовреждения
rdf:about="МеханизмПовреждения ВодородноеОхрупчивание">
  <ОбуславливаетКинетикуПовреждения
rdf:resource="#КинетикаПовреждения ВодородноеОхрупчивание" />
</МеханизмПовреждения>
<КинетикаПовреждения
rdf:about="КинетикаПовреждения ВодородноеОхрупчивание">
  <ПриводитКСобытию
rdf:resource="#НасыщениеВнешнимВодородомМатериалаИЗонТермическоговлияния" />
</КинетикаПовреждения>
...
```

Уточненная онтологическая модель использовалась для построения БЗ, при этом понятия были преобразованы в факты и правила. В свою очередь для редактирования и согласования продукций с экспертом использовался редакторы RVML диаграмм KBDS и PKBD (Рис. 6.1.17). В качестве целевых платформ для генерации кодов были выбраны CLIPS, специфкаций – PKBD.

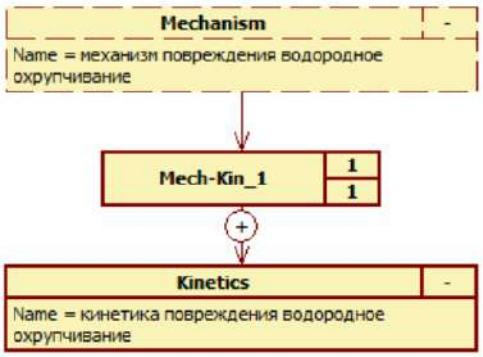


Рис. 6.1.17. Фрагмент продукционной БЗ: диаграмм RVML с описанием правила, описывающего связь механизма и кинетики ДП «водородное охрупчивание»

6.1.4 Использование диаграмм Исикавы для проектирования баз знаний

При построении БЗ ИАС «Экспертиза ПБ» помимо ДС и концепт-карт рассматривался вариант использования концептуальных моделей в форме диаграмм Исикавы («рыбьей кости» или «рыбьих скелетов») [187, 278]. Диаграммы Исикавы [303] – это концептуальные модели, которые изначально были созданы для моделирования причинно-следственных связей, они отражают совокупность факторов, влияющих на определенный объект. При этом «голова рыбы» (корень) на диаграмме – это исследуемый объект или проблема, а «кости» – это прямо или косвенно влияющие на объект факторы и причины, оказывающие как позитивное воздействие (нейтрализующие), так и негативное (усугубляющие). «Кости» могут быть представлены на нескольких уровнях: основном (первом) – связаны с «хребтом рыбы», обычно на этом уровне представлены категории или классы причин; вторичных или производных (второй, третий и т.д.) – связаны с предыдущим уровнем «костей», на этих уровнях представлены причины и их детализация (параметры и значения). Основная цель применения диаграмм Исикавы в рамках диссертационного исследования – описать ДП с точки зрения конечного пользователя, включая их механизмы и кинетику. При этом для снижения сложности процесса описания ДП специалистами-предметниками был создан специальный шаблон ДП (Рис. 6.1.18).

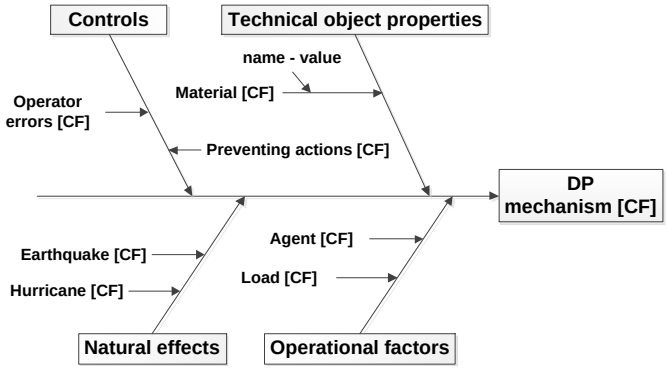


Рис. 5.2.18. Шаблон диаграммы Исикавы с категориями основных причин ДП для заполнения специалистом-предметником

Для реализации трансформаций и разработки специальных модулей-конверторов в соответствии с разработанным подходом была разработана метамодель диаграмм Исикавы (Рис. 6.1.19).

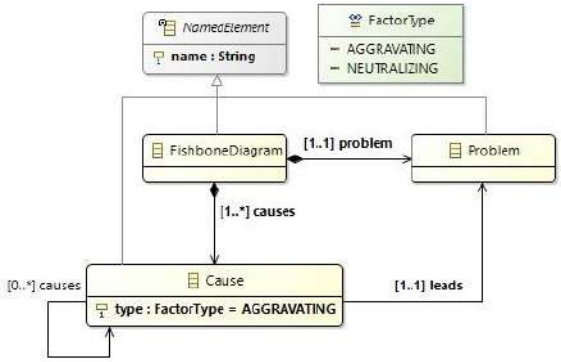


Рис. 5.2.19. Метамодель диаграммы Исикавы

Созданный шаблон ДП (Рис. 6.1.18) были конкретизирован для разных опасных процессов, в частности, «коррозионного растрескивания» (Рис. 6.1.20) и других. Конкретизированные шаблоны были преобразованы в логические правила БЗ и программные коды на CLIPS.

Для построения диаграмм Исикавы было применено программное средство XMind, сохраняющее свои модели в XML-подобном формате «*.xmind». По сути этот файл – это архив с XML-документами и другими каталогами. При реализации трансформации извлекались элементы файла с именем «content.xml», в частности: информация о блоках (узел «topics») и связях между ними (узел «relationships»). В свою очередь, узлы «topic» содержали информацию об идентификаторе блока (тег «id») и его наименовании (тег «title»), а узлы «relationship» – информацию о блоках, между которыми была установлена связь (теги «end1» и «end2») с определенным наименованием (тег «title»). В процессе трансформации на основе

содержания упомянутых узлов были сформированы множества понятий, свойств и отношений, которые в дальнейшем были отображены в структуры продукционной БЗ. Формат XMind не предусматривает возможности индикации определенной семантики узлов «topics», что требовалось в рамках решаемо задачи. Поэтому для однозначного автоматического разделения узлов на понятия предметной области и их свойства, было решено ввести требование на именование элементов (узлов): использовать первую заглавную букву в именах узлов при обозначении понятий и строчную – их свойств и значений.

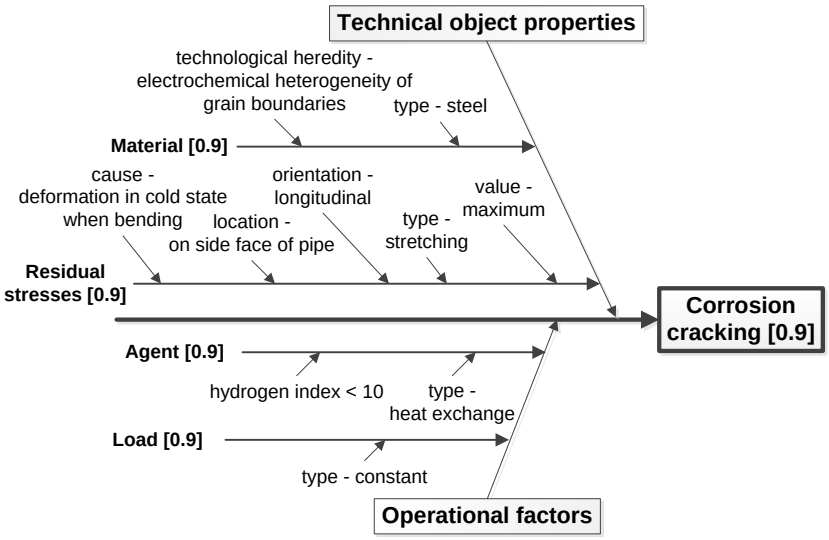


Рис. 6.1.20. Фрагмент диаграммы Исикавы ДП «коррозионное растрескивание» на стадии «повреждение»

Используя приведенное выше описание формата XMind был разработан программный модуль трансформации диаграмм Исикавы для PKBD.

По результатам использования данных диаграмм в процессе моделирования ДП был сделан вывод о сложности их понимания задействованными специалистами-предметниками, которые отдали предпочтение другим видам концептуальных моделей, в частности, ДС.

Ниже представлен фрагмент одного из прототипов БЗ для задач прогнозирования развития ДП «коррозионная усталость»:

```
;*****************************************************************
; Наименование базы знаний: Knowledge base 10234235031
;*****************************************************************
; Описание:
;*****************************************************************
;********************* Templates ******************************
(deftemplate incident-object ;object of the incident
(slot cf (default 1)) ;certainty factor
(slot caption (default "НЕТ ДАННЫХ")) ;name of the object
)
```

```
(deftemplate object-properties ;property of the object
(slot cf (default 1)) ;certainty factor
(slot caption) ;name of the object
(slot caption-incident-object) ;ref to the object
)
(deftemplate material ;material of the object
(slot cf) ;certainty factor
(slot caption) ;наименование
(slot type) ;вид
(slot mechanical-prop-strength-limit) ;strength limit
(slot mechanical-prop-yield-limit) ;свойства стойкости
(slot resistance-prop-corrosion) ;Corrosion resistance
(slot resistance-prop-temperature) ;temperature resistance
(slot resistance-prop-wear) ;chemical properties
(slot chemical-prop-alloying (default "LOW-ALLOY STEEL")) ;structure
(slot structure-prop-class (default "1")) ;structural class / martensitic / ferritic/
)
(deftemplate technological-heredity ;технологическая наследственность
(slot cf (default 1)) ;коэффициент уверенности
(slot caption) ;наименование
)
(deftemplate making-defects ;дефекты изготовления
(slot caption-technological-heredity (default "DEFECT OF WELDING")) ;ссылка на
технологическую наследственность
(slot type) ;вид /микротрещины/дефекты изготовления/
(slot cause) ;причина /сварка/
(slot location) ;месторасположение /зона терм-го воздействия/
(slot cf (default 1)) ;коэффициент уверенности
)
(deftemplate mechanical-stress-const ;механические нагрузки - статические/постоянные/
(slot stress-const-type (default "HIGH")) ;вид статических нагрузок/внутреннее
давление/мпа//сосредоточенная нагрузка /мн//распределенная нагрузка/
(slot stress-value (default "0")) ;величина нагрузок
(slot tension-type) ;вид напряжения /растягивающие / сжимающие / сдвига/
(slot tension-value) ;величина напряжения
(slot cycle-amplitude) ;амплитуда цикла
(slot cycle-frequency) ;частота цикла
(slot cycle-asymmetry) ;коэффициент асимметрии цикла
(slot cycle-average-value) ;среднее значение цикла
(slot speed) ;скорость
(slot speed-up) ;ускорение
(slot max-stress-value) ;максимальное значение нагрузок
(slot cf (default "1")) ;коэффициент уверенности
)
(deftemplate technological-environment ;/рабочая/ технологическая среда
(slot contents-molecular-hydrogen) ;содержание молекулярного водорода /меньше 10%
об/от 10% до 50% об/больше 50 % об/
(slot ph) ;водородный показатель /меньше 0-7 /рабочая среда кислая, кислотность
увеличивается к нулю/, = 7 /рабочая среда нейтральная/, больше 7-15 /рабочая среда
щелочная, щелочные свойства увеличиваются к 15//
(slot radiation) ;радиация /термическая / электромагнитная / ионизирующая/
(slot properties-alternation (default "YES")) ;чередование свойств среды /да/нет/
(slot environment-humidity) ;влажность среды
(slot environment-flash) ;температура вспышки паров среды
(slot cf (default "1")) ;коэффициент уверенности
)
(deftemplate heat-exchange-technological-environment ;теплообменная технологическая
среда
(slot environment-type) ;вид /оборотная вода /слабокоррозионная/ / химически-очищенная
вода /не коррозионная/ / химически-очищенная вода /слабощелочная//
(slot ph (default "НЕЙТРАЛЬНАЯ")) ;водородный показатель
(slot cf (default 1)) ;коэффициент уверенности
)
(deftemplate exist-event ;наблюдаемое событие кинетики
(slot cf) ;коэффициент уверенности
(slot caption) ;наименование события
(slot caption-kin) ;ссылка на кинетику
(slot probabilityrel (default 0)) ;вероятность возникновения
(slot id (default "0"))
)
```

```
(deftemplate exist-dam ;наблюдаемые повреждения
(slot id-dam (default "CRACK"))
(slot caption-dam) ;наименование повреждения
(slot dam-type) ;тип /в частном случает тип трещинообразования/
(slot dam-stress-type) ;тип напряжения
(slot dam-mesto-zarozdenia) ;место зарождения /например, трещины/
(slot dam-istochnik (default "CORROSION FATIGUE")) ;источник
(slot dam-mestopolozenie) ;местоположение
(slot dam-orientacia) ;ориентировка /например, трещины/
(slot dam-glubina) ;глубина
(slot dam-dlina) ;длина
(slot dam-diametr) ;диаметр
(slot dam-forma) ;форма в изломе
(slot dam-kolichestvo) ;количество повреждений близких по размеру
(slot dam-velichina-raskritia) ;велична раскрытия
(slot dam-tech-pered-razrusheniem) ;критерий течь перед разрушением
(slot dam-napravlenie) ;направление по отношению к главной оси
(slot caption-def)
(slot id-def)
(slot caption-meh)
(slot dam-cf (default 1)) ;коэффициент уверенности
)
(deftemplate exist-des ;наблюдаемые разрушения
(slot id-des (default "BRITTLE FAILURE"))
(slot caption-des) ;наименование
(slot des-istochnik) ;источник
(slot des-orientacia) ;ориентировка /например, трещины/
(slot des-napravlenie) ;направление по отношению к главной оси
(slot des-forma) ;форма
(slot des-kolichestvo) ;количество
(slot des-koncentracia-naprazheniy)
(slot des-mestopolozenie (default "CRACK")) ;местоположение
(slot caption-dam)
(slot id-dam)
(slot caption-meh)
(slot des-cf (default 1)) ;коэффициент уверенности
)
(deftemplate exist-def ;наблюдаемые дефекты
(slot id-def)
(slot caption-def) ;наименование
(slot caption-meh)
(slot def-cf (default 1)) ;коэффициент уверенности
)
(deftemplate exist-meh ;наблюдаемая механизм
(slot caption-meh (default "CORROSION FATIGUE")) ;наименование
(slot meh-cf (default 1)) ;коэффициент уверенности
)
(deftemplate exist-kin ;наблюдаемая кинетика отказа
(slot caption (default "KINETICS 'CORROSION FATIGUE'")) ;наименование
(slot caption-meh (default "CORROSION FATIGUE")) ;ссылка на механизм
(slot cf (default 1)) ;выявления событий -
)
;********************** Facts ******************************
(deffacts initial-settings
(mechanical-stress-const ;mechanical-stress-const
(stress-value 0)
(cycle-frequency "HIGH")
(cf 1)
)
(technological-environment ;technological-environment
(ph "ACTIVE")
(properties-alternation "YES")
(cf 1)
)
(incident-object ;incident-object
(cf 1)
(caption "PIPE INTO PIPE")
)
(material ;material
(cf 1)
```

```
(type "STEEL")
(chemical-prop-alloying "LOW-ALLOY STEEL")
)
)
;************************ Rules ******************************
(defrule fai-mechanism-ky-1001 "правило выявления /механизма/: если имеются
механические нагрузки высокой частоты и активная технологическая среда с чередующимися
свойствами и материал низколегированная сталь и имеются дефекты изготовления то может
возникнуть механизм коррозионная усталость"
(mechanical-stress-const ;механические нагрузки
(cycle-frequency "HIGH") ;-----------
)
(technological-environment ;технологическая среда
(ph "ACID") ;водородный показатель
(properties-alternation "YES") ;чередование свойств среды /да/нет/
)
(material ;материал
(type "STEEL") ;вид
(chemical-prop-alloying "LOW-ALLOY STEEL") ;легированность
(structure-prop-class "1") ;структурный класс /мартенситный / ферритный/
)
(making-defects ;дефекты изготовления
(caption-technological-heredity DEFECT OF WELDING) ;ссылка на технологическую
наследственность
)
=>
(assert
(exist-event ;exist-event
(caption "MECHANISM 'CORROSION FATIGUE'") ;может возникнуть /механизм/
))
(assert
(exist-meh ;exist-meh
(caption-meh "CORROSION FATIGUE") ;может возникнуть /механизм/
))
)
(defrule fai-kinatics-ky-1001 "правило выявления /кинетики/: если наблюдается механизм
коррозионная усталость то может возникнуть кинетика коррозионная усталость"
(exist-event ;exist-event
(caption "MECHANISM 'CORROSION FATIGUE'") ;наблюдается /механизм/
)
=>
(assert
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;может возникнуть /кинетика/
(caption-meh "CORROSION FATIGUE")
))
)
(defrule fai-kinatic-events-ky-1001 "правило выявления /параметры/события кинетики/:
если наблюдается кинетика коррозионная усталость то могут наблюдаться события: утечка
рабочей среды через сквозные трещины и полный выброс рабочей среды"
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;наблюдается /кинетика/
)
=>
(assert
(exist-event ;exist-event
(caption "LEAKAGE OF THE WORKING ENVIRONMENT THROUGH THE THROUGH CRACKS") ;может
возникнуть /событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
(assert
(exist-event ;exist-event
(caption "COMPLETE RESET OF THE WORKING ENVIRONMENT") ;может возникнуть /событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
```

```
(defrule fai-kinatic-events-ky-1002 "правило выявления /последующее событие кинетики/:
если наблюдает-ся кинетика коррозионная усталость и событие утечка рабочей среды через
сквозные трещины то может наблюдаться событие образование сквозной трещины"
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;наблюдается /кинетика/
)
(exist-event ;exist-event
(caption "LEAKAGE OF THE WORKING ENVIRONMENT THROUGH THE THROUGH CRACKS") ;/событие/
(caption-kin "KINETICS 'CORROSION FATIGUE'")
)
=>
(assert
(exist-event ;exist-event
(caption "FORMATION OF THE THROUGH CRACKS") ;может возникнуть /событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule fai-kinatic-events-ky-1003 "правило выявления /последующее событие кинетики/:
если наблюдает-ся кинетика коррозионная усталость и событие полный выброс рабочей
среды то может наблюдаться событие хрупкое разрушение"
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;наблюдается /кинетика/
)
(exist-event ;exist-event
(caption "COMPLETE RESET OF THE WORKING ENVIRONMENT")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
)
=>
(assert
(exist-event ;exist-event
(caption "BRITTLE FAILURE") ;может возникнуть /событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule des-mechanism-ky-1001 "Описание правила: des-mechanism-ky-1001"
(exist-dam ;exist-dam
(caption-dam "CRACK")
(dam-orientacia "PERPENDICULARLY")
(dam-napravlenie "LONGITUDINAL")
(caption-meh "CORROSION FATIGUE")
(id-dam ?id)
)
=>
(assert
(exist-des ;exist-des
(caption-des "MACROCRACK")
(des-istochnik "SURFACE DAMAGE")
(des-orientacia "PERPENDICULARLY")
(des-napravlenie "LONGITUDINAL")
(des-kolichestvo "SINGLE")
(caption-meh "CORROSION FATIGUE")
(caption-dam "CRACK")
(id-dam ?id)
(id-des ?id)
))
(assert
(exist-event ;exist-event
(caption "FORMATION OF THE MACROCRACKS OF THE UNACCEPTABLE SIZE") ;может возникнуть
/событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule des-mechanism-ky-1002 "Описание правила: des-mechanism-ky-1002"
(exist-dam ;exist-dam
(caption-dam "CRACK")
(dam-istochnik "CONSTRUCTIVE STRESS CONCENTRATOR FORMED BY THE INTERSECTION OF HOLES")
(dam-orientacia "PERPENDICULARLY")
```

```
(dam-napravlenie "LONGITUDINAL")
(dam-forma "SEMI-ELLIPTIC")
(caption-meh "CORROSION FATIGUE")
(id-dam ?id)
)
(exist-event ;exist-event
(caption "FORMATION OF THE THROUGH CRACKS")
)
=>
(assert
(exist-des ;exist-des
(caption-des "MACROCRACK")
(des-istochnik "HOLEHOLE")
(des-orientacia "PERPENDICULARLY")
(des-napravlenie "LONGITUDINAL")
(des-kolichestvo "SINGLE")
(des-koncentracia-naprazheniy "LESS THAN 2")
(caption-meh "CORROSION FATIGUE")
(caption-dam "CRACK")
(id-dam ?id)
(id-des ?id)
))
(assert
(exist-event ;exist-event
(caption "FORMATION OF THE MACROCRACKS OF THE UNACCEPTABLE SIZE") ;может возникнуть
/событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule des-mechanism-ky-1003 "Описание правила: des-mechanism-ky-1003"
(exist-event ;exist-event
(caption "BRITTLE FAILURE")
)
(exist-des ;exist-des
(caption-des "BRITTLE FAILURE")
(des-istochnik "CONSTRUCTIVE STRESS CONCENTRATOR FORMED BY THE INTERSECTION OF HOLES")
(des-orientacia "PERPENDICULARLY")
(des-napravlenie "LONGITUDINAL")
(des-forma "SEMI-ELLIPTIC")
(caption-meh "CORROSION FATIGUE")
(id-dam ?id)
(id-des ?id2)
)
=>
(assert
(exist-des ;exist-des
(caption-des "MACROCRACK")
(des-istochnik "HOLE")
(des-orientacia "PERPENDICULARLY")
(des-napravlenie "LONGITUDINAL")
(des-kolichestvo "SINGLE")
(des-koncentracia-naprazheniy "LESS THAN 2")
(caption-meh "CORROSION FATIGUE")
(caption-dam "CRACK")
(id-dam ?id)
(id-des ?id2)
))
(assert
(exist-event ;exist-event
(caption "FORMATION OF THE MACROCRACKS OF THE UNACCEPTABLE SIZE") ;может возникнуть
/событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule des-kinatic-events-ky-1004 "Описание правила: des-kinatic-events-ky-1004"
(exist-dam ;exist-dam
(caption-dam "CRACK")
(dam-istochnik "SURFACE DAMAGE")
```

```
(dam-orientacia "PERPENDICULARLY")
(dam-napravlenie "LONGITUDINAL")
(dam-kolichestvo "SINGLE")
(caption-meh "CORROSION FATIGUE")
)
(exist-event ;exist-event
(caption "FORMATION OF THE MACROCRACKS OF THE UNACCEPTABLE SIZE")
(caption-kin "KINETICS OF DESTRUCTION 'CORROSION FATIGUE 1'")
)
=>
(assert
(exist-event ;exist-event
(caption "MACROCRACKS DEVELOPMENT OF STRUCTURAL DAMAGE IN ZONES OF STRESS
CONCENTRATORS 'FROM THE HOLE IN THE SAME PLANE IN TWO DIRECTIONS'") ;может возникнуть
/событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule des-kinatic-events-ky-1005 "Описание правила: des-kinatic-events-ky-1005"
(exist-dam ;exist-dam
(caption-dam "CRACK")
(dam-istochnik "HOLE")
(dam-orientacia "PERPENDICULARLY")
(dam-napravlenie "LONGITUDINAL")
(dam-kolichestvo "SINGLE")
(caption-meh "CORROSION FATIGUE")
)
(exist-event ;exist-event
(caption "FORMATION OF THE MACROCRACKS OF THE UNACCEPTABLE SIZE")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
)
=>
(assert
(exist-event ;exist-event
(caption "MACROCRACKS DEVELOPMENT OF STRUCTURAL DAMAGE IN ZONES OF STRESS
CONCENTRATORS 'FROM THE HOLE IN THE SAME PLANE IN TWO DIRECTIONS'") ;может возникнуть
/событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule des-kinatic-events-ky-1006 "Описание правила: des-kinatic-events-ky-1006"
(exist-dam ;exist-dam
(caption-dam "CRACK")
(dam-istochnik "HOLE")
(dam-orientacia "PERPENDICULARLY")
(dam-napravlenie "LONGITUDINAL")
(dam-kolichestvo "SINGLE")
(caption-meh "CORROSION FATIGUE")
)
(exist-event ;exist-event
(caption "FORMATION OF THE MACROCRACKS OF THE UNACCEPTABLE SIZE")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
)
=>
(assert
(exist-event ;exist-event
(caption "MACROCRACKS DEVELOPMENT OF STRUCTURAL DAMAGE IN ZONES OF STRESS
CONCENTRATORS 'FROM THE HOLE IN THE SAME PLANE IN TWO DIRECTIONS'") ;может возникнуть
/событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule dam-mechanism-ky-1001 "правило выявления /механизма/"
(mechanical-stress-const ;механические нагрузки
(cycle-frequency "HIGH") ;-----------
)
(technological-environment ;технологическая среда
```

```
(ph "ACTIVE") ;водородный показатель
(properties-alternation "YES") ;чередование свойств среды /да/нет/
)
(incident-object ;объект инцидента
(caption ?id-inc-obj) ;код
)
(material ;материал
(type "STEEL") ;вид
(chemical-prop-alloying "LOW-ALLOY STEEL") ;легированность
)
=>
(assert
(exist-event ;exist-event
(caption "MECHANISM 'CORROSION FATIGUE'") ;может возникнуть /механизм/
))
)
(defrule dam-mechanism-ky-1002 "правило выявления /механизма/"
(mechanical-stress-const ;механические нагрузки
(cycle-frequency "HIGH") ;-----------
)
(technological-environment ;технологическая среда
(ph "ACTIVE") ;водородный показатель
(properties-alternation "YES") ;чередование свойств среды /да/нет/
)
(incident-object ;объект инцидента
(caption ?id-inc-obj) ;код
)
(material ;материал
(type "STEEL") ;вид
(chemical-prop-alloying "LOW-ALLOY STEEL") ;легированность
)
(exist-event ;exist-event
(caption "MACROCRACKS DEVELOPMENT OF STRUCTURAL DAMAGE IN ZONES OF STRESS
CONCENTRATORS 'FROM THE HOLE IN THE SAME PLANE IN TWO DIRECTIONS'")
)
=>
(assert
(exist-event ;exist-event
(caption "MECHANISM 'CORROSION FATIGUE'") ;может возникнуть /механизм/
))
)
(defrule dam-kinatics-ky-1001 "правило выявления /кинетики/"
(exist-event ;exist-event
(caption "MECHANISM 'CORROSION FATIGUE'") ;наблюдается /механизм/
)
=>
(assert
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;может возникнуть /кинетика/
))
)
(defrule dam-kinatic-events-ky-1001 "правило возникновения повреждения: если
наблюдается кинетика коррозионная усталость то может возникнуть повреждение трещина,
количество - одиночные, продольные, ориентированные перпендикулярно"
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;наблюдается /кинетика/
)
=>
(assert
(exist-dam ;exist-dam
(caption-dam "CRACK")
(dam-orientacia "PERPENDICULARLY")
(dam-napravlenie "LONGITUDINAL")
(dam-kolichestvo "SINGLE")
(caption-meh "CORROSION FATIGUE")
(id-dam "КУ-1")
))
(assert
(exist-event ;exist-event
```

```
(caption "MACROCRACKS DEVELOPMENT OF STRUCTURAL DAMAGE IN ZONES OF STRESS
CONCENTRATORS 'FROM THE HOLE IN THE SAME PLANE IN TWO DIRECTIONS'") ;может возникнуть
/событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule dam-kinatic-events-ky-1002 "Описание правила: dam-kinatic-events-ky-1002"
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;наблюдается /кинетика/ /
)
(exist-dam ;exist-dam
(caption-dam "CRACK")
(dam-orientacia "PERPENDICULARLY")
(dam-napravlenie "LONGITUDINAL")
(dam-kolichestvo "SINGLE")
(caption-meh "CORROSION FATIGUE")
)
(exist-event ;exist-event
(caption "FORMATION OF THE MACROCRACKS OF THE UNACCEPTABLE SIZE")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
)
=>
(assert
(exist-event ;exist-event
(caption "MERGING MICROCRACKS") ;может возникнуть /событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule dam-kinatic-events-ky-1003 "Описание правила: dam-kinatic-events-ky-1003"
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;наблюдается /кинетика/ /
)
(exist-event ;exist-event
(caption "MERGING MICROCRACKS")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
)
=>
(assert
(exist-event ;exist-event
(caption "PREFERENTIAL DEVELOPMENT OF MICROCRACKS ORIENTED IN PLANES PERPENDICULAR TO
THE MAXIMUM TENSILE STRESS") ;может возникнуть /событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule dam-kinatic-events-ky-1004 "Описание правила: dam-kinatic-events-ky-1004"
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;наблюдается /кинетика/ /
)
(exist-event ;exist-event
(caption "PREFERENTIAL DEVELOPMENT OF MICROCRACKS ORIENTED IN PLANES PERPENDICULAR TO
THE MAXIMUM TENSILE STRESS")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
)
=>
(assert
(exist-event ;exist-event
(caption "FORMATION OF THE MICROCRACKS") ;может возникнуть /событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule dam-kinatic-events-ky-1005 "Описание правила: dam-kinatic-events-ky-1005"
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;наблюдается /кинетика/ /
)
(exist-event ;exist-event
(caption "FORMATION OF THE MICROCRACKS")
```

```
(caption-kin "KINETICS 'CORROSION FATIGUE'")
)
=>
(assert
(exist-event ;exist-event
(caption "DEVELOPMENT OF INTRAGRANULAR SUBMICROCRACKS DUE TO THE DISSOLUTION IN THE
SLIP PLANES") ;может возникнуть /событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule dam-kinatic-events-ky-1006 "Описание правила: dam-kinatic-events-ky-1006"
(exist-kin ;exist-kin
(caption "KINETICS 'CORROSION FATIGUE'") ;наблюдается /кинетика/ /
)
(exist-event ;exist-event
(caption "DEVELOPMENT OF INTRAGRANULAR SUBMICROCRACKS DUE TO THE DISSOLUTION IN THE
SLIP PLANES")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
)
=>
(assert
(exist-event ;exist-event
(caption "FORMATION OF LOCAL DEFORMATION ZONES IN THE SURFACE DUE TO CYCLIC LOADING")
;может возникнуть /событие/
(probabilityrel "1")
(caption-kin "KINETICS 'CORROSION FATIGUE'")
))
)
(defrule R021 "Description of the rule: Generalized rule 1 1"
(mechanical-stress-const ;mechanical-stress-const
(stress-const-type "HIGH")
)
(technological-environment ;technological-environment
(ph "ACID")
(properties-alternation "YES")
)
(material ;material
(type "STEEL")
(chemical-prop-alloying "LOW-ALLOY STEEL")
(structure-prop-class "1")
)
(making-defects ;making-defects
(caption-technological-heredity "DEFECT OF WELDING")
)
=>
(assert
(exist-event ;exist-event
(caption "MECHANISM 'CORROSION FATIGUE'")
))
(assert
(exist-meh ;exist-meh
(caption-meh "CORROSION FATIGUE")
))
)
```

6.1.5 Использование сгенерированных баз знаний и спецификаций в составе ИАС «Экспертиза ПБ»

Результатом применения метода являются программные коды CLIPS и спецификации PKBD. При этом коды используются в процессе логического вывода и поиска решений, спецификации – программой-интпретатором (специальным программным модулем) при синтезе пользовательского интерфейса для создания,

чтения, обновления и удаления (CRUD) элементов продукционной БЗ и взаимодействия программных компонентов.

На Рис. 6.1.21 представлена архитектура ИАС «Экспертиза ПБ» [228, 234, 309], включающая следующие основные модули:

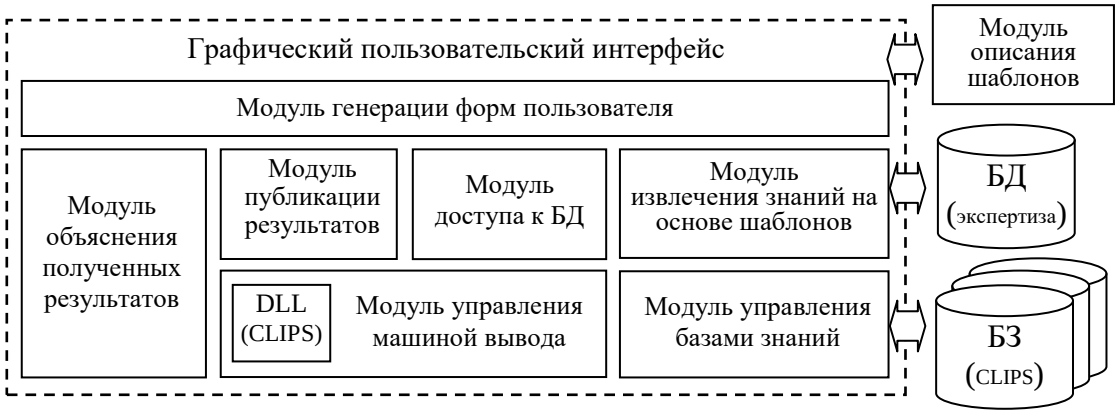


Рис. 6.1.21. Архитектура ИАС «Экспертиза ПБ»

- база знаний, включающая описание основных шаблонов и правил;

- машина логического вывода, обеспечивающая поиск решения на основе информации из БЗ и БД, реализована в виде динамической библиотеки;

- модуль взаимодействия с БД, обеспечивающий извлечение информации из БД для описания фактов;

- модуль генерации форм пользователя для ввода данных о текущем состоянии объекта экспертизы. Формы динамически создаются на основе спецификаций PKBD. содержащих формализованные описания образцов (шаблонов) основных понятий (фактов), необходимых для срабатывания правил;

- модуль формализованного описания шаблонов;

- модуль публикации результатов, обеспечивающий отображение результатов поиска решения (прогнозирования) в табличной форме в виде цепочек: Деградационный процесс – Дефект – Повреждение – Разрушение. При этом для элементов цепочки возможен просмотр их описания (свойства и значения);

- модуль объяснения результатов, обеспечивающий объяснение результатов поиска, путем публикации описаний активизированных правил из БЗ;

- модуль извлечения знаний, формирующий базы знаний на основе созданных экспертом шаблонов и данных (фактов), описанных в БД, а также позволяющий определять коэффициенты уверенности, поддерживая эксперта

статистическими данными, полученными в результате анализа БД;

- модель управления базами знаний позволяет сохранять созданные базы и подключать соответствующую в зависимости от решаемой пользователем задачи.

На Рис. 6.1.22 приведен пример экранной формы ИАС «Экспертиза ПБ» [228] с результатами логического вывода и динамической формой для ввода данных.

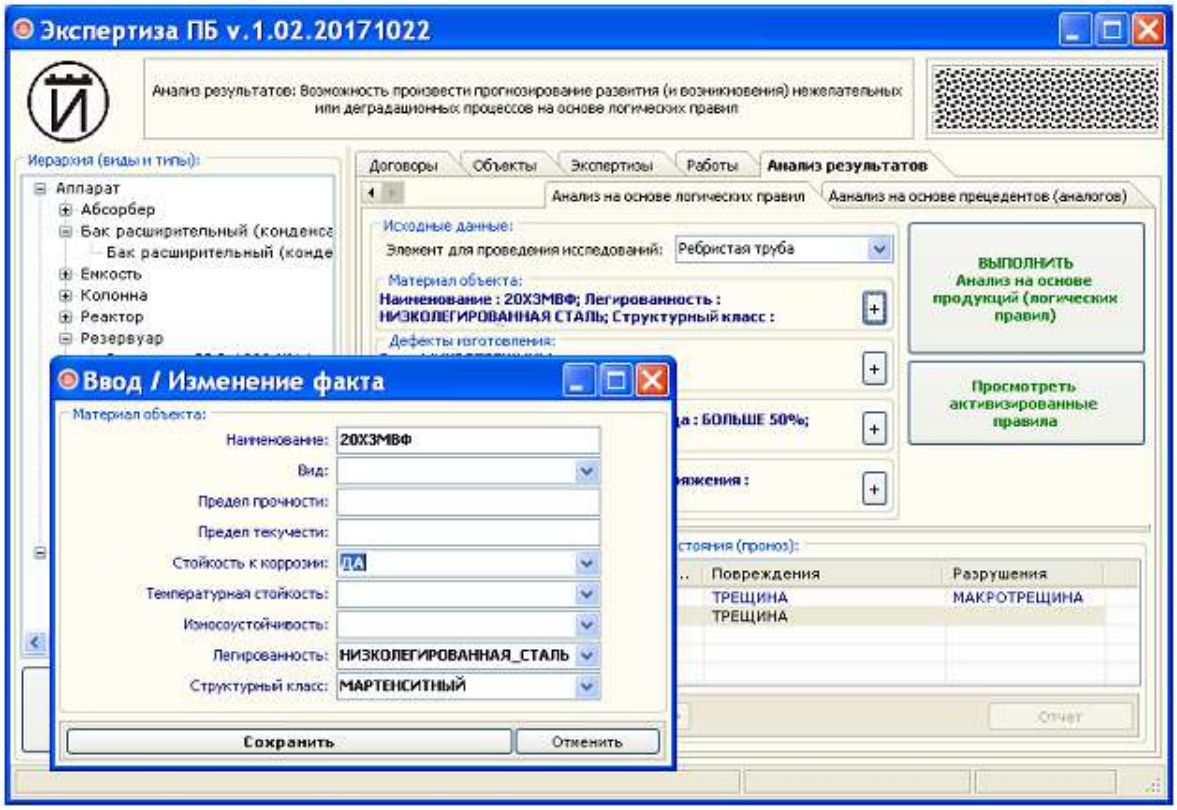


Рис. 6.1.22. Пример динамически создаваемой на основе спецификаций формы в ИАС «Экспертиза ПБ»

Разработанные БЗ используются при обосновании разделов «Программы проведения ЭПБ», при планировании видов и объемов диагностирования конкретных объектов, а также при определении причин и скорости деградационных процессов, необходимых для обоснования остаточного ресурса.

ИАС «Экспертиза ПБ» проходит опытную эксплуатацию в Иркутском научно-исследовательском и конструкторском институте химического и нефтяного машиностроения (ИркутскНИИхиммаш). Основной эффект от ее применения достигается как при проведении исследований для выявления закономерности изменения технического состояния рассматриваемых объектов, так и при организации и проведении экспертизы промышленной безопасности.

Работы по разработке ИАС «Экспертиза ПБ» проводились в рамках договора 052013НИР от 19.09.2013 с ОАО «ИркутскНИИхиммаш».

6.1.6 Использование сгенерированных баз знаний и спецификаций в проблемно-ориентированном редакторе для диагностических задач нефтехимии

Одной из задач поддержки ИАС «Экспертиза ПБ» являлась обеспечение возможности наполнения созданных баз знаний кончными пользователям, которые в общем случае не являются специалистами в области инженерии знаний или программирования. При этом модельные трансформации использовались при создании CFM-спецификаций, которые применялись не только в рамках ИАС «Экспертиза ПБ» для генерации пользовательского интерфейса, но и для проблемно-ориентированного редактора (рег.№ 2012614093) [186, 230], создаваемого на основе авторского инструментария PKBD [184, 195, 202, 269].

Основная задача использования подобного редактора – предоставить ограниченную среду разработки с предварительно формализованными структурами предметной области. Данные структуры описывали:

- основные предметные понятия в форме шаблонов фактов, основные: «событие», «материал», «воздействующие факторы», «нежелательный процесс»
- основные предметные отношения между понятиями в форме шаблонов правил, основные: связь между нежелательными состояниями.

Уточним некоторые из использованных понятий. Динамика процесса представлялась как последовательность классов состояний или стадий некоторого процесса (Рис. 6.1.23), где каждый класс/стадия описывается механизмом и кинетикой. Механизм – это совокупность воздействующих факторов и свойств объекта, определяющих направление развития процесса. Объект является носителем/обладателем процесса. Кинетика – совокупность последовательностей состояний/событий процесса.

Основным понятием в области оценки технического состояния и остаточного ресурса является понятие «нежелательный процесс» [326], как совокупность объективных физико-химических процессов, обусловленных как протеканием различных технологических (рабочих) процессов, так и

несовершенствами и нарушениями конструктивного, производственного и эксплуатационного происхождения. Структура данного понятия представлена на Рис. 6.1.24.

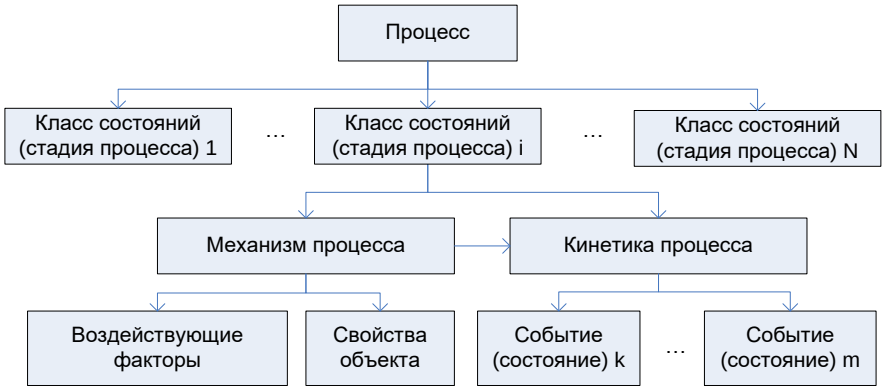


Рис. 6.1.23. Описание динамики процесса

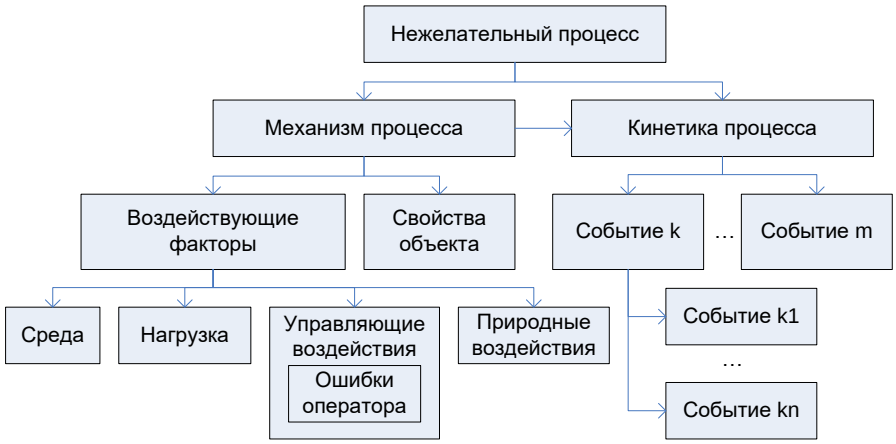


Рис. 6.1.24. Структура понятия «Нежелательный процесс» (ДП)

Динамика нежелательного процесса может быть представлена в виде причинно-следственной цепочки классов состояний: исходная дефектность, поврежденность, разрушение, отказ. На основе данной причинно-следственной цепочки классов состояний определена последовательность этапов описания деградационных процессов (Рис. 6.1.25).

Для основных предметных понятий и отношений были сгенерированы CFM спецификации. Приведем пример описания нежелательного (деградационного) процесса «коррозионное растрескивание», шаблон правила выявления механизма повреждения и соответствующий интерфейс (Рис. 6.1.26):

ЕСЛИ Материал **И** Остаточные напряжения **И** Механические нагрузки – статические **И** Теплообменная технологическая среда
ТО Механизм повреждения – Коррозионное растрескивание.

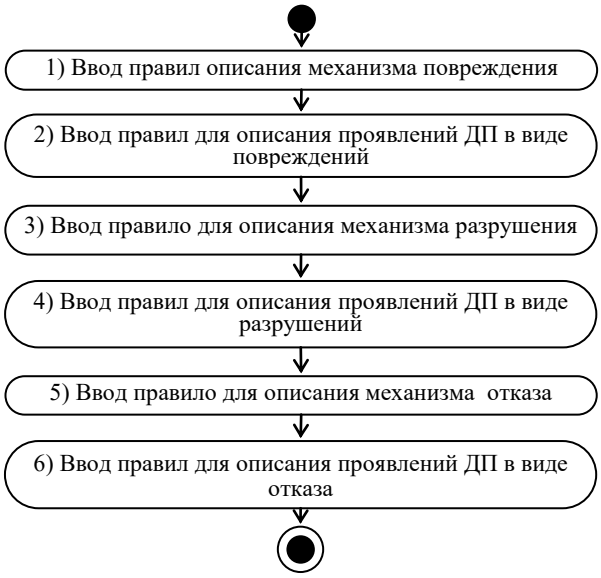


Рис. 6.1.25. Пример этапов описания ДП в проблемно-ориентированном редакторе

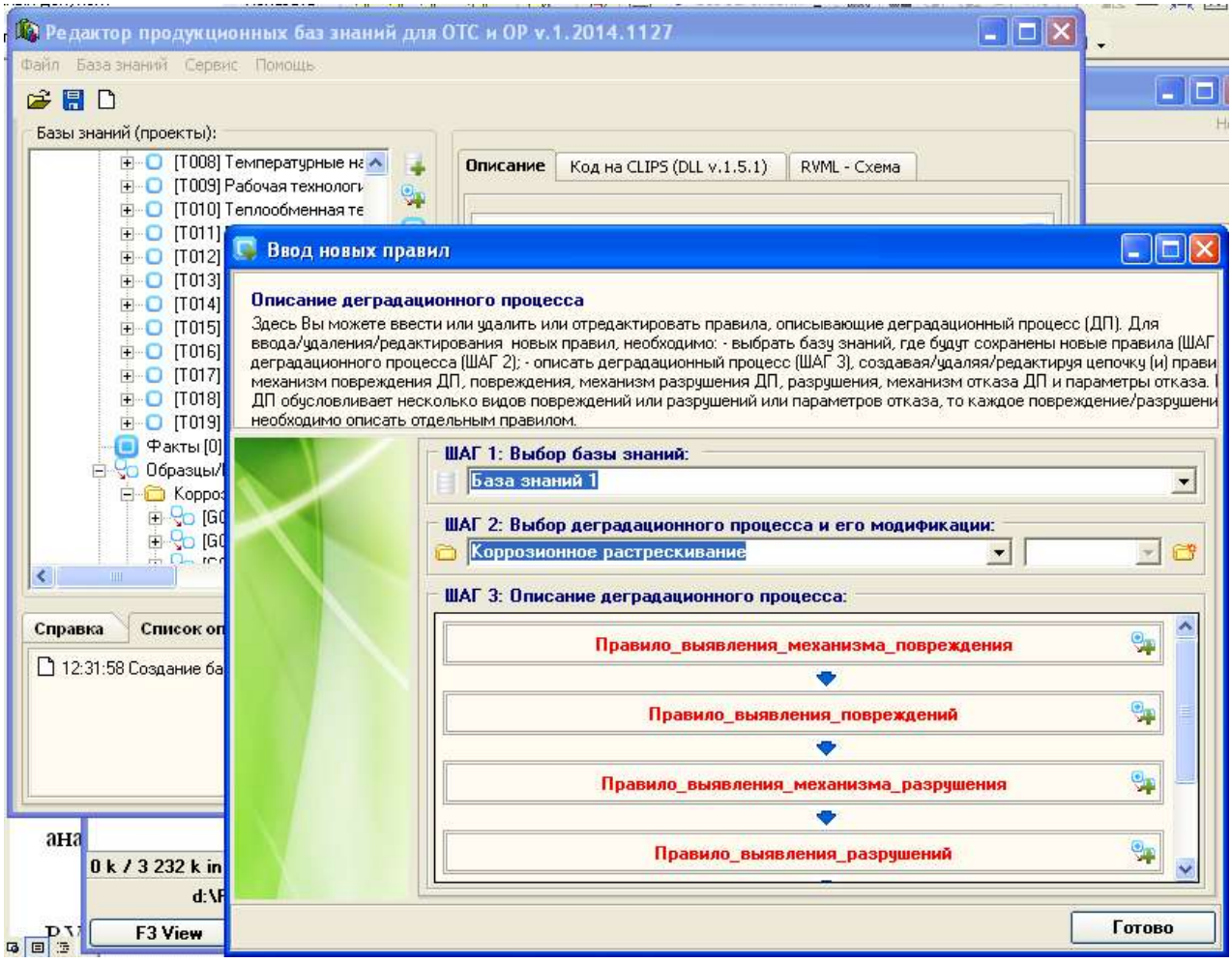


Рис. 6.1.26. Пример интерфейса проблемно-ориентированного редактора с предварительно определенным набором шаблонов правил, описывающих динамику технического состояния

Спецификация шаблона правила:

```
[Generalized rules]
#Коррозионное растрескивание
##1
dam-mechanism-dam-
ky=Правило_выявления_механизма_повреждения:material,structural-
heredity,making-defects,mechanical-stress-const,thermal-stress,technological-
environment,heat-exchange-technological-environment,flow-technological-
environment,surface-damage-from-corrosive-environment:exist-meh-dam
dam-damage-ky=Правило_выявления_повреждений:exist-meh-dam:exist-dam
des-mechanism-des-ky=Правило_выявления_механизма_разрушения:exist-meh-
dam,exist-dam:exist-meh-des
des-destruction-ky=Правило_выявления_разрушний:exist-meh-des:exist-des
```

Более полный пример спецификаций приведен в приложении В.

Помимо общей схемы исследования, конфигурационные правила определяют также последовательность описания (ввода) отдельных компонентов правила: условий и действий. Для их описания также динамически создаются элементы интерфейса пользователя.

6.2 Прототипы продукционной и прецедентной интеллектуальных систем в области конструкционных материалов

В разделе рассмотрены примеры применения предлагаемых методов и средств для прототипирования ИС с использованием PKBD. В зависимости от типа создаваемых ИС существуют некоторые особенности выполнения отдельных этапов, однако, в целом их последовательность сохраняется. По этой причине процесс создания продукционной ИС для выявления причин повреждений и разрушения элементов технических систем [16, 184, 185, 257] описан детально, в то время как для прецедентной ИС для подбора конструкционных материалов на этапе проектирования [18, 229] он рассмотрен схематично.

6.2.1 Продукционная интеллектуальная система определения причин повреждения и разрушения элементов технических систем

Проблема выявления причин повреждений и разрушения конструкционных материалов в нефтехимии является критической, поскольку связана с проблемой обеспечения надежности и безопасности опасных техногенных объектов. В большинстве случаев объектом исследования являются сложные технические системы и их компоненты, которые считаются уникальными механическими системами в силу следующих факторов: небольшое количество экземпляров (копий), воздействие экстремальных давлений и температур в процессе

эксплуатации, риск для людей и окружающей среды. Определение причин повреждения и разрушения этих систем требует анализа и обработки информации, отражающей опыт работы экспертов, отчеты об отказах, математических моделей, онтологий и т.д.

В данном разделе рассмотрена разработка прототипа БЗ и ИС для решения данной проблемы с использованием рассмотренного подхода [16, 188].

Для решения задач первого этапа связанных с анализом предметной области используется модель динамики технических состояний [217], которая описывает основные понятия в области развития деградационных процессов в нефтехимии. Эта модель включает описание и взаимосвязи следующих основных понятий:

- механические напряжения (свойства: величина, цикл, амплитуда и др.);
- материалы (свойства: имя, тип, температура сопротивления и др.);
- контактная среда (свойства: форма, давление, температура, ионы хлора, растворенный кислород и др.);
- процесс деградации (свойства: название механизма, кинетика и др.).

В дополнение к этой модели, была использована информация из базы данных об отказах нефтехимического оборудования [237], а также результаты экспертизы промышленной безопасности [15, 228, 234, 309]. В результате была получена модель предметной области, которая в дальнейшем рассматривалась как вычислительно-независимая модель БЗ.

IHMC CmapTools использовался как программный инструментарий для построения моделей данного этапа.

Как упоминалось выше, разработка концептуальной модели продукционной ЭС предполагала использование уже готового шаблона, который определяет основные архитектурные элементы и включает соответствующие понятия.

Построение платформо-независимых моделей осуществлялось с использованием PKBD, который обеспечил импорт вычислительно-независимой модели из формата IHMC CmapTools (Concept Mapping Extensible Language, CXL-формат) (Рис. 6.2.1) и ее анализ. Фактически осуществляется трансформация $T_{CIM-to-PIM} : CIM \rightarrow PIM$. В результате этой трансформации понятий и отношений

концептуальной модели были сопоставлены с шаблонами для фактов и правил (Рис. 6.2.2) и причинно-следственных связей для дальнейшего изменения пользователем, с последующим созданием конкретных правил (Рис. 6.2.3).

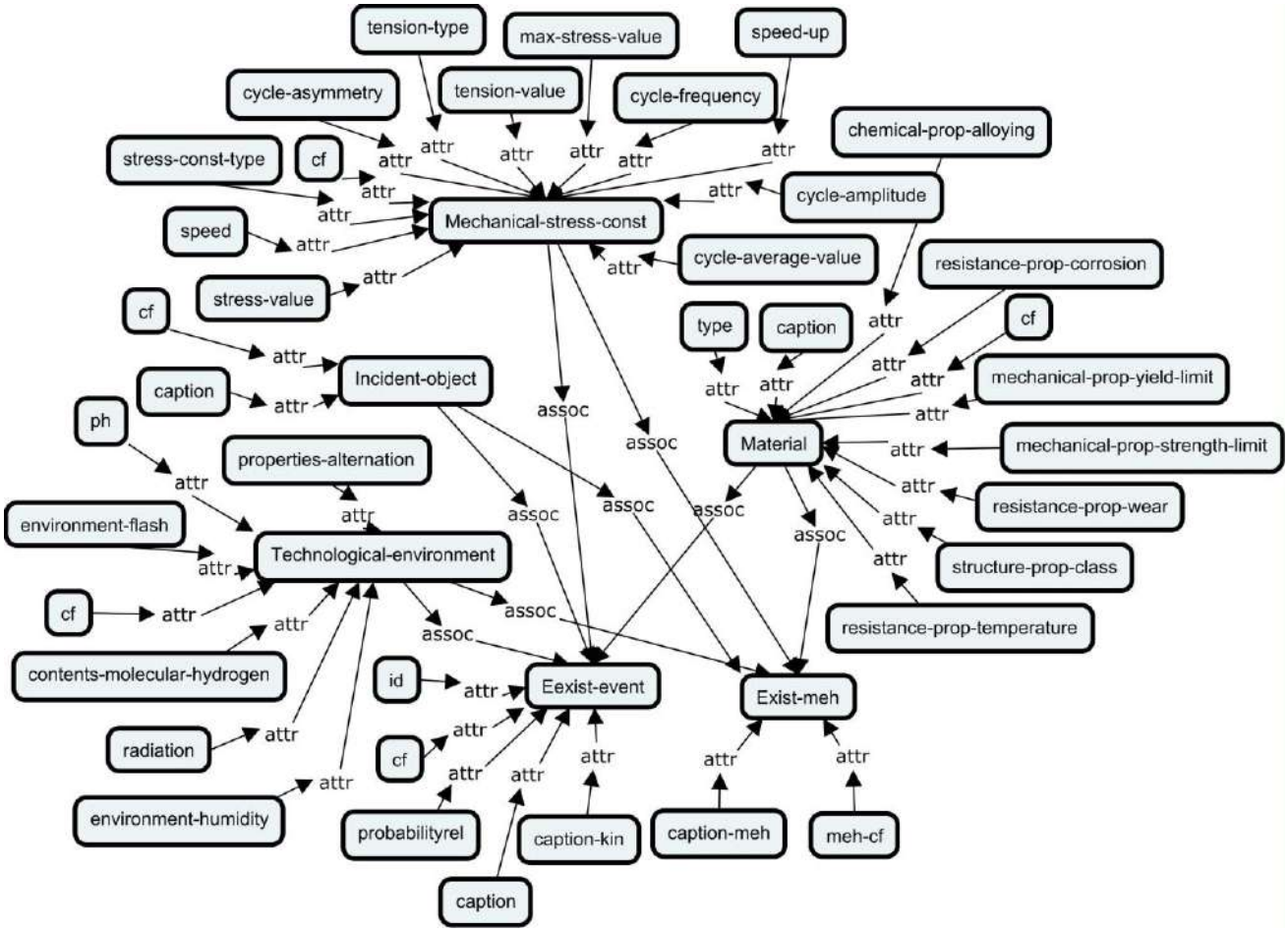


Рис. 6.2.1. Фрагмент вычислительно-независимой модели в виде концепт карты (IHMC CmapTools)

Следует отметить, что, как правило, в концепт-картах, в том числе в формате CXL, отсутствует четкое разделение понятий на понятия (классы) и их свойства (атрибуты), поэтому обеспечить автоматическое обнаружение таких сущностей сложно. Одним из решений этой проблемы является использование дополнительных ограничений для именования сущностей в концептуальных картах, а именно: при именовании понятий классов должна использоваться начальная заглавная буква, а названия понятий-свойств должны начинаться со строчных букв.

В дополнение к платформо-независимой модели БЗ формируется платформо-независимая модель ЭС. Платформо-независимая модель ЭС включает описание основных форм графического интерфейса, и эта модель создается на

основе анализа цепочки правил.

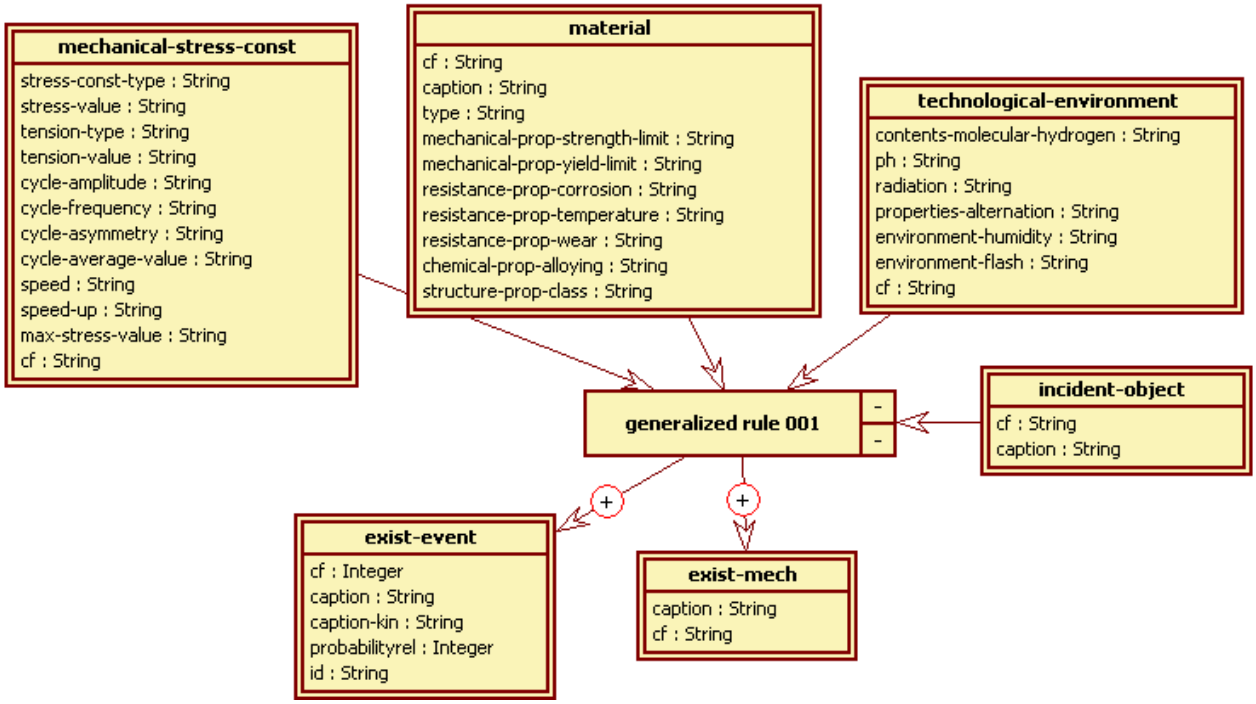


Рис. 6.2.2. Пример шаблона правила в форме RVML

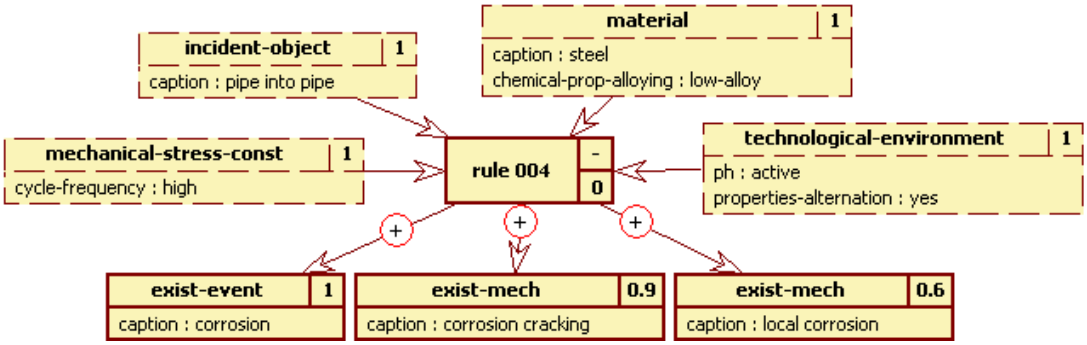


Рис. 6.2.3. Пример конкретизированного правила в форме RVML

В частности, получены следующие правила (при формировании платформо-независимой модели БЗ):

1. **ЕСЛИ** вода под давлением 3-4 МПа и температуре около 300С^0, ионы хлора и растворенный кислород (КУ=1,0) **ТО** контактная среда активная ($К_у$=0,9).
2. **ЕСЛИ** деформация низколегированной стали в холодном состоянии величиной до 5% **ТО** материал чувствительный (КУ=0,9) **И** остаточные напряжения.
3. **ЕСЛИ** $\sigma_{\text{раст}} \le 0{,}7\sigma_{0,2}$ (КУ=1,0) **ТО** значительные растягивающие напряжения ($К_у$=1,0).
4. **ЕСЛИ** значительные растягивающие напряжения, в том числе остаточные ($К_у$=0,9) **И** чувствительный материал ($К_у$=0,9) **И** активная контактная среда ($К_у$=0,9) **ТО** механизм «локальная коррозия» ($К_у$=0,8) **или** механизм «коррозионное растрескивание» ($К_у$=0,9).
5. **ЕСЛИ** механизм – «коррозионное растрескивание» **ТО** поглощение в субмикрообъемах материала энергии остаточных напряжений и

внешнего механического воздействия **И** повышение энергии кристаллической решетки до критического уровня **И** деформация кристаллической решетки **И** разрыв межатомных связей.

6. **Если** механизм – «коррозионное растрескивание» **И** кинетика «коррозионного растрескивания» **ТО** протекание локальной электрохимической коррозии, где дефекты поверхности, образовавшиеся на предыдущей стадии, являются локальными анодными участками **И** слияние субмикротрещин.
7. **Если** кинетика «коррозионного растрескивания» на микроуровне **ТО** питтинги (местоположение – на поверхности;) **И** язвы (местоположение – на поверхности;) **И** трещины (источник – питтинги;).

Эти правила формируют цепочку логического вывода, не требующую дополнительной информации (т.е. эта цепочка использует содержимое рабочей памяти и исходные данные, введенные пользователем). Платформо-независимая модель для архитектуры ЭС, соответствующая этой цепочке, включает следующие программные компоненты:

- формы ввода фактов, обеспечивающие ввод информации о: технологической среде, напряжениях, исследуемом объекте и материале;
- машину логического вывода (этот элемент является частью ЭС и скрыт от пользователя);
- выходную форму, которая показывает измененные, добавленные или удаленные факты;
- выходную форму, которая показывает активированные правила.

Количество платформо-зависимых моделей определяется количеством платформ, для которых создается ЭС. Поскольку целевыми платформами в нашем случае являются CLIPS (для описания БЗ) и PKBD (для интерпретации модели ЭС), то требуется лишь небольшая модификация платформо-независимой модели БЗ в форме RVML. В частности, необходимо определить приоритеты правил и значения слотов «по умолчанию». Таким образом, трансформация $T_{PIM\text{-}to\text{-}PSM}$: $PIM \rightarrow PSM$ практически незаметна для пользователя. Некоторые примеры экранных форм PKBD приведены в главе 4. Уже на данном этапе можно осуществить «прогон» разработанной базы знаний в PKBD с помощью встроенной машины вывода.

Генерация кодов и спецификаций (преобразование $T_{PSM\text{-}to\text{-}Code}$: $PSM \rightarrow Code$) осуществляется автоматически с помощью PKBD. Основные результаты этого этапа:

- коды CLIPS;

- спецификации ЭС для интерпретатора PKBD, которые обеспечивают генерацию пользовательского интерфейса для создания, чтения, обновления и удаления (CRUD) элементов БЗ и взаимодействия программных компонентов.

В дальнейшем тестирование БЗ и ЭС осуществляется экспертами с помощью логического вывода в уже синтезированной интеллектуальной системе (см. глава 4). При этом можно вернуться к одному из предыдущих этапов в соответствии с результатами тестирования.

6.2.2 Прецедентная интеллектуальная система подбора конструкционного материала

Для решения широкого круга инженерных задач, связанных с многокритериальной оптимизацией параметров, многовариантным планированием и последующим выбором, применяются различные методы многокритериального выбора (оптимизации) [245, 261, 264, 337, 346]. В результате проведенного обзора было отмечено, что при решении конструкторских задач в области машиностроения, например, при проектировании нового изделия (детали), требуется решить задачу выбора конструкционного материала. Данная задача является многовариантной, так как для изготовления какой-либо детали можно выбрать некоторое множество материалов, удовлетворяющих условиям ее эксплуатации, изготовления, и обслуживания в составе конструкции или машины. Как правило, при этом необходимо учитывать целый ряд условий: назначение детали; условия эксплуатации, технологию изготовления и ремонта, стоимость и доступность и др., которые неоднозначно оцениваются специалистами, принимающими решения.

В настоящее время задача выбора конструкционных материалов решается либо «по аналогии» [318], либо по описанию эксплуатационных и технологических свойств материала [210], либо методом экспертного оценивания [318] с дальнейшим применением методов многокритериальной оптимизации. При этом отсутствует программное обеспечение, автоматизирующее решение данной задачи.

В данном разделе описан процесс прототипирования прецедентной ИС для выбора конструкционного материала [18, 229] с использованием предлагаемых методов и средств. При этом общие принципы подхода в контексте создания

прецедентных интеллектуальных систем рассмотрены также в [44, 191, 273, 280, 322, 333], а в [370] рассмотрено его применение в учебном процессе.

Для решения задач первого этапа, связанного с разработкой модели предметной области, используется информация о конструкционных материалах [210], которая позволяет сформировать образ или прецедент, т.е. выделить свойства, составляющие описательную часть проблемы и ее решение (см. раздел 1.1). В данном случае в описательную часть проблемы входят основные свойства, описывающие материал, а в часть решение – его наименование (Табл. 6.2.1). Пример описания прецедента – материал 20Х3МВФ приведен в Таблице 6.2.2.

Необходимо отметить, что на данном этапе развития подхода модель прецедентов не содержит логических отношений между понятиями, хотя в рамках классического фреймового представления слот фрейма, в свою очередь, может являться вложенным фреймом. В данной задаче подобная ситуация не рассматривается.

Таблица 6.2.1. Описание структуры прецедента для задачи выбора конструкционного материала

Часть прецедента	**Имя свойства**	Описание
Описание проблемы	Structural component	конструктивный элемент
	High-temperature strength	жаропрочность
	Operation temperature	температура применения
	Service life	срок работы
	Corrosion resistance	коррозионная стойкость
	Crack resistance	трещиностойкость
	Manufacturability	технологичность
	Cost	стоимость
	Pressure (expl.)	давление (эксплуатационное)
	Temperature (expl.)	температура (эксплуатационная)
	Agent (expl.)	среда (эксплуатационная)
	Heat resistance	теплостойкость
	High-temperature resistance	температурная стойкость
Решение проблемы	Name	Наименование стали

В результате выполнения этапа была получена модель предметной области (Рис. 6.2.4), содержащая единственное понятие, которая в дальнейшем рассматривалась как вычислительно-независимая модель БЗ. Для построения моделей данного типа использовалось CASE-средство StarUML.

Таблица 6.2.2. Пример описания прецедента: материал 20Х3МВ

жаропрочность:	Да (обладает этим свойством)
температура применения:	до 500-560°С
ресурс:	от 1000 до 10000 ч. при максимальной

		температуре. При температуре до 350^0C ресурс по жаропрочности неограничен.
коррозионная стойкость:		незначительная
трещиностойкость:		Да (обладает этим свойством)
технологичность:		низкая
стоимость:		высокая
конструктивный элемент:		Теплообменная труба аппарата типа «труба в трубе»
условия эксплуатации:	давление:	160 МПа
	температура:	300^0C
	среда:	слабоагрессивная

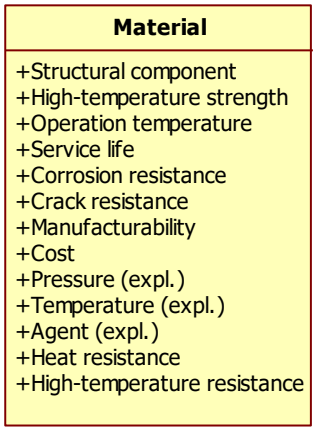


Рис. 6.2.4. Вычислительно-независимая модель в виде класса UML (StarUML)

Построение платформо-независимых моделей осуществлялось с использованием PKBD (преобразование $T_{CIM\text{-}to\text{-}PIM}$:$CIM \rightarrow PIM$), который обеспечил импорт вычислительно-независимой модели из формата StarUML и ее анализ. Результатом этапа является модель, содержащая описание шаблона факта (Рис. 6.2.5).

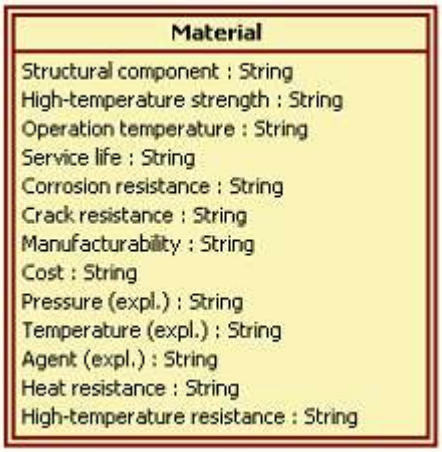


Рис. 6.2.5. Шаблон факта в форме элемента RVML

Других моделей на данном этапе строить не потребовалось, поскольку используемый инструментарий обеспечивает поддержку создания определенного

класса программных систем и содержит эти модели «по умолчанию».

Целевой платформой в нашем случае являлась PKBD, поэтому никаких модификаций платформо-независимой модели БЗ в форме RVML (для реализации преобразования $T_{PIM\text{-}to\text{-}PSM}$:*PIM* → *PSM*) не потребовалось.

Генерация кодов и спецификаций (преобразование $T_{PSM\text{-}to\text{-}Code}$:*PSM* → *Code*) осуществляется автоматически с помощью PKBD. Основной результат этого этапа – спецификации ЭС для интерпретатора PKBD, которые обеспечивают генерацию пользовательского интерфейса для создания, чтения, обновления и удаления (CRUD) элементов БЗ (Рис. 6.2.6) и взаимодействия программных компонентов.

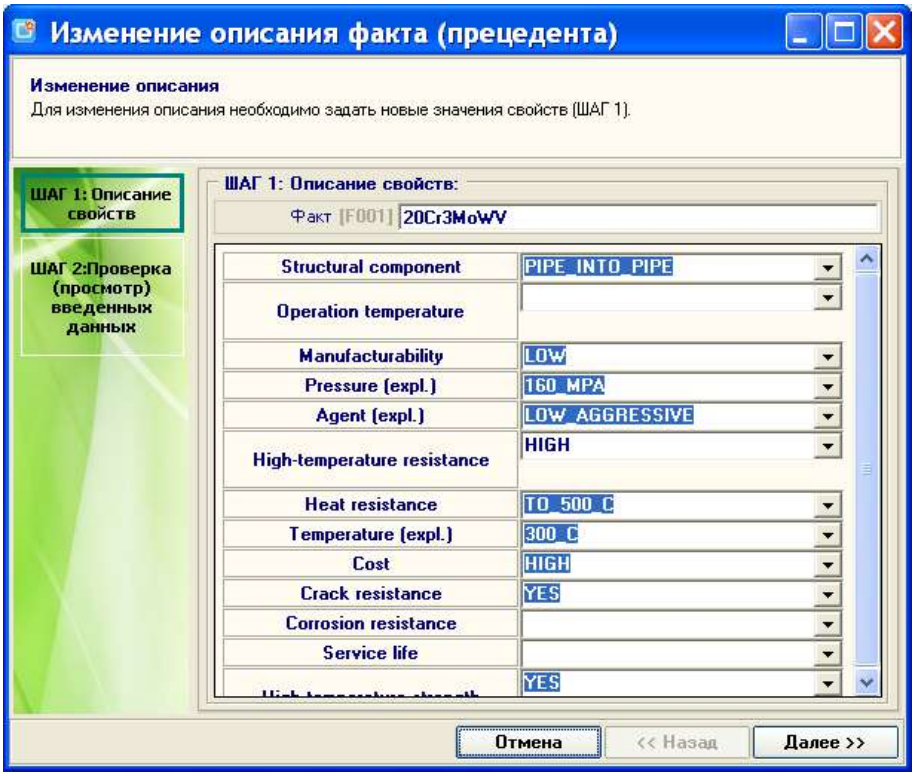


Рис. 6.2.6. Примеры пользовательского интерфейса PKBD: добавление прецедента

На данном этапе происходит наполнение БЗ информацией о прецедентах, используя сгенерированный интерфейс.

Тестирование ИС и ЭС осуществлялось экспертами с помощью вывода на основе прецедентов. В частности, для этого необходимо было описать целевую (текущую) проблемную ситуацию, например, условия которым должен удовлетворять материал с указанием относительной важности каждого условия. Затем инициировать процесс проверки и зафиксировать результат (Рис. 6.2.7).

При этом можно было вернуться к одному из предыдущих этапов в соответствии с результатами тестирования.

\

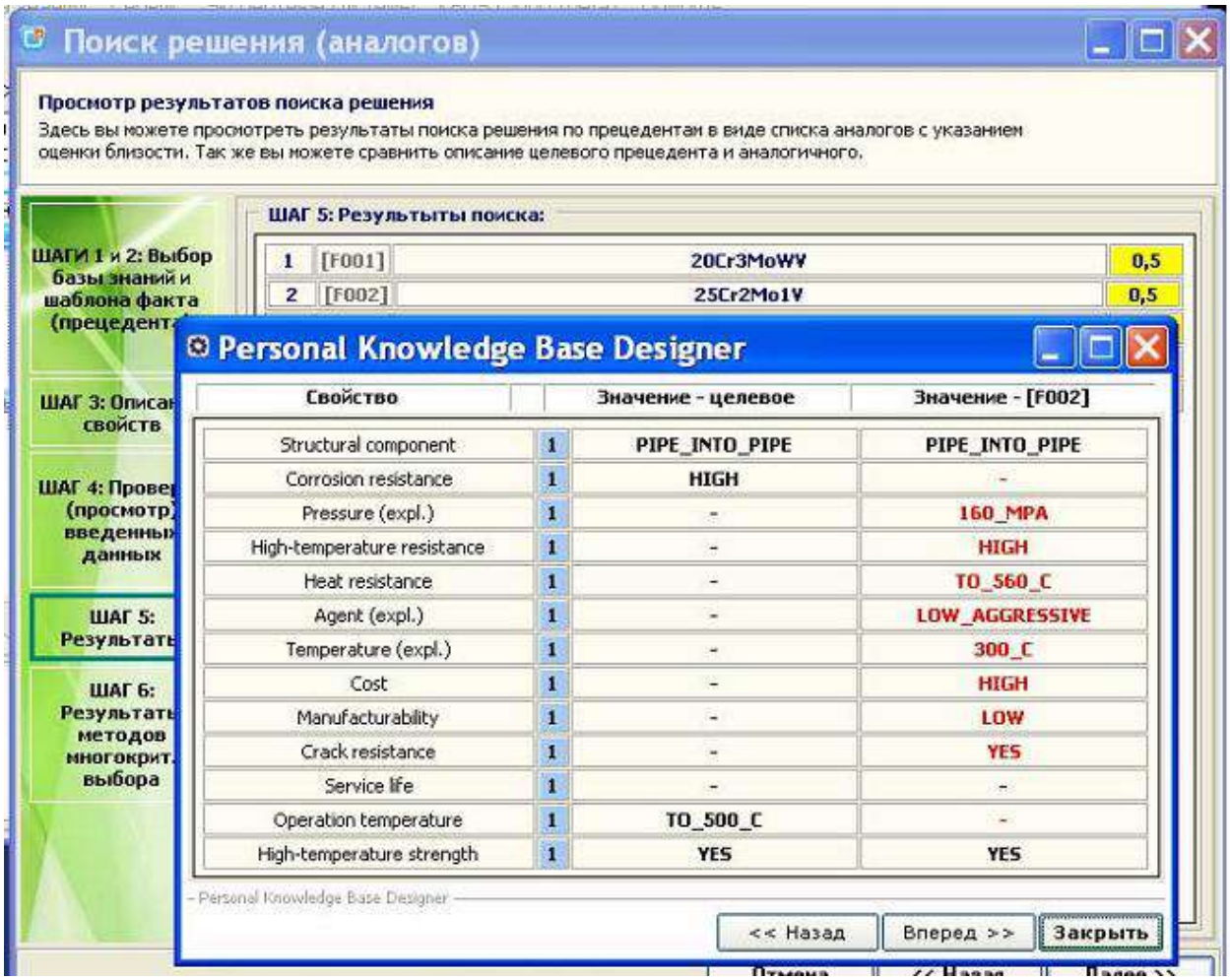


Рис. 6.2.7. Результаты поиска решения и наглядное сравнение с аналогом

Данная ИС была апробирована при выборе материала труб, предназначенных для работы в качестве теплообменных под высоким внутренним давлением, при температуре 300^0C, под воздействием на наружную поверхность слабо агрессивной среды, при требовании значительного ресурса и трещиностойкости в составе реактора типа «труба в трубе». База прецедентов включала описание материалов труб в соответствие со справочниками [210, 318].

6.3 Интеллектуальный планировщик анализа отказов ИС «INFOT-3»

Одним из первых примеров использования модельных трансформаций для создания спецификаций и программного обеспечения в рамках предлагаемого подхода является решение задачи динамического формирования планов проведения исследований и обеспечения свойств адаптивности интеллектуальной системы (ИС) «INFOT-3» [218]. Отработка принципов подхода осуществлялась на примере решения задачи анализа отказов механических систем [12, 13, 223]. Планы

формировались с использованием продукционного подхода специальным программным модулем – планировщиком. Последовательность этапов процесса создания базы знаний планировщика совпадает с основными этапам метода и методики (см.4.1) (Рис.6.3.1).

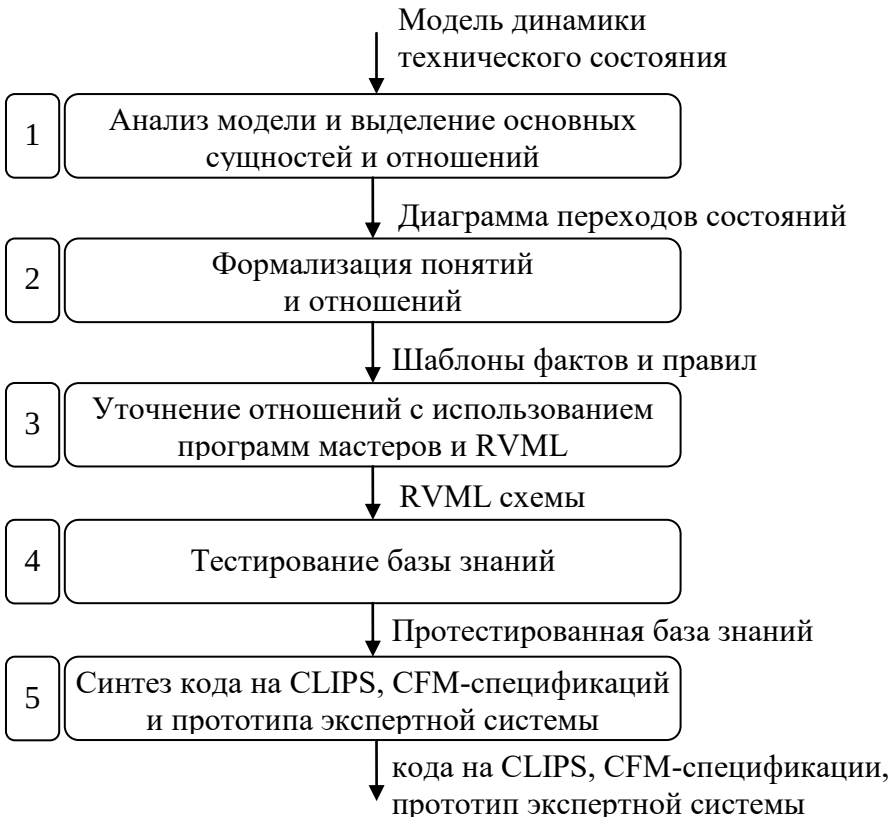


Рис. 6.3.1. Схема разработки базы знаний для планировщика ИС «INFOT-3»

В качестве вычислительно-независимой модели использовалась модель динамики технического состояния аппарата или конструкции [217, 324], представленная в теоретико-множественной форме. Далее, на основе модели динамики осуществлялось построение диаграммы (графа) переходов состояний, описывающей возможные последовательности этапов алгоритма. В свою очередь диаграмма переходов преобразовывалась в продукционную модель, для представления которой использовались RVML диаграммы, отображаемые в программные коды на CLIPS.

6.3.1 Модель динамики технического состояния

Для обеспечения адаптивности ИС «INFOT-3» информация о формировании отказа была декомпозирована согласно структуре объекта и классификации динамики состояний. Введены информационные уровни, отражающие классы

динамики состояний [221, 326]. Такая декомпозиция информации, с одной стороны обеспечивает решение задач некоторой совокупностью научных отраслей и дисциплин, с другой – позволяет любому исследователю ознакомиться с решениями, принятыми специалистами смежных дисциплин. В дальнейшем, для упрощения изложения, рассматривается структурный аспект объекта и классификация состояний на одном информационном уровне.

Представляем причинно-следственную цепочку технических состояний на одном из информационных уровней:

$$S_I \to S_A \to S_L \to S_F,$$

где S_I – исходное состояние объекта; S_A – допустимое состояние объекта; S_L – предельное состояние элемента; S_F – состояние отказа.

Модель процесса изменения технического состояния УМС отражает структурную иерархию исследуемого объекта: деталь (*D*); сборочная единица (*AU*); специфицированное изделие (*SP*); механическая система (*MS*). Отношение «часть-целое» между элементами структуры обусловливает причинно-следственные отношения их технических состояний: деталь является составной частью сборочной единицы и одновременно деталь является причиной возникновения того или иного состояния сборочной единицы и т.д. Исходя из этого, можно говорить об иерархическом пространстве технических состояний, описывающем исходное, допустимое, предельное и состояние отказа структурных элементов. Таким образом, иерархическая структура рассматриваемого объекта может быть представлена в виде матрицы состояний, где, например, пространство состояний отказа: S_F^D – детали, S_F^{AU} – сборочной единицы, S_F^{SP} – специфицированного изделия, S_F^{MS} – уникальной механической системы. При этом в границах пространства состояния структурных элементов являются также взаимообусловленными:

$$\begin{array}{cccc} S_I^{MS} \rightarrow S_A^{MS} \rightarrow S_L^{MS} \rightarrow S_F^{MS} \\ \uparrow \quad\quad \uparrow \quad\quad \uparrow \quad\quad \uparrow \\ S_I^{SP} \rightarrow S_A^{SP} \rightarrow S_L^{SP} \rightarrow S_F^{SP} \\ \uparrow \quad\quad \uparrow \quad\quad \uparrow \quad\quad \uparrow \\ S_I^{AU} \rightarrow S_A^{AU} \rightarrow S_L^{AU} \rightarrow S_F^{AU} \\ \uparrow \quad\quad \uparrow \quad\quad \uparrow \quad\quad \uparrow \\ S_I^{D} \rightarrow S_A^{D} \rightarrow S_L^{D} \rightarrow S_F^{D} \end{array} \qquad (6.3.1)$$

6.3.2 Алгоритм анализа отказов

Согласно матрице состояний (6.3.1), анализ отказа состоит в последовательном отнесении состояния объекта к одному из классов состояний, определении параметров состояния объекта и причин их изменения на основе данных о внешних признаках рассматриваемого состояния объекта и информации о подобных состояниях, содержащейся в базах знаний, представленных прецедентами и продукционными правилами.

Алгоритм анализа отказов базируется на рассмотренной модели 6.3.1. Например, этапы анализа для S_i^j – состояния представлены следующим образом:

$$\begin{array}{cccc} AS_I^j \leftrightarrow \ldots \leftrightarrow AS_{i-1}^j \leftrightarrow AS_i^j \\ \updownarrow \quad\quad \updownarrow \quad\quad \updownarrow \quad\quad \updownarrow \\ AS_I^{j-1} \leftrightarrow \ldots \leftrightarrow AS_A^{j-1} \leftrightarrow AS_i^{j-1} \\ \updownarrow \quad\quad \updownarrow \quad\quad \updownarrow \quad\quad \updownarrow \\ \ldots \quad\quad \ldots \quad\quad \ldots \quad\quad \ldots \\ \updownarrow \quad\quad \updownarrow \quad\quad \updownarrow \quad\quad \updownarrow \\ AS_I^D \leftrightarrow \ldots \leftrightarrow AS_A^D \leftrightarrow AS_L^D \end{array} \qquad (6.3.2)$$

где AS_i^j – этап; → и ← – направления выполнения последовательности этапов.

Анализ может осуществляться в прямом направлении (→): от текущего состояния к последующему, в том числе к конечному состоянию – отказу уникальной механической системы. Такая последовательность обеспечивает прогнозирование состояний. Анализ может осуществляться в обратном направлении (←): от текущего состояния к предыдущему, в том числе к исходному состоянию. Такая последовательность обеспечивает определение причин изменения состояний, то есть происхождение (генезис) состояний, в том числе отказов.

6.3.3 Структура ИС «INFOT-3»

Для реализации свойства адаптивности ИС «INFOT-3» [218] в ее состав был введен модуль управления (планировщик). В этом случае ее структура может быть представлена следующим образом:

$$FAS = \left\langle ES^{CBR}, ES^{RBR}, SS, CS \right\rangle,$$

где FAS – интеллектуальная система анализа отказов; ES^{CBR} – прецедентная экспертная система; ES^{RBR} – продукционная экспертная система; SS – хранилище этапов анализа; CS – модуль управления анализом отказов.

При этом $ES^{RBR} = \left\langle KB^{RBR}, IM^{RBR}, GUI^{RBR} \right\rangle$; $ES^{CBR} = \left\langle KB^{CBR}, IM^{CBR}, GUI^{CBR} \right\rangle$, где KB^{RBR}, KB^{CBR} – базы знаний экспертных систем; IM^{RBR}, IM^{CBR} – машины вывода экспертных систем; GUI^{RBR}, GUI^{CBR} – интерфейсы пользователя экспертных систем, $GUI^{RBR} = \left\{GUI\_Form_{rf}\right\}_{rf}$, $GUI^{CBR} = \left\{GUI\_Form_{cf}\right\}_{cf}$, GUI_Form_f – f-форма пользовательского интерфейса.

Хранилище этапов анализа (SS) – это совокупность программных модулей, реализующих отдельные этапы алгоритма анализа отказа, где:

$$SS = \left\{Stage_s\right\}_s,\ Stage_s = \left\langle Name_s^{Stage}, M_s^{Stage}, GUI_s^{Stage} \right\rangle,$$

где $Stage_s$ – s-этап алгоритма анализа отказов, $Name_s^{Stage}$ – имя s-этапа, $M_s^{Stage} = \left\langle M_{Ref}, AS_i^j \right\rangle$ – параметры модуля, используемого на s-этапе анализа для обработки информации, $M_{Ref} = \left\{M_{Ref}^{CBR} \mid M_{Ref}^{RBR}\right\}$ – ссылки на прецедентный или продукционный модули, AS_i^j – описание этапа анализа j-го структурного уровня и i-состояния, $GUI_s^{Stage} = \left\{GUI^{CBR} \mid GUI^{RBR}\right\}$ – пользовательский интерфейс модуля на s-этапе анализа, обеспечивающий ввод информации и отображение результатов обработки введенной информации.

Модуль управления анализом отказов (CS) выполняет следующие основные функции:

1) Формирование плана анализа отказов с учетом: модели динамики изменения состояний; решаемой задачи (генезис, прогнозирование); имеющейся у пользователя информации и промежуточных результатов анализа отказов. План

отражает последовательность использования (вызовов) программных модулей, обеспечивающих выполнение отдельных этапов анализа отказа.

2) Интерпретация плана, состоящая из синтеза: входных данных, интерфейса пользователя, методов обработки информации и запуска на выполнение результата синтеза. В результате интерпретации осуществляется динамическое формирование интерфейса системы анализа отказов из отдельных программных модулей в соответствии с планом.

Для реализации функций предлагается следующая структура модуля:

$$CS = \left\langle ES_{Plan}^{RBR}, Inerpr \right\rangle,$$

где ES_{Plan}^{RBR} – продукционная экспертная система планирования анализа отказов (планировщик); $Inerpr$ – интерпретатор результатов планировщика (плана).

$$ES_{Plan}^{RBR} = \left\langle KB_{Plan}^{RBR}, IM^{RBR} \right\rangle,$$

где KB_{Plan}^{RBR} – база знаний планировщика, отражающая модель динамики состояний (6.3.1) и последовательность этапов анализа (6.3.2), IM^{RBR}– машина вывода.

Основная задача планировщика – на основе данных s-этапа анализа отказа сформировать s+1-этап: $ES_{Plan}^{RBR} : Stage_s \to Stage_{s+1}$.

База знаний планировщика KB_{Plan}^{RBR} содержит правила, которые отражают:

- последовательность изменения технического состояния;
- структурную иерархию УМС.

Правила, в качестве вывода (заключения), содержат вызов определенного программного модуля, реализующего определенный этап анализа отказа.

Приведем пример правила на естественном языке:

ЕСЛИ параметры допустимого состояния детали не введены
ИЛИ параметры допустимого состояния детали невозможно описать
ИЛИ аналоги по параметрам допустимого состояния детали с оценкой близости > 0.5 не обнаружены
ТО переход к этапу анализа параметров недопустимого состояния с помощью прецедентов

Машина вывода IM^{RBR} обеспечивает логический вывод, результат которого – этап плана анализа отказов.

Принципиальные схема функционирования и структура адаптивной интеллектуальной системы анализа отказов представлена на Рис. 6.3.2.

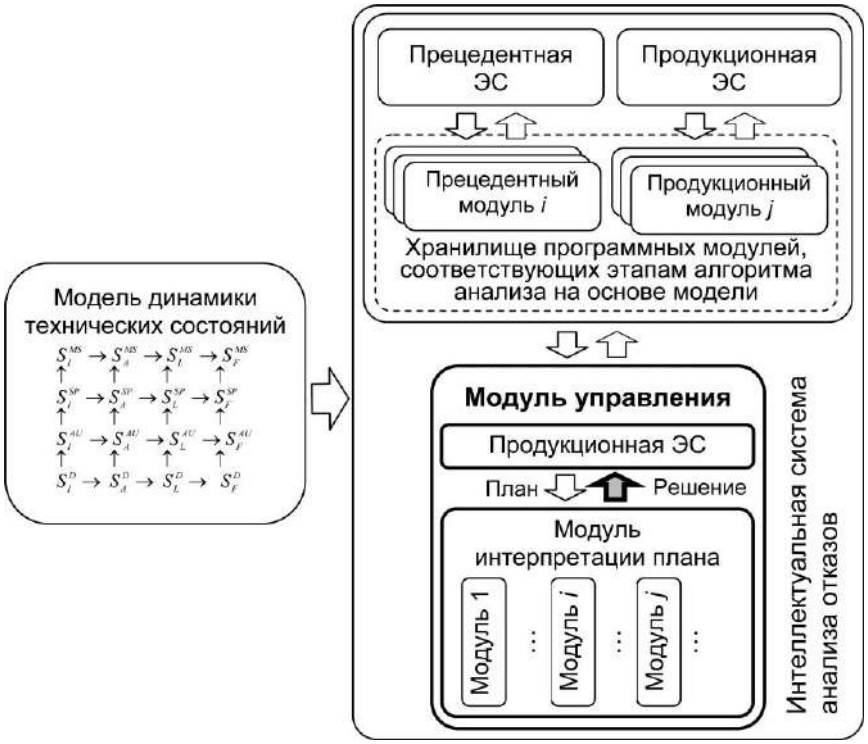


Рис. 6.3.2. Принципиальные схема функционирования и структура адаптивной интеллектуальной системы анализа отказов

План, сформированный планировщиком ES_{Plan}^{RBR}, интерпретируется интерпретатором $Inerpr$, т.е. формируется последовательность исполнения программных модулей и осуществляется их выполнение. При этом программные модули обладают унифицированным интерфейсом. Интерфейс обеспечивает ввод определенных данных (соотвествующих этапу), запуск процесса поиска решения и просмотр результатов.

6.3.4 Алгоритм планирования

Алгоритм планирования представлен на Рис. 6.3.3 где: «*Начало*» – шаг ввода начальной информации, включая выбор объекта исследования (указания его структурной принадлежности) и типа решаемой задачи. На данном шаге происходит автоматическое создание плана анализа отказа в виде начального этапа $Stage_0$, на котором выполняется анализ структурной принадлежности и ввод пользователем начальных данных.

Рис. 6.3.3. Обобщенный алгоритм анализа отказов

«*Шаг s*1»: ЭС (ES^{R}_{Plan}) делает вывод о следующем (новом) этапе плана.

«*Шаг s*2»: интерпретатор ($Inerpr$) интерпретирует описание нового этапа плана. Результат: последовательность исполнения программных модулей и динамически синтезированный пользовательский интерфейс.

«*Шаг s*3»: анализ технического состояния на основе результатов предыдущего этапа и водимых пользователем признаков анализируемого состояния с помощью ЭС (ES^{CBR} или ES^{RBR} – согласно плану). Результаты этапа используются для принятия решения о необходимости продолжения исследования. В случае продолжения – происходит переход на шаг s_1, который осуществляется на качественно новом уровне с использованием последних результатов анализа отказа и новой информации, введенной пользователем. При этом шаги s_1-s_3 могут повторяться (в цикле) вплоть до принятия решения о прекращении исследования.

«*Шаг D*»: формирование рекомендаций либо по исключению деградационных процессов или снижению их скорости, либо по снижению возможных последствий каждого состояния.

«*Шаг V*»: оценка пользователем достоверности полученного решения. При

этом система может выдать рекомендации о необходимости проведения дополнительных лабораторных исследований (испытаний) с целью получения дополнительной информации. Например, определить локальные механических свойств материала элемента механической системы или микроструктуру материала в зоне повреждения и т.д.

6.3.5 Моделирование базы знаний планировщика и трансформации

На основе модели динамики технического состояния, алгоритма анализа отказа и с учетом структуры программы и алгоритма планирования с помощью специального редактора EETE [287] была построена визуальная модель базы знаний в форме диаграммы переходов состояний (Рис.6.3.4).

Диаграмма переходов состояний была преобразована в продукционную модель (Рис.6.3.5), при этом использовался модуль расширения PKBD.STD, реализующий преобразования XML-подобного формата диаграмм. В упрощенном виде соответствия между форматами представлены в таблице 6.3.1. Фрагмент кода диаграммы переходов состояний:

```
<?xml version="1.0" encoding="UTF-8"?>
<Diagram id="3" name=" Д 5.2.3. База знаний планировщика анализа отказа"
description="Диаграмма для базы знаний планировщика анализа отказов">
  <State id="26" name="Анализ структурной принадлежности" type="Initial
state" description=""/>
  <State id="27" name="Анализ внешних проявлений отказа механической
системы" type="State" description=""/>
…
  <Transition id="28" name="ASB-&gt;AMSF" state-from="26" state-to="27"
description="">
    <TransitionProperty id="30" name="Свойства введены" operator="="
value="да" description=""/>
    <TransitionProperty id="32" name="Мин.Близость" operator="&gt;"
value="0.5" description=""/>
    <TransitionProperty id="40" name="Количество прецедентов"
operator="&gt;" value="0" description=""/>
  </Transition>
…
```

Таблица 6.3.1. Фрагмент таблицы отображения элементов модели (вычислительно-независимой модели в платформо-независимую и платформо-независимой в CLIPS)

Элементы диаграммы (вычислительно-независимая модель)	**Элементы продукционной БЗ (платформо-независимая модель)**	**Элементы CLIPS кода**
Diagram (id, name, description)	Knowledge base (name, description)	-
State (id, name, description)	Template (name, description)	deftemplate
StateProperty (id, name, operator, value, description)	Slot (name, value, description)	slot
Transition (id, name, state-from,	Rule (nodal element)	defrule

state-to, description)		
TransitionProperty (id, name, operator, value, description)	Slot (name, value, description)	slot

При преобразовании вычислительно-независимой модели свойства из описания перехода были добавлены в описание состояний (Рис.6.3.5), т.е. «TransitionProperty» интерпретировались как «StateProperty».

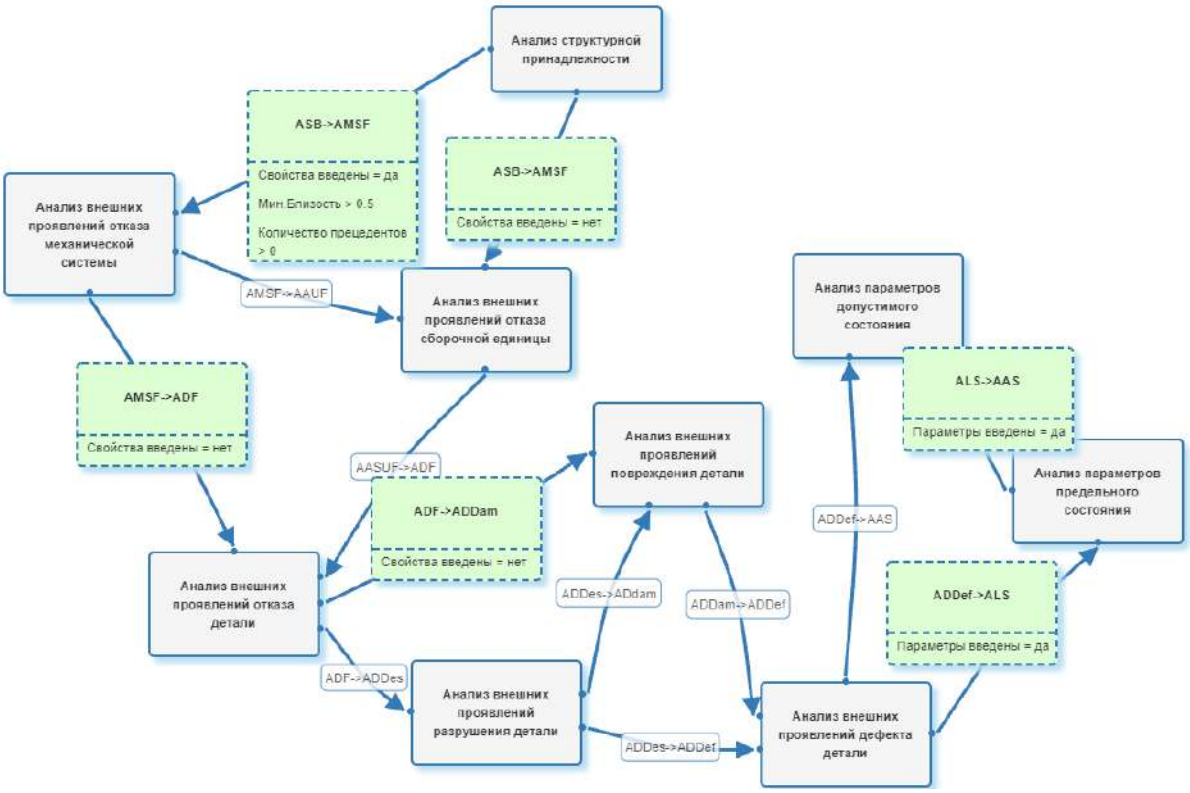


Рис. 6.3.4. Фрагмент диаграммы переходов состояний, описывающей алгоритм анализа отказов

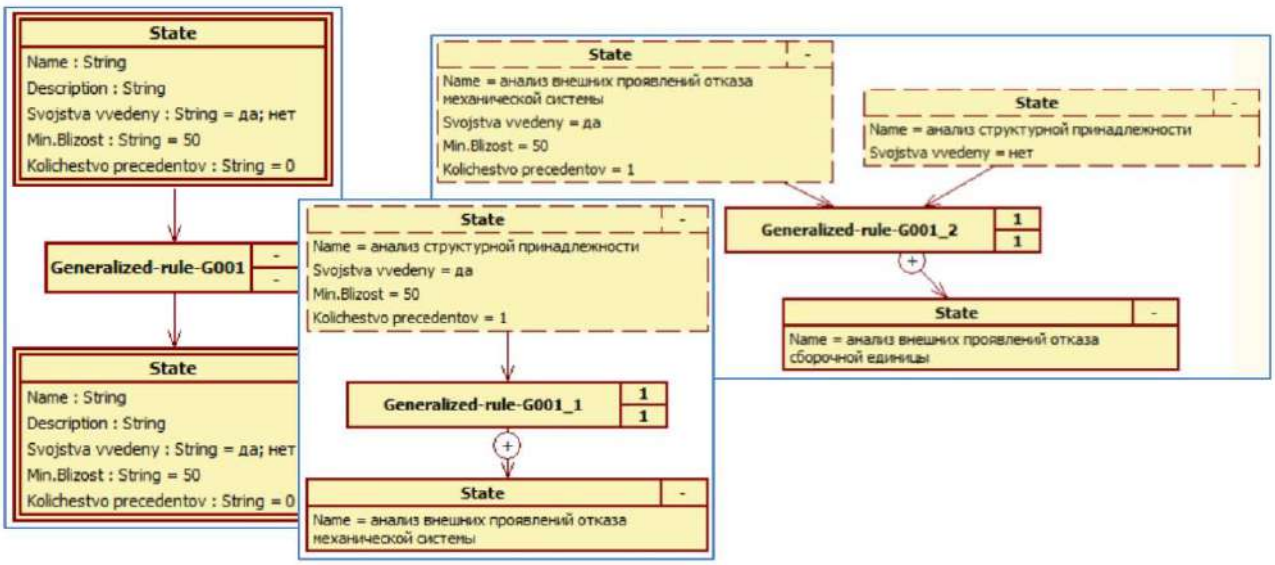


Рис. 6.3.5. Примеры полученных диаграмм RVML (платформо-независимых моделей): шаблона правила и пример двух конкретных правил

Для учета особенностей ИС «INFOT-3», в частности, возможности вызова определенного программного модуля для описания состояния, в RVML диаграммы была добавлена информация о наименовании соответствующего модуля в форме значения слота «GUI-form-name» для элементов «State». Дополненные диаграммы RVML были отображены в код на CLIPS с уточнением знаков операторов, определяющих ограничения на значения слотов, и добавлением дополнительных условий. Пример одного из полученных правил:

```
      (defrule Generalized-rule-G001-2 "Description of the rule: Generalized-
rule-G001 2"
      (declare (salience 1))
       (State ;State
        (Name "АНАЛИЗ ВНЕШНИХ ПРОЯВЛЕНИЙ ОТКАЗА МЕХАНИЧЕСКОЙ СИСТЕМЫ")
        (Svojstva-vvedeny "ДА")
        (Min.Blizost >50)
        (Kolichestvo-precedentov >0)
       )
       (State ;State
        (Name "АНАЛИЗ СТРУКТУРНОЙ ПРИНАДЛЕЖНОСТИ")
        (Svojstva-vvedeny "НЕТ")
       )
      (State ;State
        (Name "АНАЛИЗ ВНЕШНИХ ПРОЯВЛЕНИЙ ОТКАЗА МЕХАНИЧЕСКОЙ СИСТЕМЫ")
        (Svojstva-vvedeny "ДА")
        (Min.Blizost <50)
       )
      =>
      (assert
       (State ;State
        (Name "АНАЛИЗ ВНЕШНИХ ПРОЯВЛЕНИЙ ОТКАЗА СБОРОЧНОЙ ЕДИНИЦЫ")
        (GUI-form-name "AAUF")
       ))
      )
```

6.3.6 Формирование плана анализа отказа трубопровода подачи синтез-газа

Апробация подхода осуществлялась на примере анализ отказа трубопровода подачи синтез-газа в колонну синтеза аммиака при решении задачи генезиса. Рассмотрим процесс формирования плана.

При запуске ИС «INFOT-3» был сформирован план в виде начального этапа ($Stage_0$) для описания исследуемого объекта. Была введена информация о структурной принадлежности и воздействующих факторах.

Далее был осуществлен поиск прецедентов. Для формирования базы прецедентов использовалась информация из базы данных по отказам нефтехимического оборудования [237]. Результат этапа: извлечено 25 прецедентов по структурной принадлежности, 15 из них имеют оценку близости более 0,9. При

данных условиях активизируется правило, обеспечивающее переход на этап «Анализ внешних проявлений отказа механической системы», в план решения задачи добавляется этап $Stage_1$ и система формирует соответствующий интерфейс. В контексте решаемой задачи механической системой является трубопровод.

Пользователь выбирает одно из внешних проявлений отказа механической системы: «Выброс взрыво-пожароопасной среды». Осуществляется поиск прецедентов по внешним проявлениям отказа. Результат: 18 прецедентов, 12 из них имеют оценку близости более 0,9. При данных условиях активизируется правило, обеспечивающее переход на этап «Анализ внешних проявлений отказа сборочной единицы», в план решения задачи добавляется этап $Stage_2$ и система формирует соответствующий интерфейс.

При описании отказа сборочной единицы пользователь выбирает: «Потеря герметичности сварного соединения». Далее осуществляется поиск прецедентов по внешним проявлениям отказа сборочной единицы. Результат: 10 прецедентов, 8 из них имеют оценку близости более 0,8. Активизируется правило, обеспечивающее переход на этап «Анализ внешних проявлений отказа детали» ($Stage_3$).

Пользователь выбирает «Истечение среды в окружающее пространство». Далее осуществляется поиск прецедентов по внешним проявлениям отказа детали. Результат: 5 прецедентов, 4 из них имеют оценку близости более 0,8.

Подобным образом пользователь последовательно проходит этапы описания внешних проявлений разрушений, повреждений и дефектов ($Stage_4 - Stage_6$). Результат: сформирован следующий план решения задачи:

$Stage_0$: Анализ структурной принадлежности;

$Stage_1$: Анализ внешних проявлений отказа механической системы;

$Stage_2$: Анализ внешних проявлений отказа сборочной единицы;

$Stage_3$: Анализ внешних проявлений отказа детали;

$Stage_4$: Анализ внешних проявлений разрушения детали;

$Stage_5$: Анализ внешних проявлений повреждения детали;

$Stage_6$: Анализ внешних проявлений дефекта детали.

В результате работы системы, на основании информации в прецедентах, определяются причины отказа (механизм) и набор решений, направленных на ликвидацию последствий отказа и снижение вероятности (риска) подобных отказов в будущем.

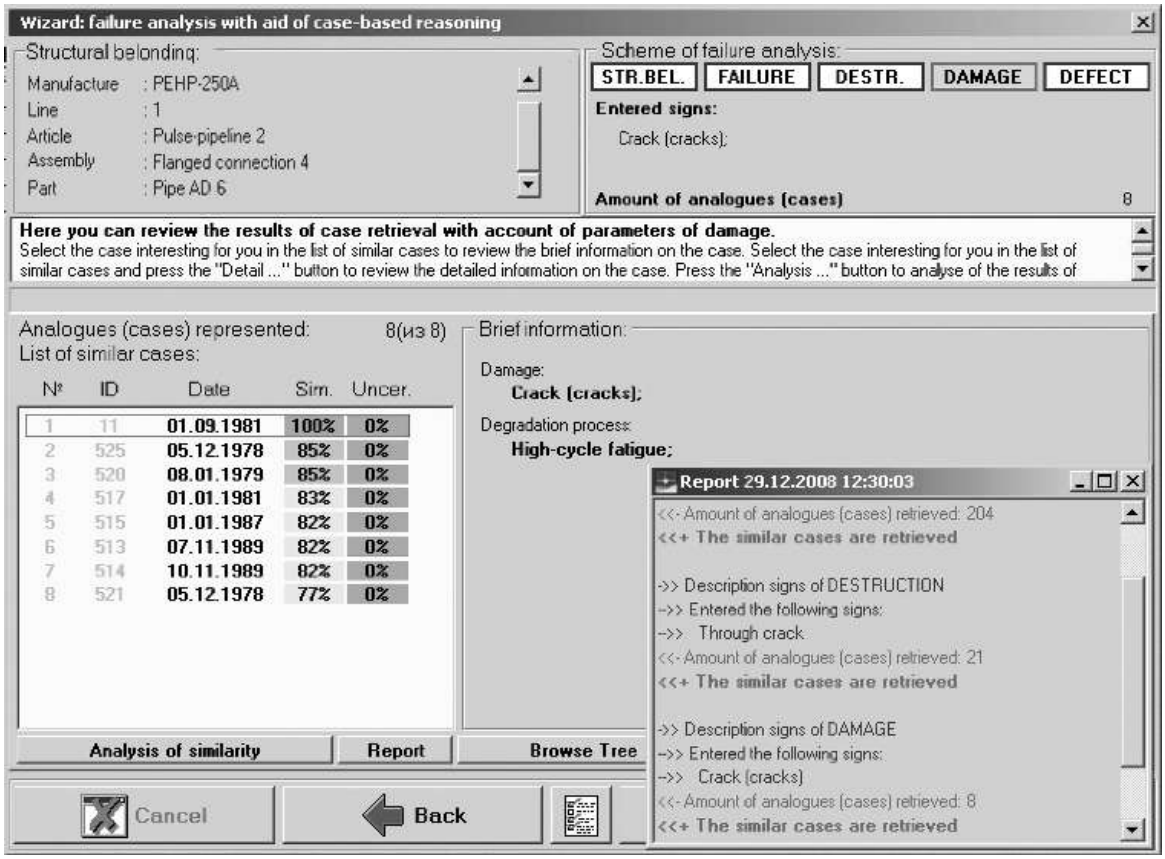


Рис. 6.3.6. Пример экранной формы адаптивной программной системы анализа отказов

В частности, рассматриваемый отказ трубопровода, обусловлен недопустимым дефектом – наличием трещины в сварном шве и недопустимым уровнем вибрации при пуске технологической линии после очередного ремонта. На Рис. 6.3.6 представлен пример интерфейса ИС «INFOT-3»: в правом верхнем углу – схема процесса исследования, динамически сформированная на основе введенных данных; в правом нижнем углу – отчет (лог) процесса анализа отказа.

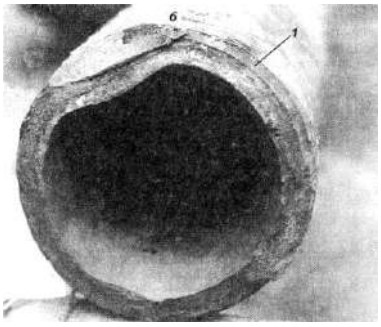


Рис. 6.3.7. Труба с недопустимым дефектом изготовления: 1 – трещина в сварном шве

На Рис. 6.3.7 представлена фотография отказавшей детали с недопустимым дефектом изготовления.

В рамках данной задачи использование модельных трансформаций позволило сформировать базу знаний планировщика и обеспечить не только динамическое формирование последовательности этапа анализа, но и адаптивные свойства программной системы.

6.4 E-INFOT: Программное средство создания интеллектуальных систем диагностики технических состояний конструкций

Оригинальное программное инструментальное средство «E-INFOT» [361] типа «оболочка» предназначено для создания проблемно-ориентированных интеллектуальных систем диагностики (идентификации) технических состояний деталей машин и конструкций [14, 116, 216, 219, 220, 226, 231, 232, 361]. Первая версия данного средства и реализуемый в нем метод были разработаны в рамках кандидатской диссертации соискателя.

Метод создания интеллектуальных систем диагностики (идентификации) технических состояний конструкций соответствует общим принципам модельно-ориентированной разработки и предлагаемому методу прототипирования БЗ и интеллектуальных систем. Основным результатом применения метода является создание определенных концептуальных моделей предметной области, обеспечивающих учет особенностей реализуемых методов и дальнейшую разработку инструментального программного средства типа «оболочка» – «E-INFOT» [361].

6.4.1 Реализуемые в E-INFOT подходы к решению задачи диагностики

Решение задачи диагностики (идентификации) технических состояний осуществляется в результате совместного применения прецедентных (Case-Based Reasoning) [1] и продукционных (Rule-Based Reasoning) [262, 263, 331, 356] экспертных систем.

Прецедентный подход. Процесс решения задачи при данном подходе представляет собой идентификацию (распознавание) проблемной ситуации по заданному (вводимому пользователем) набору идентификационных признаков, что

предполагает использование элементов теории распознавания образов. Распознавание [211, 238, 251, 298, 350] представляет собой задачу преобразования входной информации, представленной в виде признаков распознаваемого объекта (образа), в выходную информацию в виде заключения о принадлежности описываемого объекта (образа) к определенному классу.

Под образом понимают структурированное приближенное (частичное) описание изучаемого объекта или явления, причем частичная определенность описания является принципиальным свойством образа [251, 298]. При автоматизированном решении задачи диагностики (идентификации) технического состояния понятию «образ» соответствует понятие «прецедент».

Успешное решение задачи распознавания (поиска и извлечения) требует разработки соответствующей системы распознавания и решения следующих основных задач [251]: разработки словаря признаков; разработки алфавита классов (включая описание классов на языке словаря признаков); выбора алгоритма распознавания; формирования решающих правил.

Под словарем признаков понимается набор признаков, описывающих распознаваемые образы. В нашем случае признаки характеризуют динамику технического состояния и иерархическую структуру объекта (машины или конструкции) [236], что позволяет их разделить на группы: признаки, описывающие структурную принадлежность; признаки, описывающие техническое состояние объекта (Рис. 6.4.1).

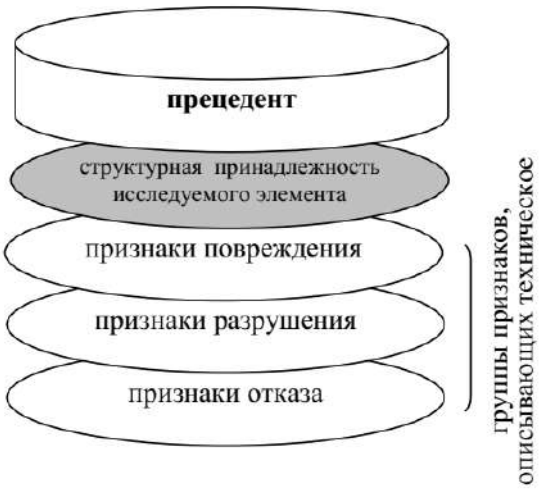


Рис. 6.4.1. Группы признаков, описывающих прецедент

В рамках предлагаемого решения рассматриваемые образы (прецеденты), рассматривались как классы, содержащие по одному экземпляру.

При *выборе алгоритма распознавания* была учтена особенность словаря признаков - то, что он содержит признаки смешанных типов: детерминированные и логические [298]. *Детерминированные признаки* - признаки, принимающие конкретные числовые значения *(количественные)*, например, параметр: глубина трещины – значение: 5 мм. *Логические признаки* можно рассматривать как элементарные высказывания, принимающие два значения истинности: «да», «нет» или «истина», «ложь». Это признаки, не имеющие количественного выражения, например, параметр: наличие поля ориентированных трещин – значение: да.

Обработка признаков смешанных типов требует создания *комбинированных* систем распознавания. Журавлевым Ю. И. предложен класс алгоритмов распознавания, используемых при построении комбинированных систем, называемый *алгоритмами распознавания, основанными на вычислении оценок* (АВО) [251, 298]. В основе АВО лежит эвристический принцип, которым пользуется человек, - принцип прецедентности или частичной прецедентности.

АВО сравнивает описание распознаваемого образа с описаниями всех образов, вычисляя оценку близости в два этапа:

- получение оценки близости для каждого образа;
- получение суммарной оценки по каждому из классов образов (при решении задачи идентификации технического состояния данный пункт опускается, так как не производилось выделение классов образов).

Оценки близости рассчитывается по формуле, позволяющей свернуть многомерное признаковое пространство, учесть важность (значимость, вес) признаков, описывающих образ [141, 251, 293, 298] с нормированием полученного значения:

$$d_G(\bar{x},\bar{y}) = \sum_{i=1}^{N} w_i h_G(x_i, y_i) \Big/ N,\ h_G(x_i, y_i) = \begin{cases} \textit{количественные} \begin{cases} 1, \textit{если} |x_i - y_i| < \xi \\ 0, \textit{иначе} \end{cases} \\ \textit{качественные} \begin{cases} 1, x_i = y_i \\ 0, x_i \neq y_i \end{cases} \end{cases},$$

где w_i – вес признака, ξ – ограничение на отличие значений признаков.

Для приведения всех признаков к единому масштабу выполняется процедура стандартизации:

$$x_{ik} \to \frac{x_{ik} - \min\limits_{k} x_{ik}}{\max\limits_{k} x_{ik} - \min\limits_{k} x_{ik}}.$$

Процедура вычисления оценки близости обладает *большой вычислительной мощностью (сложностью)* [251, 298], по этой причине была использована структуру прецедента (выделенные ранее группы признаков) для ее снижения: для решения задачи диагностики (идентификации) по прецедентам были применены элементы процедуры *последовательных* решений [251] и поиск решения производился в несколько этапов [116], по отдельным фрагментам (группам признаков) прецедента с целью постепенного *понижения размерности* пространства признаков и «стягивания» пространства прецедентов гиперсферой или гиперкубом.

Интерпретация полученных оценок близости осуществлялась при помощи *решающих правил* – правил, на основании которых вырабатывается заключения о принадлежности образа к определенному классу [251, 298]. В системе распознавания технического состояния предлагается использовать решающее правило следующего вида [116]: образы, которым соответствуют *максимальные* оценки близости по отношению к распознаваемому образу, являются его аналогами.

На этапе *повторного использования* решений принятых в *похожих* ситуациях происходит *адаптация* решения аналогичной проблемной ситуации к особенностям *рассматриваемой* проблемной ситуации. При этом использовалась адаптация *трансформационного* типа (transformational adaptation) [1, 125]. При данном типе адаптации решение для *новой* проблемной ситуации формируется путем *копирования* элементов решения по аналогичной проблемной ситуации и его дальнейшего *преобразования*. Достоинством [125] данного подхода к адаптации является то, что для своей работы он требует *только* описания проблемной ситуации, в отличие от *производной* адаптации (derivational adaptation) [125], которая хотя и позволяет генерировать принципиально новые решения, но требует для своего эффективного использования описание всего процесса (trace) получения

решения, что в свою очередь требует проведения *детальной* формализации предметной области.

Процесс адаптации [101, 102, 116, 182, 183, 362, 364, 365] осуществляется пользователем и может принимать две основные формы:

- Преобразование *описания* рассматриваемой проблемной ситуации. Осуществляется путем повторной конкретизации (качественного переопределения описания прецедента) и (или) путем уточнения значений параметров. Приводит к повторному поиску новых аналогов и новых решений.
- Преобразование *решения*, принятого по проблемной ситуации. Осуществляется путем изменения элементов решения на основании экспертных знаний о процессах и явлениях, воплощенных в модели предметной области.

В том случае, если подходящий прецедент не найден или найденное решение является неудовлетворительным по ряду критериев, то для решения задачи используется второй подход – продукционная экспертная система.

Продукционный подход. В основе решения задачи диагностики (идентификации) технического состояния по правилам лежит структура причинно-следственного комплекса формирования изменения технических состояний объекта (Рис. 6.4.2) [236].

Рис. 6.4.2. Структурная схема процесса формирования отказа и аварии.

На основе данной схемы были предопределены модели предметной области,

описывающие определенные понятия и отношения с точки зрения структурного аспекта. Рассмотрим предопределенные модели подробнее.

6.4.2 Концептуальные модели предметной области

Для построения моделей был произведен анализ работ [236, 328], по результатам которого были выделены следующие понятия и отношения [14, 116, 216, 325], формирующие модели предметной области (вычислительно-независимые модели):

- *Диагностируемый объект* рассмотрен как механическая система (машина или конструкция), имеющая сложную иерархическую структуру: специфицированное изделие – сборочные единицы – детали, при этом каждый уровень иерархии обладает соответствующим набором технических свойств (Рис. 6.4.3).

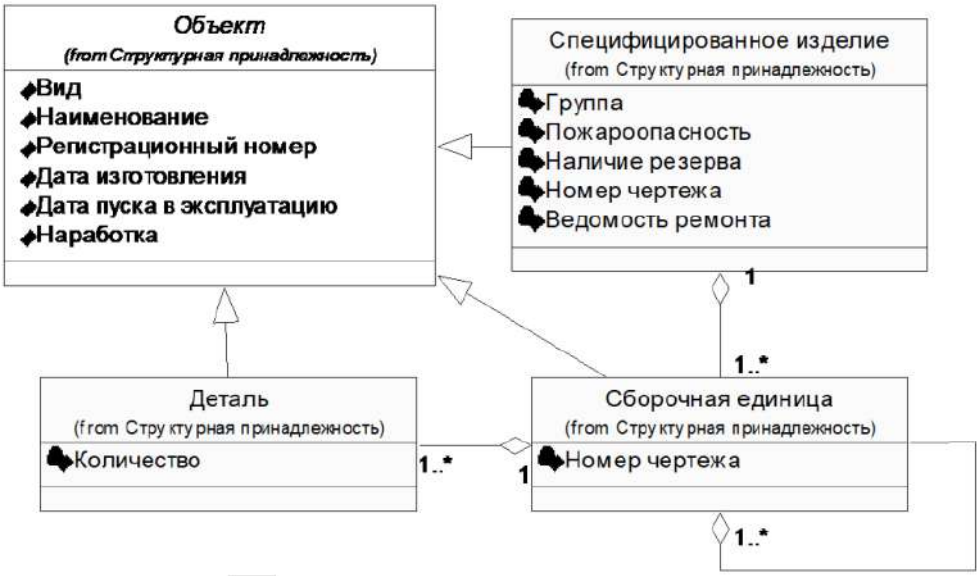


Рис. 6.4.3. Концептуальная модель понятия «Объект»

- *Процесс изменения технического состояния* представлен в виде непрерывно-дискретной последовательности технических состояний: исходной дефектности, поврежденности, разрушения, отказа и т.д. (Рис. 6.4.4). Состояния объекта взаимообусловлены. Состояние объекта (машины или конструкции) в целом определяется состояниями составляющих систему элементов (деталей).
- *Признаки/параметры технических состояний*. Состояние исходной дефектности описывается набором дефектов, каждый дефект, в свою

очередь, характеризуется набором параметров со значениями. Состояние поврежденности описывается набором повреждений (повреждение – изменение состояния элемента и свойств материала, из которого он изготовлен, вследствие внешних воздействий), каждое повреждение, в свою очередь, характеризуется набором параметров со значениями. Состояние разрушения описывается набором разрушений (разрушение – это изменение состояния, связанное с разделением элемента на части или достижением значений параметров разрушения недопустимых пределов), каждое разрушение, в свою очередь, характеризуется набором параметров со значениями. Состояние отказа (отказ – событие (изменение состояния), заключающееся в нарушении работоспособного состояния объекта) характеризуется набором параметров отказа со значениями.

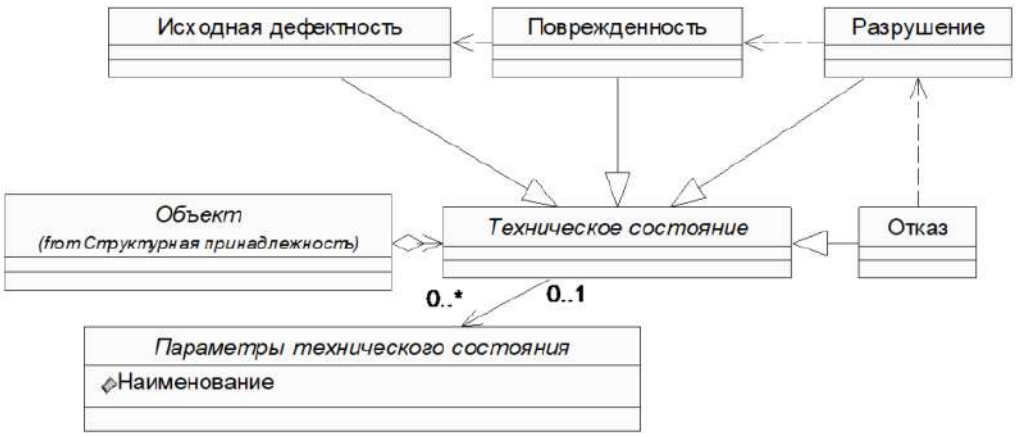


Рис. 6.4.4. Концептуальная модель понятия «Состояние»

- *Процесс деградации*. Вследствие взаимообусловленности состояний объекта можно говорить о взаимообусловленности параметров, характеризующих эти состояния. Наличие определенного дефекта (состояние исходной дефектности) приводит к развитию определенного повреждения (состояние поврежденности), которое, в свою очередь, вызывает разрушение (состояние разрушения), приводящее к отказу (состояние отказа). Таким образом, можно сделать вывод, что взаимообусловленность параметров, характеризующих состояния, является следствием того факта, что эти параметры являются проявлением протекающего во времени процесса деградации объекта (Рис. 6.4.5), являющегося объективной причиной нарушения его функционирования.

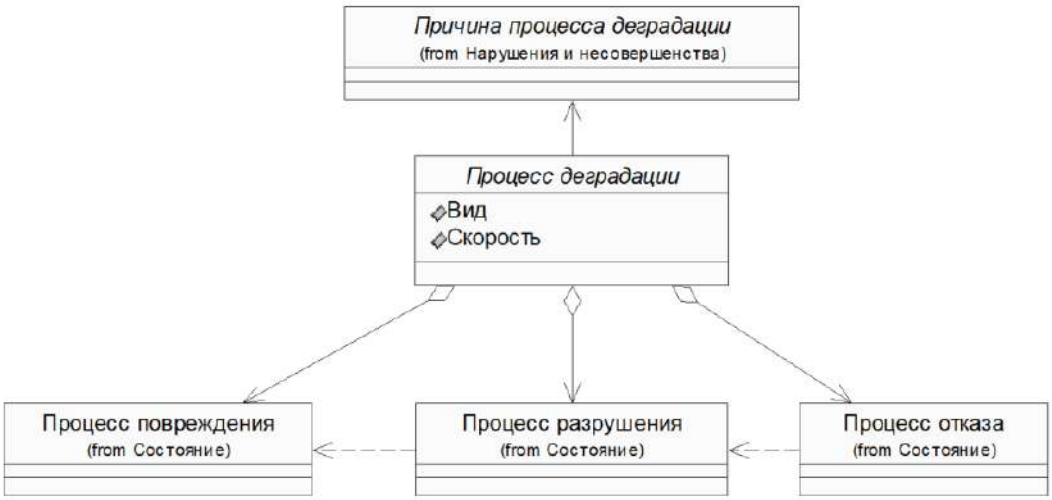


Рис. 6.4.5. Концептуальная модель понятия «Процесс деградации»

- *Причина процесса деградации*, как факторы, влияющие на изменение технического состояния – несовершенства и нарушения технологии, являющиеся причинами повреждения, разрушения, отказа.
- *Последствия* процесса деградации экологического, экономического и социального характера.

На основе построенных предметных моделей были созданы *платформо-независимые* модели, отражающие специфику использованных методов решения задачи диагностики: прецедетного и продукционного подходов.

Построенные предметные вычислительно-независимые модели были агрегированы (преобразовано) в единую модель, соответствующую понятию прецедент (Рис. 6.4.6) и включающую идентифицирующую и обучающую части. С предметной точки зрения это понятие наиболее близко понятию «*инцидент»* [328], под которым понимается отказ или повреждение (разрушение) технических устройств, применяемых на опасном производственном объекте, а также отклонение от режима технологического процесса и другие нарушения.

Фрагменты модели прецедента были использованы для формирования продукций, описывающих причинно-следственные связи, описывающие формирование и развитие деградационных процессов на основе причинно-следственного комплекса.

В качестве целевой платформы для прецедентного компонента интеллектуальной системы рассматривалась СУБД Cache. В связи с этим

платформо-зависимая модель структурно соответствует платформо-независимой модели, но содержат уточненные типы данных.

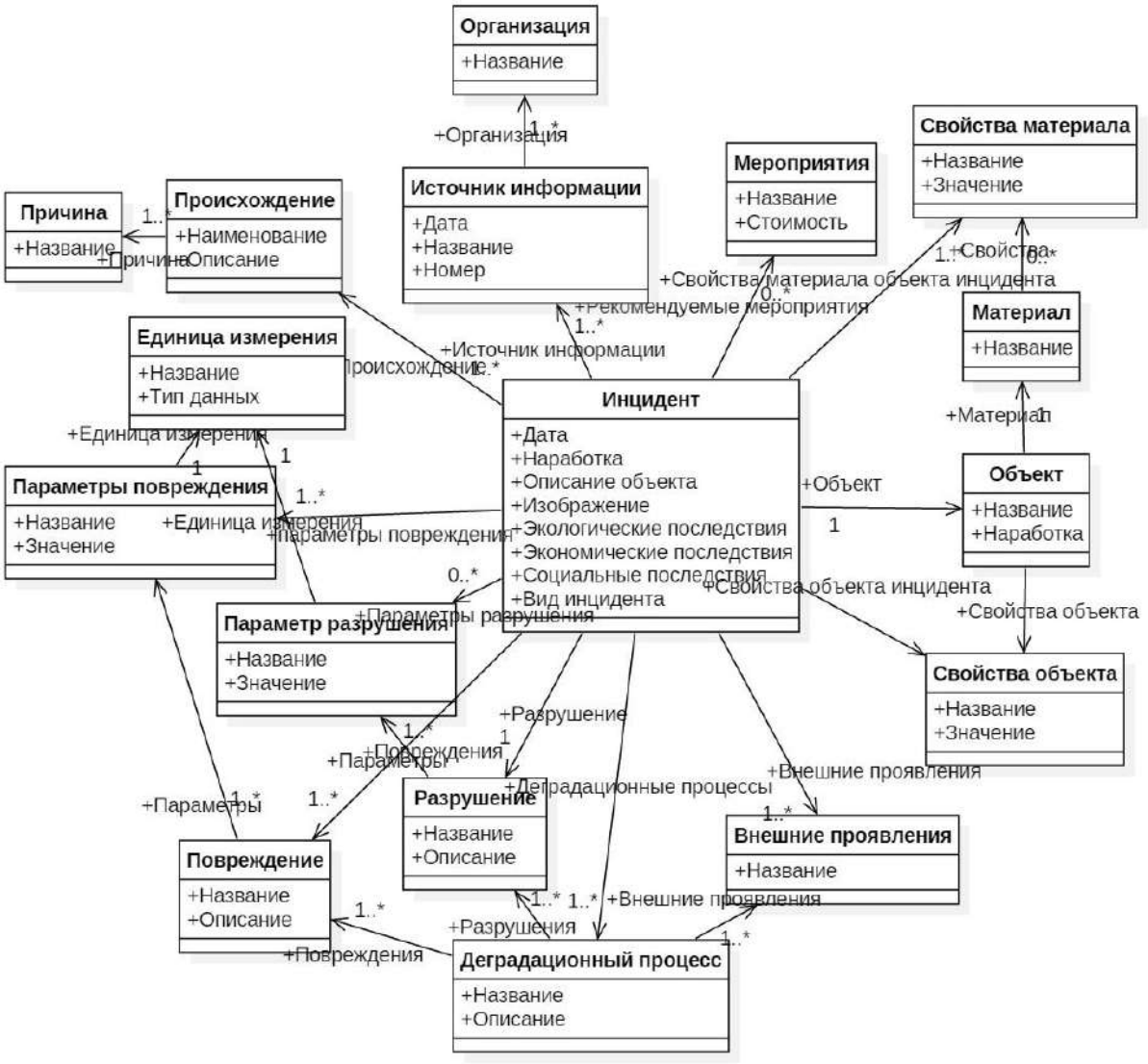


Рис. 6.4.6. Платформо-независимая модель базы прецедент для задачи идентификации технических состояний

В качестве целевой платформы для продукционного компонента рассматривался CLIPS. В связи с этим платформо-зависимые модели содержат уточненные типы данных и возможные значения «по умолчанию».

6.4.3 Модельные трансформации

Трансформации созданных концептуальных моделей осуществлялось непосредственно пользователем с использованием сторонних средств:

- для построения вычислительно- и платформо-независимых моделей использовалось CASE-средство IBM Rational Rose;

- для Cache использовался дополнительный модуль-расширение: «Rose-Cache-Link».
- для преобразования полученных моделей в программные структуры CLIPS был разработан модуль, обеспечивающий анализ MDL-формата. При этом при формировании фактов и обобщенных правил классам предметной области ставились в соответствие фреймы-образцы CLIPS, атрибутам классов – слоты фрейма-образца (Рис.6.4.7), отношениям между классами – продукционные правила (Рис.6.4.8). реализация трансформаций осуществлялась с использованием языка общего назначения Object Pascal.

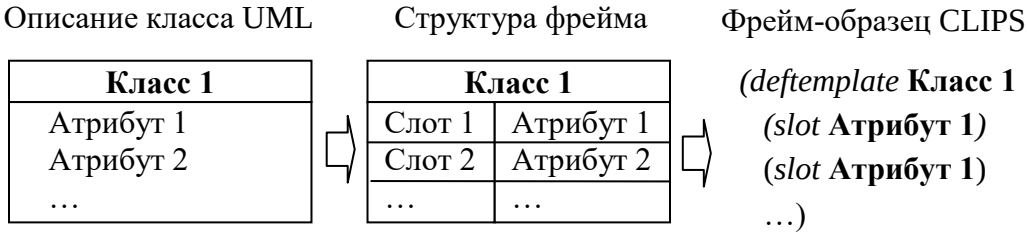


Рис.6.4.7. Преобразование описания класса на UML (Rational Rose) в описание фрейма образца базы знаний CLIPS

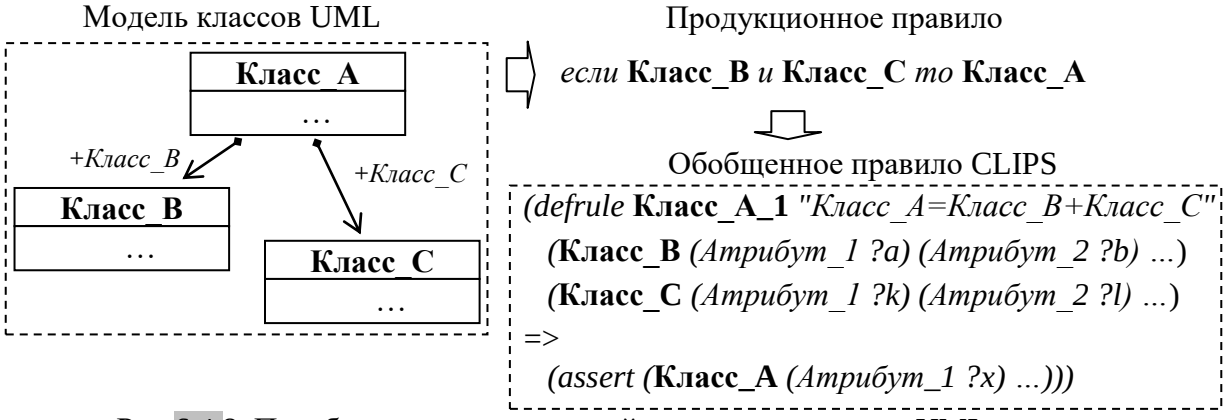


Рис.6.4.8. Преобразование отношений между классами на UML диаграмме (Rational Rose) в обобщенное правило базы знаний CLIPS

В настоящий момент созданы программные модули реализующие данные трансформации для PKBD и KBDS.

6.4.4 Программная реализация E-INFOT

E-INFOT представляет собой проблемно-ориентированную программу-оболочку, обладающую следующим основными функциями:

- Наполнение предопределенных (обусловленных используемыми методами решения задачи и моделями предметной области) структур базы данных

предметной информацией и обеспечение доступа к ним, включая: описание свойств объекта исследования; технических состояний объекта (МС); причин изменения состояний деталей, сборочных единиц и специфицированных изделий на стадиях жизненного цикла; описание мероприятий по предотвращению инцидентов.

- Наполнение предопределенных структур базы знаний предметной информацией и обеспечение доступа к ним, включая: определение свойств модели деградационных процессов; определение причинно-следственных зависимостей между причинами технических состояний объекта и необходимыми мероприятиями по предотвращению инцидентов.
- Настройка алгоритмов поиска решений;

Создаваемые при помощи E-INFOT прикладные системы обладают следующим набором функций:

- описание технического состояния детали по имеющейся информации;
- выявление причинно-следственного комплекса факторов обусловивших техническое состояние;
- планирование работ по уточнению причин изменения технического состояния (если необходимо);
- обоснование мероприятий: по предупреждению причин изменения технического состояния (рекомендуемые мероприятия); по обеспечению и восстановлению работоспособности системы (рекомендуемые меры); по минимизации опасности и снижению экономических, экологических и социальных потерь (рекомендуемые меры);
- диагностирование (идентификация) технического состояния детали (с регистрацией результатов в базе данных);
- прогнозирование изменений технического состояния;
- генерирование специализированной, краткой, полной отчетной информации о техническом состоянии детали.

С целью реализации рассмотренных функций была предложена архитектура E-INFOT, которая включает следующие основные модули (Рис. 6.4.9): создания и управления базами знаний, генерации отчетов и интерфейсного модуля.

Интерфейсный модуль обеспечивает доступ к основным функциям системы, а также реализует возможность динамического формирования графического

пользовательского интерфейса.

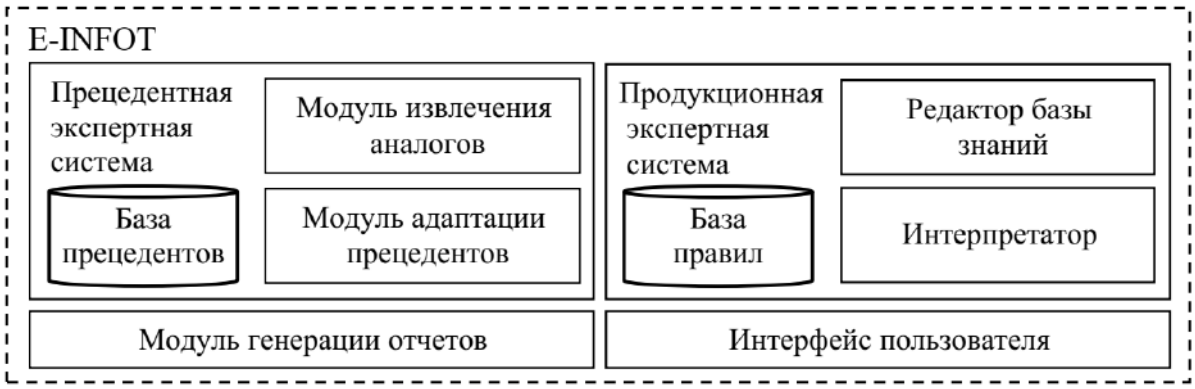


Рис. 6.4.9. Архитектура E-INFOT

Модуль *генерации отчетов* обеспечивает генерирование отчетной информации в формате HTML. Предусмотрено формирование следующих видов отчетов: (а) сведения по инциденту (прецеденту); (б) сравнительная характеристика двух инцидентов (прецедентов); (в) список отобранных аналогов; (г) отобранные по определенному признаку инциденты (прецеденты).

Особое внимание при проектировании архитектуры программной системы уделяется модулю *создания и управления базами знаний*. Ядро модуля - гибридная экспертная система, реализующая основные методы решения задачи идентификации. При этом предполагается двухстороннее взаимодействие: обучение продукционной ЭС за счет базы прецедентов, а также использование знаний продукционной базы знаний для адаптации решений полученных по аналогии.

Способность к обучению будет реализована посредством модуля *извлечения знаний*. В задачи данного модуля входит: (а) формирование шаблонов (шаблон состоит из перечня классов, связанных отношением); (б) анализ базы прецедентов при помощи сформированных шаблонов; (в) сохранение результатов анализа в продукционной базе знаний. Разработка данного модуля входит в планы дальнейших исследований.

В рамках прецедентной экспертной системы выделяются: база прецедентов; модуль извлечения модуль адаптации решений.

Составными частями продукционной экспертной системы являются: база знаний – содержит шаблоны фактов, факты, правила, описывающие предметные сущности, процессы, явления (модель предметной области); интерпретатор; редактор базы знаний.

6.4.5 Методика создания прикладных систем с помощью E-INFOT

Методика создания интеллектуальных систем диагностики (идентификации) технических состояний конструкций [116, 227, 280, 370] реализуется функциями E-INFOT и состоит из следующих этапов:

1) Уточнение моделей предметной области путем указания значений свойств отдельных элементов через графический интерфейс E-INFOT путем заполнения специальных справочников. В частности, описываются:

- техническое оборудование, с детальным описанием деталей, сборочных единиц, изделий, линий и производств;

- прецеденты (конкретные отказы) произошедшие на данном оборудовании;

- возможные деградационные процессы и стадии их развития в форме отдельных технических состояний;

- возможные причины деградационных процессов и мероприятия по их устранению.

2) Описание причинно-следственных зависимостей между техническими состояниями, причинами их изменения и последствиями средствами внешних систем. В частности, рекомендуется использовать IBM Rationl Rose для построения визуальных моделей в форме диаграмм классов UML.

3) Уточнение построенных UML диаграмм классов, в части конкретизации типов данных и возможных значений свойств выделенных классов.

4) Преобразование построенных UML диаграмм классов в программные коды CLIPS и перенесение полученных кодов в E-INFOT.

5) Проверка (прогон) программной системы по введенным данным.

Примеры интерфейса созданной интеллектуальной системы идентификации технических состояний (INFOT-3) приведены в приложении Г.

6.5 Трансформация электронных таблиц из отчетов по ЭПБ для создания онтологий в области нефтехимии

В данном разделе представлено применение трансформаций в рамках подхода к полуавтоматической формализации и представлению инженерных знаний в области нефтехимии в виде концептуальных моделей и электронных таблиц [47, 48, 54, 196, 197, 289, 290]. Исследование проводилось в рамках проекта РНФ 18-71-10001, решалась задача синтеза баз знаний с генерацией программных

кодов, в том числе в форме онтологий (на OWL 2 DL). Последовательность основных этапов решения данной задачи соответствует методу и методике (см. 4.1) (Рис.6.5.1).

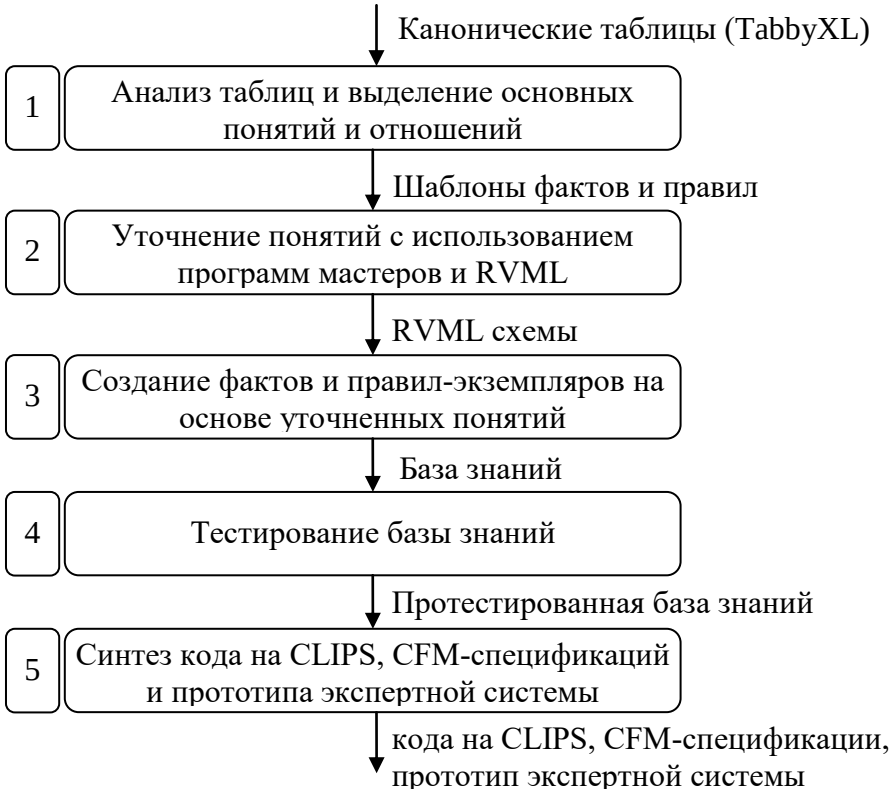


Рис. 6.5.1. Схема разработки базы знаний на основе анализа электронных таблиц

В качестве исходной информации использовались канонические таблицы, как особая форма представления электронных таблиц.

6.5.1 Электронные таблицы как источник знаний

В настоящее время остается актуальным разработка новых методов и инструментов формализации и представления сложных инженерных знаний в контексте создания интеллектуальных предметно-ориентированных систем для решения различных практических инженерных задач [171]. При этом можно отметить тенденцию к автоматизации формализации и представления знаний с использованием полуавтоматического преобразования различных источников информации (текстов, документации, баз данных, электронных таблиц, веб-ресурсов и т.д.), а также принципов визуального, концептуального и когнитивного моделирования [97, 165, 171].

В данном контексте электронные таблицы являются одним из наиболее

удобных способов структурирования и представления статистических и других данных. По этой причине они широко распространены, и их число достигает 150 миллионов только в сети Интернет [97]. Одним из направлений развития подходов к автоматизации разработки баз знаний является использование результатов анализа электронных таблиц в формате CSV и Excel, а также веб-таблиц [81, 158]. Эти подходы направлены на автоматизированное извлечение знаний из табличных данных и, как правило, ориентированы на конкретную структуру (модель) таблицы.

В рамках выполнения работ по проекту РНФ 18-71-10001 был предложен оригинальный подход, обеспечивающий автоматизированный анализ электронных таблиц с произвольной компоновкой с целью извлечения знаний. При этом были решены следующие задачи:

- Извлечение исходных произвольных электронных таблиц и преобразование их в оригинальную каноническую форму.
- Формализация и представление фрагментов модели предметной области, извлеченных из канонизированных таблиц, и объединение их в полную модель предметной области.
- Синтез кодов базы знаний на целевом языке представления знаний из полной модели предметной области.

В данной работе основное внимание уделено решению задач 2 и 3 при помощи авторского инструментального средства PKBD [184, 185, 195, 202, 257, 258, 269, 283, 367], которое использовалось для формализации и представления знаний в виде моделей предметной области и генерации программных кодов баз знаний. В качестве предметной области рассматривалась экспертиза промышленной безопасности, а наличие моделей и баз знаний в данной области позволило произвести содержательную оценку полученных решений.

6.5.2 Предобработка электронных таблиц

Источником исходных таблиц являлись отчеты по экспертизе промышленной безопасности. В рамках исследования было обработано 6 отчетов и выделено 216 таблиц, предварительный анализ которых выявил невозможность их прямого использования и необходимость предварительной обработки с целью

унификации их компоновки (формы).

В качестве унифицированного представления была выбрана так называемая каноническая форма:

$$CS^{CSV} = \{D, RH, CH\},$$

где D – блок данных, который описывает конкретные значения данных (записи), принадлежащие к одному и тому же типу данных (например, числовые, текстовые и т.д.); RH – набор заголовков строк, т.е. $RH = \{rh^1, ..., rh^n\}$, где rh^i – i-ячейка заголовка строки; CH – набор заголовков столбцов, т.е. $CH = \{ch^1, ..., ch^m\}$, где ch^j – j-ячейка заголовка столбца. Значения в ячейках блоков заголовков могут быть разделены символом «|», с помощью которого осуществляется представление иерархических отношений между заголовками (разделение заголовков на подзаголовки). Эта структура основана на каноническом представлении таблиц, предложенном в работе [158] и адаптированном для программной системы TabbyXL [148], которая используется в качестве средства предобработки исходных данных. Метамодель канонических таблиц представлена на Рис.6.5.2.

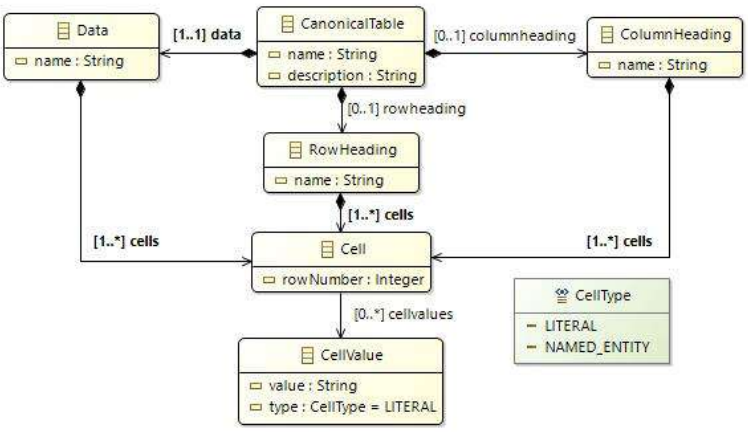


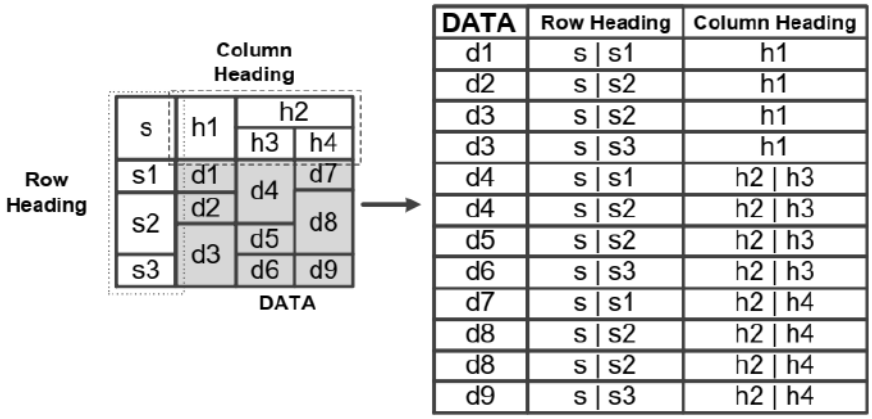
Рис. 6.5.2. Метамодель канонических таблиц.

В ходе анализа исходных таблиц были выделены две основные компоновки (формы) (Рис.6.5.3-6.5.4), которые и подверглись преобразованию. Более подробно преобразование рассмотрено в [48, 197].

6.4.3 Трансформация электронных таблиц

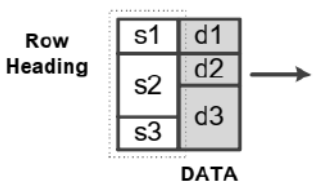
Унифицированные (канонизированные) таблицы в дальнейшем рассматриваются как вычислительно-независимые модели, содержащие структурированную информацию о понятиях и отношениях предметной области. В

дальнейшем они были преобразованы в структуры баз знаний (платформо-независимую модель) при помощи модуля расширения PKBD.TabbyXL.



DATA	Row Heading	Column Heading
d1	s | s1	h1
d2	s | s2	h1
d3	s | s2	h1
d3	s | s3	h1
d4	s | s1	h2 | h3
d4	s | s2	h2 | h3
d5	s | s2	h2 | h3
d6	s | s3	h2 | h3
d7	s | s1	h2 | h4
d8	s | s2	h2 | h4
d8	s | s2	h2 | h4
d9	s | s3	h2 | h4

Рис. 6.5.3. Пример преобразования таблицы первой формы в каноническую форму



DATA	Row Heading	Column Heading
d1	s1	
d2	s2	
d3	s2	
d3	s3	

Рис. 6.5.4. Пример преобразования таблицы второй формы в каноническую форму

В процесс преобразования каждая таблица соответствует одному фрагменту знаний, который может быть формализован и представлен в виде концептуальной модели, описывающей ограниченное подмножество понятий, свойств понятий и отношений предметной области. В зависимости от особенностей заполнения полученных канонических таблиц данными было выделено 5 основных вариантов типовых преобразований [48]. При этом преобразования 1-4 соответствуют первой форме исходных таблиц.

Преобразования 1 и 2 (Рис. 6.5.5) позволяют на основе канонической таблицы получить структуру одиночных понятий предметной области, отличающихся набором свойств (необходимо отметить, что UML диаграммы классов далее используется только для наглядной визуализации). Так в случае преобразования 1 отсутствует вложенность (иерархия) в заголовках строк, а в случае преобразования 2 данная иерархия используется для создания именующего свойства («name») с определенным значением.

Преобразование 3 позволяет интерпретировать вложенность (иерархию) в заголовках строк и столбцов с созданием набора из двух понятий, связанных (в

частном случае, иерархическим) отношением (Рис.6.5.6).

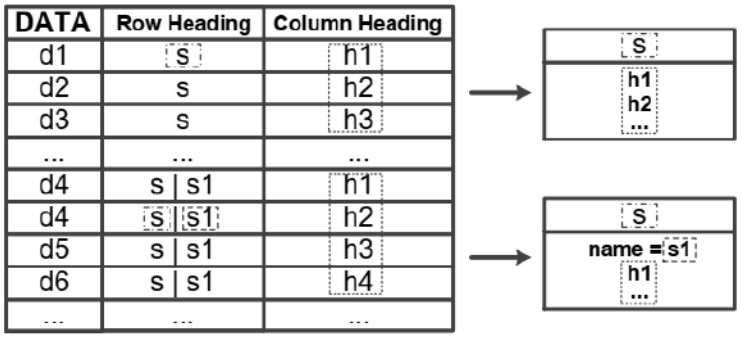


Рис. 6.5.5. Генерация фрагментов концептуальных моделей из канонической таблицы: преобразования 1 и 2

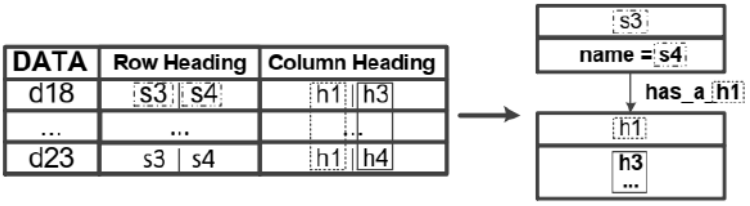


Рис. 6.5.6. Генерация фрагментов концептуальных моделей из канонической таблицы: преобразование 3

Преобразование 4 позволяет интерпретировать множественную вложенность (иерархию) в заголовках столбцов с созданием набора из трех понятий, связанных отношениями (Рис.6.5.7).

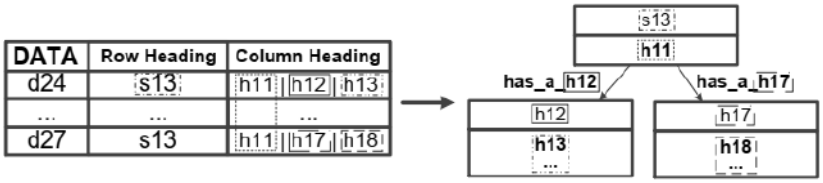


Рис. 6.5.7. Генерация фрагментов концептуальных моделей из канонической таблицы: преобразование 4

Преобразование 5 позволяет интерпретировать канонические таблицы, соответствующие второй форме исходных таблиц (Рис.6.5.8).

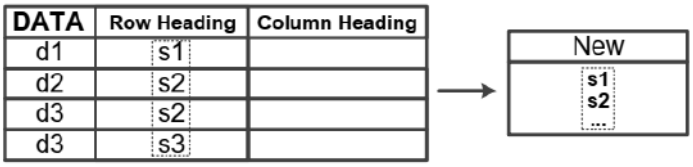


Рис. 6.5.8. Генерация фрагментов концептуальных моделей из канонической таблицы: преобразование 5

Алгоритм обработки заголовков строк (*RH*), реализующий рассмотренные преобразования, может быть описан следующим образом:

```
01 Create new RH class
02 New RH class name = first RH label
03 If count of RH labels == 1 then
04   Create new property of new RH class
05   New property name = "Name"
```

```
06 If count of RH labels == 2 then
07   Create new property of new RH class
08   New property name = second RH label
```

Алгоритм обработки заголовков столбцов (*CH*), реализующий рассмотренные преобразования, может быть описан следующим образом:

```
01 If count of CH labels == 1 then
02   Create new property of new RH class
     // created on the step 1 of
     // the row heading analysis algorithm
03   New property name = first CH label
04 If count of CH labels == 2 then
05   Create new CH class
06   New CH class name = first CH label
07   Create a new property of new CH class
     // created on the previous step
08   New property name = second CH label
09   Create new relationship
10   New relationship name = "has a"
       + first CH label
11   New relationship right entity = New RH class
     // created on the step 1 of
     // the row heading analysis algorithm
12   New relationship left entity = New CH class
     // created on the step 5 of
     // the column heading analysis algorithm
13 If count of CH labels == 3 then
14   Create new property of new RH class
15   New property name = first CL label
16   Create new CH class
17   New CH class name = second CH label
18   Create new relationship
19   New relationship name = "has a"
       + second CH label
20   New relationship right entity = New RL class
     // created on the step 1 of
     // the row heading analysis algorithm
21   New relationship left entity = New CH class
     // created on the step 16 of
     // the column heading analysis algorithm
22   Create a new property of new CH class
     // created on the step 16 of
     // the column heading analysis algorithm
23   New property name = third CH label
```

Все иерархические отношения в заголовках канонических таблиц интерпретируются как ассоциации без указания кратности (в терминах UML диаграмм классов), а не отношения обобщения (наследования), так как преобразования рассматриваются в контексте создания баз знаний, содержащих логические правила, в частности, для CLIPS, который не поддерживает подобных отношений.

Все значения свойств понятий (атрибутов классов) формируются с использованием информации из столбца данных (*DATA*) и обозначаются

строковым типом данных.

В результате преобразования отдельных канонических таблиц был получен набор фрагментов концептуальной модели, которые необходимо объединить в полную модель предметной области.

Объединение модели осуществлено с помощью следующих эвристических правил:

- объединение понятий (классов) с одинаковыми (совпадающими) или похожими именами, при этом для сравнения имен использовалась метрика редактирования (Левинштайна) [98] в диапазоне от 0 до 3 с учетом частичного совпадения структуры;
- объединение понятий с одинаковой структурой (наборами свойств);
- создание нового отношения между понятиями, если существуют одноименные понятия и свойства, когда имя одного понятия соответствует имени свойства другого понятия;
- удаление повторяющихся отношений между понятиями.

Сформированные понятия и отношения уточняются, и на их основе происходит генерация программных кодов целевой базы знаний.

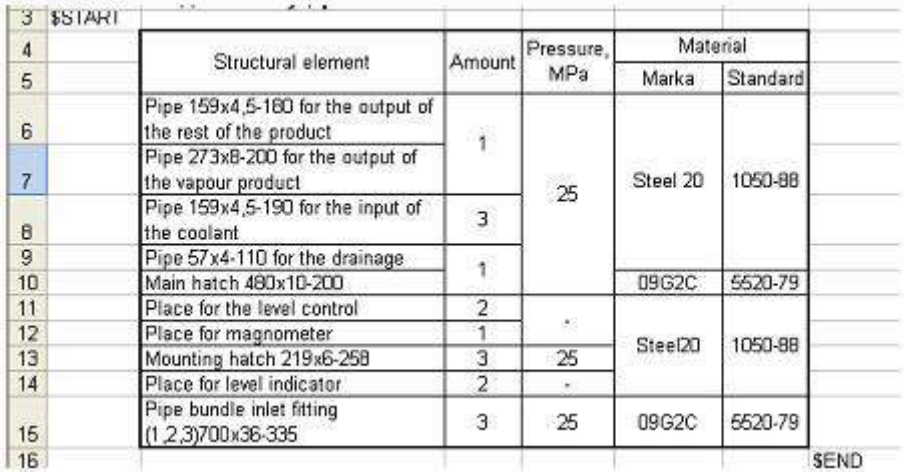
6.5.4 Создание онтологии по экспертизе промышленной безопасности

Для апробации подхода был использован набор данных из 216 таблиц, извлеченных из 6 отчетов по экспертизе промышленной безопасности формы «Ф. 2.04-01/03-08», 173 таблицы из набора имели уникальную компоновку и содержали 5817 ячеек. Была отобрана 161 таблица с уникальным содержанием и преобразована в каноническую форму с использованием описанных выше преобразований, получившийся набор данных был зарегистрирован под именем ISI-161 [189].

На Рис. 6.5.9 приведен пример одной из исходных таблиц, а на Рис. 6.5.10 – соответствующая ей таблица в канонической форме.

В результате преобразования канонических таблиц из набора ISI-161 с помощью PKBD.TabbyXL было получено 429 сущностей, включая 59 понятий (классов), 338 свойств (атрибутов) и 32 отношения (ассоциации). Необходимо отметить, что в результате экспертной оценки только 56% из полученных

сущностей были полезны для дальнейшей обработки. Объединение полученных фрагментов в полную концептуальную модель сократило их количество до 242 сущностей, включая 25 понятий, 196 свойств и 21 отношение.



		Structural element	Amount	Pressure, MPa	Material	
					Marka	Standard
3	$START					
6		Pipe 159x4,5-180 for the output of the rest of the product	1	25	Steel 20	1050-88
7		Pipe 273x8-200 for the output of the vapour product				
8		Pipe 159x4,5-190 for the input of the coolant	3			
9		Pipe 57x4-110 for the drainage	1			
10		Main hatch 480x10-200			09G2C	5520-79
11		Place for the level control	2	-	Steel20	1050-88
12		Place for magnometer	1			
13		Mounting hatch 219x6-258	3	25		
14		Place for level indicator	2	-		
15		Pipe bundle inlet fitting (1,2,3)700x36-335	3	25	09G2C	5520-79
16						$END

Рис. 6.5.9. Пример исходной таблицы из отчетов по экспертизе промышленной безопасности.

	1	2	3	4
1	DATA	Row Heading	Column Heading	
2	1	structural element | pipe 159x4,5-180 for the output of the rest of the product	amount	
3	25	structural element | pipe 159x4,5-180 for the output of the rest of the product	pressure, mpa	
4	steel 20	structural element | pipe 159x4,5-180 for the output of the rest of the product	material | marka	
5	1050-88	structural element | pipe 159x4,5-180 for the output of the rest of the product	material | standard	
6	1	structural element | pipe 273x8-200 for the output of the vapour product	amount	
7	25	structural element | pipe 273x8-200 for the output of the vapour product	pressure, mpa	
8	steel 20	structural element | pipe 273x8-200 for the output of the vapour product	material | marka	
9	1050-88	structural element | pipe 273x8-200 for the output of the vapour product	material | standard	
10	3	structural element | pipe 159x4,5-190 for the input of the coolant	amount	
11	25	structural element | pipe 159x4,5-190 for the input of the coolant	pressure, mpa	
12	steel 20	structural element | pipe 159x4,5-190 for the input of the coolant	material | marka	
13	1050-88	structural element | pipe 159x4,5-190 for the input of the coolant	material | standard	
14	1	structural element | pipe 57x4-110 for the drainage	amount	
15	25	structural element | pipe 57x4-110 for the drainage	pressure, mpa	
16	steel 20	structural element | pipe 57x4-110 for the drainage	material | marka	
17	1050-88	structural element | pipe 57x4-110 for the drainage	material | standard	

Рис. 6.5.10. Фрагменты канонической таблицы из набора ISI-161, соответствующей таблице на Рис. 6.5.9

Для содержательной (качественной) оценки полученной модели был использован набор моделей экспертизы промышленной безопасности ISI-Models [190], разработанный ранее в сотрудничестве с экспертами АО «ИркутскНИИХимМаш» при работе над ИАС «Экспертиза ПБ».

При сравнении моделей наборов ISI-161 и ISI-Models было обнаружено, что 17 % (69 из 400) понятий из набора ISI-Models могут быть соотнесены с понятиями, полученными в результате анализа набора данных ISI-161, включая сущности (Таблица 6.5.1), свойства и отношения (Таблица 6.5.2). На рисунке 6.5.11 представлен фрагмент из набора данных ISI-Models (файл 01.mdl) и

соответствующие понятия из результатов преобразования таблиц.

Таблица 6.5.1. Пример соотнесенных понятий из наборов ISI-161 и ISI-Models

ISI-161	**ISI-Models**
material (материал)	material (материал)
inspection method (метод проверки)	inspection result (результаты проверки)
organization (организация)	expert organization; enterprise (экспертная организация; экспертиза)
parameter (параметр)	parameters; operation parameters (параметры, параметры функционирования)
dimensions (размеры)	geometric parameters (геометрические параметры)
welded joint (сварной шов)	information about welding (информация о сварке)
hardness (прочность)	the results of hardness measurement (результаты измерения прочности)
thickness (толщина)	thickness measurements (измерения толщины)
point (place) of measurement (точка (место) измерения)	the results of thickness measurements at points (результаты измерения толщины в точке)
fitting (фиттинг)	fittings; flanges; information about fittings and flanges for MID (фитинги, фланцы, информация о фитингах и фланцах)
expert (эксперт)	experts and specialists (эксперты и специалисты)
element (элемент)	elements of a technical object; technical object (элементы технического объекта; технический объект)

Таблица 6.5.2. Пример соотнесенных отношений между понятиями из наборов ISI-161 и ISI-Models

ISI-161	**ISI-Models**
parameter – value (параметр – згначение)	operation parameters – technical characteristics (параметр функционирования – технические характеристики)
point (place) of measurement – material (точка (место) измерения – материал)	the results of thickness measurements at points – material (результаты измерения в точке – материал)
element – material (элемент – материал)	elements of a technical object – material (элементы технического объекта – материал)
element – parameter (элемент – параметр)	technical object – operation parameters (технический объект – параметры функционирования)

Качественная оценка полученной модели показала, что ее использование для построения баз знаний без дополнения предметной информацией (например, из ISI-Models) затруднительно. Был проведен эксперимент по дополнению полученной модели, в результате которого процент соответствия понятий в наборах достиг 24 % (106 из 400). В таблице 6.5.3 показаны количественные характеристики сравниваемых наборов данных.

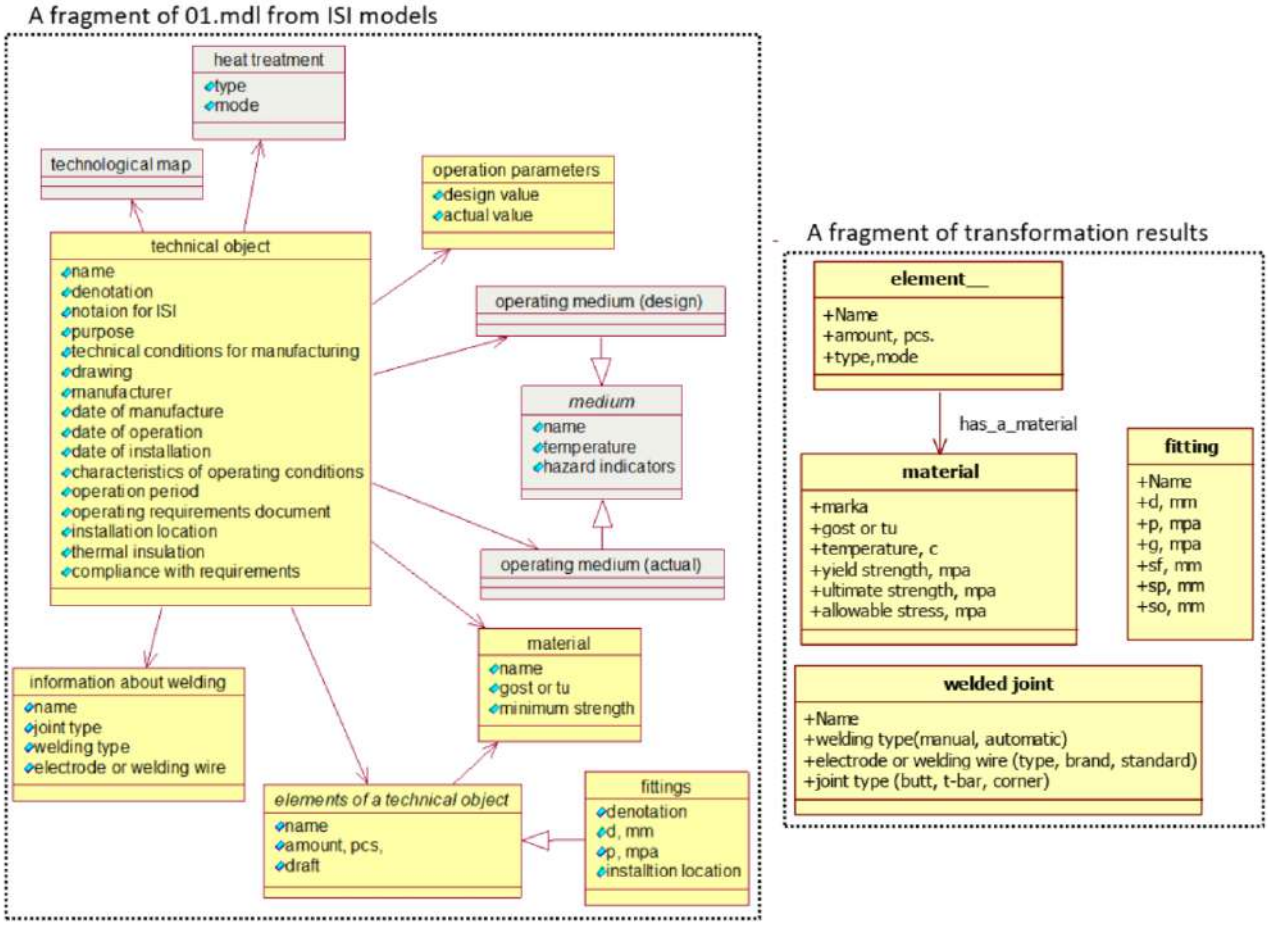


Рис. 6.5.11. Фрагмент модели из набора ISI-Models (файл 01.mdl) и соответствующие понятия из результатов преобразования таблиц из набора ISI-161

Таблица 6.5.3. Количественные характеристики наборов ISI-161 и ISI-Models

Набор данных	**Критерий**			
	Элементы	Понятия	Свойства	Отношения
ISI-161	429	59	338	32
Отредактированный ISI-161 (56% от ISI-161)	242	25	196	21
ISI-Models (фрагмент из 21 модели)	400	98	249	53
Модели, полученные на основе анализа отредактированного ISI-161 и соответствующие им ISI-Models (17% от ISI-Models)	69	14	51	4
Дополненные модели, полученные на основе анализа отредактированного ISI-161 и соответствующие им ISI-Models (24% от ISI-Models)	106	14	88	4

В дальнейшем, полученные концептуальные модели (понятия и отношения) были преобразованы в структуры базы знаний с использованием PKBD. На

рисунке 6.5.12 показаны примеры полученных уточненных структур в форме диаграмм RVML.

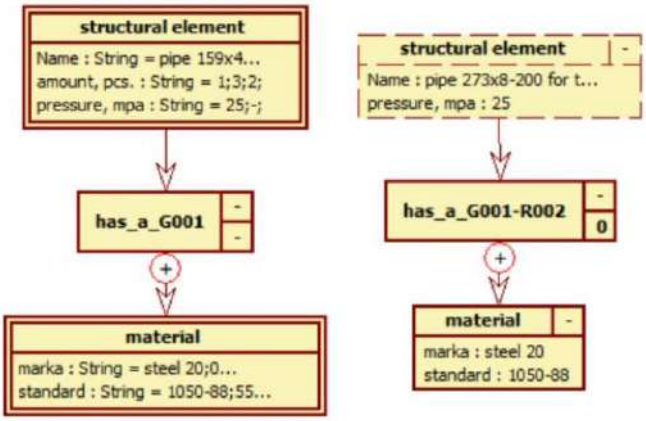


Рис. 6.5.12. Примеры полученных элементов базы знаний: шаблон правила и конкретное правило (экземпляр шаблона правила) в нотации RVML

Уточненные структуры использовались для синтеза программных кодов прототипов онтологий и баз знаний для ИАС «Экспертиза ПБ» (Рис.6.5.13).

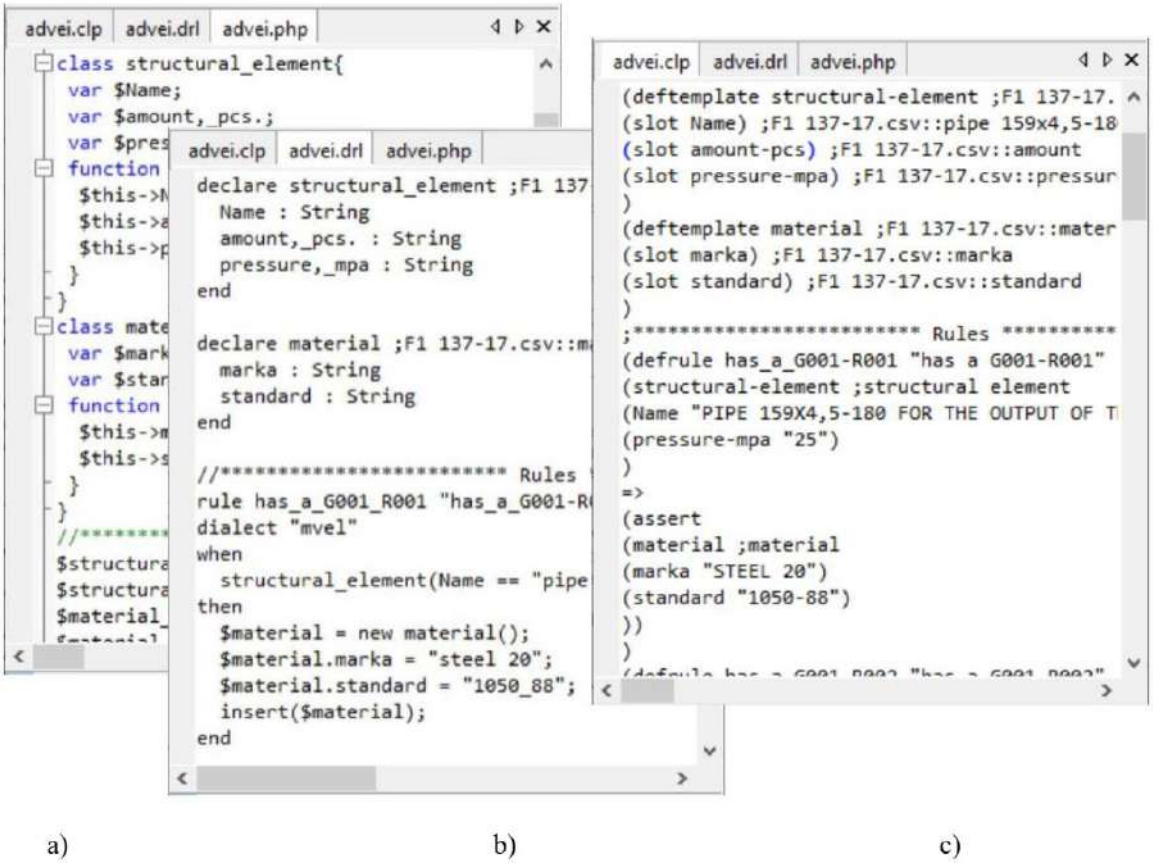


Рис. 6.5.13. Примеры синтезированных программных кодов на: a) PHP; b) DROOLS; c) CLIPS

6.5.5 Экспериментальная оценка трансформаций таблиц

Была произведена оценка правильности произведенных трансформаций с формальной точки зрения, т.е. учитывая только синтаксический аспект. Для этого

были использованы классические формулы определения полноты, точности и F-меры:

$$recall = \frac{TP}{TP + FN}, \quad precision = \frac{TP}{TP + FP},$$

где *TP* – множество правильно преобразованных сущностей (в том числе значений ячеек), *FP* – множество некорректно преобразованных сущностей, *FN* – множество непреобразованных сущностей, которые могли быть преобразованы. В таблице 6.5.4 представлены результаты оценки трансформаций с количественной точки зрения.

Таблица 6.5.4. Результаты оценки трансформаций

Трансформация	**Оценка**		
	Recall	Precision	F1
Канонические таблицы в концептуальные модели	0.96	0.97	0.97
Концептуальные модели в программные коды	0.99	0.99	0.99

Оценки показали, что с помощью предлагаемого подхода удается преобразовать большинство канонических таблиц и концептуальных моделей. В результате анализа таблиц из отчетов по экспертизе промышленной безопасности удалось создать концептуальные модели, полезность которых с содержательной (семантической) точки зрения была оценена как 17% и 24%. Основная причина таких оценок – разный акцент (направленность) используемых наборов данных: набор данных ISI-161 содержит информацию о результатах технической диагностики оборудования, в то время как ISI-Models ориентирован на описание (моделирование) всей процедуры промышленной безопасности, включая такие задачи, как разработка программы исследования, анализ и интерпретация результатов диагностики, а также принятие решений по ремонту и формирование заключения (отчета). По этой причине 173 сущности из ISI-161 в дальнейшем не использовались для построения баз знаний.

В качестве основных факторов, влияющих на качество трансформаций канонических таблиц, выделены следующие:

- несовершенство правил преобразования для обработки иерархии понятий, в частности, пропуск третьего уровня иерархии понятий для заголовков строк;
- несовершенство стратегий агрегирования для фрагментов концептуальной модели (например, объединение понятий “gt_20, MPA”

и “gb_20, MPA”, которые синтаксически похожи, но различаются семантически);

- недопустимый тип ячейки во входных данных электронной таблицы.

Общим недостатком трансформации концептуальных моделей и генерации программных кодов является ограничение на сложность создаваемых логических правил, которые не поддерживают такие элементы, как переменные, вычисляемые выражения, функции и т.д. Однако такие структуры могут быть добавлены после синтеза кодов при их отладке и интеграции.

Выводы

Произведена апробация и оценка эффективности разработанных языков, методов и программных средств при решении задач в области техногенной безопасности для нефтехимии, в частности при создании: ИАС «Экспертиза ПБ» (рег.№ 2016610757) для выявления причин повреждений и разрушения элементов технических систем в нефтехимии; БЗ прецедентной интеллектуальной системы для подбора конструкционных материалов; программы для интеллектуального планировщика анализа отказов в ИС «INFOT-3» (рег.№ 2007613715); спецификаций для проблемно-ориентированного редактора (рег.№ 2012614093) и БЗ системы идентификации технических состояний конструкций, созданной с помощью программы-оболочки «E-INFOT» (рег.№ 2005611217); онтологии в области экспертизы промышленной безопасности.

Применение предлагаемых в диссертации методов и программных средств позволило повысить эффективность решения данных задач за счет использования визуального программирования, концептуальных моделей, автоматической кодогенерации и вовлечения в процесс разработки конечных пользователей.

Глава 7. Оценка эффективности разработанных методов и средств

В качестве критерия для оценки эффективности предлагаемых языков, методов и средств был выбран временной критерий. Проведение модельных экспериментов, оценивающих реальное время разработки экспертных систем (когда разработка выполняется стандартным методом, а потом предлагаемым) является затратным и не всегда возможно. По этой причине оценка эффективности была произведена как косвенным способом – путем умозрительной оценки затрат на отдельных этапах процесса разработки, так и прямым – на примере решения учебных задач. При этом прямой способ оценивал временные затраты на разработку только БЗ, т.к. использованный инструментарий представлял собой, в том числе, программу-оболочку, соответственно, программирование ЭС не требовалось.

7.1 Косвенный способ

При косвенном способе оценки рассматривались основные этапы разработки ЭС и время их выполнения. При этом этап концептуализации был декомпозирован на этапы идентификации проблемы, получения и структурирования знаний, согласно [244, 263] (Табл. 7.6.1).

Таблица 7.6.1. Оценка времени разработки ЭС косвенным способов

Этапы создания ЭС	**Стандартный метод (недель)**	**Предлагаемый метод (недель)**
Идентификация проблемы	1-2	0.75-1.5
Получение знаний	4-12	3-8
Структурирование знаний	2-4	1.5-3
Формализация знаний	4-8	3-6
Реализация	4-8	-
Тестирование	1-2	1-2
ИТОГО:	**16-36**	**9.25-20.5**

Таким образом (Табл. 7.6.1), разработка ЭС стандартным методом в среднем продолжается от 4 до 9 месяцев. При применении подходов, основанных на модельных трансформациях, из процесса разработки исключается этап реализации, т.к. используется автоматическая кодогенерация, что обеспечивает сокращение времени разработки ЭС на 1-2 месяца.

Кроме того, при реализации этапов получения, структурирования и формализации знаний из процесса разработки исключается (или сокращается время

участия) инженера по знаниям, т.е. сокращается время необходимое:

- для организации этапов извлечения знаний, осуществляемых инженером по знаниям и экспертом во взаимодействии;
- для структуризации и формализации знаний за счет слияния данных этапов, т.к. в предлагаемом подходе структуризация (или установление связей) осуществляется одновременно с формализацией, где связи описывают структуру продукции;
- для ликвидации «языковых ножниц» между инженером по знаниям и экспертом;
- для детализации моделей представления знаний эксперта и инженера по знаниям (исключение эффекта «испорченного телефона»).

Исходя из данных фактов и худших предположений о достижимости перечисленных эффектов, время выполнения данных этапов сократится на 25%, т.е. на 0,5-1,5 месяца.

Итоговое сокращение времени составит – от 1,5 до 3,5 месяцев.

Необходимо также отметить, что согласно концепции модельно-ориентированных подходов, основной акцент при разработке приложений переносится с собственно этапа программирования на этап создания модели. При этом, создав один раз модель, разработчик получает принципиальную возможность генерации приложений для разных аппаратных и программных платформ.

Поэтому дополнительный эффект возникает при необходимости переноса разработки на новую технологическую платформу, так как при этом можно использовать старую платформо-независимую модель и разрабатывать заново только платформо-зависимую. При использовании предлагаемого подхода время разработки ЭС составит всего 1,5 до 2,5 месяца, т.к. потребует повторного выполнения работ только на этапе тестирования. Конечно, это возможно только при наличии программных средств (модулей), обеспечивающих генерацию кодов под целевую технологическую платформу.

В свою очередь, использование онтологического описания предметной области позволяет исключить (или сократить) этап концептуализации при разработке программной системы для решения близких по тематике задач в конкретной предметной области (или повторное использование).

Кроме того, применение нового подхода позволяет :

- уменьшить риск ошибки проектирования: на относительно простой и полностью основанной на требованиях заказчика платформо-независимой модели (в отличие от традиционной модели, загромождённой деталями реализации) легко находить и исправлять подобные ошибки;
- использовать когнитивную графику на этапе извлечения знаний;
- упростить создание документации;
- исключить вероятность возникновения ошибок программирования, за счет автоматической кодогенерации;
- упростить интеграцию систем и создание гетерогенных систем.

7.2 Прямой способ: решение учебных задач

Помимо косвенного способа, оценка эффективности разработанных в диссертации языков, методов и средств была произведена прямым способом, т.е. в результате создания прототипов БЗ ИС для решения тестовых задач диагностики и прогнозирования из учебного процесса.

Экспериментальное исследование [184, 270] проводилось с привлечением 60 обучаемых Института информационных технологий и анализа данных (ИТиАД) Иркутского национального исследовательского технического университета, в частности, были задействованы студенты 4 курса групп АСУз-10, АСУбз-11, АСУб-12, ЭВМбзс-12 в рамках выполнения лабораторных работ по дисциплинам «CASE-средства», «Инструментальные средства информационных технологий» и «Технологии программирования». На момент проведения исследования, его участники уже обладали навыками и компетенциями в областях: проектирование и моделирование программного обеспечения, интеллектуальные системы, банки и базы знаний, разработка на языках высокого уровня.

Основная цель исследования заключалась в оценке трудоемкость создания БЗ ИС разными способами по временному критерию. При этом было определено три способа разработки:

- **П1** – основан на использовании разработанных диссертантом языков, методов и средств, предполагает проведение концептуального моделирования, трансформацию полученных концептуальных моделей в БЗ, доработку уточненной

БЗ, ее отладку и генерацию кодов в формате CLIPS; способ в большей степени ориентирован на конечных пользователей;

- **П2** – основан на использовании сторонних средств концептуального моделирования и программирования БЗ, предполагает проведение концептуального моделирования, программирование и отладку БЗ CLIPS; способ требует определенных навыков программирования;
- **П2** – основан на использовании сторонних программирования БЗ, предполагает проведение программирование и отладку БЗ CLIPS без использования концептуальных моделей; способ ориентирован на программистов.

Определен следующий набор инструментальных средств для проведения экспериментального исследования:

- средства концептуального моделирования: IBM Rational Rose Enterprise (для UML построения диаграмм классов) и IHMC CmapTools (для построения концепт-карт);
- средства программирования на CLIPS: ClipsWin, PKBD, KBDS;
- средства поддержки модельных трансформаций: PKBD и KBDS.

Был определен набор из 20 вариантов учебных (тестовых) заданий (Табл. 7.6.2), целью которых являлась разработка БЗ для статических ЭС диагностического или прогностического типа в разных предметных областях (например, прогнозирование риска наводнения, диагностика неисправности электрочайника и др.).

Таблица 7.2.1. Описание тестовых заданий

Вариант №	Предметные сущности, кол. экз.	Связи, кол. шт.	Причинно-следственные связи, кол. шт.	Правила, кол. шт.
1	6	5	3	10
2	5	6	3	10
3	8	5	3	10
4	5	8	4	11
5	8	7	3	12
6	9	5	3	10
7	5	6	3	14
8	8	7	4	14
9	6	5	3	15
10	7	10	3	12
11	5	6	3	11
12	5	6	3	12

13	7	7	3	14
14	8	5	3	11
15	7	6	3	18
16	6	8	3	14
17	6	5	3	11
18	8	7	3	12
19	7	8	3	10
20	5	7	3	12

С целью обеспечения возможности повторения процесса выполнения задания за приемлемое время были определены количественные ограничения на разрабатываемые предметные концептуальные модели:

- количество сущностей: 5-10;
- количество свойств сущностей: до 3;
- количество связей между сущностями: 5-10;
- количество причинно-следственных связей, которые могут быть интерпретированы как принципиальные логические зависимости: 3-4;
- количество экземпляров причинно-следственных связей, которые представляют собой конкретные правила, построенные на основе принципиальных логических зависимостей: 10-15.

При этом первые четыре пункта приведенного выше списка связаны с концептуальным моделированием и представляют собой ограничения на количество, состав и структуру элементов UML диаграммы классов и концепт-карт, тогда как последний связан с уточнением БЗ и ее программированием и представляет собой ограничение на количество возможных правил в БЗ.

В качестве основного критерия для оценки эффективности выбран временной критерий, который позволял определить затраты времени на выполнение следующих этапов процесса разработки БЗ [244, 263]:

1) Концептуализация (структурирование) знаний, в том числе, подэтапы:

a. выделение основных предметных понятий и отношений между ними, включая характеристику различных видов используемых данных, анализ информационных потоков и структур в предметной области в терминах, причинно-следственных связей, отношений частное/целое, постоянное/временное и т.п.;

b. построение концептуальной модели.

2) Формализация знаний, в том числе, подэтапы:

a. перевод ключевых понятий и отношений на некоторый формальный ЯПЗ;

b. оценка полноты и степени достоверности (неопределенности) информации и других ограничений, накладываемых на логическую интерпретацию данных, таких как зависимость от времени, надежность и полнота различных источников информации.

3) Кодирование (программирование) БЗ.

Результаты оценки временных затрат для вариантов заданий при использовании UML представлены в Таблице 7.2.2 и на Рис. 7.2.1; для концепт-карт – в Таблице 7.2.3 и на Рис. 7.2.2.

При этом выделены минимальные и максимальные процентные значения относительной разницы между П1 и П3, П1 и П2.

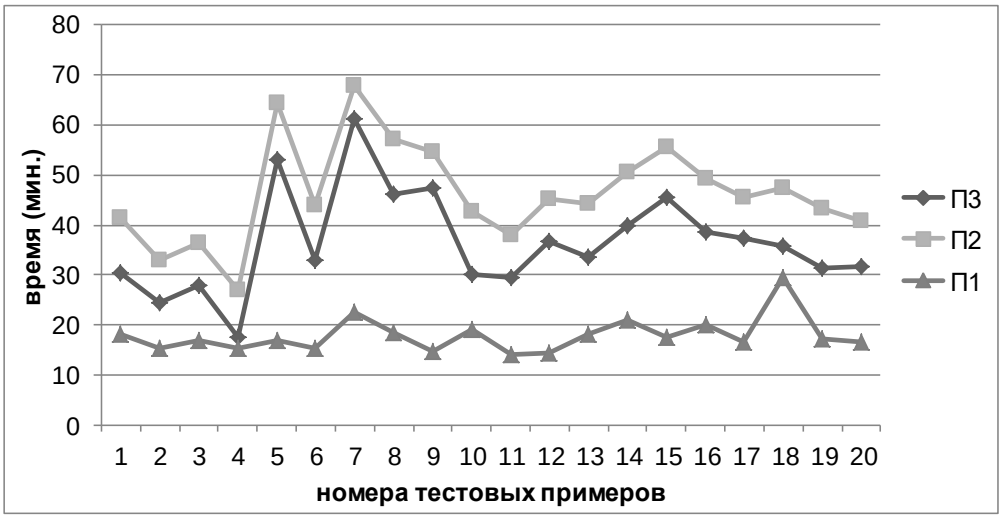


Рис. 7.2.1. Результаты оценки временных затрат (использование UML)

Рис. 7.2.2. Результаты оценки временных затрат (использование концепт-карт)

Таблица 7.2.2. Результаты оценки временных затрат (использование UML)

Вариант	**Rational Rose, мин.**	**KBDS, мин.**	**П1, мин.**	**П2, мин.**	**П3, мин.**	**П3: Ошибки програм., шт.**	**Относительная разница, % П1 и П3**	**Относительная разница, % П1 и П2**
1	10,89	7,2	18,09	41,29	30,4	2	40,49	56,19
2	8,36	7,1	15,46	32,86	24,5	0	36,89	52,95
3	8,58	8,3	16,88	36,46	27,88	1	39,45	53,70
4	9,36	5,83	15,19	26,82	17,46	0	*13,00*	43,36
5	11,25	5,52	16,77	64,41	53,16	3	68,45	**73,96**
6	10,78	4,6	15,38	43,8	33,02	1	53,42	64,89
7	6,6	15,82	22,42	68	61,4	5	63,48	67,03
8	10,95	7,56	18,51	57,23	46,28	3	60,00	67,66
9	7,37	7,2	14,57	54,71	47,34	3	**69,22**	73,37
10	12,58	6,6	19,18	42,7	30,12	0	36,32	55,08
11	8,69	5,5	14,19	38,01	29,32	0	51,60	62,67
12	8,36	6	14,36	45,22	36,86	2	61,04	68,24
13	10,64	7,42	18,06	44,31	33,67	2	46,36	59,24
14	10,66	10,23	20,89	50,57	39,91	1	47,66	58,69
15	10,01	7,56	17,57	55,5	45,49	4	61,38	68,34
16	10,92	8,96	19,88	49,42	38,5	1	48,36	59,77
17	8,14	8,36	16,5	45,59	37,45	1	55,94	63,81
18	11,55	18	29,55	47,43	35,88	3	17,64	*37,70*
19	11,85	5,2	17,05	43,28	31,43	1	45,75	60,61
20	9,12	7,44	16,56	40,95	31,83	2	47,97	59,56
Итоговое среднее значение относительной разницы:							**48,2**	**60,3**

Таблица 7.2.3. Результаты оценки временных затрат (использование концепт-карт)

Вариант	**Cmap Tools, мин.**	**KBDS, мин.**	**П1, мин.**	**П2, мин.**	**П3, мин.**	**П3: Ошибки програм., шт.**	**Относительная разница, % П1 и П3**	**Относительная разница, % П1 и П2**
1	7,6	7,2	14,8	38	30,4	2	51,31	61,05
2	4,95	7,1	12,05	29,45	24,5	0	50,82	59,08
3	6,14	8,3	14,44	34,02	27,88	1	48,21	57,55
4	6,4	5,83	12,23	23,86	17,46	0	29,95	48,74
5	8,42	5,52	13,94	61,58	53,16	3	**73,78**	**77,36**
6	8,12	4,6	12,72	41,14	33,02	1	61,48	69,08
7	4,68	15,82	20,5	66,08	61,4	5	66,61	68,98
8	7,38	7,56	14,94	53,66	46,28	3	67,72	72,16
9	7,28	7,2	14,48	54,62	47,34	3	69,41	73,49
10	10,36	6,6	16,96	40,48	30,12	0	43,69	58,1
11	5,84	5,5	11,34	35,16	29,32	0	61,32	67,75
12	5,07	6	11,07	41,93	36,86	2	69,97	73,6
13	8,87	7,42	16,29	42,54	33,67	2	51,62	61,71
14	9,55	10,23	19,78	49,46	39,91	1	50,44	60,01
15	9,85	7,56	17,41	55,34	45,49	4	61,73	68,54
16	8,62	8,96	17,58	47,12	38,5	1	54,34	62,69
17	5,6	8,36	13,96	43,05	37,45	1	62,72	67,57
18	9,28	18	27,28	45,16	35,88	3	*23,97*	*39,59*
19	9,25	5,2	14,45	40,68	31,43	1	54,02	64,48
20	7,24	7,44	14,68	39,07	31,83	2	53,88	62,43
Итоговое среднее значение относительной разницы:							***55,3***	***63,7***

Необходимо отметить ключевые особенности способов, выявленные при их

использовании:

1) **П1:** При разработке БЗ применялись разработанные диссертантом программные средства PKBD и KBDS, содержащие программные компоненты-конверторы анализа диаграмм классов UML и концепт-карт XTM, которые позволили автоматизировать создание структур БЗ, приближенных к моделям предметной области, а также автоматизировать кодогенерацию и отладку.

2) **П2:** Данный способ показал самые большие временные затраты даже при условии концептуального моделирования предметной области средствами IBM Rational Rose Enterprise и IHMC CmapTools, что обсуловлено разрывом в технологической цепочке: по причине отсутствия у ClipsWin возможности интеграции с системами концептуального моделирования разработанные модели вручную переносились (перенабирались) в среду программирования.

3) **П3:** Функциональные ограничения ClipsWin в части удобства редактирования программного кода обусловили применение дополнительного программного обеспечения в форме текстового редактора Programmer’s Notepad (PN). По этой причине программный код первоначально формировался в PN с использованием механизмов копирования и вставки отдельных блоков, а также подцветки синтаксиса, затем он переносился в ClipsWin для дальнейшей отладки. Несмотря на введение дополнительного программного обеспечения, временные затраты на создание БЗ уменьшились в 1,5 раза по сранению с использованием только ClipsWin.

Сравнительные анализ времени выполнения тестовых заданий различными способами показал, что эффективность разработки БЗ способом П1 (с использованием UML-моделирования) может быть повышена в среднем на 60.3% по сравнению с П2 (и на 63.7% при использовании концепт-карт) и на 48.2% по сравнению с П3 (и на 55.3% при использовании концепт-карт) за счет автоматической кодогенерации на основе визуальных моделей, что, в свою очередь, позволяет:

- использовать результаты этапов концептуализации и формализации в форме диаграмм классов и концепт-карт, рассматривая последние не как статические графические артефакты, а как основу для формирования программного кода в соответствии с идеологией модельно-управляемого подхода (MDD/MDA);

- снизить риск ошибок проектирования за счет быстрого прототипирования БЗ, интепретации и кодогенерации;
- исключить ошибки программирования за счет автоматического отображения элементов концептуальной модели в языковые конструкции CLIPS или OWL.

В целом, результаты косвенной и прямой оценки эффективности позволяют сделать вывод о возможности увеличения скорости разработки БЗ ИС с помощью предлагаемых языков, методов и средств в некоторых случаях до 60%. Однако, данный показатель не учитывает временные затраты на изучение инструментария, а также сложность создаваемых баз знаний, которые в экспериментальном исследовании были достаточны просты.

Выводы

Помимо задач в области техногенной безопасности, апробация и оценка эффективности разработанных языков, методов и программных средств произведена при решении задач в следующих предметных областях (см Приложения Д-И): идентификации лицевых признаков (БЗ системы «EmSi-Interpreter»); обнаружения нежелательных сообщений (БЗ модуля «Детектор» (рег.№ 2020614257)); поддержки технического персонала при поиске и устранении неисправностей системы электроснабжения воздушного судна (БЗ ИС «АвиаТехПом»); анализа и прогнозирования риска (опасности) лесного пожара на основе информации о классе пожароопасности лесов, метеоусловий и других факторов. Были разработаны программные компоненты для трансформации концептуальных моделей в форме диаграмм классов UML, концепт-карт CmapTools (XTM) и ДС.

Оценка эффективности языков, методов и средств произведена на тестовых примерах (из учебного процесса ИрНИТУ) по временному критерию в сравнении с классическим методом, который не предусматривает синтез программного кода БЗ на основе трансформации концептуальных моделей. Решение учебных задач показало, что эффективность в отдельных случаях может быть повышена на 48-50% благодаря использованию концептуальных моделей, генераторов и интерпретаторов кодов.

Разработанные языки, методы и системы показали свою применимость для разработки интеллектуальных прикладных систем пользователями с низкими навыками программирования. Универсальность языков, методов и средств позволила решать задачи диагностики и прогнозирования в разных предметных областях. При этом ключевыми факторами являются: возможность визуального программирования, повтороное использование разработанных ранее концептуальных моделей, автоматическая кодогенерация, а также интерпретация результирующих кодов и спецификаций в специализированных системах-оболочках.

Практическая значимость результатов подтверждена полученными актами внедрения и справками использования программных систем АО «ИркутскНИИхиммаш», ООО «Смарт Технологии», ООО «ЦентраСиб», ИрНИТУ, МГТУ ГА.

Заключение

Важная научно-техническая проблема повышения эффективности создания интеллектуальных систем с декларативными базами знаний, в том числе конечными пользователями, не теряет свою актуальность. При ее решении в данной работе получены следующие новые результаты:

- метод проектирования декларативных баз знаний интеллектуальных систем, основанный на принципах модельно-ориентированного подхода и использующий новые модели, языки и платформы, ориентированные на конечных пользователей;
- визуальный язык программирования продукционных баз знаний – RVML (Rule Visual Modeling Language), основанный на UML и расширяющий его выразительные способности в контексте моделирования декларативных баз знаний;
- оригинальный текстовый декларативный язык описания трансформаций концептуальных моделей – TMRL (Transformation Model Representation Language), включающий конструкции для описания не только преобразуемых структур и связей между ними, но и механизма взаимодействия с внешними программными компонентами трансформаций;
- методы проектирования программ трансформаций концептуальных моделей и программных компонентов-конверторов концептуальных моделей, отличающиеся от известных использованием языка описания трансформаций моделей TMRL и принципов визуального программирования;
- программные средства, обеспечивающие поддержку разработанных языков и методов, объединенные общей идеологией модельных трансформаций и формирующие технологическую платформу создания интеллектуальных систем с декларативными базами знаний.

Разработанные языки, методы и средства обеспечивают повышение эффективности обработки данных и знаний на основе модельных трансформаций в контексте создания программного обеспечения систем ИИ конечными (непрограммирующими) пользователями. Определенный уровень универсальности разработанных элементов позволяет их применять при создании прикладных интеллектуальных систем для решения задач диагностики и прогнозирования в разных предметных областях.

Литература

1. Aamodt, A. Case-based reasoning: Foundational issues, methodological variations, and system approaches / A. Aamodt, E. Plaza // AI Communications. – 1994. – Vol. 7, № 1. – P. 39–59.

2. Abdullah, M.S. A UML profile for knowledge-based systems modelling / M.S. Abdullah, R. Paige, C. Kimble, I. Benest // Processing of the 5th ACIS International Conference on Software Engineering Research, Management & Applications. – 2007. – P. 871–878.

3. Alberts, R. An Integrated Method Using Conceptual Modelling to Generate an Ontology-based Query Mechanism / R. Alberts, E. Franconi // Proceedings of the OWL: Experiences and Directions Workshop 2012 (OWLED). – 2012. – Vol. 849.

4. An, Y. Constructing Complex Semantic Mappings between XML Data and Ontologies / Y. An, A. Borgida, J. Mylopoulos // Lecture Notes in Computer Science. – 2005. – Vol. 3729. – P. 6–20.

5. Arendt, T. Henshin: advanced concepts and tools for in-place EMF model transformations / T. Arendt, E. Biermann, S. Jurack, C. Krause, G. Taentzer // Lecture Notes in Computer Science. – 2010. – Vol. 6394. – P. 121–135.

6. Balasubramanian, D. The graph rewriting and transformation language: GReAT / D. Balasubramanian, A. Narayanan, C. Buskirk, G.Karsai // Electronic Communications of the EASST. – 2006. – Vol. 1. – P. 1–8.

7. Barricelli, B.R. End-user development, end-user programming and end-user software engineering: A systematic mapping study / B.R. Barricelli, F. Cassano, D. Fogli, A. Piccinno // Journal of Systems and Software. – 2019. – Vol.149. – P.101–137.

8. Bassiliades, N. R-DEVICE: an object-oriented knowledge base for RDF metadata / N. Bassiliades, I. Vlahavas // International Journal on Semantic Web and Information Systems. – 2006. – Vol. 2, № 2. – P. 24–90.

9. Baumeister, J. Knowledge-driven systems for episodic decision support / J. Baumeister, A. Striffler // Knowledge-Based Systems. – 2015. – Vol. 88. – P. 45–56.

10. Bedini, I. Transforming XML Schema to OWL Using Patterns / I. Bedini, C. Matheus, P.F. Patel-Schneider, A. Boran, B. Nguyen // Proceedings of the 2011 IEEE Fifth International Conference on Semantic Computing. – 2011. – P. 102–109.

11. Belghiat, A. An Approach based AToM3 for the Generation of OWL Ontologies from UML Diagrams / A. Belghiat, M. Bourahla // International Journal of Computer Applications. – 2012. – Vol. 41, № 3. – P. 41–48.

12. Berman, A.F. Intelligent planner for control of failures analysis of unique mechanical systems / A.F. Berman, O.A. Nikolaychuk, A.Yu. Yurin // Expert Systems with Applications. – 2010. – Vol. 37. – P. 7101–7107.

13. Berman, A.F. Intellectual data system for analyzing failures / A.F. Berman, O.A. Nikolaychuk, A.Yu. Yurin // Journal of Machinery Manufacture and Reliability. – 2012. – Vol. 41, №4. – P. 337–343.

14. Berman, A.F. A methodology for the investigation of the reliability and safety of unique technical systems / A.F. Berman, O.A. Nikolaychuk, A.Yu. Yurin, A.I. Pavlov // Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability. – 2014. – Vol. 228. – P.29–38.

15. Berman, A.F. Support of Decision-Making Based on a Production Approach in the Performance of an Industrial Safety Review / A.F. Berman, O.A. Nikolaichuk, A.Yu. Yurin, K.A. Kuznetsov // Chemical and Petroleum Engineering. – 2015. – Vol.50, № 1-2. – P.730–738.

16. Berman, A.F. A model-driven approach and a tool to support creation of rule-based expert systems for industrial safety expertise / A.F. Berman, M.A. Grishchenko, N.O. Dorodnykh, O.A. Nikolaychuk, A.Y. Yurin // Proceedings of the 12th International Forum on Knowledge Asset Dynamics (IFKAD-2017), Russia, St. Petersburg: Graduate School of Management of St. Petersburg University. – 2017. – P. 2034–2050.

17. Berman, A.F. Knowledge bases engineering on the basis of fault trees analysis / A.F. Berman, N.O. Dorodnykh, O.A. Nikolaychuk, A.Yu. Yurin // CEUR Workshop Proceedings. – 2018. – Vol. 2221. – P.25–31.

18. Berman, A.F. Application of case-based reasoning and multi-criteria decision-making methods for material selection in petrochemistry / A.F. Berman, G.S. Maltugueva, A.Yu. Yurin // Proceedings of the Institution of Mechanical Engineers, Part L: Journal of Materials: Design and Applications. – 2018. – Vol. 232, №3. – P. 204–212.

19. Berman, A.F. Event trees transformation for rule bases engineering / A.F. Berman, N.O. Dorodnykh, O.A. Nikolaychuk, A.Yu. Yurin // Proceedings of the 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). – 2019. – P. 1138-1143.

20. Berman, A.F. Computer-aided event tree synthesis on the basis of case-based reasoning / A.F. Berman, O.A. Nikolaychuk, A.Yu.Yurin // Advances in Intelligent Systems and Computing. – 2019. – Vol. 875. – P. 3–12.

21. Berman, A.F. The validation system for reliability and survivability of unique mechanical systems. / A.F. Berman, O.A. Nikolaichuk, A.Yu. Yurin // IOP Conference Series:

Materials Science and Engineering. – 2020. – Vol. 1061. – 012007.

22. Berman, A.F. A Module for Industrial Safety Inspection Planning Based on Self-organization / A.F Berman., O.A. Nikolaychuk, A.I. Pavlov, A.Y. Yurin // Lecture Notes in Artificial Intelligence. – 2021. – Vol.12948. – P. 365–379.

23. Bohring, H. Mapping XML to OWL Ontologies / H. Bohring, S. Auer // In: Jantke K, Fähnrich K, Wittig W. Marktplatz Internet: Von e-Learning bis e-Payment: Leipziger Informatik-Tage (LIT2005). – 2005. – P. 147–156.

24. Brambilla, M. Model Driven Software Engineering in Practice. / M. Brambilla, J. Cabot, M. Wimmer. Morgan & Claypool Publishers. – 2012.

25. Brilhante, V. Heuristic transformation of well-constructed conceptual maps into OWL preliminary domain ontologies / V. Brilhante, G.T. Macedo, S.F. Macedo // Proceedings of the Second Workshop on Ontologies and their Applications, CEUR-WS. – 2006.

26. Brockmans, S. A Model-Driven Approach for Building OWL DL and OWL Full Ontologies / S. Brockmans, R.M. Colomb, P. Haase, E.F. Kendall, E.K. Wallace, C. Welty, G.T. Xie // Lecture Notes in Computer Science. – 2006. – Vol. 4273. – P. 187–200.

27. Bychkov, I.V. A method and tools for prototyping components of intelligent systems based on transformations / I.V. Bychkov, A.Yu. Yurin // Journal of Physics: Conference Series. – 2021. – Vol.1864. – 012042.

28. Bychkov, I.V. Organization of digital monitoring of the Baikal natural territory / I.V. Bychkov, G.M. Ruzhnikov, R.K. Fedorov, A.E. Khmelnov, A.K. Popova // IOP Conference Series: Earth and Environmental Science. – 2021. – Vol.629, №1. – 012067.

29. Cabello, M.E. Baseline-oriented modeling: An MDA approach based on software product lines for the expert systems development / M.E. Cabello, I. Ramos, A. Gomez, R. Limon // Proceedings of the 1st Asian Conference on Intelligent Information and Database Systems, Apr. 1-3, IEEE Xplore Press, Dong Hoi, Vietnam. – 2009. – P. 208–213.

30. CAKE – Computer Aided Knowledge Engineering Technique [Электронный ресурс]. – Режим доступа: https://www.ida.liu.se/divisions/aiics/publications/ECAI-2002-CAKE-Computer-Aided.pdf (дата обращения: 18.03.2022).

31. Canadas, J. InSCo-Gen: A MDD Tool for Web Rule-Based Applications / J. Canadas, J. Palma, S.Tunez // Lecture Notes in Computer Science. – 2009. – Vol. 5648. – P. 523–526.

32. Chaur, G.W. Modeling Rule-Based Systems with EMF. Eclipse Corner Articles [Электронный ресурс]. – Режим доступа: http://www.eclipse.org/articles/Article-Rule%20Modeling%20With%20EMF/article.html (дата обращения: 18.03.2022).

33. CLIPS: A Tool for Building Expert Systems [Электронный ресурс]. – Режим доступа: http://clipsrules.sourceforge.net/ (дата обращения: 18.03.2022).

34. ClipsWin: CLIPS Rule Based Programming Language [Электронный ресурс]. – Режим доступа: https://sourceforge.net/p/clipsrules/news/2008/01/clipswin-6241/ (дата обращения: 18.03.2022).

35. Coronado, E. Visual Programming Environments for End-User Development of Intelligent and Social Robots, a Systematic Review / E. Coronado, F. Mastrogiovanni, B. Indurkhya, G. Venture // Journal of Computer Languages. – 2020. – Vol. 58. – 100970.

36. Corsar, D. Developing Knowledge-Based Systems using the Semantic Web / D. Corsar, D.H Sleeman // Proceedings of the 2008 International Conference on Visions of Computer Science: BCS International Academic Conference. – 2008. – P. 29–40.

37. Cranefield, S. Bridging the gap between the model-driven architecture and ontology engineering / S. Cranefield, J. Pan // International Journal of Human-Computer Studies. – 2007. – Vol. 65, №7. – P. 595–609.

38. Cretu, L.G. Model-Driven Engineering of Information Systems: Principles, Techniques, and Practice. / L.G. Cretu, D. Florin. Apple Academic Press. – 2014.

39. Czarnecki, K. Overview of generative software development / K. Czarnecki // Unconventional Programming Paradigms. – 2005. – P. 326–341.

40. Czarnecki, K. Feature-based survey of model transformation approaches / K. Czarnecki, S. Helsen // IBM Systems Journal. – 2006. – Vol. 45, №3. – P. 621–645.

41. Da Silva, A.R. Model-driven engineering: A survey supported by the unified conceptual model / A.R. da Silva // Computer Languages, Systems & Structures. – 2015. – Vol. 43. – P. 139–155.

42. Djurić, D. The Tao of Modeling Spaces / D. Djurić, D. Gašević, V. Devedžić // Journal of Object Technology. – 2006. – Vol. 5, №8. – P. 125–147.

43. Djurić, D. Ontology modeling and MDA / D. Djurić, D. Gašević, V. Devedžić // Journal of Object technology. – 2005. – Vol. 4, №1. – P. 109–128.

44. Dorodnykh, N.O. About the specialization of model-driven approach for creation of case-based intelligence decision support systems / N.O. Dorodnykh, A.Yu. Yurin // Открытые семантические технологии проектирования интеллектуальных систем. – 2017. – №7. – С. 151–154.

45. Dorodnykh, N.O. Ontology Driven Development of Rule-Based Expert Systems / N.O Dorodnykh., A.Y. Yurin, A.B. Stolbov // Proceedings of the 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC). – 2018. – P. 1-6.

46. Dorodnykh, N.O. A domain-specific language for transformation models / N.O. Dorodnykh, A.Y. Yurin // CEUR Workshop Proceedings. – 2018. – Vol. 2221. – P. 70–75.

47. Dorodnykh, N.O. Towards Ontology Engineering Based on Transformation of Conceptual Models and Spreadsheet Data: A Case Study / N.O. Dorodnykh, A.Yu. Yurin // Advances in Intelligent Systems and Computing. – 2019. – Vol. 1046. – P. 233–247.

48. Dorodnykh, N.O. Conceptual Model Engineering for Industrial Safety Inspection Based on Spreadsheet Data Analysis / N.O. Dorodnykh, A.Yu. Yurin, A.O. Shigarov // Communications in Computer and Information Science. – 2020. – Vol. 1126. – P. 51–65.

49. Dorodnykh, N.O. A Transformation-Based Approach for Fuzzy Knowledge Bases Engineering / N.O. Dorodnykh, O.A. Nikolaychuk, A.Yu. Yurin // Studies in Systems, Decision and Control. – 2021. – Vol. 337. – P. 76–90.

50. Dorodnykh, N.O. Towards a universal approach for semantic interpretation of spreadsheets data / N.O. Dorodnykh, A.Yu. Yurin // IDEAS'20: Proceedings of the 24th Symposium on International Database Engineering & Applications. – 2020. – Vol. 22. – P. 1–9.

51. Dorodnykh, N.O. PKBD.Onto: A Plugin for Ontological Schemas Generation / N.O. Dorodnykh, A.Yu. Yurin, A.V. Vidiya // CEUR Workshop Proceedings. – 2020. – Vol. 2677. – P. 84–94.

52. Dorodnykh, N.O. An RVML extension for modeling fuzzy rule bases / N.O. Dorodnykh, A.Yu. Yurin // CEUR Workshop Proceedings. – 2021. – Vol. 2858. – P. 34–45.

53. Dorodnykh, N.O. End-user development of knowledge bases for semi-automated formation of task cards / N.O. Dorodnykh, Y.V. Kotlov, O.A. Nikolaychuk, V.M. Popov, A.Yu. Yurin // CEUR Workshop Proceedings. – 2021. – Vol. 2913. – P. 60–73.

54. Dorodnykh, N.O. Spreadsheet Data Transformation for Ontology Engineering in Petrochemical Equipment Inspection Tasks / N.O. Dorodnykh, A.Yu.Yurin // Lecture Notes in Networks and Systems. – 2021. – Vol 330. – P. 562–571.

55. Dorodnykh, N.O. Using UML class diagrams for content ontology design patterns engineering / N.O. Dorodnykh, O.A. Nikolaychuk, A.Yu. Yurin // Journal of Physics: Conference Series. – 2021. – Vol. 1801. – 012026.

56. Drools [Электронный ресурс]. – Режим доступа: https://drools.org/ (дата обращения: 18.03.2022).

57. Dubois, D. Readings in Fuzzy Sets for Intelligent Systems / D. Dubois, H. Prade, R.Yager. Amsterdam: Elsevier. – 2014. – 928 p.

58. Dunstan, N. Generating domain-specific web-based expert systems / N. Dunstan // Expert Systems with Applications. – 2008. – Vol. 35, №3. – P. 686–690.

59. Eclipse Modeling Framework (EMF) [Электронный ресурс]. – Режим доступа: http://www.eclipse.org/modeling/emf/ (дата обращения: 18.03.2022).

60. Ecore [Электронный ресурс]. – Режим доступа: http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html (дата обращения: 18.03.2022).

61. Eichhoff, J.R. RuleML 1.02: Deliberation, Reaction and Consumer Families / J.R. Eichhoff, D. Roller // Proceedings of the RuleML 2015 Challenge, the Special Track on Rule-based Recommender Systems for the Web of Data, the Special Industry Track and the RuleML 2015 Doctoral Consortium hosted by the 9th International Web Rule Symposium (RuleML 2015) http://ceur-ws.org/Vol-1417/paper6.pdf

62. Epsilon [Электронный ресурс]. – Режим доступа: http://www.eclipse.org/epsilon/ (дата обращения: 18.03.2022).

63. Eriksson, H. The JESSTAB approach to Protégé and JESS integration / H. Eriksson // Proceedings of the IFIP 17th World Computer Congress – TC12 Stream on Intelligent Information Processing. – 2002. – P. 237–248.

64. ES-Builder [Электронный ресурс]. – Режим доступа: http://www.mcgoo.com.au/esbuilder/index.php (дата обращения: 18.03.2022).

65. Expert System Designer [Электронный ресурс]. – Режим доступа: https://www.winsite.com/Utilities/Miscellaneous/Expert-System-Designer/ (дата обращения: 18.03.2022).

66. Exsys Corvid: Expert System Development Tool [Электронный ресурс]. – Режим доступа: http://www.exsys.com/exsyscorvid.html (дата обращения: 18.03.2022).

67. Felfernig, A. UML as domain specific language for the construction of knowledge-based configuration systems / A. Felfernig, G. E. Friedrich, D. Jannach // International Journal of Software Engineering and Knowledge Engineering. – 2000. – Vol. 10, №4. – P. 449–469.

68. Felfernig, A. Configuration knowledge representation using UML/OCL / A. Felfernig, G. Friedrich, D. Jannach, M.Zanker // Proceedings of the International Conference on the Unified Modeling Language. – 2002. – P. 49–62.

69. Frankel, D. Model Driven Architecture: Applying MDA to Enterprise Computing. / D. Frankel. New York: Wiley. – 2003.

70. Gascueña, J.M. Model-to-model and model-to-text: looking for the automation of VigilAgent / J.M. Gascueña, E. Navarro, A. Fernández-Caballero, R. Martínez-Tomás // Expert Systems. – 2014. – Vol. 31, №3. – P. 199–212.

71. Gašević, D. Converting UML to OWL ontologies / D. Gašević, D. Djurić, V. Devedžić, V. Damjanović // Proceedings of the 13th international World Wide Web conference. – 2004. – P. 488–489.

72. Gašević, D. Model driven engineering and ontology development (2nd ed.) / D. Gašević, D. Djurić, V. Devedžić. New York: Springer-Verlag. – 2009.

73. Graphical Modeling Framework (GMF) Tooling – Eclipse [Электронный ресурс]. – Режим доступа: http://www.eclipse.org/gmf-tooling/ (дата обращения: 18.03.2022).

74. Golenkov, V. Principles of organization and automation of the semantic computer systems development / V. Golenkov, D. Shunkevich, I. Davydenko, N. Grakova // Open Semantic Technologies for Intelligent Systems. – 2019. – 53–90

75. Greenfield, J. Software factories: assembling applications with patterns, models, frameworks, and tools / J. Greenfield, K. Short, S. Cook, S. Kent, J. Crupi. Wiley Publishing. – 2004.

76. Gribova, V. The methods and the IACPaaS Platform tools for semantic representation of knowledge and development of declarative components for intelligent systems / V. Gribova, A. Kleschev, P. Moskalenko, V. Timchenko, L. Fedorischev, E. Shalfeeva // Open Semantic Technologies for Intelligent Systems. – 2019. – 21–24

77. Grissa-Touzi, A. VISUAL JESS: An expandable visual generator of oriented object expert systems / A. Grissa-Touzi, H. Ounally, A. Boulila // International Journal of Computer and Information Engineering. – 2007. – Vol.1(11). – P. 1668–1671.

78. Gruber, T.R. A translation approach to portable ontologies / T.R. Gruber // Knowledge Acquisition. – 1993. – Vol. 5, №2. – P. 199–220.

79. Guarino, N. Formal Ontology in Information Systems / N. Guarino // Proceedings of the first international conference (FOIS'98). – 1998. – Vol. 46. – P. 3–15.

80. Hatzilygeroudis, I. A Tool for Automatic Creation of Rule-Based Expert Systems with CFs / I. Hatzilygeroudis, K. Kovas // Processing of the International Conference on Artificial Intelligence Applications and Innovations (AIAI) / Advances in Information and Communication Technology. – 2010. – Vol. 339. – P. 195–202.

81. Hung, V. Spreadsheet-based complex data transformation / V. Hung, B. Benatallah, R. Saint-Paul // Proc. 20th ACM Int. Conf. Inf. and Know. Management. Glasgow Scotland, UK. – 2011. P. 1749–1754.

82. IBM WebSphere ILOG JRules [Электронный ресурс]. – Режим доступа: https://www.ibm.com/support/knowledgecenter/ru/SSZJPZ_11.7.0/

com.ibm.swg.im.iis.conn.jrules.use.doc/topics/ilog_jrules_container.html (дата обращения: 18.03.2022).

83. JESS: The Rule Engine for the Java Platform [Электронный ресурс]. – Режим доступа: http://www.jessrules.com/ (дата обращения: 18.03.2022).

84. Jouault, F. KM3: a DSL for Metamodel Specification / F. Jouault, J. Bézivin // Lecture Notes in Computer Science. – 2006. – Vol. 4037. – P. 171–185.

85. Jouault, F. ATL: A model transformation tool / F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev // Science of Computer Programming. – 2008. – Vol. 72, №1. – P. 31–39.

86. Kadhim, M.A. Design and implementation of Intelligent Agent and Diagnosis Domain Tool for Rule-based Expert System / M.A. Kadhim, M.A. Alam, H. Kaur // Processing of the International Conference on Machine Intelligence Research and Advancement. – 2013. – P. 619–622.

87. Kang, K.C. Feature-Oriented Domain Analysis (FODA) feasibility study / K.C. Kang, S.G. Cohen, J.A. Hess, W.E. Novak, A.S. Peterson // Technical Report CMU/SEI-90-TR-21 ESD-90-TR-222. – 1990.

88. Knowledge Base Development System (KBDS). URL: http://kbds.knowledge-core.ru/ (дата обращения: 18.03.2022).

89. Kelly, S. Domain-Specific Modeling: Enabling Full Code Generation / S. Kelly, J.-P.Tolvanen. Wiley-IEEE Computer Society. – 2008.

90. Khan, A. Introducing Design Patterns in XML Schemas / A. Khan, M. Sum [Электронный ресурс]. – Режим доступа: http://www.oracle.com/technetwork/java/design-patterns-142138.html (дата обращения: 18.03.2022).

91. Kiko, K. A Detailed Comparison of UML and OWL / K. Kiko, C. Atkinson. Germany: University of Mannheim. – 2008.

92. Kleppe, A. MDA Explained: The Model-Driven Architecture: Practice and Promise (1st ed.) / A. Kleppe, J. Warmer, W. Bast. Addison-Wesley Professional. – 2003.

93. The Knowledge Core [Электронный ресурс]. – Режим доступа: http://knowledge-core.ru (дата обращения: 18.03.2022).

94. Kosko, B. Fuzzy Cognitive Maps / B. Kosko // International Journal of Man-Machine Studies. – 1986. – Vol.24. – P.65–75.

95. Kotlov, Yu.V. Towards designing knowledge bases for aircraft malfunctions diagnostics based on model trans-formations / Yu.V. Kotlov, V.M. Popov, A.Yu. Yurin // Journal of Physics: Conference Series. – 2021. – Vol.2060. – 012016.

96. Laera, L. SweetProlog: A system to integrate ontologies and rules / L. Laera, V.

Tamma, T. Bench-Capon, G. Semeraro // Lecture Notes in Computer Science. – 2004. – Vol. 3323. – P. 188–193.

97. Lehmberg, O. A large public corpus of web tables containing time and context metadata / O. Lehmberg, D. Ritze, R. Meusel, C. Bizer // Proc. of the 25th International Conference Companion on World Wide Web. – 2016. – P. 75–76.

98. Levenshtein, V.I. Binary codes capable of correcting deletions, insertions, and reversals / V.I. Levenshtein // Tech. Rep. 8, Soviet Physics Doklady. – 1966.

99. Lukichev, S. UML-based Rule Modeling with Fujaba / S. Lukichev, G. Wagner[Электронный ресурс]. – Режим доступа: http://www.rewerse.net/I1/oxygen.informatik.tu-cottbus.de/rewerse-i1/default.htm (дата обращения: 18.03.2022).

100. Lukichev, S. Using UML-based rules for web services modeling / S. Lukichev, A. Giurca, G. Wagner, D. Gasevic, M. Ribaric // Proc. the Second International Workshop on Services Engineering. – 2007. – P. 290–297.

101. Maltugueva, G.S. Case-based reasoning for the multi-method decision making / G.S. Maltugueva, A.Y.Yurin // CEUR Workshop Proceedings. – 2018. – Vol. 2221. – P. 32-36.

102. Maltugueva, G.S. Improving case-based reasoning with the aid of multi-criteria and group decision-making methods / G.S. Maltugueva, A.Yu. Yurin // Proceedings of the 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). – 2019. – P. 1031-1036.

103. MDA Specifications. URL: http://www.omg.org/mda/specs.htm (дата обращения: 18.03.2022).

104. Meditskos, G. CLIPS-OWL: A framework for providing object-oriented extensional ontology queries in a production rule engine / G. Meditskos, N. Bassiliades // Data & Knowledge Engineering. – 2011. – Vol. 70. – P. 661–681.

105. Mehrolhassani, M. Developing Ontology Based Applications of Semantic Web Using UML to OWL Conversion / M. Mehrolhassani, A. ELÇİ // Communications in Computer and Information Science. – 2008. – Vol. 19. – P. 566–577.

106. Mei, J. OWL2Jess: A transformational implementation of the OWL semantics / J. Mei, E.P. Bontas, Z. Lin // Lecture Notes in Computer Science. – 2005. – Vol. 3759. – P. 599–608.

107. Mens, T. A Taxonomy of Model Transformations / T. Mens, P.V. Gorp // Electronic Notes in Theoretical Computer Science. – 2006. – Vol. 152. – P. 125–142.

108. Model-Integrated Computing (MIC). URL:

http://www.isis.vanderbilt.edu/research/MIC (дата обращения: 18.03.2022).

109. Miguel, M. Practical Experiences in the Application of MDA / M. Miguel, J. Jourdan, S. Salicki // Lecture Notes in Computer Science. – 2002. – Vol. 2460. – P. 128–139.

110. Milanović, M. On interchanging between OWL/SWRL and UML/OCL / M.Milanović, D.Gašević, A.Giurca, G.Wagner, V.Devedžić // Proceedings of the 6th Workshop on OCL for (Meta-) Models in Multiple Application Domains (OCLApps) at the 9th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS). – 2006. – P. 81–95.

111. Milanović, M. Bridging concrete and abstract syntaxes in model-driven engineering: a case of rule languages / M. Milanović, D.Gašević, A.Giurca, G.Wagner, V.Devedžić // Software: Practice and Experience. – 2009. – Vol. 39, №16. – P. 1313–1346.

112. Meta Object Facility (MOF) Core. OMG Formally Released Versions of MOF. URL: http://www.omg.org/spec/MOF/ (дата обращения: 18.03.2022).

113. Myroshnichenko, I. Mapping ER Schemas to OWL Ontologies / I. Myroshnichenko, M.C. Murphy // Proceedings of the 2009 IEEE International Conference on Semantic Computing. – 2009. – P. 324–329.

114. Na, H.-S. A method for building domain ontologies based on the transformation of UML models / H.-S. Na, O-H. Choi, J.-E. Lim // Proceedings of 4th International Conference on Software Engineering Research, Management and Applications (SERA 2006). – 2006. – P. 332–338.

115. Nalepa, G.J. UML representation for rule-based application models with XTT2-based business rules / G.J. Nalepa, K. Kluza // International Journal of Software Engineering and Knowledge Engineering. – 2012. – Vol. 22, №4. – P. 485–524.

116. Nikolaychuk, O.A. Computer-aided identification of mechanical system's technical state with the aid of case-based reasoning / O.A. Nikolaychuk, A.Y. Yurin // Expert Systems with Applications. – 2008. – Vol. 34. – P. 635–642.

117. Nofal, M. Developing Web-Based Semantic Expert Systems / M. Nofal, K.M. Fouad // International Journal of Computer Science. – 2014. – Vol. 11, №1. – P. 103–110.

118. O'Connor, M.J. Developing a Web-Based Application using OWL and SWRL / M.J. O'Connor, R.D. Shankar, C. Nyulas, S.W. Tu, A. Das // Proceedings of the AI Meets Business Rules and Process Management, AAAI Spring Symposium. – 2008.

119. O'Connor, M.J. Acquiring OWL Ontologies from XML Documents / M.J. O'Connor, A.K. Das // Proceedings of the 6th International Conference on Knowledge Capture (K-CAP'11). – 2011. – P. 17–24.

120. Object Constraint Language (OCL) Version 2.4 URL: http://www.omg.org/spec/OCL/2.4/ (дата обращения: 18.03.2022).

121. OWL 2 Web Ontology Language Conformance (Second Edition) [Электронный ресурс]. – Режим доступа: https://www.w3.org/TR/2012/REC-owl2-conformance-20121211/ (дата обращения: 18.03.2022).

122. Parreiras, F.S. Using ontologies with UML class-based modeling: The TwoUse approach . F.S. Parreiras, S. Staab // Data & Knowledge Engineering. – 2010. – Vol. 69, №11. – P. 1194–1207.

123. Partsch, H. Program Transformation Systems / H. Partsch, R. Steinbruggen // ACM Computing Surveys. – 1983. – Vol. 15, № 3. – P. 199–236.

124. Paschke, A. A Representational Analysis of the API4KP Metamodel / A. Paschke, T. Athan, D. Sottara, E. Kendall, R. Bell // Lecture Notes in Business Information Processing. – 2015. – Vol. 225.

125. Plaza, E. Constructive Adaptation / E. Plaza, J.L. Arcos // Lecture Notes on Artificial Intelligence. – 2002. – Vol.2416. – P.306–320.

126. Pop, D. Expert system creator: A visual tool for expert systems construction / D. Pop, V. Negru, C. Sandru // Proceedings of the 24th International Conference on Information Technology Interfaces. – 2002. – P. 217–222.

127. Pourghasemi, H.R. Application of learning vector quantization and different machine learning techniques to assessing forest fire influence factors and spatial modeling / H.R. Pourghasemi, A. Gayen, R. Lasaponara, J.P. Tiefenbacher // Environ Res. – 2020. – Vol.184. – 109321.

128. Query/View/Transformation (QVT) Version 1.3 [Электронный ресурс]. – Режим доступа: http://www.omg.org/spec/QVT/1.3/ (дата обращения: 18.03.2022).

129. Ramos, L. Semantic Web for manufacturing, trends and open issues: Toward a state of the art / L. Ramos // Computers & Industrial Engineering. – 2015. – Vol. 90. – P. 444–460.

130. RDF/XML Syntax Specification (Revised) [Электронный ресурс]. – Режим доступа: https://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/ (дата обращения: 18.03.2022).

131. RELAX NG [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/RELAX_NG (дата обращения: 18.03.2022).

132. Reynares, E. A set of ontology design patterns for reengineering SBVR statements into OWL/SWRL ontologies / E. Reynares, M. L. Caliusco, M.R. Galli // Expert

Systems with Applications. – 2015. – Vol. 42, №5. – P. 2680–2690.

133. Riesbeck, C.K. Inside Case-based Reasoning. / C.K. Riesbeck, R. Schank. Erlbaum. Northvale, NJ. – 1989.

134. Rodrigues, T. Mapping XML to exiting OWL ontologies / T. Rodrigues, P. Rosa, J. Cardoso // Proceedings of the International Conference WWW/Internet. – 2006. – P. 72–77.

135. Rodrigues, T. Moving from syntactic to semantic organizations using JXML2OWL / T. Rodrigues, P. Rosa, J. Cardoso // Computers in Industry. – 2008. – Vol. 59, №8. – P. 808–819.

136. Ruiz-Mezcua, B. An expert system development tool for non AI experts / B. Ruiz-Mezcua, A. Garcia-Crespo, J.L. Lopez-Cuadrado, I. Gonzalez-Carrasco // Expert Systems with Applications. – 2011. – Vol. 38, № 1. – P. 597–609.

137. Specification of RuleML 1.03 [Электронный ресурс]. – Режим доступа: http://wiki.ruleml.org/index.php/Specification_of_RuleML_1.03 (дата обращения: 18.03.2022).

138. Rybina, G. The use of temporal inferences in dynamic integrated expert systems / G. Rybina, A. Mozgachev // Scientific and Technical Information Processing. – 2014. – Vol. 6. – P. 390–399.

139. Rybina, G.V. Dynamic integrated expert systems: automated construction features of temporal knowledge bases with using problem-oriented methodology / G.V. Rybina, I.A. Sorokin, D.O. Sorokin // Open Semantic Technology for Intelligent Systems. – 2019. – P.129–132

140. Sami, B. Model-Driven Software Development / B. Sami, M. Book, V.Gruhn. Springer. – 2005.

141. San, O.M. An alternative extension of the k-means algorithm for clustering categorical data / O.M. San, V. Huynh, Y. Nakamori // Applied mathematics and computer science. – 2004. – Vol. 14, №2. – P.241–247.

142. Santos, M. Characterizing end-user development solutions: A systematic literature review / M. Santos, M.L.B. Villela // International Conference on Human-Computer Interaction. – 2019. – P. 194–209.

143. SBVR: Semantics Of Business Vocabulary And Business Rules [Электронный ресурс]. – Режим доступа: https://www.omg.org/spec/SBVR/About-SBVR/ (дата обращения: 18.03.2022).

144. Schmidt, D.C. Model-Driven Engineering / D.C. Schmidt // IEEE Computer. – 2006. – Vol. 39, №2. – P. 25–31.

145. Schreiber, G. Knowledge Engineering and Management. The CommonKADS

methodology / G. Schreiber, H. Akkermans, A. Anjewierden, de R. Hoog, N.R. Shadbolt, W.V. Velde, B. Wielinga. MIT Press, Cambridge, MA. – 2000.

146. Schreiber, G. CML: The commonKADS conceptual modelling language / G. Schreiber, B. Wielinga, H.Akkermans, W. Van de Velde, A. Anjewierden // Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence). – 1994. – Vol. 867.

147. Sendall, S. Model Transformation: The Heart and Soul of Model-Driven Software Development / S. Sendall, W. Kozaczynski // IEEE Software. – 2003. – Vol. 20, № 5. – P. 42–45.

148. Shigarov, A. TabbyXL: Software platform for rule-based spreadsheet data extraction and transformation / A. Shigarov, V. Khristyuk, A. Mikhailov // SoftwareX. – 2019. – Vol. 10. – 100270.

149. Shue, L. The development of an ontology-based expert system for corporate financial rating / L. Shue, C. Chen, W.Shiue // Expert Systems with Applications. – 2009. – Vol. 36, №2. – P. 2130–2142.

150. Sicilia, M.A. Integrating fuzziness in object oriented modeling language: towards a fuzzy-UML / M.A.Sicilia, E.Garcia, J.A. Gutierrez // Proceedings of International Conference on Fuzzy Sets Theory and its Applications. – 2002. – P. 66–67.

151. Simón, A. Generation of OWL Ontologies from Concept Maps in Shallow Domains / A. Simón, L. Ceccaroni, A. Rosete // Lecture Notes in Computer Science. – 2007. – Vol. 4788. – P. 259–267.

152. Starr, R.R. Concept maps as the first step in an ontology construction method / R.R. Starr, J.M. Parente de Oliveira // Information Systems. – 2013. – Vol. 38. – P. 771–783.

153. Stokes, M. Managing engineering knowledge: MOKA: methodology for knowledge based engineering applications (6th ed.) / M. Stokes. New York: ASME Press. – 2001.

154. Surakratanasakul, B. Lightweight CommonKADS / B. Surakratanasakul // 9th International Conference on Information Technology and Electrical Engineering (ICITEE). – 2017. – 1-5

155. SweetRules [Электронный ресурс]. – Режим доступа: http://sweetrules.projects.semwebcentral.org/ (дата обращения: 18.03.2022).

156. Semantic Web Rule Language [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Semantic_Web_Rule_Language (дата обращения: 18.03.2022).

157. Thi Thu Thuy, P. DTD2OWL: Automatic Transforming XML Documents into OWL Ontology / P. Thi Thu Thuy, Y.K. Lee, S.Y.Lee // Proceedings of the 2nd International

Conference on Interaction Sciences: Information Technology, Culture and Human. – 2009. – P. 125–131.

158. Tijerino, Y.A. Towards Ontology Generation from Tables / Y. A. Tijerino, D. W. Embley, D.W. Lonsdale, Y. Ding, G. Nagy // World Wide Web: Internet and Web Information Systems. – 2005. – Vol. 8, №8. – P. 261–285.

159. Tolvanen, J.-P. Model-Driven Development Challenges and Solutions – Experiences with Domain-Specific Modelling in Industry / J.-P. Tolvanen, S.Kelly // Processing of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016). – 2016. – P. 711–719.

160. Touzi, A. New Approach for Conception and Implementation of Object Oriented Expert System Using UML / A. Touzi, M.B. Messaoud // The International Arab Journal of Information Technology. – 2009. – Vol. 6, №1. – P. 99–106.

161. Truyen, F. The Fast Guide to Model Driven Architecture: The Basics of Model Driven Architecture / F.Truyen. Cephas Consulting Corp. – 2006.

162. Unified Modeling Language (UML) Version 2.5 [Электронный ресурс]. – Режим доступа: http://www.omg.org/spec/UML/2.5/ (дата обращения: 18.03.2022).

163. van Assem, M. A Method for Converting Thesauri to RDF/OWL / M. van Assem, M.R.Menken, G.Schreiber, J.Wielemaker, B.Wielinga // Lecture Notes in Computer Science. – 2004. – Vol. 3298. – P. 17–31.

164. Varro, D. The model transformation language of the VIATRA2 framework / D.Varro, A.Balogh // Science of Computer Programming. – 2007. – Vol. 63, №3. – P. 214–234.

165. Verhagen, W.J.C. A critical review of Knowledge-Based Engineering: An identification of research challenges / W.J.C. Verhagen, P. Bermell-Garcia, R.E.C. van Dijk, R. Curran // Advanced Engineering Informatics. – 2012. – Vol.26. – P.5-15.

166. VisiRule. Logic Programming Associates [Электронный ресурс]. – Режим доступа: http://www.lpa.co.uk/ind_hom.htm (дата обращения: 18.03.2022).

167. Visual Rules BRM [Электронный ресурс]. – Режим доступа: https://www.bosch-si.com/bpm-and-brm/visual-rules/business-rules-management.html (дата обращения: 18.03.2022).

168. Volter, M. Model-Driven Software Development: Technology, Engineering, Management / M.Volter, T.Stahl, J.Bettin, A.Haase, S.Helsen, K.Czarnecki. John Wiley & Sons. – 2006.

169. Visual Prolog [Электронный ресурс]. – Режим доступа: https://www.visual-prolog.com/ (дата обращения: 18.03.2022).

170. W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures [Электронный ресурс]. – Режим доступа: https://www.w3.org/TR/xmlschema11-1/ (дата обращения: 18.03.2022).

171. Wagner, W.P. Trends in expert system development: A longitudinal content analysis of over thirty years of expert system case studies / W.P. Wagner // Expert Systems with Applications. – 2017. – Vol. 76. – P. 85–96.

172. Wang, E. A Teaching Strategies Engine Using Translation from SWRL to Jess / E.Wang, Y.S. Kim // Lecture Notes in Computer Science. – 2006. – Vol. 4053. – P. 51–60.

173. Wang, W. Development of a rule-based diagnostic platform on an object-oriented expert system shell / W. Wang, M. Yang, P.H. Seong // Annals of Nuclear Energy. – 2016. – Vol.88. – P. 252-264.

174. Wang, F. Knowledge representation using non-parametric Bayesian networks for tunneling risk analysis / F. Wang, H.Li, C. Dong, L. Ding // Reliability Engineering & System Safety. – 2019. – Vol.191. – 106529.

175. Watson, I. Case-based reasoning is a methodology not a technology / I. Watson // Knowlidge-based systems. – 1999. – Vol. 12. – P. 303–308.

176. XML Metadata Interchange (XMI) Version 2.5.1 [Электронный ресурс]. – Режим доступа: http://www.omg.org/spec/XMI/2.5.1/ (дата обращения: 18.03.2022).

177. Extensible Markup Language (XML) [Электронный ресурс]. – Режим доступа: https://www.w3.org/XML/ (дата обращения: 18.03.2022).

178. XSL Transformations (XSLT) Version 2.0 [Электронный ресурс]. – Режим доступа: http://www.w3.org/TR/xslt20 (дата обращения: 18.03.2022).

179. XML Topic Maps (XTM) 1.0. 2001 [Электронный ресурс]. – Режим доступа: http://www.topicmaps.org/xtm/ (дата обращения: 18.03.2022).

180. Xu, Z. Automatic extraction of OWL ontologies from UML class diagrams: a semantics-preserving approach / Z.Xu, Y.Ni, W.He, L.Lin, Q.Yan // World Wide Web. – 2012. – Vol. 15, №5. – P. 517–545.

181. Yahia, N. Automatic Generation of OWL Ontology from XML Data Source / N.Yahia, S.A. Mokhtar, A.Ahmed // International Journal of Computer Science Issues. – 2012. – Vol. 9, №2. – P. 1–7.

182. Yurin, A.Yu. An approach for definition of recommendations for prevention of repeated failures with the aid of case-based reasoning and group decision-making methods / A.Yu.Yurin // Expert Systems with Applications. – 2012. – Vol. 39. – P. 9282–9287.

183. Yurin, A.Yu. Group Decision Making Methods for Adapting Solutions Derived

from Case-Based Reasoning / A.Yu. Yurin // Scientific and Technical Information Processing. – 2015. – Vol.42, №5. – P. 375–381.

184. Yurin, A.Yu. Designing rule-based expert systems with the aid of the model-driven development approach / A.Yu.Yurin, N.O.Dorodnykh, O.A.Nikolaychuk, M.A.Grishenko // Expert Systems. – 2018. – Vol. 35, №5. – 12291.

185. Yurin, A.Yu. Prototyping Rule-Based Expert Systems with the Aid of Model Transformations / A.Yu.Yurin, N.O.Dorodnykh, O.A.Nikolaychuk, M.A.Grishenko // Journal of Computer Science. – 2018. – Vol. 14, №5. – P. 680–698.

186. Yurin, A.Yu. The domain-specific editor for rule-based knowledge bases / A.Yu.Yurin, A.F.Berman, O.A.Nikolaychuk, N.O.Dorodnykh, M.A.Grishenko // Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). – 2018. – P. 1130–1135.

187. Yurin, A.Yu. Fishbone diagrams for the development of knowledge bases / A.Yu.Yurin, A.F.Berman, N.O.Dorodnykh, O.A.Nikolaychuk, N.Yu.Pavlov // Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). – 2018. – P. 1136-1141.

188. Yurin, A.Yu. Knowledge Base Engineering for Industrial Safety Expertise: A Model-Driven Development Approach / A.Yu.Yurin, A.F.Berman, O.A.Nikolaychuk, N.O.Dorodnykh // Studies in Systems, Decision and Control. – 2019. – Vol. 199. – P. 112-124.

189. Yurin, A.Yu. ISI-161: Spreadsheet tables / A.Yu. Yurin, A.O. Shigarov, N.O. Dorodnykh, V.V. Khristyuk // Mendeley Data, v1, 2019 [Электронный ресурс]. – Режим доступа: http://dx.doi.org/10.17632/8zdymg4y96.1 (дата обращения: 18.03.2022).

190. Yurin, A.Yu. ISI models / A.Yu. Yurin, N.O. Dorodnykh, O.A. Nikolaychuk, A.F. Berman, A.F. Pavlov // Mendeley Data, v1, 2019 [Электронный ресурс]. – Режим доступа: http://dx.doi.org/10.17632/f9h2t766tk.1 (дата обращения: 18.03.2022).

191. Yurin, A.Yu. Application of a Case-Based Approach for Tasks of Industrial Safety Inspection / A.Yu.Yurin, A.F.Berman, O.A.Nikolaychuk, K.A.Kuznetsov // CEUR Workshop Proceedings. – 2019. – Vol. 2463. – P. 32–39.

192. Yurin, A.Yu. Technology for Prototyping Expert Systems Based on Transformations (PESoT): A Method / A.Yu.Yurin // CEUR Workshop Proceedings. – 2020, – Vol. 2677. – P. 36–50.

193. Yurin, A.Yu. Development of Software Decision-Making Modules Based on a Model-Driven Approach / A.Yu.Yurin, N.O. Dorodnykh // CEUR Workshop Proceedings. – 2020. – Vol. 2648. – P. 265–279.

194. Yurin, A.Yu. A Reverse Engineering Process for Inferring Conceptual Models from Canonicalized Tables / A.Yu.Yurin, N.O. Dorodnykh // Proceedings of the 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). – 2020. – P. 485–490.

195. Yurin, A.Y. Personal knowledge base designer: Software for expert systems prototyping / A.Yu.Yurin, N.O. Dorodnykh // SoftwareX. – 2020. – Vol. 11. – 100411.

196. Yurin, A.Yu. Experimental Evaluation of a Spreadsheets Transformation in the Context of Domain Model Engineering / A.Yu.Yurin, N.O. Dorodnykh // Proceedings of the 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT). – 2020. – P. 388–391.

197. Yurin, A.Yu. Semi-Automated Formalization and Representation of the Engineering Knowledge Extracted from Spreadsheet Data / A.Yu.Yurin, N.O. Dorodnykh, A.O.Shigarov // IEEE Access. – 2021. – Vol. 9. – P. 157468-157481.

198. Yurin, A.Yu. The rapid development of knowledge bases using UML class diagrams / A.Yu.Yurin, N.O.Dorodnykh, O.A.Nikolaychuk // International Journal of Computer Aided Engineering and Technology. – 2021. – Vol. 14, №1. – P. 39–61.

199. Yurin, A.Yu. The conception of an intelligent system for troubleshooting an aircraft / A.Yu.Yurin, Y.V.Kotlov, V.M.Popov // CEUR Workshop Proceedings. – 2021. – Vol. 2984. – P. 42–48.

200. Yurin, A.Yu. Application of decision tables transformations for prototyping knowledge bases in the case of forest fire risk forecasting / A.Yu.Yurin, N.O.Dorodnykh, O.A.Nikolaychuk // CEUR Workshop Proceedings. – 2021. – Vol. 2984. – P. 34–41.

201. Yurin, A.Yu., Creating Web Decision-Making Modules on the Basis of Decision Tables Transformations / A.Yu.Yurin, N.O.Dorodnykh // Communications in Computer and Information Science. – 2021. – Vol. 1341. – P. 167–184.

202. Yurin, A.Yu. Update (4.2020.0303) to “Personal Knowledge Base Designer: Software for expert systems prototyping”, (PII: S2352711019303334) / A.Yu.Yurin, N.O.Dorodnykh, O.A.Nikolaychuk // SoftwareX. – 2021. – Vol. 16. – 100825.

203. Zedlitz, J. From UML to OWL 2 / J.Zedlitz, J.Jörke, N.Luttenberger // Communications in Computer and Information Science. – 2012. – Vol. 295. – P. 154–163.

204. Zedlitz, J. Conceptual Modelling in UML and OWL-2 / J.Zedlitz, N.Luttenberger // International Journal on Advances in Software. – 2014. – Vol. 7,№1. – P. 182–196.

205. Zhang, F. Representation and reasoning of fuzzy ER model with description logic / F.Zhang, Z.M.Ma, L.Yan // Proceedings of IEEE International Conference on Fuzzy Systems. –

2008. – P.1358–1365.

206. Аверкин, А.Н. Толковый словарь по искусственному интеллекту / А.Н.Аверкин, М.Г.Гаазе-Рапопорт, Д.А.Поспелов. – М.: Радио и связь, 1992. – 256 с.

207. Агафонов, В.Н. Требования и спецификации в разработке программ / В.Н. Агафонов. – М.: Мир, 1984. – 344 с.

208. Агафонов, В.Н. Спецификация программ: понятийные средства и их организация / Агафонов В.Н. – Новосибирск: Наука, 1987. – 240 с.

209. Аликин, С.С. Разработка платформы создания экспертных систем с применением метапрограммирования / С.С.Аликин, В.П.Жидаков // Фундаментальные проблемы радиоэлектронного приборостроения. – 2012. – Т.12, № 6. – С. 80–83.

210. Арзамасов, Б.Н. Справочник по конструкционным материалам / Б.Н. Арзамасов и др.. – М.: МГТУ им Н.Э. Баумана, 2005.

211. Аркадьев, А.Г. Обучение машины классификации объектов / А.Г. Аркадьев, Э.М.Браверман. – М.: Наука, 1971. – 192 с.

212. Аршинский, Л.В. Логика с векторной семантикой как средство верификации баз знаний / Л.В. Аршинский, А.А. Ермаков, М.С. Нитежук // Онтология проектирования. – 2019. – Т. 9. – № 4 (34). – С. 510-521.

213. Ахо, А. В. Компиляторы: принципы, технологии и инструментарий, 2-е изд / А.В.Ахо, М.С. Лам, Р.Сети, Дж.Д.Ульман. – М.: Вильямс, 2008. – 1184 с.

214. Башлыков, А.А. Основы конструирования интеллектуальных систем поддержки принятия решений в атомной энергетике / А.А.Башлыков, А.П.Еремеев. – М.: Инфра-М, 2018. – 351 с.

215. Берман, А.Ф. Онтология надежности механических систем / А.Ф. Берман, О.А. Николайчук, А.И. Павлов, А.Ю. Юрин // Искусственный интеллект. – 2004. – №3. – С.266–271.

216. Берман, А.Ф. Интеллектуальная система поддержки принятия решений при определении причин отказов и аварий в нефтехимической промышленности / А.Ф. Берман, О.А. Николайчук, А.И. Павлов, А.Ю. Юрин // Автоматизация в промышленности. – 2006. – №6. – С.15–17.

217. Берман, А.Ф. Пространство технических состояний уникальных механических систем / А.Ф. Берман, О.А. Николайчук // Проблемы машиностроения и надежности машин. – 2007. – №1. – С.14–22.

218. Берман, А.Ф. Программная система для идентификации технического состояния машин и аппаратов / А.Ф. Берман, О.А. Николайчук, А.И. Павлов, А.Ю. Юрин

// Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2007613715 от 31.08.07.

219. Берман, А.Ф. Обеспечение безопасности технических объектов методом прецедентных экспертных систем / А.Ф. Берман, О.А. Николайчук, А.Ю. Юрин // Проблемы безопасности и чрезвычайных ситуаций. – 2008. – №5. – С.83–93.

220. Берман, А.Ф. Автоматизация прогнозирования технического состояния и остаточного ресурса деталей уникальных машин и аппаратуры / А.Ф. Берман, О.А. Николайчук, А.Ю. Юрин // Заводская лаборатория. Диагностика материалов. – 2009. – №3. – С.48–57.

221. Берман, А. Ф. Модели, знания и опыт для управления техногенной безопасностью / А.Ф. Берман, О.А. Николайчук // Проблемы управления. – 2010. – № 2. – С. 53–60.

222. Берман, А.Ф. Автоматизированное построение деревьев отказов и событий на основе модели динамики технического состояния и методов искусственного интеллекта / А.Ф. Берман, О.А. Николайчук, А.Ю. Юрин, Н.Ю. Павлов // Проблемы безопасности и чрезвычайных ситуаций. – 2011. – №1. – С. 40–52.

223. Берман, А.Ф. Интеллектуальная информационная система анализа отказов / А.Ф. Берман, О.А. Николайчук, А.Ю. Юрин // Проблемы машиностроения и надежности машин. – 2012. – №4. – С.88–96.

224. Берман, А.Ф. Интеллектуальная программная система автоматизированного построения деревьев событий / А.Ф. Берман, О.А. Николайчук, А.Ю. Юрин, Н.Ю. Павлов // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2012614092 от 04.05.12.

225. Берма,н А.Ф. Методы и средства автоматизированного построения деревьев событий и отказов / А.Ф. Берман, О.А. Николайчук, А.Ю. Юрин, Н.Ю. Павлов // Автоматизация и современные технологии. – 2013. – № 9. – С. 8–16.

226. Берман, А.Ф. Система поддержки принятия решений по предупреждению и ликвидации техногенных ЧС на основе прецедентного подхода / А.Ф. Берман, О.А. Николайчук, А.И. Павлов, А.Ю. Юрин // Технологии техносферной безопасности: Интернет-журнал. – Вып. 5 (51). – 3.11.2013. – 9 с.

227. Берман, А.Ф. Применение прецедентного подхода для поддержки принятия решений при определении причин и прогнозировании инцидентов и аварий / А.Ф. Берман, О.А. Николайчук, Г.С. Малтугуева, А.Ю. Юрин // Безопасность труда в промышленности. – 2014. – №11. – С.18-26.

228. Берман, А.Ф. Поддержка принятия решений на основе продукционного подхода при проведении экспертизы промышленной безопасности / А.Ф. Берман, О.А. Николайчук, А.Ю. Юрин, К.А. Кузнецов // Химическое и нефтегазовое машиностроение. – 2014. – № 11. – С. 28–35.

229. Берман, А.Ф. Поддержка принятия решений при выборе конструкционных материалов для обеспечения безопасной эксплуатации оборудования / А.Ф. Берман, Г.С. Малтугуева, А.Ю. Юрин // Заводская лаборатория. Диагностика материалов. – 2015. – №11. – С.73-80.

230. Берман, А.Ф. Проблемно-ориентированный редактор продукционных баз знаний / А.Ф. Берман, М.А. Грищенко, О.А. Николайчук, А.Ю. Юрин // Программные продукты и системы. – 2015. – №2. – С.13–19.

231. Берман, А.Ф. Обеспечение надежности и безопасности химических и нефтехимических производств методами искусственного интеллекта. Часть 1 / А.Ф. Берман, О.А. Николайчук, А.И. Павлов, А.Ю. Юрин // Прикладная информатика. – 2016. – Т.11. – №5(65). – С.63-75.

232. Берман, А.Ф. Обеспечение надежности и безопасности химических и нефтехимических производств методами искусственного интеллекта. Часть 2 / А.Ф. Берман, О.А. Николайчук, А.И. Павлов, А.Ю. Юрин // Прикладная информатика. – 2017. – Т.12. – №1(67). – С.26-38.

233. Берман, А.Ф. Метод синтеза и анализа деревьев отказов на основе понятий механизма и кинетики событий / А.Ф. Берман, О.А. Николайчук, Н.Ю. Павлов // Проблемы анализа риска. – 2018. – Т.15. – №3. – С.62–77.

234. Берман, А.Ф. Информационно-аналитическая поддержка экспертизы промышленной безопасности объектов химии, нефтехимии и нефтепереработки / А.Ф. Берман, К.А. Кузнецов, О.А. Николайчук, А.И. Павлов, А.Ю. Юрин // Химическое и нефтегазовое машиностроение. – 2018. – №8. – С.30–36.

235. Берман, А.Ф. Принципы информационной технологии решения междисциплинарных задач обеспечения техногенной безопасности на основе самоорганизации / А.Ф. Берман, О.А. Николайчук, А.И. Павлов, А.Ю. Юрин // Информационные и математические технологии в науке и управлении. – 2019. – №2(14). – С.5–15.

236. Берман, А.Ф. Деградация механических систем / А.Ф. Берман. – Новосибирск: Наука, 1998. – 320 с.

237. Берман, А.Ф. База данных по отказам оборудования высокого давления

химико-технологических линий по производству полиэтилена / А.Ф. Берман, В.К. Храмова, О.А. Николайчук // Свидетельство об официальной регистрации Базы Данных. – М. – Рег. № 990010 от 26.02.99 г.

238. Бонгард, М.М. Проблема узнавания / М.М.Бонгард. – М.: Физматгиз, 1967. – 320 с.

239. Бычков, И.В. Подход к разработке программных компонентов для формирования баз знаний на основе концептуальных моделей / И.В. Бычков, Н.О. Дородных, А.Ю. Юрин // Вычислительные технологии. – 2016. – Т. 21, № 4. – С. 16–36.

240. Варшавский, П.Р. Моделирование рассуждений на основе прецедентов в интеллектуальных системах поддержки принятия решений / П.Р. Варшавский, А.П. Еремеев // Искусственный интеллект и принятие решений. – 2009. – №2. – С.45–57.

241. Ворожцова, Т.Н. Онтологический подход к моделированию программного комплекса / Т.Н. Ворожцова, С.К. Скрипкин // Вестник Иркутского государственного технического университета. – 2006. – Т. 26, № 2. – С. 72–78.

242. Гаврилова, Т.А. Онтологический подход к управлению знаниями при разработке корпоративных информационных систем / Т.А. Гаврилова // Новости искусственного интеллекта. – 2003. – № 2. – С. 24–29.

243. Гаврилова, Т.А. Визуальные методы работы со знаниями: попытка обзора / Т.А. Гаврилова, Н.А. Гулякина // Искусственный интеллект и принятие решений. – 2008. – № 1. – С. 15–21.

244. Гаврилова, Т.А. Инженерия знаний. Модели и методы / Т.А. Гаврилова, Д.В. Кудрявцев, Д.И. Муромцев. – СПб.: Лань, 2016. – 324 с.

245. Гаибова, Т.В. Многокритериальная оптимизация инвестиционных проектов развития промышленных предприятий / Т.В. Гаибова // Автореферат н соискание ученой степени кандидата технических наук: 05.13.01. – Самара: Самарский государственный технический университет, 2004.

246. Голенков, В.В. Принципы построения массовой семантической технологии компонентного проектирования интеллектуальных систем / В.В. Голенков, Н.А. Гулякина // Материалы международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем OSTIS-2011». – 2011. – С. 21–58.

247. Голенков, В.В. Проект открытой семантической технологии компонентного проектирования интеллектуальных систем. часть 1: принципы создания / В.В. Голенков, Н.А. Гулякина // Онтология проектирования. – 2014. – №1. – С. 42–64.

248. Голубева, Л.В. Исследование влияния метеорологических факторов на возникновения и распространение лесных пожаров в Иркутской области / Л.В. Голубева, А.В. Латышева, К.А. Лощенко, А.С. Щевлыкин // Известия Иркутского государственного университета. Серия: Науки о Земле. – 2017. – №22. – С.30–40.

249. Горбунов-Посадов, М.М. Облик многократно используемого компонента / М.М. Горбунов-Посадов // Открытые системы. – 1998. – № 3. – С. 45–49.

250. Горбунов-Посадов, М.М. Расширяемые программы / / М.М. Горбунов-Посадов. – М.: Полиптих, 1999. – 336 с.

251. Горелик, А.Л. Методы распознавания / А.Л. Горелик, В.А.Скрипкин. – М.: ВШ, 2004. – 261 с..

252. Грибова, В.В. Проект IACPaaS. Комплекс для интеллектуальных систем на основе облачных вычислений / В.В. Грибова, А.С. Клещев, Д.А. Крылов, Ф.М. Москаленко, С.В. Смагин, В.А. Тимченко, М.Б. Тютюнник, Е.А. Шалфеева // Искусственный интеллект и принятие решений. – 2011. – № 1. – С. 27–35.

253. Грибова, В.В. Онтология диагностики процессов / В.В.Грибова, Е.А.Шалфеева // Онтология проектирования. – 2019. – 9(4/34). – С.449–461.

254. Грибачев, К.Г. Delphi и Model Driven Architecture. Разработка приложений баз данных / К.Г. Грибачев. – СПб: Питер, 2004. – 352 с.

255. Грищенко, М.А. Редактор баз знаний CLIPS / М.А.Грищенко, А.Ю.Юрин // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2012614093 от 04.05.12.

256. Грищенко, М.А. Разработка экспертных систем на основе трансформации информационных моделей предметной области / М.А. Грищенко, А.Ю. Юрин, А.И. Павлов // Программные продукты и системы. – 2013. – № 3. – С.143–147.

257. Грищенко, М.А. Применение модельно-управляемого подхода для создания продукционных экспертных систем и баз знаний / М.А. Грищенко, Н.О. Дородных, О.А. Николайчук, А.Ю. Юрин // Искусственный интеллект и принятие решений. – 2016. – № 2. – С. 16–29.

258. Грищенко, М.А. Система для прототипирования продукционных баз знаний / М.А. Грищенко, Н.О. Дородных, А.Ю. Юрин // Труды Пятнадцатой национальной конференции по искусственному интеллекту с международным участием (КИИ-2016). – 2016. – Т. 3. – С. 254–260.

259. Грищенко, М.А. Разработка интеллектуальных диагностических систем на основе онтологий / М.А. Грищенко, Н.О.Дородных, С.А.Коршунов, А.Ю.Юрин //

Онтология проектирования. – 2018. – Т.8. – №2(28). – С.265–284.

260. Гуськов, Г.Ю. Ontological Mapping for Conceptual Models of Software System / Г.Ю. Гуськов, А.М. Наместников // Материалы VII Международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем OSTIS-2017». – 2017. – С. 111–116.

261. Денисова, Л.А. Автоматизация параметрического синтеза системы регулирования на основе многокритериальной оптимизации с использованием генетического алгоритма / Л.А.Денисова // Автоматизация в промышленности. – 2013. – №12.

262. Джарратано, Дж. Экспертные системы: принципы разработки и программирования / Дж. Джарратано, Г.Райли. – М. Вильямс, 2007. – 1152 с.

263. Джексон, П. Введение в экспертные системы / П. Джексон. – М: Вильямс, 2001. – 624 с.

264. Дилигенский, Н.В. Нечеткое моделирование и многокритериальная оптимизация производственных систем в условиях неопределенности: технология, экономика, экология / Н.В.Дилигенский, Л.Г.Дымова, П.В.Севастьянов. – М.: Машиностроение, 2004.

265. Добрынин, В.Ю. Технологии компонентного программирования / В.Ю. Добрынин. – СПб.: СПбГУ, 2003. – 121 с.

266. Дородных, Н.О. Использование диаграмм классов UML для формирования продукционных баз знаний / Н.О. Дородных, А.Ю. Юрин // Программная инженерия. – 2015. – № 4. – С. 3–9.

267. Дородных, Н.О. Автоматизация создания продукционных баз знаний на основе концептуальных моделей / Н.О. Дородных, О.А. Николайчук, А.Ю. Юрин // Труды Шестой международной конференции «Системный анализ и информационные технологии». – 2015. – Т.1. – С.281–288.

268. Дородных, Н.О. Подход к автоматизации создания баз знаний на основе трансформации концептуальных моделей / Н.О. Дородных, А.Ю. Юрин // Материалы VI Международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем OSTIS-2016». – 2016. – С. 203–208.

269. Дородных, Н.О. Система программирования продукционных баз знаний: Personal Knowledge Base Designer / Н.О. Дородных, М.А. Грищенко, А.Ю. Юрин // Материалы VI международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем OSTIS-2016». –

2016. – С. 209–212.

270. Дородных, Н.О. Формирование баз знаний продукционного типа на основе UML-моделей / Н.О. Дородных, А.Ю. Юрин // Информатика и кибернетика. – 2016. – Т.5, № 3. – С. 44–50.

271. Дородных, Н.О. Концепция подхода к созданию программных компонентов генерации баз знаний на основе трансформации концептуальных моделей / Н.О. Дородных, С.А. Коршунов, А.Ю. Юрин // Информационные и математические технологии в науке и управлении. – 2016. – № 2. – С. 111–120.

272. Дородных, Н.О. Разработка программных компонентов для формирования баз знаний на основе трансформации концептуальных моделей / Н.О. Дородных, А.Ю. Юрин // Труды Пятнадцатой национальной конференции по искусственному интеллекту с международным участием (КИИ-2016). – 2016. – Т. 1. – С. 33–40.

273. Дородных, Н.О. About the specialization of model-driven approach for creation of case-based intelligent decision support systems / Н.О. Дородных, А.Ю. Юрин // Материалы VII Международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем OSTIS-2017». – 2017. – С. 151–154.

274. Дородных, Н.О. Автоматизированное создание продукционных баз знаний на основе деревьев событий / Н.О. Дородных, О.А. Николайчук, А.Ю. Юрин // Информационные и математические технологии в науке и управлении. – 2017. – Т. 6, № 2. – С. 30–41.

275. Дородных, Н.О. Использование онтологий при разработке продукционных экспертных систем / Н.О. Дородных, А.Ю. Юрин // Материалы VI Всероссийской конференции с международным участием «Знания-Онтологии-Теория ЗОНТ-2017». – 2017. – Т. 1. – С. 129–138.

276. Дородных, Н.О. Использование концепт-карт для автоматизированного создания продукционных баз знаний / Н.О. Дородных, А.Ю. Юрин // Программные продукты и системы. – 2017. – №4, – С. 658–662.

277. Дородных, Н.О. Автоматизированное создание продукционных баз знаний на основе коцепт-карт CmapTools / Н.О. Дородных, А.Ю. Юрин // Труды Седьмой международной конференции «Системный анализ и информационные технологии». – 2017. – С. 337–341.

278. Дородных, Н.О. Подход автоматизированной разработки баз знаний на основе трансформации диаграмм Исикавы / Н.О. Дородных, О.А. Николайчук, А.Ю. Юрин // Вестник компьютерных и информационных технологий. – 2018. – №4. – С.41–51.

279. Дородных, Н.О. Средства поддержки моделирования логических правил в нотации RVML / Н.О. Дородных, С.А. Коршунов, А.Ю. Юрин // Программные продукты и системы. – 2018. – №4. – С. 667–672.

280. Дородных, Н.О. A Model-Driven Development Approach for Case Bases Engineering / Н.О. Дородных, А.Ю. Юрин // Открытые семантические технологии проектирования интеллектуальных систем. – 2019. – Т.3. – №9. – С. 179–182.

281. Дородных, Н. О. Knowledge Base Development System / Н.О. Дородных, А.Ю. Юрин // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2019661803 от 09.09.19.

282. Дородных, Н.О. Разработка и использование метамоделей для синтеза спецификаций и кодов баз знаний / Н.О. Дородных, О.А. Николайчук, А.Ю. Юрин, С.А. Коршунов // Информационные и математические технологии в науке и управлении. – 2019. – №2(14). – С. 26–39.

283. Дородных, Н.О. Технология создания продукционных экспертных систем на основе модельных трансформаций / Н.О. Дородных, А.Ю. Юрин. – Новосибирск: СО РАН, 2019. – 144 с.

284. Дородных, Н.О. Model Transformations for Intelligent Systems Engineering / Н.О.Дородных, С.А.Коршунов, Н.Ю. Павлов, Д.Ю. Сопп, А.Ю. Юрин // Открытые семантические технологии проектирования интеллектуальных систем. – 2018. – Т.2. – №8. – С.77–81.

285. Дородных, Н.О. Разработка метамоделей для поддержки синтеза нечетких баз знаний / Н.О. Дородных, О.А. Николайчук, А.Ю. Юрин // Вестник компьютерных и информационных технологий. – 2020. – №1. – С. 34–47.

286. Дородных, Н.О. Создание нечетких баз знаний на основе преобразования нечетких концептуальных моделей / Н.О. Дородных, О.А. Николайчук, А.Ю. Юрин, С.А. Коршунов // Информационные и математические технологии в науке и управлении. – 2020. – №2(18). – С. 19–35.

287. Дородных, Н.О. Extended Event Tree Editor / Н.О. Дородных, Д.Н. Шпаченко // Свидетельство о государственной регистрации программ для ЭВМ №2020660788 от 11.09.2020. – М.: ФИПС, 2020.

288. Дородных, Н.О. Средство визуального моделирования и генерации кода нечетких продукций / Н.О. Дородных, А.Ю. Юрин, С.А. Коршунов, Д.Ю. Сопп, Д.Н. Шпаченко // Информационные и математические технологии в науке и управлении. – 2021. – №1(21). – С. 121–131.

289. Дородных, Н.О Подход к созданию онтологий на основе электронных таблиц с произвольной структурой / Н.О. Дородных, А.В. Видия, А.Ю. Юрин // Онтология проектирования. – 2021. – Т.11. – №2. – С.212–226.

290. Дородных, Н.О. Разработка схем онтологий на основе преобразования электронных таблиц / Н.О. Дородных, А.В. Видия, А.Ю. Юрин // Программные продукты и системы. – 2021. – №1. – С. 124–131.

291. Дородных, Н.О. Knowledge Graph Augmentation System / Н.О. Дородных, А.Ю. Юрин // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2021664724 от 13.09.21.

292. Дубинин, В.Н. Проектирование управляющих приложений на основе трансформации онтологий с использованием языков логического программирования / В.Н. Дубинин, В.В. Вяткин // Труды Международной научно-технической конференции «Современные информационные технологии». – 2012. – № 16. – С. 6–25.

293. Дюк, В.А. Компьютерная психодиагностика / В.А. Дюк. – СПб.: Братство, 1994. – 364 с.

294. Еремеев, А.П. Инструментальный программный комплекс СИМПР-2015 и его применение при обучении студентов по направлению «Прикладная математика и информатика» / А.П. Еремеев, Н.В. Чибизова // В сборнике: Информатизация инженерного образования. Труды Международной научно-практической конференции - ИНФОРИНО-2016. – 2016. – С. 125-130.

295. Еремеев, А.П. Инструментальный комплекс проектирования систем поддержки принятия решений реального времени СИМПР-2015 / А.П. Еремеев, Н.В. Чибизова // Свидетельство о государственной регистрации программы для ЭВМ. – М: Роспатент. – Рег. № 2017619525 от 25.08.2017.

296. Ермаков, А.Е. Инструментальное средство для автоматизированного создания экспертных систем / А.Е. Ермаков, К.А. Найденова // Программные продукты и системы. – 2013. – № 3. – С. 107–114.

297. Ершов, А.П. Предварительные соображения о лексиконе программирования / А.П. Ершов // Кибернетика и вычислительная техника. – 1985. – № 1.

298. Журавлев, Ю.И. Распознавание, классификация, прогноз. Математические методы и их применение. Вып. 2. / Под ред. Ю.И. Журавлева – М.: Наука, 1989. – 302 с.

299. Загорулько, Ю.А. Семантическая технология разработки интеллектуальных систем, ориентированная на экспертов предметной области / Ю.А. Загорулько // Онтология проектирования. – 2015. – Т. 15, № 1. – С. 30–46.

300. Загорулько, Ю.А. Подход к реализации паттернов содержания при разработке онтологий научных предметных областей / Ю. А. Загорулько, О.И. Боровикова // Системная информатика. – 2018. – №12. – С. 27–40.

301. Залесов, С.В.Уточненная шкала распределения участков лесного фонда по классам природной пожарной опасности / С.В.Залесов, Г.А. Годовалов, Е.Ю. Платонов // Аграрный вестник Урала. – 2013. – № 10 (116). – С. 45–49.

302. Игошин, В.А. Инструментальные средства создания баз знаний, экспертных систем и интеллектуальных систем поддержки принятия решений / В.А. Игошин, М.В. Игошин // В сборнике: Искусственный интеллект в решении актуальных социальных и экономических проблем XXI века. – 2018. – С. 291–293.

303. Исикава, К. Японские методы управления качеством / К. Исикава. – М: Экономика, 1988. – 214 с.

304. Касьянов, В.Н. Методы построения трансляторов / В.Н. Касьянов, И.В. Поттосин. – Н.: Наука, 1986. – 344 с.

305. Касьянов, В.Н. Инструментальные средства преобразования программ / В.Н. Касьянов, В.К. Сабельфельд. – Препр. ВЦ СО АН СССР №765. – Новосибирск, 1987.

306. Кознов, Д.В. Основы визуального моделирования / Д.В. Кознов. – М: БИНОМ, 2012. – 246 с.

307. Котова, В.Е. Смешанные вычисления и преобразование программ // Сб. научных трудов под ред. В. Е.Котова. – Новосибирск: ВЦ СО АН СССР, 1991. – 259 с.

308. Котлов, Ю.В. Онтология поиска и устранения неисправностей системы электроснабжения воздушного судна / Ю.В. Котлов, В.М. Попов, А.Ю. Юрин // Материалы всероссийской конференции с международным участием «Знания-Онтологии-Теории (ЗОНТ-21)» (8-12 ноября 2021 г., Новосибирск). – Новосибирск: ИМ СО РАН, 2021. – С. 142–149.

309. Кузнецов, А.М. Информационно-аналитическая система «Экспертиза промышленной безопасности оборудования химической, нефтехимической и нефтеперерабатывающей промышленности» / А.М. Кузнецов, А.Ф. Берман, К.А. Кузнецов, О.А. Николайчук, А.И. Павлов, А.Ю. Юрин // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2016610757 от 19.01.16.

310. Кузнецов, А.М. Сосуды и трубопроводы высокого давления: Справочник / под ред. А.М. Кузнецова и В.И. Лившица. – Иркутск. Типография №1, 1999.

311. Лаврищева, Е.М. Сборочное программирование. Основы индустрии программных продуктов / Е.М. Лаврищева, В.Н.Грищенко. – К.: Наук. думка, 2009. – 372

с.

312. Лаврищева, Е.М. Программная инженерия. Парадигмы, технологии и CASE-средства: Учебник / Е.М. Лаврищева. – М.: Юрайт, 2016. – 282 с.

313. Люгер, Дж. Ф. Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е издание / Дж. Ф. Люгер. – М.: Вильямс, 2003. – 864 с.

314. Макаров, Н.Н. Синтез алгоритма функционирования информационно-управляющей системы контроля и диагностики состояния общесамолетного оборудования. Известия вузов / Н.Н. Макаров // Авиационная техника. – 2008. №1. – С.46–50.

315. Малтугуева, Г.С. Система поддержки принятия решений в задачах группового выбора / Г.С. Малтугуева, И.А. Наумов, А.Ю. Юрин // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2009614243 от 12.08.09.

316. Мартыненко, Б.К. Языки и трансляции: Учеб. Пособие / Б.К. Мартыненко. – СПб.: СПбТУ, 2008. – 257 с.

317. Массель, Л.В. Построение интеллектуальных систем для исследований энергетики на основе алгебраических сетей и онтологий: подход и реализация / Л.В. Массель, А.Н. Копайгородский, В.Л. Аршинский // Вычислительные технологии. – 2008. – Т. 13, спец. выпуск 1. – С. 50–58.

318. Машиностроение. Энциклопедия / Ред. Совет: К.В. Фролов (пред.) и др. – М.: Машиностроение. Машины и аппараты химических производств. Т. IV-12 / М.Б. Генералов и др. 2004.

319. Минский, М. Структура для представления знания / М. Минский // Психология машинного зрения. – М.: Мир, 1978. – С. 249–338.

320. Наместников, А.М. Программная система преобразования UML-диаграмм в онтологии на языке OWL / А.М. Наместников, Г.Ю. Гуськов // Труды Пятнадцатой национальной конференции по искусственному интеллекту с международным участием (КИИ-2016). – 2016. – Т. 3. – С. 270–278.

321. Николайчук, О.А. Инструментальное средство создания информационных систем автоматизации процесса исследования и обеспечения надежности механических систем / О.А. Николайчук, А.И. Павлов, А.Ю. Юрин // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2005611218 от 25.05.05.

322. Николайчук, О.А. Программный модуль правдоподобного вывода по прецедентам / О.А. Николайчук, А.Ю. Юрин // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2007613714 от 31.08.07.

323. Николайчук, О.А. Управление опытом при исследовании динамики технического состояния уникальных машин и конструкций: моделирование опыта / О.А. Николайчук, А.Ю. Юрин // Информационные технологии. – 2008. – №6. – С.30–37.

324. Николайчук, О.А. Автоматизация исследований технического состояния опасных механических систем / О.А. Николайчук // Проблемы машиностроения и надежности машин. – 2008. – №6. – С.72–78.

325. Николайчук, О.А. Применение прецедентного подхода для автоматизированной идентификации технического состояния деталей механических систем / О.А. Николайчук, А.Ю. Юрин // Автоматизация и современные технологии. – 2009. – №5. – С.3–12.

326. Николайчук, О.А. Моделирование знаний для исследования динамики технического состояния уникальных объектов / О.А. Николайчук // Проблемы управления. – 2009. – № 4. – С.58-65.

327. Николайчук, О. А. Методы, модели и инструментальное средство для исследования надежности и безопасности сложных технических систем: автореф. дис. докт. тех. наук: 05.13.01 / Николайчук Ольга Анатольевна. – М. ИСА РАН, 2011. – 37 с.

328. НТЦ «Промышленная безопасность» [Электронный ресурс]. – Режим доступа:http://www.safety.ru/ (дата обращения: 18.03.2022).

329. Опарин, Г.А. Сатурн – метасистема для построения пакетов прикладных программ / Г.А. Опарин // Пакеты прикладных программ. Методы и разработки. – Новосибирск: Наука, 1982. – С. 130–160.

330. Осипов, Г.С. Методы искусственного интеллекта / Г.С. Осипов. – М.: ФИЗМАТЛИТ, 2011. – 296 с.

331. Осуга, С. Приобретение знаний / С. Осуга, Ю.Саэки, Х.Судзуки, Х.Кобаяси, С. Оцуки, Т.Китихаси, Ю.Танака, С.Арикава, Т.Синохара, Т.Мияхара, М.Харагути. – М.: Мир, 1990. – 304 с.

332. Павлов, А.И. Прототип системы поддержки проектирования агентов для имитационных моделей сложных систем / А.И. Павлов, А.Б. Столбов // Программные продукты и системы. – 2016. – № 3. – С. 79-84.

333. Павлов, А.И. Компонентный подход: модуль правдоподобного вывода по прецедентам / А.И. Павлов, А.И. Юрин // Программные продукты и системы. – 2008. – №3. – С.55–58.

334. Перфильев, О.В. Интеллектуальная система поиска неисправности на самолете / О.В.Перфильев, С.Г.Рыжаков, В.А.Должиков // Известия Самарского научного

центра Российской академии наук. – 2018. – № 4(3). – С. 326–331.

335. Поспелов, Д.А. Инженерия знаний / Д.А. Поспелов // Наука и жизнь. – 1987. – №6. – С. 11–24.

336. Рассел, С. Искусственный интеллект: современный подход, 2-е издание / С.Рассел, П.Норвиг. – М.: Вильямс, 2006. – 1408 с.

337. Ройзензон, Г.В. Многокритериальный выбор вычислительных кластеров / Г.В. Ройзензон // Труды Института системного программирования РАН. – М.: ИСА РАН, 2005. – Том 12.

338. Рубцов, А.В. Системный анализ погодной пожарной опасности при прогнозировании крупных пожаров в лесах Сибири / А.В. Рубцов, А.И. Сухинин, Е.А. Ваганов // Исследование земли из космоса. – 2010. – № 3. – С. 62–70.

339. Рыбина, Г.В. Методы и алгоритмы верификации баз знаний в интегрированных экспертных системах / Г.В. Рыбина, В.В. Смирнов // Известия Российской академии наук. Теория и системы управления. – 2007. – № 4. – С. 91–102.

340. Рыбина, Г.В. Теория и технология построения интегрированных экспертных систем / Г.В.Рыбина. − М: Научтехлитиздат, 2008. – 482 с.

341. Рыбина, Г.В. Основы построения интеллектуальных систем / Г.В.Рыбина. − М.: Финансы и статистика; ИНФРА-М, 2010. – 432 с.

342. Рыбина, Г.В. Инструментальные средства построения динамических интегрированных экспертных систем: развитие комплекса АТ-Технология / Г.В. Рыбина // Искусственный интеллект и принятие решений. – 2010. – № 1. – С. 41–48.

343. Саввина, А.М. Предложение по модернизации бортовой системы технического обслуживания самолета SSJ 100 / А.М. Саввина // Crede Experto: транспорт, общество, образование, язык. – 2019. – № 3(22). – С. 27–35.

344. Слободюк, А.А. О подходе к созданию онтологий на основе системно-объектных моделей предметной области / А.А. Слободюк, С.И. Маторин, С.Н. Четвериков // Научные ведомости БелГУ. Сер. История. Политология. Экономика. Информатика. – 2013. – Т. 165, № 22. – С. 186–194.

345. СМС-Органайзер [Электронный ресурс]. – Режим доступа: http://centrasib.ru/index.php?p=smso (дата обращения: 18.03.2022).

346. Соболь, И.М. Выбор оптимальных параметров в задачах со многими критериями / И.М. Соболь, Р.Б. Статников. – М.: Дрофа, 2006.

347. Софронова, А.В. Оценка природной пожарной опасности лесных участков на территории нефтегазовых комплексов с использованием данных дистанционного

зондирования земли / А.В. Софронова, А.В. Волокитина // Сибирский лесной журнал. – 2017. – № 5. – С. 84–94.

348. Стенников, В.А. Применение онтологий при реализации концепции модельно-управляемой разработки программного обеспечения для проектирования теплоснабжающих систем / В.А. Стенников, Е.А. Барахтенко, Д.В. Соколов // Онтология проектирования. – 2014. – Т. 14, № 4. – С. 54–68.

349. Указ Президента РФ от 01.12.2016 N 642 (ред. от 15.03.2021) О Стратегии научно-технологического развития Российской Федерации [Электронный ресурс]. – Режим доступа: https://sudact.ru/law/ukaz-prezidenta-rf-ot-01122016-n-642/strategiia-nauchno-tekhnologicheskogo-razvitiia-rossiiskoi-federatsii/ (дата обращения: 18.03.2022).

350. Ту, Дж. Принципы распознавания образов.: Пер. с англ / Дж.Ту, Р. Гонсалес. – М.: Мир, 1978. – 411 с.

351. Тыугу, Э.Х. Концептуальное программирование / Э.Х. Тыугу. – М.: Наука, 1984. – 256 с.

352. Федеральный закон «О рекламе» [Электронный ресурс]. – Режим доступа: https://base.garant.ru/12145525/ (дата обращения: 18.03.2022).

353. Фримен, А. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов, 4-е издание / А.Фримен. – М.: Вильямс, 2013. – 688 с.

354. Цаленко, М.Ш. Моделирование семантики в базах данных / М.Ш. Цаленко. – М.: Наука, 1989

355. Цикритзис, Д. Модели данных / Д. Цикритзис, Ф.Лоховски. – М.: Финансы и статистика, 1985. – 168 с.

356. Частиков, А.П. Разработка экспертных систем. Среда CLIPS / А.П. Частиков, Т.А. Гаврилова, Д.Л. Белов. – СПб.: БХВ-Петербург, 2003. – 393 с.

357. Чернецки, К. Порождающее программирование: методы, инструменты, применение / К.Чернецки, У.Айзенекер. – СПб: Питер, 2005. – 736 с.

358. Черняховская, Л.Р. Формирование правил принятия решений в управлении проектами по результатам онтологического анализа / Л.Р.Черняховская, А.И. Малахова // Материалы XV Международной научной конференции «Проблемы управления и моделирования в сложных системах». – 2013. – С. 343–350.

359. Шур, Ю.З. Региональные шкалы оценки природной пожарной опасности лесов / Ю.З. Шур, В.Ю. Нешатаев, А.А. Степченко, Н.В. Шаповал // Труды Санкт-Петербургскогонаучно-исследовательскогоинституталесногохозяйства. – 2020. – №2. – С.59–69.

360. Экман, П. Узнай лжеца по выражению лица / П.Экман. – СПб.: Питер, 2013. – 272 с

361. Юрин, А.Ю. Инструментальное средство создания интеллектуальных систем поддержки принятия решений для идентификации технического состояния деталей машин и конструкций / А.Ю. Юрин, О.А. Николайчук, А.И. Павлов // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2005611217 от 25.05.05.

362. Юрин, А.Ю. Применение методов группового выбора в составе прецедентных экспертных систем для обоснования мероприятий по предотвращению повторных отказов технологического оборудования / А.Ю. Юрин, Г.С. Малтугуева // Вестник компьютерных и информационных технологий. – 2012. – №9. – С.37–44.

363. Юрин, А.Ю. Редактор баз знаний в формате CLIPS / А.Ю. Юрин, М.А. Грищенко // Программные продукты и системы. – 2012. – №4. – С.83–87.

364. Юрин, А.Ю. Обоснование мероприятий по предотвращению повторных отказов на основе прецедентов и методов группового выбора / А.Ю. Юрин, Г.С. Малтугуева // Автоматизация и современные технологии. – 2013. – №2. – С.16–23.

365. Юрин, А.Ю. Методы группового выбора для адаптации решений, полученных в результате рассуждений на основе прецедентов / А.Ю. Юрин // Искусственный интеллект и принятие решений. – 2013. – №3. – С.78–85.

366. Юрин, А.Ю. Web-сервис для автоматизированного формирования продукционных баз знаний на основе концептуальных моделей / А.Ю. Юрин., Н.О. Дородных // Программные продукты и системы. – 2014. – № 4. – С. 103–107.

367. Юрин, А.Ю. Personal Knowledge Base Designer / А.Ю. Юрин // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2016617733 от 14.07.16.

368. Юрин, А.Ю. Нотация для проектирования баз знаний продукционных экспертных систем / А.Ю. Юрин // Объектные системы. – 2016. – №12. – С.48–54.

369. Юрин, А.Ю. Разработка продукционных баз знаний для задач экспертизы промышленной безопасности / А.Ю. Юрин, А.Ф. Берман, О.А. Николайчук, Н.О. Дородных // Информатика и кибернетика. – 2018. – №4 (14). – С. 39–46.

370. Юрин, А.Ю. Прототипирование прецедентных баз знаний на основе модельных трансформаций / А.Ю. Юрин // Образовательные ресурсы и технологии. – 2019. – №2(27). – С. 45–58.

371. Юрин, А.Ю. Применение трансформаций таблиц решений при создании интеллектуального программного модуля «Детектор» для веб-приложений / А.Ю. Юрин //

Программные продукты и системы. – 2020. – №4. – С. 573–581.

372. Юрин, А.Ю. iDSS.Desktop / А.Ю. Юрин, Н.О. Дородных // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2022617130 от 19.04.22.

373. Юрин, А.Ю. Детектор / А.Ю. Юрин // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2020614257 от 27.03.20.

374. Юрин, А.Ю. СМС-Органайзер / А.Ю. Юрин, Д.Ю. Сопп // Свидетельство о государственной регистрации программы для ЭВМ. – М. – Рег. № 2022613512 от 14.03.22.

Приложение А
Акты и справки о внедрении

Акционерное общество
«Иркутский научно-исследовательский и конструкторский институт
химического и нефтяного машиностроения»
(АО «ИркутскНИИхиммаш»)

Академика Курчатова ул., д. 3, г. Иркутск, 664074
тел.: (395-2) 41-04-34 факс: (395-2) 41-05-10 **e-mail:** himmash@irk.ru http://himmash.irk.ru

ОКПО 00220227; ОГРН 1023801748596; ИНН/ КПП 3812010128/381201001

АКТ
об использовании результатов научных исследований

Результаты диссертационной работы Юрина Александра Юрьевича в форме языков, методов, алгоритмов и программного обеспечения используются для автоматизированного создания баз знаний экспертных систем определения причин инцидентов и аварий нефтехимического оборудования при проведении экспертизы промышленной безопасности. В частности, используется программное обеспечение формирования баз знаний на основе трансформации концептуальных моделей деревьев событий, отражающих знания о процессах формирования инцидентов и аварий.

Использование указанных результатов позволяет повысить качество проведения технического диагностирования объектов экспертизы промышленной безопасности.

Результаты используются в рамках продолжения НИР, осуществляемых по договору № 052013НИР от 19 сентября 2013 г. «Разработка проблемно-ориентированного редактора продукционных баз знаний для задач оценки технического состояния и остаточного ресурса».

Заведующий научно-исследовательским
и конструкторским отделом оборудования
для химической, нефтехимической,
нефтегазоперерабатывающей промышленности Г.М. Мордина

Форма
Ф.3.10-01/05-19

ООО «Смарт технологии»

Почтовый адрес: Москва, улица Панфилова, 3Б, стр 1, офис 16
Фактический адрес: Москва, Шаболовка, 34, офис 22.
Тел: +74952410693 E-mail: info@ocenkakadrov.ru www.ocenkakadrov.ru

Исх. №9 от 24.01.2022 г.

СПРАВКА
об использовании результатов интеллектуальной деятельности

Результаты интеллектуальной деятельности в форме оригинальных методов, алгоритмов и программных средств модельных трансформаций концептуальных моделей, в частности, «Personal Knowledge Base Designer» (Свидетельство о государственной регистрации программы для ЭВМ №2016617733) и «Knowledge Base Development System» (Свидетельство о государственной регистрации программы для ЭВМ №2019661803) были использованы при прототипировании модуля интерпретации признаков эмоций.

Разработка модуля осуществлюсь в рамках стартапа по созданию системы поддержки кадрового сотрудника (рекрутера) «HR-Robot». Разработанный программный модуль «EmSi-Interpreter» обеспечил возможность обработки информации о движении лицевых мышц и опорных точек с целью формирования заключения об эмоциональном состоянии респондента.

С уважением,

Генеральный директор
ООО «Смарт технологии»
Ляпунова О.А.

ОБЩЕСТВО С ОГРАНИЧЕННОЙ
ОТВЕСТВИЕННОСТЬЮ

ЦЕНТРАСИБ

ул. Лермонтова, д.277/5, кв.26 Иркутск, 664033
Тел: (3952) 603-911
E-mail: info@centrasib.ru
http://www.centrasib.ru
ОКПО 90993335, ОГРН 1113850018281
ИНН/КПП 3812133850/381201001

№ 2020/08-2

На № _____ от _____

АКТ О ВНЕДРЕНИИ ПРОГРАММЫ ДЛЯ ЭВМ

Данный акт удостоверяет, что программа для ЭВМ «Детектор» (Свидетельство о государственной регистрации программы для ЭВМ №2020614257) успешно внедрена в 2020 г. в эксплуатацию в ООО «ЦентраСиб» в виде отдельного интегрируемого модуля телекоммуникационной платформы «СМС-Органайзер».

Программный модуль предназначен для обнаружения коротких сообщений и пользователей платформы, нарушающий требования ФЗ-38 «О рекламе». В основе модуля лежит реализация методов искусственного интеллекта, обеспечивающих решение задачи на основе логических правил. Необходимо отметить также инновационный характер примененного при создании модуля «Детектор» инструментария – среды разработки «Personal Knowledge Base Designer», которая позволила сотрудникам-экспертам компании принять участие в дополнении и корректировке разработанной базы знаний путем использования табличного редактора Microsoft Excel.

Использование внедренной программы повысило защищенность нашей телекоммуникационной платформы и обеспечило снижение рисков финансовых потерь, связанных с уплатой штрафов за нарушение федерального законодательства.

Технический директор Д.Ю. Сопп

УТВЕРЖДАЮ
Проректор по учебной работе
ФГБОУ ВО «ИРНИТУ»
Смирнов В.В.
«14» 03 2022 г.

АКТ ВНЕДРЕНИЯ

Настоящим актом подтверждается, что программы для ЭВМ «Personal Knowledge Base Designer» (Свидетельство о государственной регистрации программы для ЭВМ №2016617733), «Knowledge Base Development System» (Свидетельство о государственной регистрации программы для ЭВМ №2019661803), «Система поддержки принятия решений в задачах группового выбора» (Свидетельство о государственной регистрации программы для ЭВМ №2009614243) и «Редактор баз знаний CLIPS» (Свидетельство о государственной регистрации программы для ЭВМ №2012614093) используются при проведении лабораторных занятий по дисциплинам «CASE-средства» (направление подготовки 09.03.01 «Информатика и вычислительная техника») и «Инструментальные средства информационных систем» (направление подготовки 09.03.02 «Информационные системы и технологии») для автоматизированного создания баз знаний экспертных систем на основе анализа концептуальных моделей и решения задач поддержки принятия решений в Институте информационных технологий и анализа данных ФГБОУ ВО «Иркутский национальный исследовательский технический университет (ФГБОУ ВО «ИРНИТУ»).

Директор института
ИТиАД, ИРНИТУ

Говорков А.С.

СПРАВКА
об использовании результатов диссертационного исследования

Результаты диссертационной работы Юрина Александра Юрьевича, представленные оригинальными методами, алгоритмами и комплексом взаимосвязанных программных средств автоматизации создания интеллектуальных систем и их компонентов с использованием многоступенчатых преобразований концептуальных моделей различной степени абстракции, используются на кафедре «Авиационных электросистем и пилотажно-навигационных комплексов» (АЭС и ПНК) Иркутского филиала МГТУ ГА при проведении НИР «Интеллектуальная система поиска и устранения неисправностей системы электроснабжения воздушного судна RRJ-95», связанной с автоматизацией и интеллектуализацией деятельности технического персонала, направленной на поиск и устранение неисправностей воздушных судов. Конечной целью НИР является разработка интеллектуальной системы поддержки принятия решения «АвиаТехПом».

Особенностью разработанного диссертантом инструментария является его ориентация на конечных пользователей, которые, в общем случае, не являются программистами. Использование данной особенности позволило привлечь студентов к решению задач НИР, связанных с разработкой продукционных баз знаний.

Заведующий кафедрой Иркутского
филиала МГТУ ГА, к.т.н., доцент

«19» апреля 2022 г _____ В.М. Попов

Подпись работника Попова В.М.
заверяю Специалист по
персоналу Пузь
И.Ф. Зуева

Приложение Б
Описание методов программного интерфейса KBDS

Для доступа внешних систем к основным функциям/компонентам KBDS разработан программный интерфейс, взаимодействие осуществляется на основе REST API. Определены следующие сокращения для обозначения типов компонентов:

- 0 – анализа, использующий модель онтологии;
- 1 – анализа, использующий модель продукций;
- 2 и 4 – кодогенерации на OWL;
- 3 и 5– кодогенерации на CLIPS.

Сокращения для обозначения типа БЗ: 0 – онтологическая; 1 – продукционная.

Сокращения для обозначения статуса БЗ: 0 – общедоступная; 1 – частная.

Основные методы интерфейса:

- **getAllModulesList** – полный список всех компонентов; метод обращения GET; без параметров.

  Ответ системы при наличии компонентов:

  ~id=идентификатор (номер) компонента**;name=** наименование компонента**;description=**описание компонента**;type=**тип компонента**;status=**статус компонента**~**…

  Ответ содержит информацию о компонентах, разделенную символом «**~**»; символ «**;**» используется для разделения информации о компоненте.

  При отсутствии компонентов метод возвращает «**false»**.

  Пример вызова: http://kbds.knowledge-core.ru/api/get-all-modules-list.

- **getModulesList** – список компонентов заданного типа и статуса; метод обращения GET; передаваемые параметры: тип и статус программного компонента (см. список выше).

  Ответ системы при наличии компонентов:

  ~**id=**идентификатор (номер) компонента**;name=** наименование компонента**;description=**описание компонента**~**…

  При отсутствии компонентов с данными условиями метод возвращает «**false»**.

  Пример вызова: http://kbds.knowledge-core.ru/api/get-modules-list/1/2.

- **getAllKnowledgeBasesList** – список баз знаний; метод обращения GET; баз параметров.

  Ответ системы при наличии компонентов:

```
~id=идентификатор (номер) базы знаний;name= наименование базы
знаний;description=описание базы знаний;subject_domain=наименование
предметной области базы знаний;type=тип базы знаний;status=статус базы
знаний;author=логин автора базы знаний~…
```

  При отсутствии БЗ метод возвращает «**false**».

- **getKnowledgeBasesList** – список баз знаний заданного типа и статуса; метод запроса GET; передаваемые параметры: тип и статус.

  Ответ системы при наличии БЗ:

```
~id=идентификатор (номер) базы знаний;name= наименование базы
знаний;description=описание базы знаний;subject_domain=наименование
предметной области базы знаний;author=логин автора базы знаний~…
```

  При отсутствии БЗ с данными условиями метод возвращает «**false**».

  Пример вызова: http://kbds.knowledge-core.ru/api/get-knowledge-bases-list/1/2.

- **importConceptualModel** – импортирование модели в KBDS; метод запроса POST; передаваемый параметр: наименование файла концептуальной модели.

- **exportKnowledgeBase** – экспортирование кода базы знаний из KBDS; метод запроса GET; передаваемый параметр: идентификатор базы знаний.

В Таблице Б.1 и Б.2 приведены методы взаимодействия с моделью онтологии продукций, соответственно.

Таблица Б.1. Методы взаимодействия с моделью онтологии

Название	Сигнатура	Описание
createOntology	name – наименование модели онтологии, description – описание модели онтологии, xml_file – файл концептуальной модели в формате XML.	Создание модели онтологии на основе импорта концептуальной модели.
generateOWLCode	file_name – наименование файла БЗ, ontology_name – наименование модели онтологии.	Генерация кода БЗ на ЯПЗ OWL, на основе модели онтологии.
addConcept	name – наименование понятия, description – описание понятия, ontology_id – идентификатор модели онтологии.	Добавление нового понятия.
updateConcept	id – идентификатор понятия, name – наименование понятия, description – описание понятия, ontology_id – идентификатор модели онтологии.	Изменение понятия.

removeConcept	id – идентификатор понятия.	Удаление понятия.
addAttribute	name – наименование атрибута, type – тип атрибута, value – значение атрибута, description – описание атрибута, concept_id – идентификатор понятия.	Добавление нового атрибута (свойства) понятия.
updateAttribute	id – идентификатор атрибута, name – наименование атрибута, type – тип атрибута, value – значение атрибута, description – описание атрибута.	Изменение атрибута (свойства) понятия.
removeAttribute	id – идентификатор атрибута.	Удаление атрибута (свойства) понятия.
getRelation	id – идентификатор связи.	Получение причинно-следственной связи между двумя понятиями по идентификатору.
addRelation	left_concept_id – идентификатор левого понятия связи, right_concept_id – идентификатор правого понятия связи, name – наименование связи, description – описание связи, obligation – обязательность связи, ontology_id – идентификатор модели онтологии.	Добавление нового причинно-следственного отношения между двумя понятиями.
updateRelation	id – идентификатор связи, left_concept_id – идентификатор левого понятия связи, right_concept_id – идентификатор правого понятия связи, name – наименование связи, description – описание связи, obligation – обязательность связи.	Изменение причинно-следственного отношения между двумя понятиями.
removeRelation	id – идентификатор связи.	Удаление причинно-следственного отношения между двумя понятиями.
getAllConcepts	ontology_id – идентификатор модели онтологии.	Получение списка всех понятий у данной модели онтологии.
getConcept	id – идентификатор понятия.	Получение понятия по идентификатору.
getAllAttributes	concept_id – идентификатор понятия.	Получение всех свойств понятия.
getAttribute	id – идентификатор свойства.	Получение свойства понятия по идентификатору.
getAllRelations	ontology_id – идентификатор модели онтологии.	Получение всех причинно-следственных связей между понятиями у данной модели онтологии.

Таблица Б.2. Методы взаимодействия с моделью продукций

Название	Сигнатура	Описание
createProductionModel	name – наименование модели продукций, description – описание модели продукций, xml_file – файл концептуальной модели в	Создание модели продукций на основе импорта концептуальной модели.

	формате XML.	
generateCLIPSCode	file_name – наименование файла БЗ, production_model_name – наименование модели продукций.	Генерация кода БЗ на ЯПЗ CLIPS, на основе модели продукций.
addFactTemplate	name – наименование шаблона факта, description – описание шаблона факта, production_model_id – идентификатор модели продукций.	Добавление нового шаблона факта.
updateFactTemplate	id – идентификатор шаблона факта, name – наименование шаблона факта, description – описание шаблона факта.	Изменение шаблона факта.
removeFactTemplate	id – идентификатор шаблона факта.	Удаление шаблона факта.
addFactTemplateSlot	name – наименование слота шаблона факта, datatype_id – идентификатор типа данных, default_value – значение по умолчанию, description – описание слота шаблона факта, fact_template_id – идентификатор шаблона факта.	Добавление нового слота шаблона факта.
updateFactTemplateSlot	id – идентификатор слота шаблона факта, name – наименование слота шаблона факта, datatype_id – идентификатор типа данных, default_value – значение по умолчанию, description – описание слота шаблона факта.	Изменение слота шаблона факта.
removeFactTemplateSlot	id – идентификатор слота шаблона факта.	Удаление слота шаблона факта.
addRuleTemplate	name – наименование шаблона правила, description – описание шаблона правила, salience – значимость шаблона правила, condition_id – идентификатор условия (шаблон факта), operator – оператор условия, action_id – идентификатор действия (шаблон факта), function – функция (команда) действия, production_model_id – идентификатор модели продукций.	Добавление нового шаблона правила.
updateRuleTemplate	id – идентификатор шаблона правила, name – наименование шаблона правила, description – описание шаблона правила, salience – значимость шаблона правила, condition_id – идентификатор условия (шаблон факта), operator – оператор условия, action_id – идентификатор действия (шаблон факта), function – функция (команда) действия.	Изменение шаблона правила.
removeRuleTemplate	id – идентификатор шаблона правила.	Удаление шаблона правила.
addFact	name – наименование факта, description –	Добавление нового факта.

	описание факта, initial – указание начального факта, certainty_factor – коэффициент уверенности, fact_template_id – идентификатор шаблона факта, production_model_id – идентификатор модели продукций.	
updateFact	id – идентификатор факта, name – наименование факта, description – описание факта, initial – указание начального факта, certainty_factor – коэффициент уверенности.	Изменение факта.
removeFact	id – идентификатор факта.	Удаление факта.
addFactSlot	name – наименование слота факта, datatype_id – идентификатор типа данных, value – значение, description – описание слота факта, fact_ id – идентификатор факта.	Добавление нового слота факта.
updateFactSlot	id – идентификатор слота факта, name – наименование слота факта, datatype_id – идентификатор типа данных, value – значение, description – описание слота факта.	Изменение слота факта.
removeFactSlot	id – идентификатор слота факта.	Удаление слота факта.
addRule	name – наименование правила, description – описание правила, salience – значимость правила, certainty_factor – коэффициент уверенности, condition_id – идентификатор условия (факт), operator – оператор условия, action_id – идентификатор действия (факт), function – функция (команда) действия, rule_template_id – идентификатор шаблона правила, production_model_id – идентификатор модели продукций.	Добавление нового правила.
updateRule	id – идентификатор правила, name – наименование правила, description – описание правила, salience – значимость правила, certainty_factor – коэффициент уверенности, condition_id – идентификатор условия (факт), operator – оператор условия, action_id – идентификатор действия (факт), function – функция (команда) действия.	Изменение правила.
removeRule	id – идентификатор правила.	Удаление правила.
getAllFactTemplates	production_model_id – идентификатор модели продукций.	Получение списка всех шаблонов фактов у данной модели продукций.
getFactTemplate	id – идентификатор шаблона факта.	Получение шаблона факта по

		идентификатору.
getAllFactTemplate Slots	fact_template_id – идентификатор шаблона факта.	Получение списка всех слотов шаблона факта.
getFactTemplateSlot	id – идентификатор слота шаблона факта.	Получение слота шаблона факта по идентификатору.
getAllRuleTemplate s	production_model_id – идентификатор модели продукций.	Получение списка всех шаблонов правил у данной модели продукций.
getRuleTemplate	id – идентификатор шаблона правила.	Получение шаблона правила по идентификатору.
getAllFacts	production_model_id – идентификатор модели продукций.	Получение списка всех фактов у данной модели продукций.
getFact	id – идентификатор факта.	Получение факта по идентификатору.
getAllFactSlots	fact_id – идентификатор факта.	Получение списка всех слотов факта.
getFactSlot	id – идентификатор слота факта.	Получение слота факта по идентификатору.
getAllRules	production_model_id – идентификатор модели продукций.	Получение списка всех правил у данной модели продукций.
getRule	id – идентификатор правила.	Получение правила по идентификатору.

Приложение В
Пример CFM спецификаций

```
[Metadata]
;material
tempale_name=material
edited_by_user=Yes

[Fields]
cf=string
caption=string
type=string
mechanical-prop-strength-limit=String
mechanical-prop-yield-limit=String
resistance-prop-corrosion=String
resistance-prop-temperature=String
resistance-prop-wear=String
chemical-prop-alloying=val3:1
structure-prop-class=val4:1

[Captions]
cf=cf
caption=caption
type=type
mechanical-prop-strength-limit=mechanical-prop-strength-limit
mechanical-prop-yield-limit=mechanical-prop-yield-limit
resistance-prop-corrosion=resistance-prop-corrosion
resistance-prop-temperature=resistance-prop-temperature
resistance-prop-wear=resistance-prop-wear
chemical-prop-alloying=chemical-prop-alloying
structure-prop-class=structure-prop-class

[Values]
_1;_STEEL=val0:1=STRING_
_НЕ_ОПРЕДЕЛЕН;_LOW-ALLOY_STEEL=val1:1=STRING_
_СТАЛЬ;_STEEL=val2:1=STRING_
val3:1=LOW-ALLOY_STEEL
val4:1=1

[Metadata]
;exist-des
tempale_name=exist-des
edited_by_user=Yes

[Fields]
id-des=val0:1
caption-des=string
des-istochnik=string
des-orientacia=string
des-napravlenie=string
des-forma=string
des-kolichestvo=string
des-koncentracia-naprazheniy=string
des-mestopolozenie=val8:1
caption-dam=String
id-dam=String
caption-meh=String
des-cf=val9:1

[Captions]
id-des=id-des
caption-des=caption-des
des-istochnik=des-istochnik
des-orientacia=des-orientacia
des-napravlenie=des-napravlenie
des-forma=des-forma
des-kolichestvo=des-kolichestvo
des-koncentracia-naprazheniy=des-koncentracia-naprazheniy
des-mestopolozenie=des-mestopolozenie
```

```
caption-dam=caption-dam
id-dam=id-dam
caption-meh=caption-meh
des-cf=des-cf

[Values]
val0:1=BRITTLE_FAILURE
_MACROCRACK;_CONSTRUCTIVE_STRESS_CONCENTRATOR_FORMED_BY_THE_INTERSECTION_OF_HOLES=val1
:1=STRING_
_SURFACE_DAMAGE;_HOLEHOLE;_PERPENDICULARLY;_HOLE=val2:1=STRING_
_PERPENDICULARLY;_LONGITUDINAL;_ПЕРПЕНДИКУЛЯРНО=val3:1=STRING_
_LONGITUDINAL;_SEMI-ELLIPTIC=val4:1=STRING_
_SINGLE;_CORROSION_FATIGUE=val5:1=STRING_
_CORROSION_FATIGUE;_LESS_THAN_2=val6:1=STRING_
_CRACK;_CORROSION_FATIGUE=val7:1=STRING_
val8:1=CRACK
val9:1=1

[Generalize rules]
#Knowledge base 10234235031
Generalized rule 1=mechanical-stress-const & technological-environment & material &
making-defects => exist-event & exist-meh:mechanical-stress-const,technological-
environment,material,making-defects:exist-event,exist-meh
Generalized rule 2=exist-event => exist-kin:exist-event:exist-kin
Generalized rule 3=exist-kin => exist-event & exist-event:exist-kin:exist-event,exist-
event
Generalized rule 4=exist-kin & exist-event => exist-event:exist-kin,exist-event:exist-
event
Generalized rule 5=exist-dam => exist-des & exist-event:exist-dam:exist-des,exist-
event
Generalized rule 6=exist-dam & exist-event => exist-des & exist-event:exist-dam,exist-
event:exist-des,exist-event
Generalized rule 7=exist-event & exist-des => exist-des & exist-event:exist-
event,exist-des:exist-des,exist-event
Generalized rule 8=exist-dam & exist-event => exist-event:exist-dam,exist-event:exist-
event
Generalized rule 9=mechanical-stress-const & technological-environment & incident-
object & material => exist-event:mechanical-stress-const,technological-
environment,incident-object,material:exist-event
Generalized rule 10=mechanical-stress-const & technological-environment & incident-
object & material & exist-event => exist-event:mechanical-stress-const,technological-
environment,incident-object,material,exist-event:exist-event
Generalized rule 11=exist-kin => exist-dam & exist-event:exist-kin:exist-dam,exist-
event
Generalized rule 12=exist-kin & exist-dam & exist-event => exist-event:exist-
kin,exist-dam,exist-event:exist-event
```

Приложение Г
Примеры экранных форм интеллектуальной системы идентификации техничсеких состояний конструкций INFOT-3

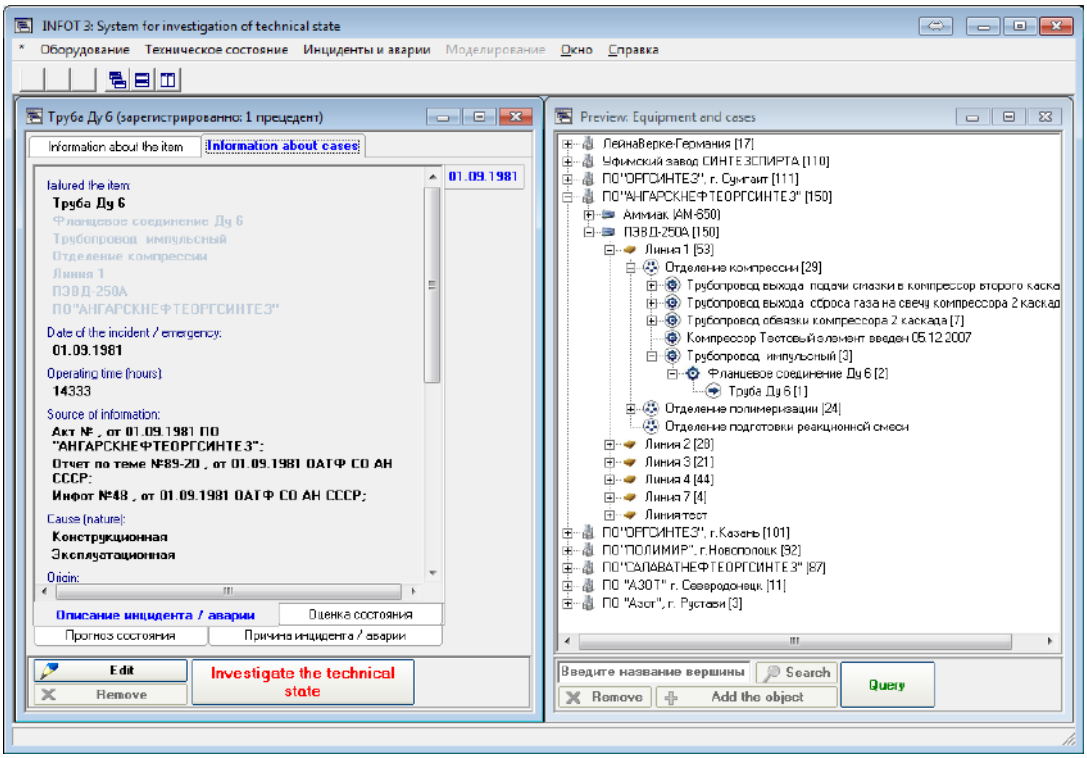


Рис. Г.1. Просмотр информации об оборудовании и прецедентах

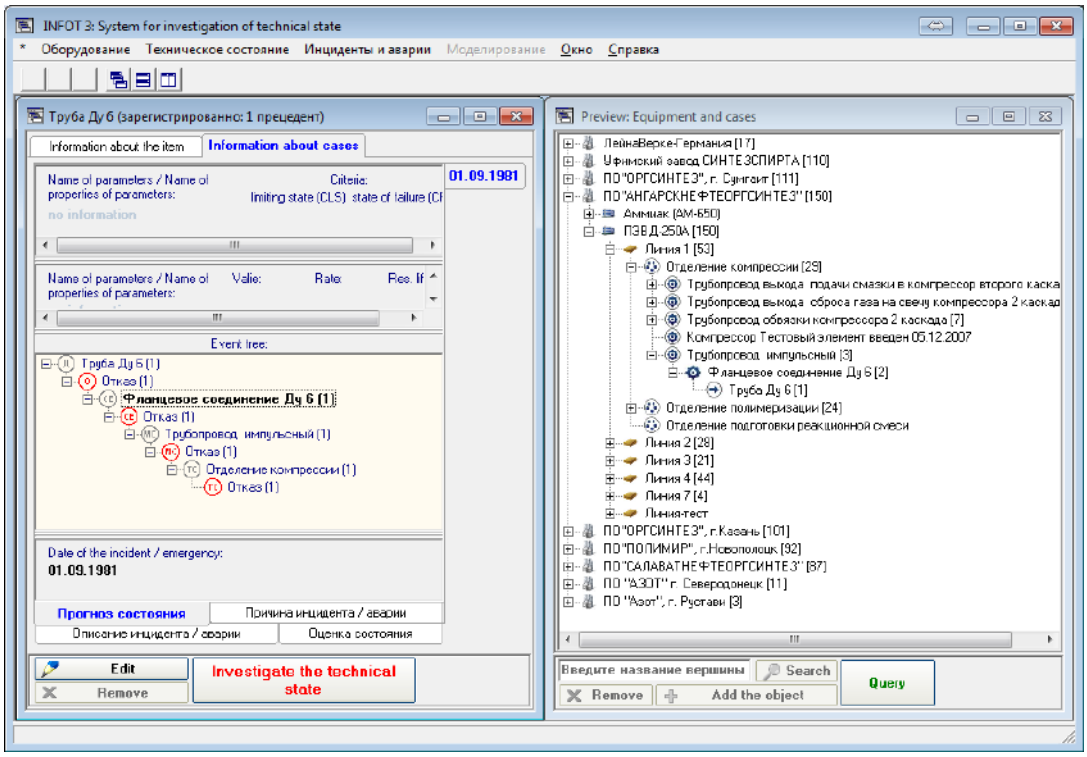


Рис. Г.2. Прогноз изменения технического состояния детали

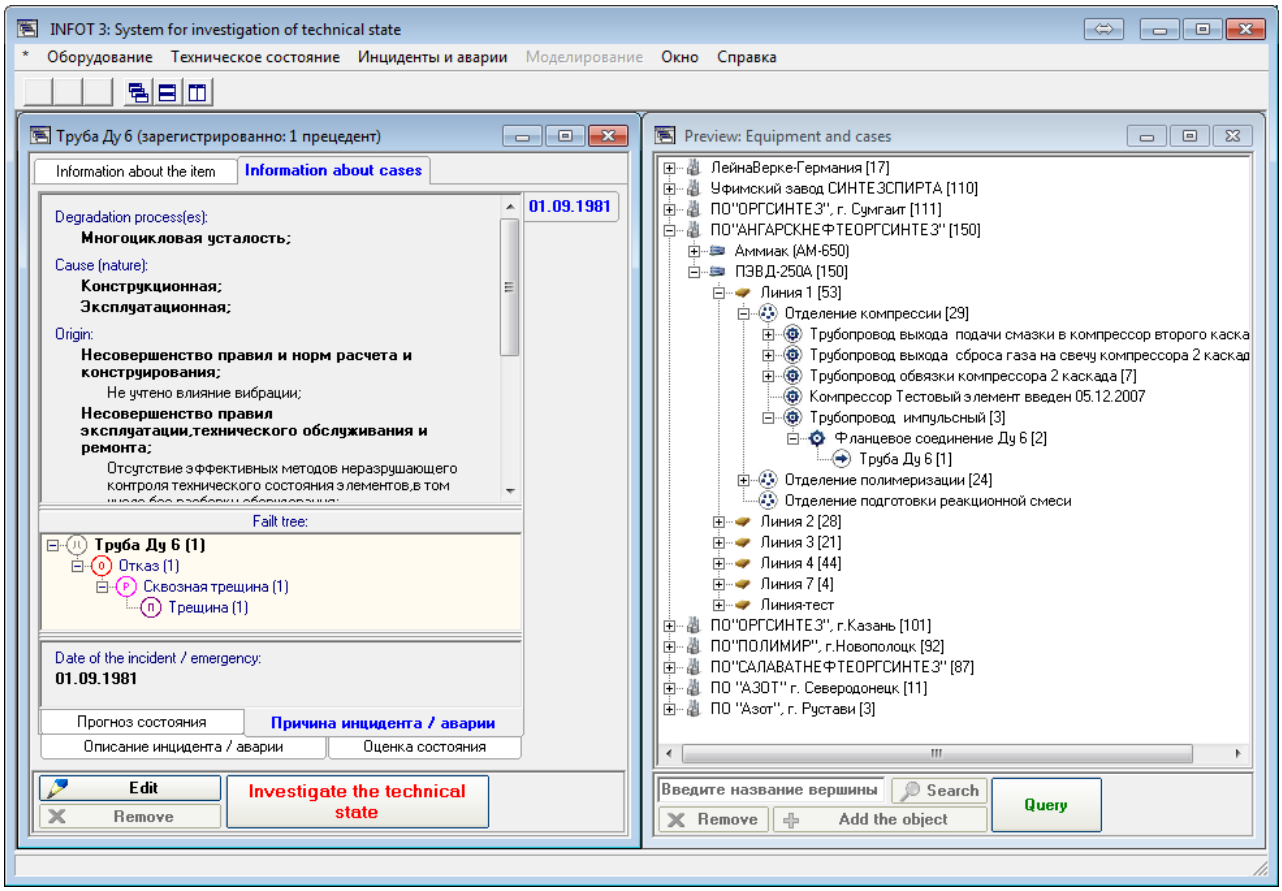


Рис. Г.3. Определение причины технического состояния детали

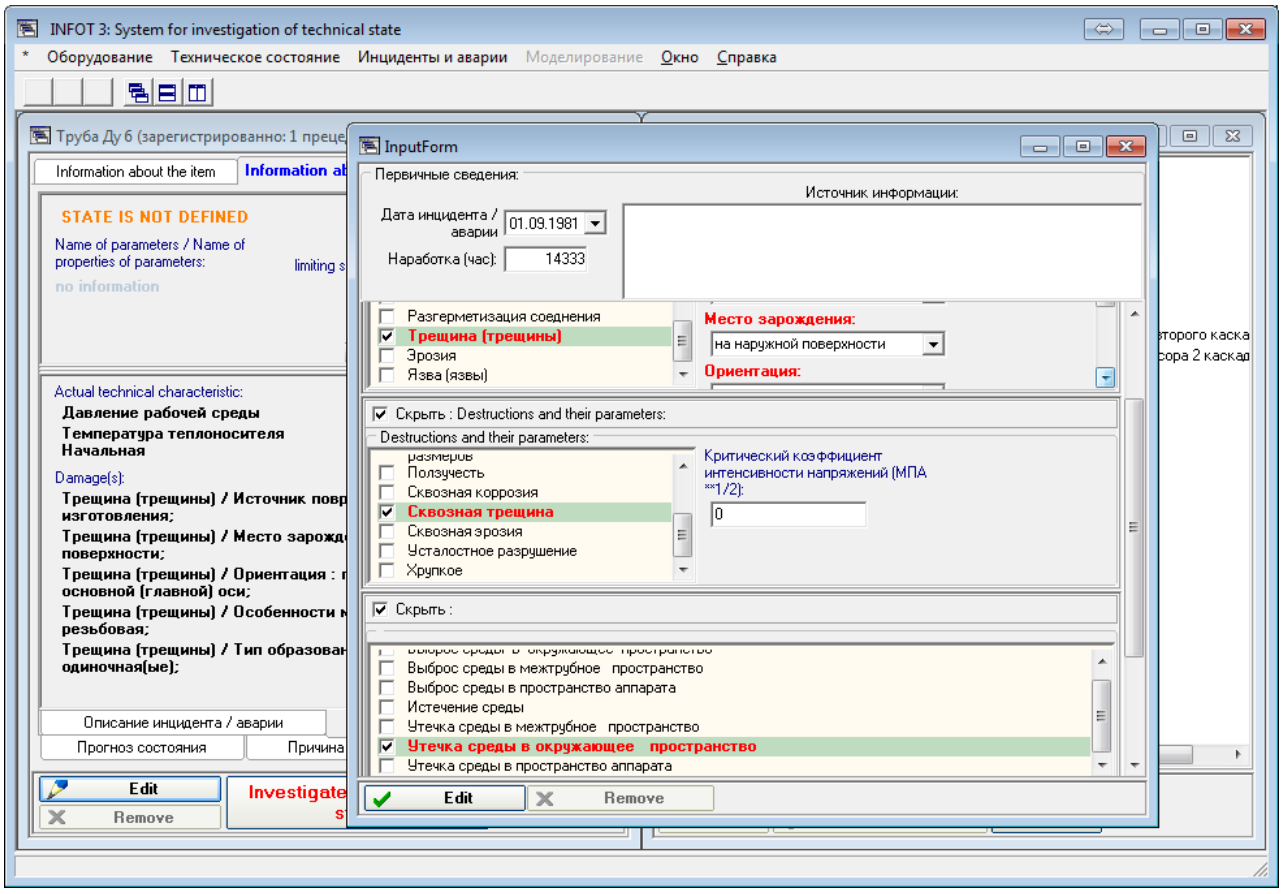


Рис. Г.4. Описание внешних проявлений состояний

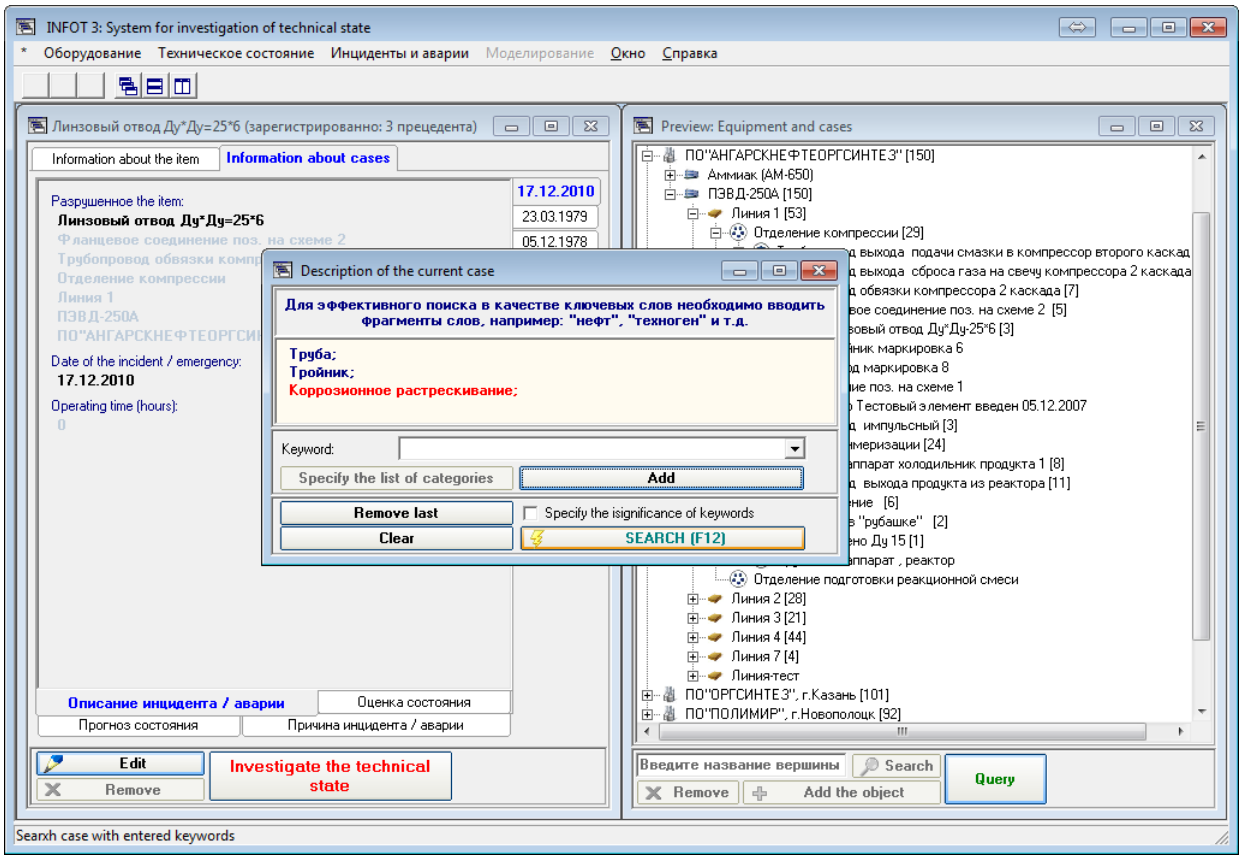


Рис. Г.5. Формирование запроса

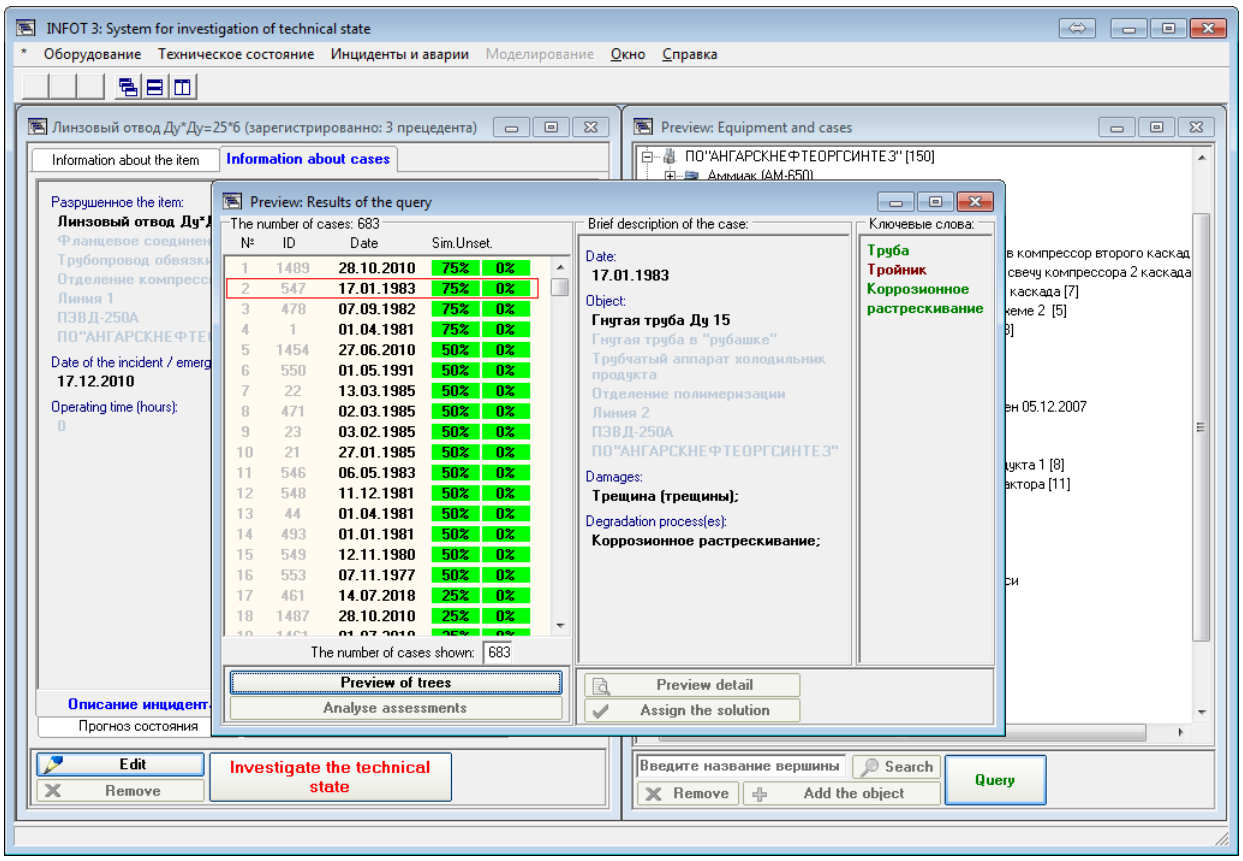


Рис. Г.6. Просмотр результатов запроса

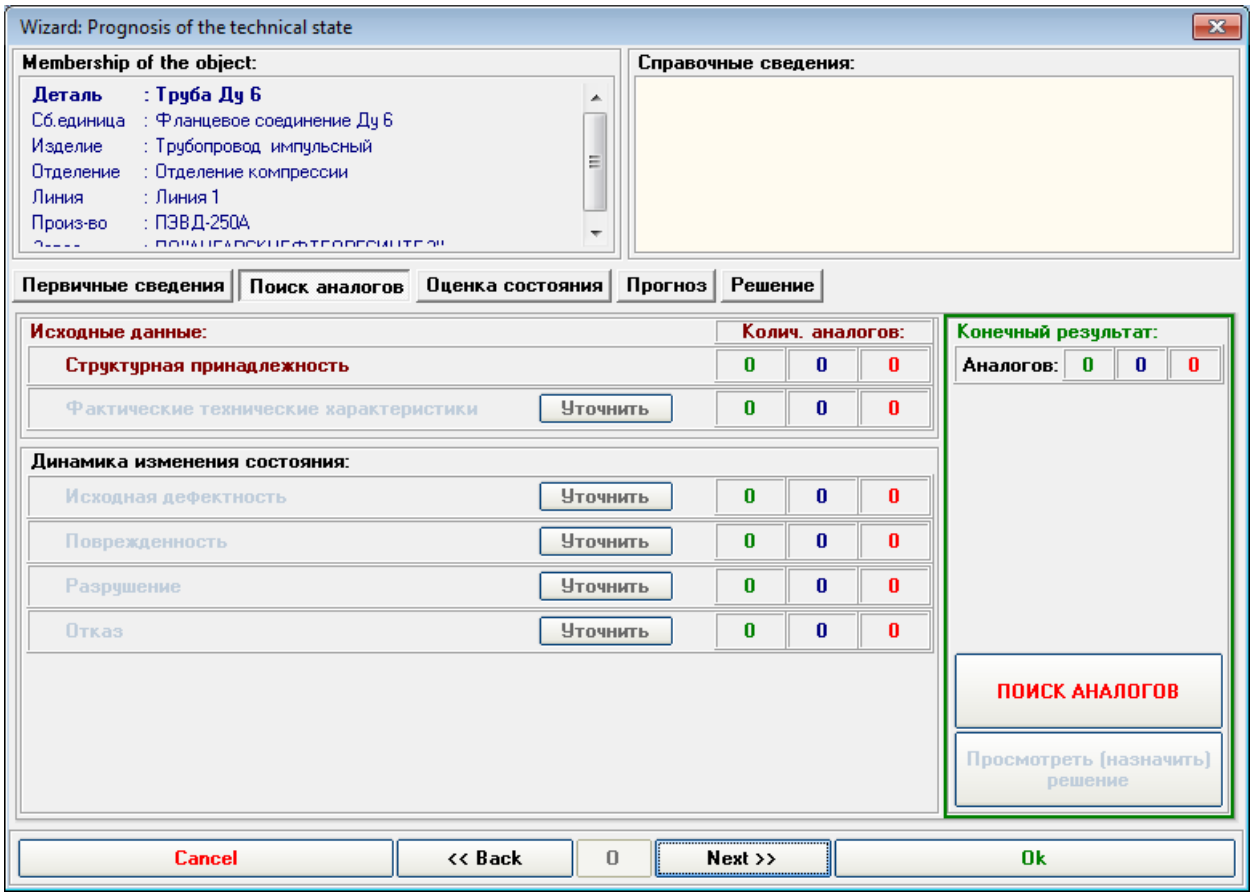


Рис. Г.7. Схема исследования технического состояния по прецедентам

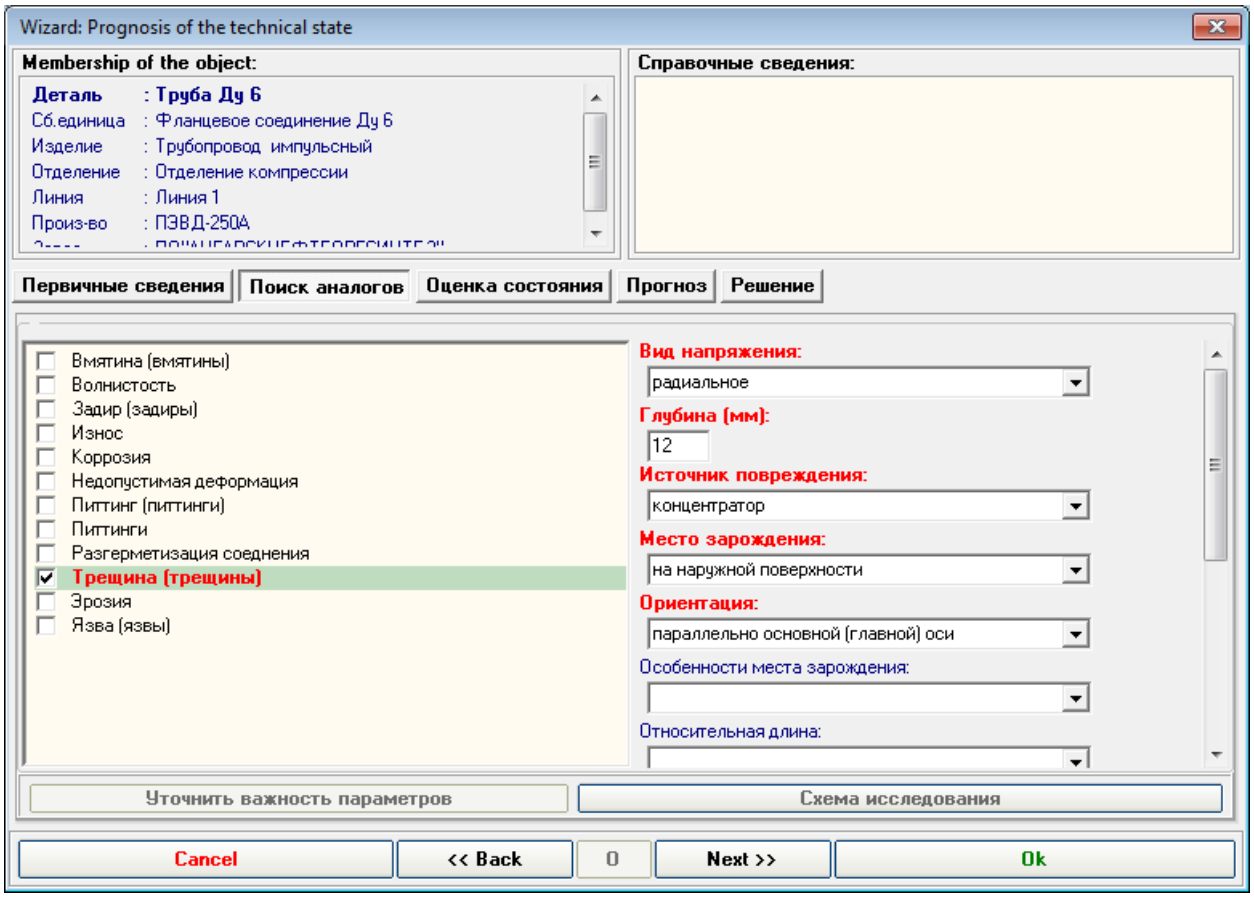


Рис. Г.8. Описание проявлений повреждений для поиска прецедентов

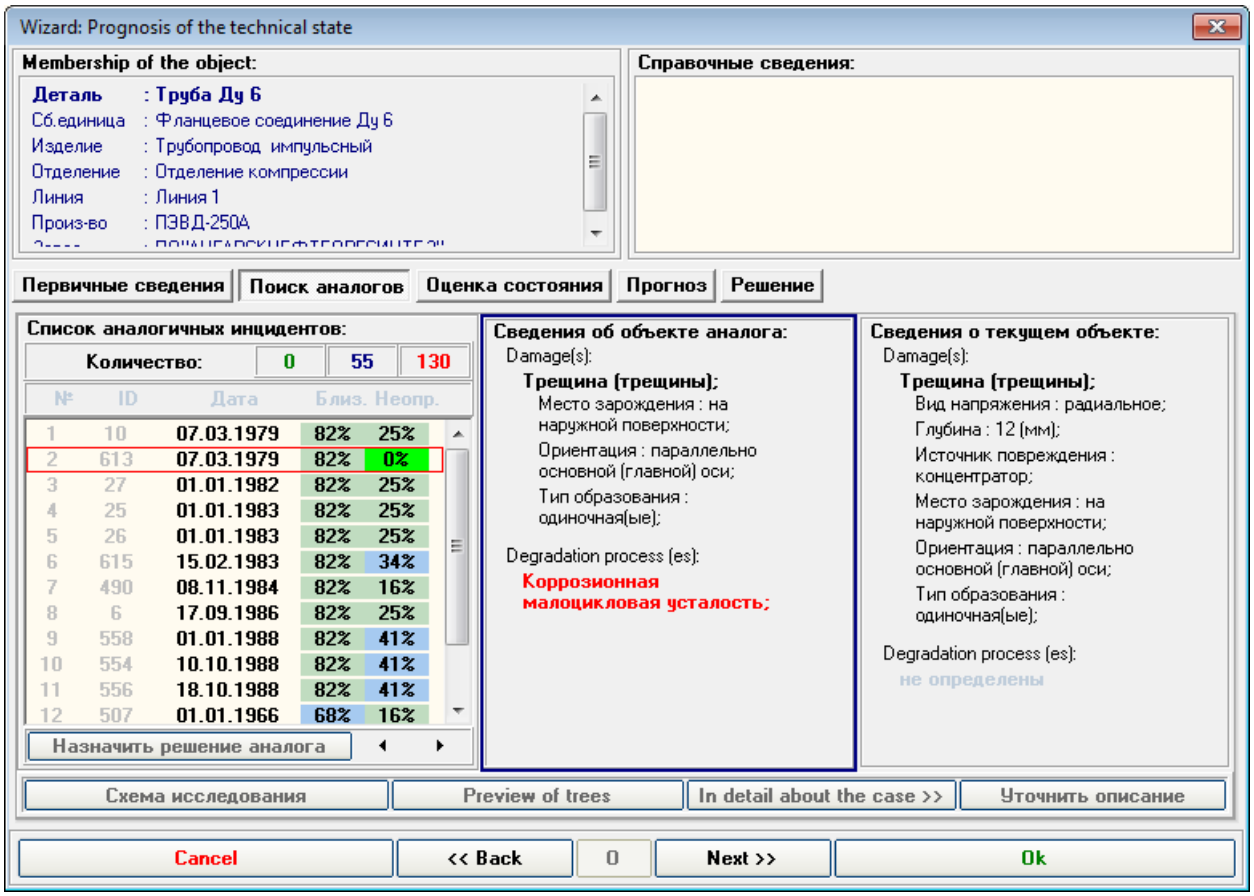


Рис. Г.9. Просмотр результатов поиска аналогов по признакам повреждений

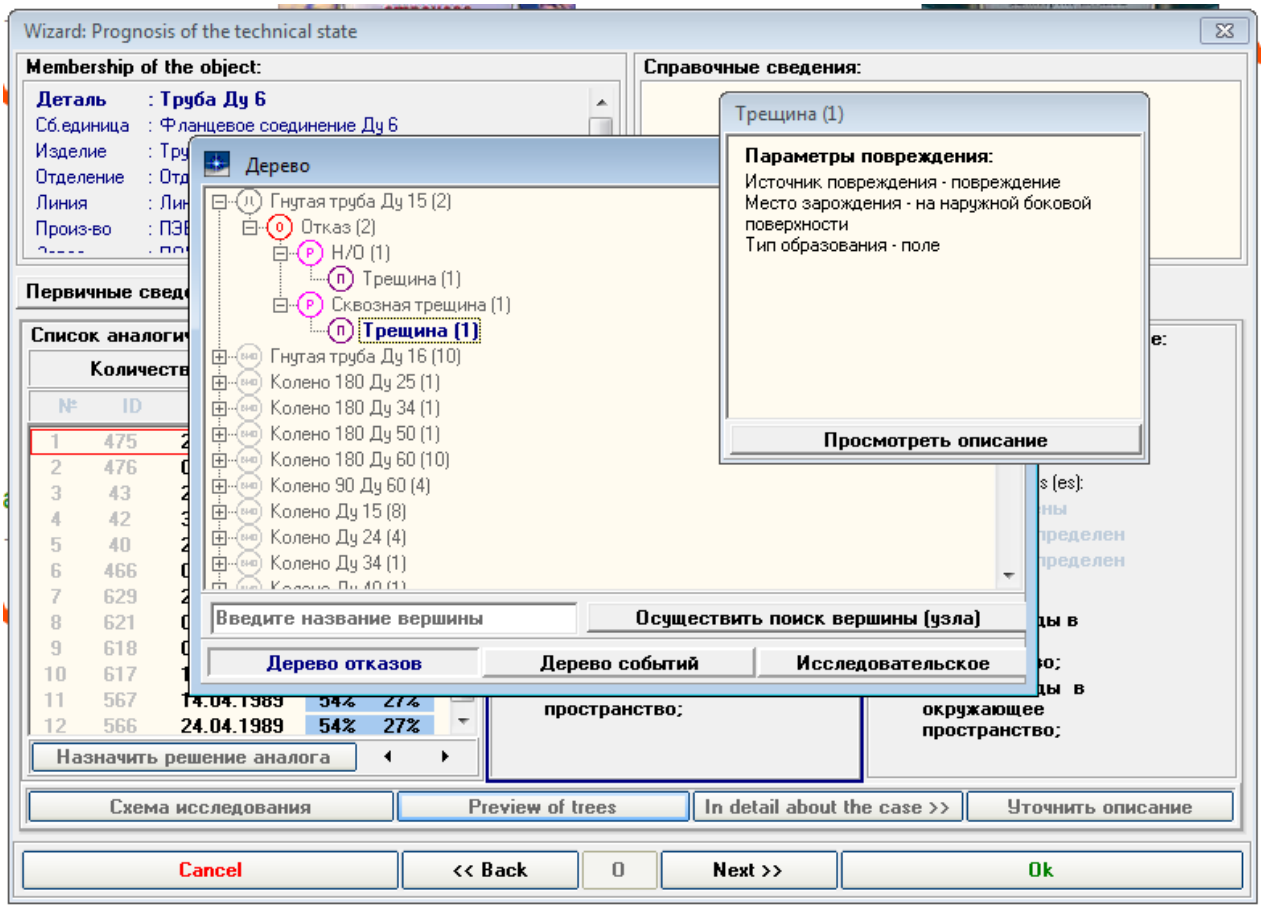


Рис. Г.10. Просмотр деревьев отказов для прецедентов

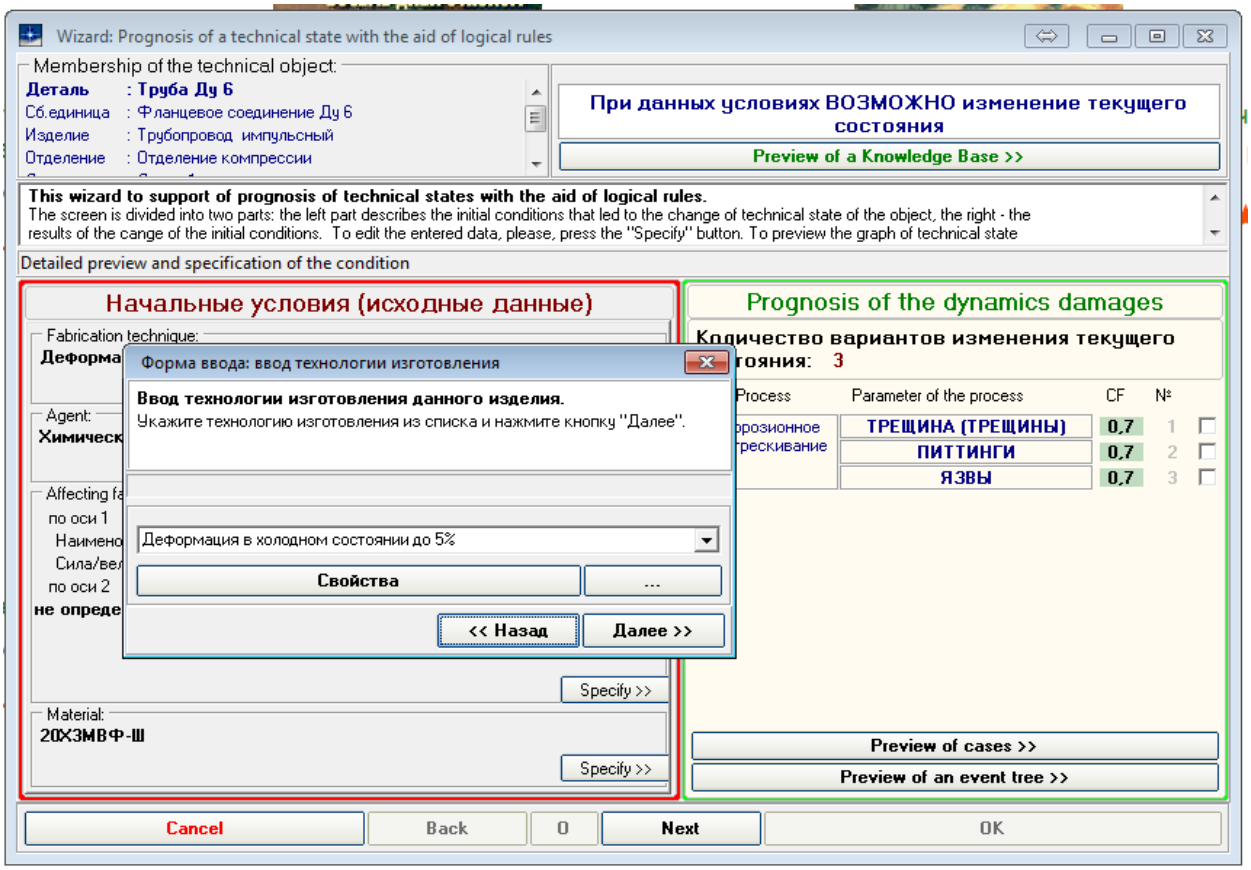


Рис. Г.11. Описание начальных условий для продукционной экспертной системы

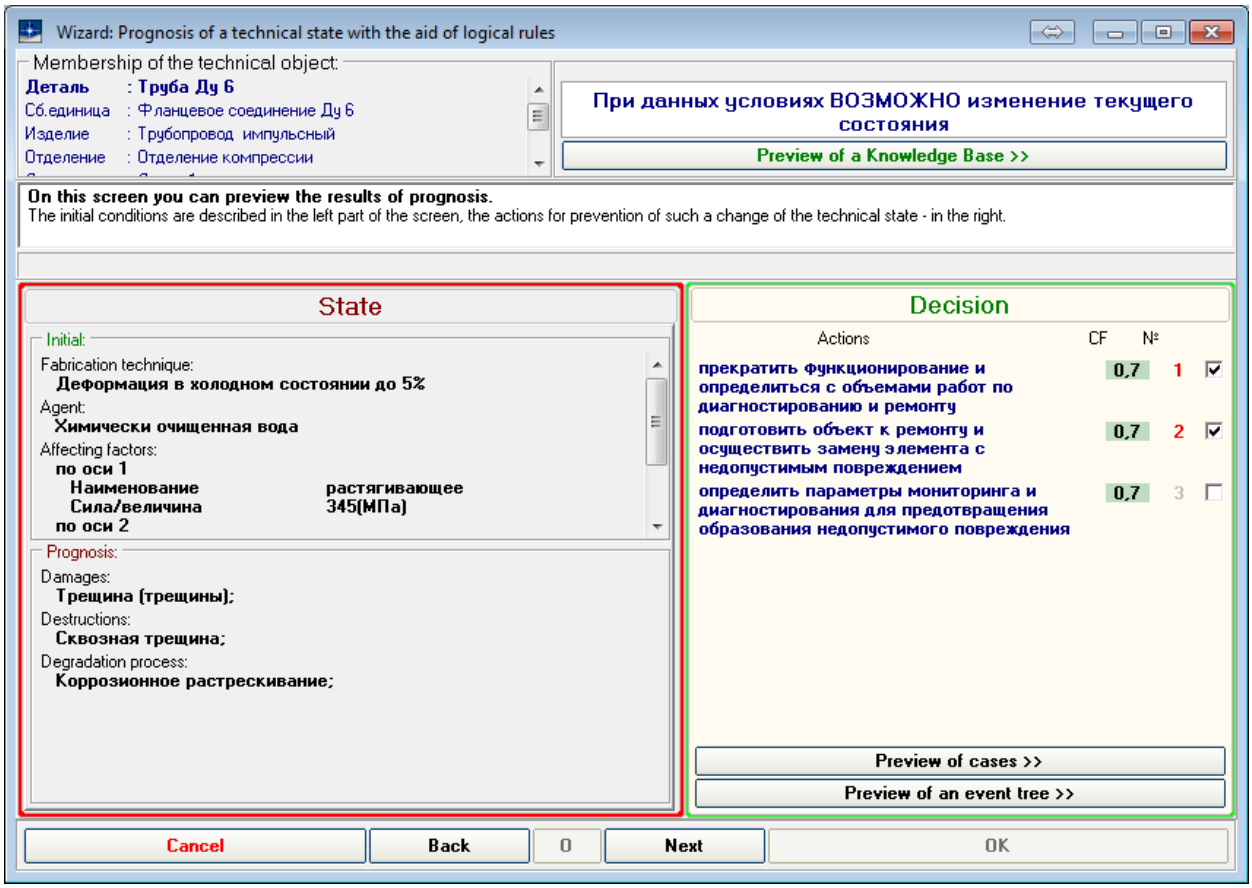


Рис. Г.12. Результаты работы продукционной экспертной системы

Приложение Д
Модуль интерпретации признаков эмоций

В приложении описано применение предлагаемых методов и средств при прототипировании одного из модулей системы поддержки кадрового сотрудника (рекрутера) HR-Robot (ООО «Смарт технологии»). Основное назначение HR-Robot – поддержка принятия решений при отборе кандидатов на вакансии и проверка персонала на мотивацию (исследование психологической ситуации в коллективе) на основе анализа эмоций. Система состоит из следующих основных модулей: обработки видео, обнаружение признаков эмоций, интерпретация признаков эмоций, отображения и валидации данных, формирования вопросов интервью, определения мотивации, анализа видео-интервью психологами, проведения экспериментов, хранения информации, модуль управления, веб-интерфейс пользователя, формирования отчетов.

При разработке модуля интерпретации признаков эмоций («EmSi-Interpreter») был применен предлагаемый подход [193, 201]. Последовательность основных этапов решения задачи прототипирования базы знаний данного модуля соответствует методу и методике (см. 4.1) (Рис.Д.1).

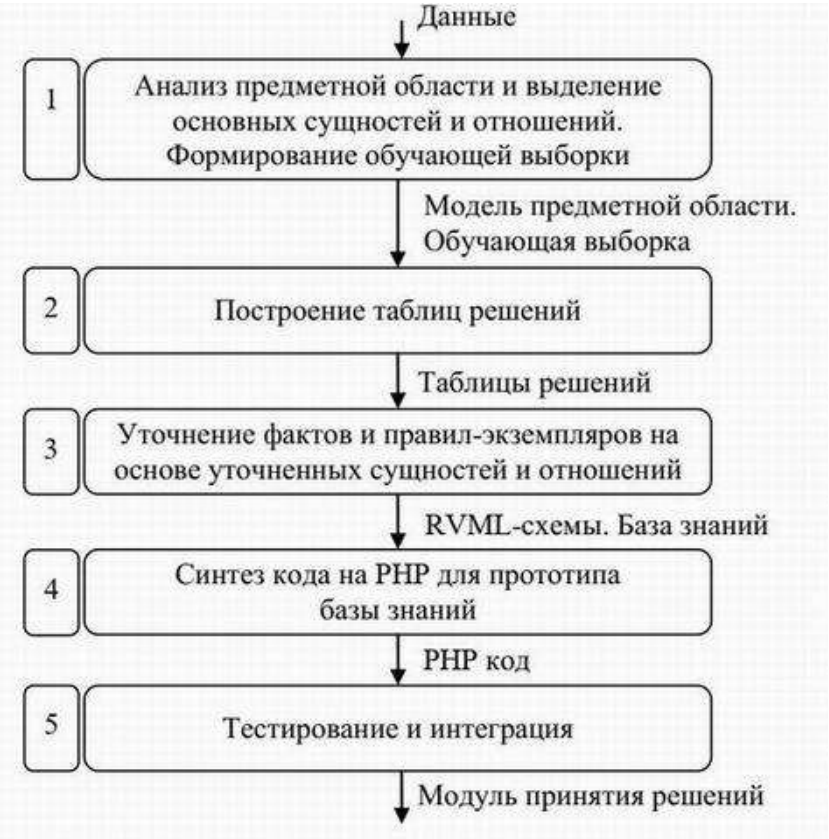


Рис. Д.1. Схема разработки прототипа базы знаний для модуля интерпретации признаков эмоций

В качестве исходной информации использовались диаграммы классов, а также экспертная информация, которая в дальнейшем была представлена в форме таблиц решений специального вида.

Признаки эмоций и их интерпретация

В качестве теоретической основы для решения задачи выявления признаков эмоций и их интерпретации использовались данные практикующего психолога-эксперта, а также элементы теории Пола Экмана [360], связанные с интерпретацией движений лицевых мышц.

В частности, результате анализа информации, предоставленной от психолога-эксперта, был получен структурированный перечень признаков эмоций человека. Пример перечня признаков представлен в таблицах ниже.

Таблица Д.1. Перечень признаков эмоции «страх»

Зона	**Описание изменений**	**Описание изменений для камеры**	**Время на оценку**
Лоб	Появление горизонтальных складок/морщин	Ширина лба – уменьшение	1-3 сек
Брови	Брови почти прямые и кажутся несколько приподнятыми, внутренние уголки бровей сдвинуты друг к другу	Линия бровей –движение вверх ИЛИ Движение к центру	1-3 сек
Глаза	Глаза при страхе раскрыты шире, чем в спокойном состоянии, становится виден белок глаза сверху, т.к. нижнее веко напряжено, а верхнее приподнято	Ширина глаз – увеличение ширины Верхнее веко – движение вверх	1-3 сек
Носогубная складка	Могут появляться носогубные складки от приоткрывания рта	Носогубная складка – движение от центра в стороны	1-5 сек
Рот (губы)	Рот приоткрыт, губы напряжены и слегка растянуты, форма рта похожа на овал	Длина рта – увеличение ИЛИ Уголки рта – движение от центра в стороны	1-5 сек

Таблица Д.2. Перечень признаков эмоции «гнев»

Зона	**Описание изменений**	**Описание изменений для камеры**	**Время на оценку**
Лоб	Мышцы лба сдвигаются внутрь и вниз, придавая лицу нахмуренное и угрожающее выражение глаз.	Ширина лба – уменьшение	1-5 сек
Брови	Брови опущены и сведены, между бровями появляются вертикальные морщины	Линия бровей – движение к центру и вниз	1-5 сек
Глаза	Нижние веки напряжены, они могут быть или не быть приподняты. Верхние веки напряжены, они могут быть или не быть опущены в результате опускания бровей. Глаза глядят пристально и могут быть слегка выкачены наружу или глаза прищурены.	Нижнее веко – движение к центру и вверх Верхнее веко – движение вниз Зрачок – фиксируется и не двигается	1-2 сек
Носогубная складка	Неярко выражена	-	-
Рот (губы)	Плотно сжаты, уголки губ прямые или спущенные вниз, либо губы могут быть раздвинуты (образуя прямоугольный рот) и напряжены.	Ширина рта – уменьшается Форма рта – прямоугольник (учитывать, если рот в нормальном состоянии не имеет форму прямоугольника)	1-5 сек

Таблица Д.3. Перечень признаков эмоции «отвращение (презрение)»

Зона	**Описание изменений**	**Описание изменений для камеры**	**Время на оценку**
Лоб	Нет выраженный реакций	-	-
Брови	Брови опущены, в результате чего происходит опускание верхних век.	Линия бровей – движение вниз	1-3 сек
Глаза	Проявляются морщинки на коже под нижними веками, а нижние веки приподняты, но не напряжены	Верхнее веко – движение вниз (можно не учитывать) Нижнее веко – движение вверх	1-3 сек
Носогубная	Нос сморщен.	Носогубная складка –	1-3 сек

складка	Носогубная складка ярко выражена, поднимается вверх. Щеки приподняты.	движение вверх одна или сразу обе.	
Крылья носа		Движение вверх	
Рот (губы)	Верхняя губа несколько приподнята, уголки губ сжаты. Нижняя губа также приподнята и придвинута к верхней губе или же опущена и слегка выдвинута вперед	Верхняя губа и нижняя губа –движение вверх Уголки рта – движение вниз Или Один уголок рта – движение вниз или без движения, Другой уголок рта – движение вверх	1-3 сек

Таблица Д.4. Перечень признаков эмоции «печаль»

Зона	**Описание изменений**	**Описание изменений для камеры**	**Время на оценку**
Лоб	Нет выраженных изменений	-	-
Брови	внутренние концы бровей приподняты и сведены к переносице	Движение бровей – к центру и вверх Внутренние уголки бровей – движение к центру и вверх Внешние уголки бровей – движение вниз	1-5 сек
Глаза	Глаза слегка сужены, уголки глаз опущены	Внешние уголки глаз – движение вниз	1-5 сек
Носогубная складка	Неярко выражена	-	-
Рот (губы)	уголки рта опущены	Уголки рта – движение вниз	1-5 сек

Таблица Д.5. Перечень признаков эмоции «радость»

Зона	**Описание изменений**	**Описание изменений для камеры**	**Время на оценку**
Лоб	Нет выраженных изменений	-	-
Брови	Нет выраженных изменений	-	-
Глаза	Нижние веки могут быть подняты, но не напряжены; под ни ми появляются морщинки.	Нижние веки – движение вверх Внешние уголки глаз – движение вверх	1-2 сек
Носогубная складка	Морщины (носогубные складки) идут вниз от	Движение от центра в стороны обеих складок	1-2 сек

	носа к областям, находящимся у краев рта. Щеки приподняты.	одновременно (можно не учитывать)	
Рот (губы)	Уголки рта оттянуты назад и вверх. Рот может быть приоткрыт или закрыт; в первом случае будут видны зубы, во втором – нет.	Длина рта – увеличивается Движение уголков рта – от центра в стороны и вверх Ширина рта – увеличивается (верхняя губа движение вверх, нижняя губа движение вниз)	1-2 сек

Таблица Д.6. Перечень признаков эмоции «удивление»

Зона	**Описание изменений**	**Описание изменений для камеры**	**Время на оценку**
Лоб	Появление горизонтальных складок/морщин	Ширина лба - уменьшение	1 сек
Брови	Брови высоко подняты и изогнуты. Кожа под бровями натянута.	Линия бровей - движение вверх Внутренние уголки бровей – движение вверх	1 сек
Глаза	Глаза расширяются и округляются. Веки открыты; верхние веки подняты, нижние опущены; белок глаз — склеру — можно видеть над радужной оболочкой, а нередко и под ней	Верхнее веко – движение вверх Нижнее веко – движение вниз Ширина глаз – увеличение	1 сек
Носогубная складка	Нет ярких выражений	-	-
Рот (губы)	Приоткрытый рот принимает овальную форму. Нижняя челюсть опускается, так что губы и зубы размыкаются, а рот находится в ненапряженном состоянии.	Ширины рта – увеличение (Верхняя губа- движение вверх И Нижняя губа –движение вниз) ИЛИ Нижняя губа –движение вниз Уголки губ – отсутствие движения Форма рта - овал	1-2 сек
Подбородок		Линия подбородка – движение вниз	

Помимо информации от психолога-эксперта в проекте была использована методика Пола Экмана в форме системы кодирования лицевых движений (СКЛиД) (Facial Action Coding System (FACS)) [360]. Данная методика представляет собой систему Action Units (AU) для классификации выражений лица человека.

Пример описания эмоции с использованием AU представлен в таблице Д.7, описание AU в таблице Д.8.

Таблица Д.7. Перечень Action Units эмоций (* - интенсивность любая)

Эмоция	Прототип	Главные варианты
Страх	1+2+4+5*+20*+25, 26, или 27 1+2+4+5*+25, 26, или 27	1+2+4+5*+L или R20*+25, 26, или 27 1+2+4+5* 1+2+5*, с/без 25, 26, 27 5*+20* с/без 25, 26, 27
Удивление	1+2+5B+26 1+2+5B+27	1+2+5B 1+2+26 1+2+27 5B+26 5B+27
Страх	1+2+4+5*+20*+25, 26, или 27 1+2+4+5*+25, 26, или 27	1+2+4+5*+L или R20*+25, 26, или 27 1+2+4+5* 1+2+5*, с/без 25, 26, 27 5*+20* с/без 25, 26, 27
Радость	6+12* 12C/D	
Печаль	1+4+11+15B с/без 54+64 1+4+15* с/без 54+64 6+15* с/без 54+64	1+4+11 с/без 54+64 1+4+15B с/без 54+64 1+4+15B+17 с/без 54+64 11+15B с/без 54+64 11+17
	25 или 26 могут встречаться со всеми прототипами и основными вариантами	
Отвращение	9 9+16+15, 26 9+17 10* 10*+16+25, 26 10+17	
Презрение	9 или U10 12 U14 или B14	L12+L14 R12+R14
Гнев	4+5*+7+10*+22+23+25, 26 4+5*+7+10*+23+25, 26 4+5*+7+23+25, 26	Любые из прототипов без любой из следующих ДЕ: 4, 5, 7 или 10.

Эмоция	Прототип	Главные варианты
	4+5*+7+17+23 4+5*+7+17+24 4+5*+7+23 4+5*+7+24	

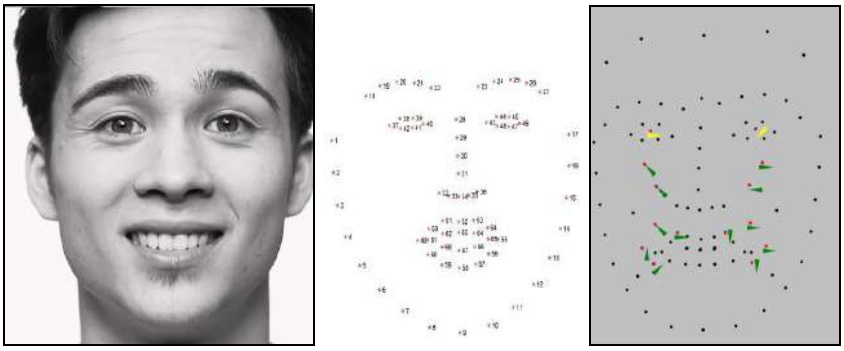
Таблица Д.8. Список основных двигательных единиц и двигательных дескрипторов (с указанием мышц)

№ ДЕ	Оригинал	Перевод	Мышечная основа
0	Neutral face	Нейтральное лицо	
1	Inner brow raiser	Подниматель внутренней части брови	лобная мышца (медиальная часть)
2	Outer brow raiser	Подниматель внешней части брови	лобная мышца (латеральная часть)
4	Brow lowerer	Опускатель брови	мышца гордецов; мышца, опускающая бровь; мышца, сморщивающая бровь
5	Upper lid raiser	Подниматель верхнего века	мышца, поднимающее верхнее веко
6	Cheek raiser	Подниматель щеки	круговая мышца глаза (глазничная часть)
7	Lid tightener	Натягиватель века	круговая мышца глаза (вековая часть)
8	Lips toward each other	Губы навстречу друг другу	круговая мышца рта
9	Nose wrinkler	Сморщиватель носа	мышца, поднимающая верхнюю губу и крыло носа
10	Upper lip raiser	Подниматель верхней губы	мышца, поднимающая верхнюю губу (также известна как квадратная мышца верхней губы), нижнеглазничная головка
11	Nasolabial deepener	Углубитель носогубной складки	малая скуловая мышца
12	Lip corner puller	Подниматель уголка губы	большая скуловая мышца
13	Sharp lip puller	Острый подниматель уголка губы	мышца, поднимающая угол рта (также известна как собачья мышца)
14	Dimpler	Ямочка	щёчная мышца (также известна как мышца трубачей)
15	Lip corner depressor	Опускатель уголка губы	мышца, опускающая угол рта (также известна как треугольная мышца рта)
16	Lower lip depressor	Опускатель нижней губы	мышца, опускающая нижнюю губу (также известна как четырёхугольная мышца нижней губы)
17	Chin raiser	Подниматель подбородка	подбородочная мышца

№ ДЕ	Оригинал	Перевод	Мышечная основа
18	Lip pucker	Сморщиватель губ	резцовая мышца верхней губы и резцовая мышца нижней губы
19	Tongue show	Показ языка	
20	Lip stretcher	Растягиватель губ	мышца смеха с/без подкожной мышцей шеи
21	Neck tightener	Натягиватель шеи	подкожная мышца шеи
22	Lip funneler	Губы воронкой	круговая мышца рта
23	Lip tightener	Натягиватель губ	круговая мышца рта
24	Lip pressor	Сжиматель губ	круговая мышца рта
25	Lips part	Губы разведены	мышца, опускающая нижнюю губу или расслабление подбородочной мышцы или круговой мышцы рта
26	Jaw drop	Челюсть опущена	жевательная мышца, расслабленные височная мышца и медиальная крыловидная мышца
27	Mouth stretch	Рот широко открыт	медиальная крыловидная мышца и латеральная крыловидная мышца, двубрюшная мышца
28	Lip suck	Втягивание губ	круговая мышца рта
29	Jaw thrust	Нижняя челюсть вперёд	
30	Jaw sideways	Челюсть в бок	
31	Jaw clencher	Сжиматель челюстей	жевательная мышца
32	Lip bite	Покусывание губы	
33	Cheek blow	Выдувание	
34	Cheek puff	Раздувание щёк	
35	Cheek suck	Втягивание щёк	
36	Tongue bulge	Язык высунут	
37	Lip wipe	Облизывание губ	
38	Nostril dilator	Расширитель ноздрей	носовая мышца (внутренняя, или крыльная часть)
39	Nostril compressor	Суживатель ноздрей	носовая мышца (наружная, или поперечная часть) и мышца, опускающая перегородку носа
41	Glabella lowerer	Опускатель надпереносья	Отдельная часть ДЕ 4: мышца гордецов
42	Inner	Опускатель внутренней	Отдельная часть ДЕ 4: мышца,

№ ДЕ	Оригинал	Перевод	Мышечная основа
	eyebrow lowerer	части брови	опускающая бровь
43	Eyes closed	Глаза закрыты	Расслабление мышцы, поднимающей верхнее веко
44	Eyebrow gatherer	Сведение бровей	Отдельная часть ДЕ 4: мышца, сморщивающая бровь
45	Blink	Моргание	Расслабление мышцы, поднимающей верхнее веко; сокращение круговой мышцы глаза (вековая часть)
46	Wink	Подмигивание	круговая мышца глаза

Для идентификации и отслеживания проявлений эмоций было разработано программное обеспечение для нахождения и распознавания лицевых опорных точек (Facial landmarks (FL)), которые являются основой для получения информации об эмоциональном состоянии человека (Рис.Д.2-Д.3).



а) б) в)

Рис. Д.2. Пример использования лицевых опорных точек: а) изображение лица; б) цифровая маска с опорными точками; в) цифровая маска с векторами движения опорных точек

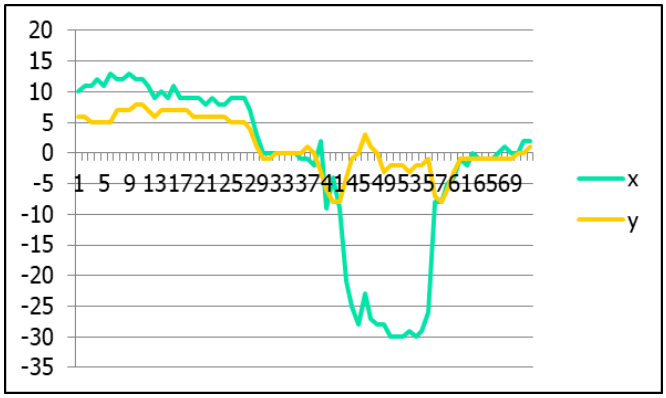


Рис. Д.3. Пример графика отслеживания движения опорной точки

В результате обобщения информации психолога-эксперта, а также результатов экспериментов с тестовым видео, были определены основные

направления движения частей лица и их элементов, соответствующие определенным эмоциям (Рис.Д.4). В дальнейшем полученные данные были использованы для создания базы знаний модуля интерпретации признаков.

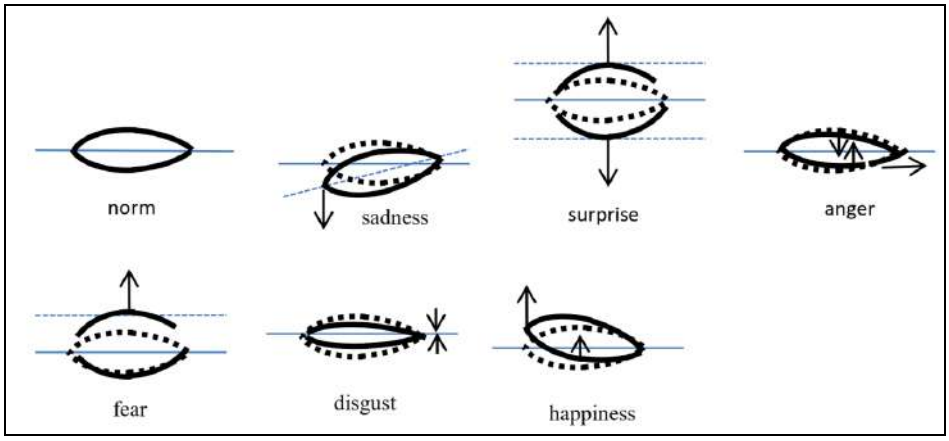


Рис. Д.4. Пример движений глаза, соответствующих проявлениям разных эмоций

Создание прототипа базы знаний

Как было описано выше процессу разработки прототипа базы знаний (Рис.Д.5) предшествовал этап анализа предметной области, включающий построение концептуальных моделей, а также сбор информации о существующих методиках оценки эмоций. Основным результатом этого этапа стала концептуальная модель в форме диаграммы классов, описывающая основные части лица и их основные элементы. Данная модель в совокупности с информацией от специалиста психолога интерпретировались как вычислительно-независимая модель (Рис. Д.5, блок 1).

Далее была произведена формализация этой модели в форме таблиц решений особого вида, которые представляли собой платформо-независимую модель, т.к. в контексте задачи обеспечивали прямое отображение информации в логические правила.

Выбор формы таблиц решений для конкретизации предметных знаний обусловлен простотой и популярностью данного метода у специалистов предметников, а также возможностью использования распространенных редакторов таблиц (например, Microsoft Excel) для их формирования с сохранением в виде CSV файлов.

При этом использовался специализированный вид таблиц решений (Рис.Д.6), обладающий следующими основными особенностями: возможностью включения столбца с именами правил в структуру таблицы; возможностью индикации

зависимых столбцов символом "#"; возможностью задания составных имен столбцов, включающих имя сущности (или имя класса) и имя ее свойства, разделенные строкой "::"; отсутствием ограничений на значения в ячейках, т. е. они могут состоять не только из набора значений {да, нет}, представляя собой вырожденную форму таблиц истинности, но позволяют использовать конкретные произвольные значения в качестве значений ячеек, а не только значения, указывающие на наличие или отсутствие определенного свойства (компонента) в структуре правила.

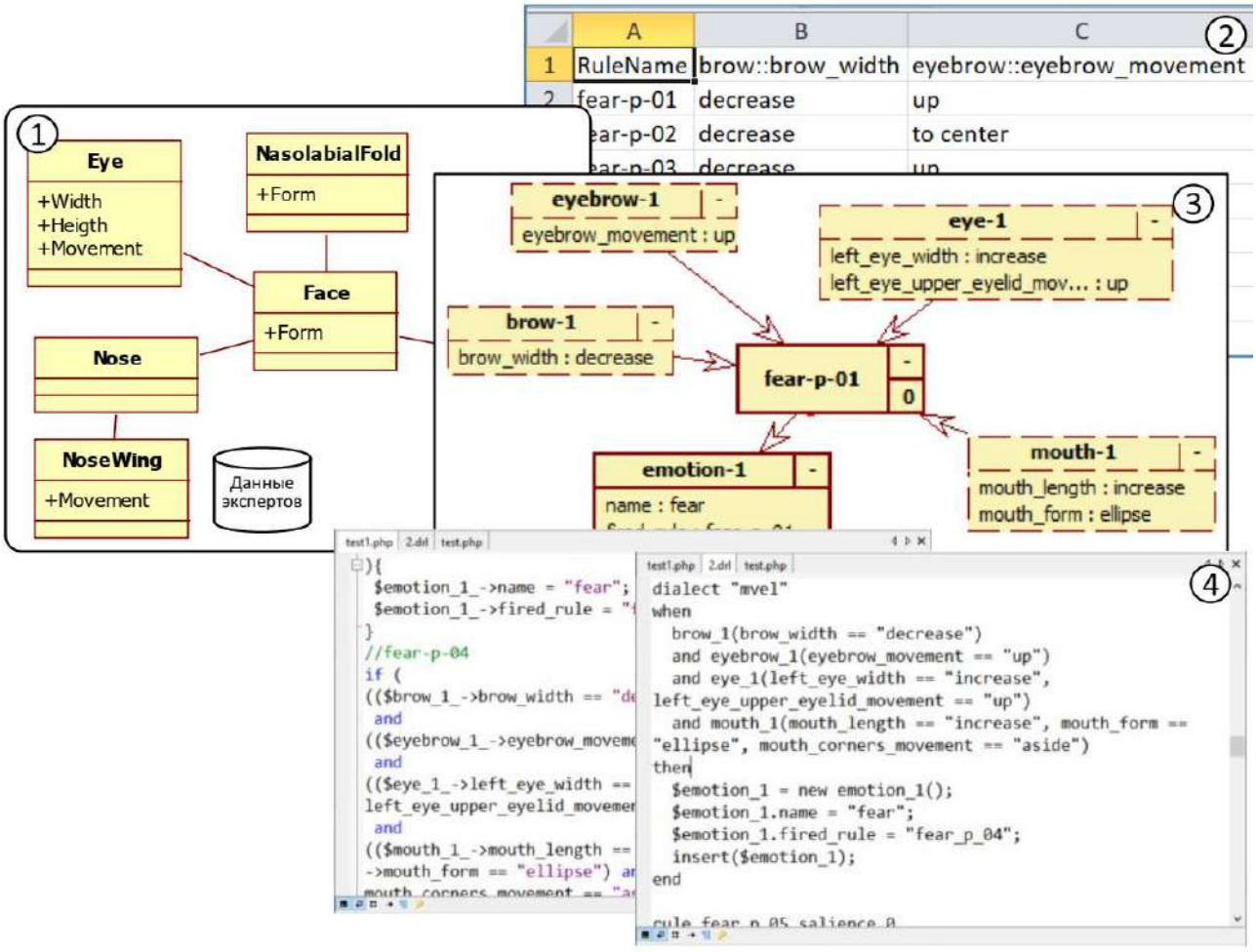


Рис. Д.5. Этапы процесса создания прототипа базы знаний модуля интерпретации признаков эмоций: 1) фрагмент исходной концептуальная модель и информация о методиках анализа эмоций; 2) таблица решений, описывающая эмоцию «страх»; 3) RVML диаграмма, соответствующая таблице решений; 4) сгенерированный программный код для PHP и DROOLS

	1	2	3	4	5	6	7	8	9
1	RuleName	Risk::grade	Risk::kind	...	Flood-haz	...	#Conclusion::grade	#Conclusion::text	#Conclusion::cf
2	Risk+Flood hazard->	low	natural		low		low	safe situation	0.9

Рис.Д.6. Пример фрагмента таблицы решений

В рамках решаемой задачи таблицы решений описывали структурный аспект предметной области (ее модели), а также знания психолога-эксперта. Они

содержали информацию о комбинациях признаков, описывающих эмоции, например, «страх» (Рис. Д.5, блок 2). При этом каждая строка таблицы представляет собой логическое правило.

Далее, с использованием модуля расширения PKBD.DnTable было произведено преобразование таблиц решений в логические правила с возможностью их отображения в виде RVML диаграмм для дальнейшего уточнения (Рис. Д.5, блок 3). На основе уточненных RVML диаграмм генерировался программный код (Рис. Д.5, блок 4).

Оценка результатов

В рамках проекта было создано несколько вариантов баз знаний для идентификации определенных эмоций, максимальный объем которых достигал 225 правил с учетом Action Units и интенсивности их проявления, листинг программного кода на CLIPS составил 1327 строк. Листинг одного из прототипов БЗ приведен ниже.

Оценка эффективности решения задачи осуществлялась с двух точек зрения: 1) эффективность создания базы знаний; 2) эффективность решения задачи распознавания эмоций.

Эффективность создания баз знаний оценивалась на примере сравнения временных затрат при использовании предлагаемого подхода с ручным кодированием и визуальным программированием при помощи PKBD. Более подробно эксперимент с тестовыми задачами рассмотрен в [201]. Его результаты представлены на Рис.Д.7, согласно которым использование таблиц решений может уменьшить временные затраты по сравнению с ручным кодированием (в среднем до 52%), при этом, они менее эффективны в среднем на 38,8% чем использование специализированных мастеров PKBD или манипулирования графическими примитивами (например, RVML диаграммами). Однако, как позже было определено, при размере баз знаний более 150 правил и автоматической генерации таблиц решений, например, на основе баз данных, этот подход оказался более предпочтителен.

Оценка эффективности распознавания эмоций с использованием разработанных баз знаний осуществилась на двух наборах данных: тестовом видео (студийные записи с актерами, явно выражающими эмоции) и интервью реальных респондентов (съемка с помощью камеры ноутбука или смартфона реального человека). В первом случае оценка составила 65%, во втором – 20%. Основные причины достаточно низких показателей: необходимость калибровки системы, т.е.

настройки на определенного респондента в части индивидуальной интенсивности проявления эмоций; необходимость повышения качества видео за счет минимизации теней, бликов, дрожания камеры при съемке.

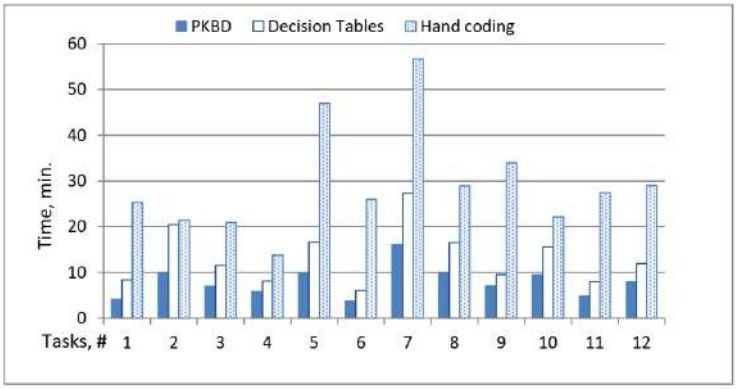


Рис.Д.7. Результаты сравнения PKBD, таблицы решений и ручное кодирование по временному критерию

По результатам применения технологии при решении данной задачи получена справка об использовании результатов интеллектуальной деятельности (см. Приложение А).

Листинг прототипа базы знаний модуля интепретации признаков эмоций

```
;*************************************
; Наименование базы знаний: Определение
эмоций
;*************************************
; Описание:
;*************************************
;********************* Templates
(deftemplate Dvigatelnaya-edinitsa
(slot Kod)
(slot Myshechnaya-osnova)
)
(deftemplate Emotsiya
(slot Naimenovanie)
(slot Intensivnost-emotsii)
(slot Dvigatelnaya-edinitsa)
)
(deftemplate Dvizhenie
(slot Naimenovanie)
(slot Chast-litsa)
(slot Dvigatelnaya-edinitsa)
(slot Intensivnost)
(slot Simmetrichnost)
)
(deftemplate Opredelenie-emotsiy
(slot Emotsiya)
(slot Obekt)
(slot Subekt)
)
;************************ Facts
(deffacts initial-settings
(Dvizhenie ;Движение "Подъем щеки"
(Naimenovanie "ПОДЪЕМ ЩЕКИ")
(Chast-litsa "ЩЁКИ")
(Dvigatelnaya-edinitsa "AU6")
(Intensivnost "C")
(Simmetrichnost "B")
)
(Dvizhenie ;Движение "Подъем уголка
губы"
(Naimenovanie "ПОДЪЕМ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU12")
(Intensivnost "D")
(Simmetrichnost "B")
)
)
;************************ Rules
(defrule Otvrashchenie-2 "Отвращение"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "СМОРЩИВАНИЕ НОСА")
(Chast-litsa "НОС")
(Dvigatelnaya-edinitsa "AU9")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ НИЖНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU16")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU15")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ОТВРАЩЕНИЕ")
))
)
```

```
(defrule Prezrenie-1 "Презрение"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОЯВЛЕНИЕ ЯМОЧКИ")
(Chast-litsa "ЩЁКИ")
(Dvigatelnaya-edinitsa "AU14")
(Simmetrichnost "L")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU12")
(Simmetrichnost "L")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПРЕЗРЕНИЕ")
))
)
(defrule Radost-1 "Радость"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ЩЕКИ")
(Chast-litsa "ЩЁКИ")
(Dvigatelnaya-edinitsa "AU6")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU12")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "РАДОСТЬ")
))
)
(defrule Udivlenie-1 "Удивление"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
(Simmetrichnost "B")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "УДИВЛЕНИЕ")
))
)
(defrule Udivlenie-2 "Удивление"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ ЧЕЛЮСТИ")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU26")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "УДИВЛЕНИЕ")
))
)
(defrule Udivlenie-3 "Удивление"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "РОТ ШИРОКО ОТКРЫТ")
(Chast-litsa "РОТ")
(Dvigatelnaya-edinitsa "AU27")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "УДИВЛЕНИЕ")
))
)
(defrule Udivlenie-4 "Удивление"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
(Simmetrichnost "B")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ ЧЕЛЮСТИ")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU26")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "УДИВЛЕНИЕ")
))
)
(defrule Udivlenie-5 "Удивление"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
(Simmetrichnost "B")
```

```
)
(Dvizhenie ;Движение
(Naimenovanie "РОТ ШИРОКО ОТКРЫТ")
(Chast-litsa "РОТ")
(Dvigatelnaya-edinitsa "AU27")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "УДИВЛЕНИЕ")
))
)
(defrule Strakh-1 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "РАСТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU20")
)
(Dvizhenie ;Движение
(Naimenovanie "ГУБЫ РАЗВЕДЕНЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU25")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-2 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "РАСТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU20")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ ЧЕЛЮСТИ")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU26")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-3 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "РАСТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU20")
)
(Dvizhenie ;Движение
(Naimenovanie "РОТ ШИРОКО ОТКРЫТ")
(Chast-litsa "РОТ")
(Dvigatelnaya-edinitsa "AU27")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-4 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
```

```
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-5 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "ГУБЫ РАЗВЕДЕНЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU25")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-6 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ ЧЕЛЮСТИ")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU26")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-7 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "РОТ ШИРОКО ОТКРЫТ")
(Chast-litsa "РОТ")
(Dvigatelnaya-edinitsa "AU27")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-8 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНЕШНЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU2")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-9 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
```

```
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "РАСТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU20")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-10 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "РАСТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU20")
)
(Dvizhenie ;Движение
(Naimenovanie "ГУБЫ РАЗВЕДЕНЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU25")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-11 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "РАСТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU20")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ ЧЕЛЮСТИ")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU26")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Strakh-12 "Страх"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "РАСТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU20")
)
(Dvizhenie ;Движение
(Naimenovanie "РОТ ШИРОКО ОТКРЫТ")
(Chast-litsa "РОТ")
(Dvigatelnaya-edinitsa "AU27")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "СТРАХ")
))
)
(defrule Radost-2 "Радость"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU12")
(Intensivnost "C")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "РАДОСТЬ")
))
)
(defrule Radost-3 "Радость"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU12")
(Intensivnost "D")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "РАДОСТЬ")
))
)
(defrule Pechal-1 "Печаль"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "РАСШИРЕНИЕ НОСОГУБНОЙ
СКЛАДКИ")
(Chast-litsa "НОС")
(Dvigatelnaya-edinitsa "AU11")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПЕЧАЛЬ")
))
)
(defrule Pechal-2 "Печаль"
(declare (salience 1))
(Dvizhenie ;Движение
```

```
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "РАСШИРЕНИЕ НОСОГУБНОЙ
СКЛАДКИ")
(Chast-litsa "НОС")
(Dvigatelnaya-edinitsa "AU11")
)
(Dvizhenie ;Движение
(Naimenovanie "ГОЛОВА ВНИЗ")
(Chast-litsa "ГОЛОВА")
(Dvigatelnaya-edinitsa "AU54")
)
(Dvizhenie ;Движение
(Naimenovanie "ГЛАЗА ВНИЗ")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU64")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПЕЧАЛЬ")
))
)
(defrule Pechal-3 "Печаль"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU15")
(Simmetrichnost "B")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПЕЧАЛЬ")
))
)
(defrule Pechal-4 "Печаль"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU15")
(Simmetrichnost "B")
)
(Dvizhenie ;Движение
(Naimenovanie "ГОЛОВА ВНИЗ")
(Chast-litsa "ГОЛОВА")
(Dvigatelnaya-edinitsa "AU54")
)
(Dvizhenie ;Движение
(Naimenovanie "ГЛАЗА ВНИЗ")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU64")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПЕЧАЛЬ")
))
)
(defrule Pechal-5 "Печаль"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU15")
(Simmetrichnost "B")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ПОДБОРОДКА")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU17")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПЕЧАЛЬ")
))
)
(defrule Pechal-6 "Печаль"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВНУТРЕННЕЙ ЧАСТИ
БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU1")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU15")
(Simmetrichnost "B")
```

```
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ПОДБОРОДКА")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU17")
)
(Dvizhenie ;Движение
(Naimenovanie "ГОЛОВА ВНИЗ")
(Chast-litsa "ГОЛОВА")
(Dvigatelnaya-edinitsa "AU54")
)
(Dvizhenie ;Движение
(Naimenovanie "ГЛАЗА ВНИЗ")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU64")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПЕЧАЛЬ")
))
)
(defrule Pechal-7 "Печаль"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "РАСШИРЕНИЕ НОСОГУБНОЙ
СКЛАДКИ")
(Chast-litsa "НОС")
(Dvigatelnaya-edinitsa "AU11")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU15")
(Simmetrichnost "B")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПЕЧАЛЬ")
))
)
(defrule Pechal-8 "Печаль"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "РАСШИРЕНИЕ НОСОГУБНОЙ
СКЛАДКИ")
(Chast-litsa "НОС")
(Dvigatelnaya-edinitsa "AU11")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU15")
(Simmetrichnost "B")
)
(Dvizhenie ;Движение
(Naimenovanie "ГОЛОВА ВНИЗ")
(Chast-litsa "ГОЛОВА")
(Dvigatelnaya-edinitsa "AU54")
)
(Dvizhenie ;Движение
(Naimenovanie "ГЛАЗА ВНИЗ")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU64")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПЕЧАЛЬ")
))
)
(defrule Pechal-9 "Печаль"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "РАСШИРЕНИЕ НОСОГУБНОЙ
СКЛАДКИ")
(Chast-litsa "НОС")
(Dvigatelnaya-edinitsa "AU11")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ПОДБОРОДКА")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU17")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПЕЧАЛЬ")
))
)
(defrule Otvrashchenie-1 "Отвращение"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "СМОРЩИВАНИЕ НОСА")
(Chast-litsa "НОС")
(Dvigatelnaya-edinitsa "AU9")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ОТВРАЩЕНИЕ")
))
)
(defrule Otvrashchenie-3 "Отвращение"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "СМОРЩИВАНИЕ НОСА")
(Chast-litsa "НОС")
(Dvigatelnaya-edinitsa "AU9")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ НИЖНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU16")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ ЧЕЛЮСТИ")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU26")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ОТВРАЩЕНИЕ")
))
)
(defrule Otvrashchenie-4 "Отвращение"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "СМОРЩИВАНИЕ НОСА")
(Chast-litsa "НОС")
(Dvigatelnaya-edinitsa "AU9")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ПОДБОРОДКА")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU17")
)
=>
```

```
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ОТВРАЩЕНИЕ")
))
)
(defrule Otvrashchenie-5 "Отвращение"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU10")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ОТВРАЩЕНИЕ")
))
)
(defrule Otvrashchenie-6 "Отвращение"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU10")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ НИЖНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU16")
)
(Dvizhenie ;Движение
(Naimenovanie "ГУБЫ РАЗВЕДЕНЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU25")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ОТВРАЩЕНИЕ")
))
)
(defrule Otvrashchenie-7 "Отвращение"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU10")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ НИЖНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU16")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ ЧЕЛЮСТИ")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU26")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ОТВРАЩЕНИЕ")
))
)
(defrule Otvrashchenie-8 "Отвращение"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU10")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ПОДБОРОДКА")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU17")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ОТВРАЩЕНИЕ")
))
)
(defrule Prezrenie-2 "Презрение"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ УГОЛКА ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU12")
(Simmetrichnost "R")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОЯВЛЕНИЕ ЯМОЧКИ")
(Chast-litsa "ЩЁКИ")
(Dvigatelnaya-edinitsa "AU14")
(Simmetrichnost "R")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ПРЕЗРЕНИЕ")
))
)
(defrule Gnev-1 "Гнев"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU7")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU10")
)
(Dvizhenie ;Движение
(Naimenovanie "ГУБЫ ВОРОНКОЙ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU22")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU23")
)
(Dvizhenie ;Движение
(Naimenovanie "ГУБЫ РАЗВЕДЕНЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU25")
)
=>
```

```
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ГНЕВ")
))
)
(defrule Gnev-2 "Гнев"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU7")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU10")
)
(Dvizhenie ;Движение
(Naimenovanie "ГУБЫ ВОРОНКОЙ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU22")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU23")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ ЧЕЛЮСТИ")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU26")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ГНЕВ")
))
)
(defrule Gnev-3 "Гнев"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU7")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU10")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU23")
)
(Dvizhenie ;Движение
(Naimenovanie "ГУБЫ РАЗВЕДЕНЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU25")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ГНЕВ")
))
)
(defrule Gnev-4 "Гнев"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU7")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕЙ ГУБЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU10")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU23")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ ЧЕЛЮСТИ")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU26")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ГНЕВ")
))
)
(defrule Gnev-5 "Гнев"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU7")
```

```
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU23")
)
(Dvizhenie ;Движение
(Naimenovanie "ГУБЫ РАЗВЕДЕНЫ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU25")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ГНЕВ")
))
)
(defrule Gnev-6 "Гнев"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU7")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU23")
)
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ ЧЕЛЮСТИ")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU26")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ГНЕВ")
))
)
(defrule Gnev-7 "Гнев"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU7")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ПОДБОРОДКА")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU17")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU23")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ГНЕВ")
))
)
(defrule Gnev-8 "Гнев"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU7")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ПОДБОРОДКА")
(Chast-litsa "ЧЕЛЮСТЬ")
(Dvigatelnaya-edinitsa "AU17")
)
(Dvizhenie ;Движение
(Naimenovanie "СЖИМАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU24")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ГНЕВ")
))
)
(defrule Gnev-9 "Гнев"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU7")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU23")
)
=>
(assert
```

```
(Emotsiya ;Эмоция
(Naimenovanie "ГНЕВ")
))
)
(defrule Gnev-10 "Гнев"
(declare (salience 1))
(Dvizhenie ;Движение
(Naimenovanie "ОПУСКАНИЕ БРОВИ")
(Chast-litsa "БРОВИ")
(Dvigatelnaya-edinitsa "AU4")
)
(Dvizhenie ;Движение
(Naimenovanie "ПОДЪЕМ ВЕРХНЕГО ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU5")
)
(Dvizhenie ;Движение
(Naimenovanie "НАТЯГИВАНИЕ ВЕКА")
(Chast-litsa "ГЛАЗА")
(Dvigatelnaya-edinitsa "AU7")
)
(Dvizhenie ;Движение
(Naimenovanie "СЖИМАНИЕ ГУБ")
(Chast-litsa "ГУБЫ")
(Dvigatelnaya-edinitsa "AU24")
)
=>
(assert
(Emotsiya ;Эмоция
(Naimenovanie "ГНЕВ")
))
)
```

Приложение Е
Детектор: Модуль обнаружения нежелательных сообщений

Разработанные модели, методы и средства использовались при разработке веб-модуля принятия решений «Детектор» (рег.№ 2020614257) [193, 201, 371, 373] для платформы «СМС-Органайзер» [345, 374] (рег.№ 2022613512) (ООО «ЦентраСиб»).

В ходе решения задачи была расширена функциональность PKBD в части поддержки импорта (трансформации) таблиц решений (разработан модуль расширения PKBD.DnTable), а также генерации программных кодов на PHP. Основной результат – разработана база знаний для модуля обнаружения сообщений, нарушающих положения 38-ФЗ «О рекламе» [352], а также отправляющих их абонентов. В качестве источника информации использовались данные об отправленных сообщений платформы «СМС-Органайзер».

Процесс разработки прототипа баз знаний для модуля соответствует методу и методике (см. 4.1) и может быть представлен в виде следующей схемы (Рис.Е.1).

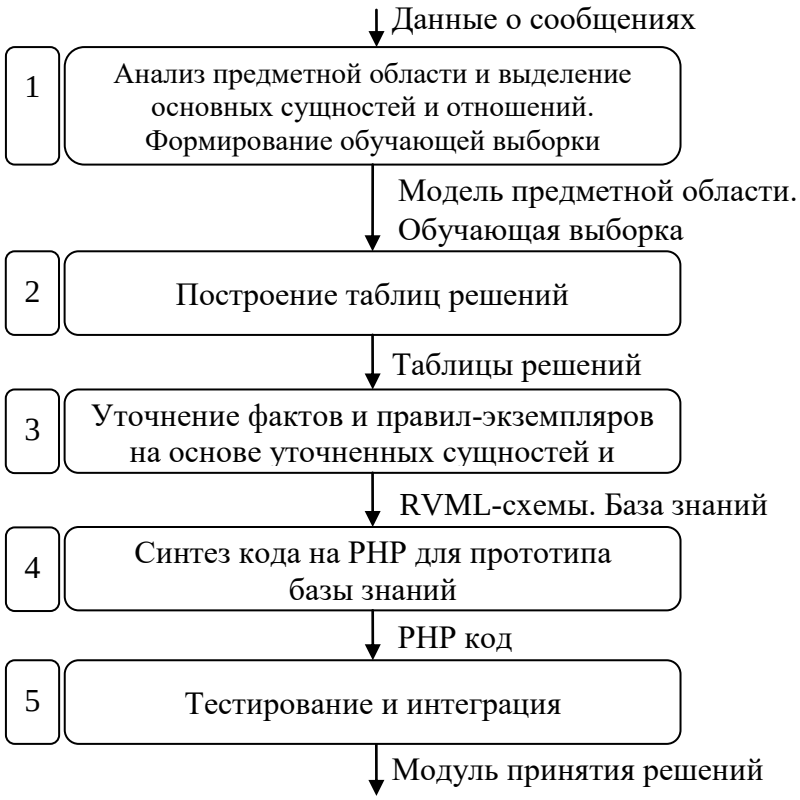


Рис.Е.1. Схема разработки модуля обнаружений нежелательных сообщений для PHP с использованием PKBD

Далее рассмотрим этапы подробнее. На этапе построения модели предметной области был выполнен ее анализ и выделены ключевые абстракции

(Рис.Е.2). На концептуальном уровне основными элементами в решаемой задаче является «СМС-сообщение» (Message), содержащее «ключевые слова» (Keyword), так называемые «маркеры» СПАМ-сообщений, и отправитель сообщений (Sender). В дальнейшем при формализации модель была упрощена с целью более полного соответствия структуре логических правил.

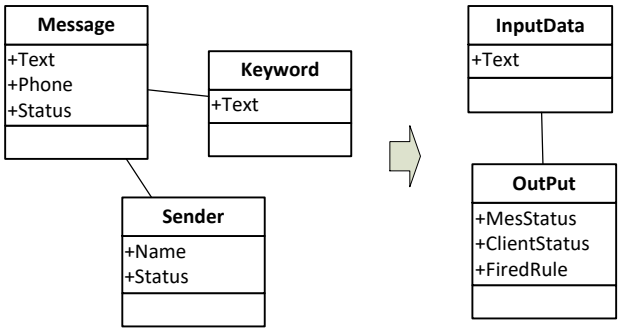


Рис.Е.2. Концептуальная модель решаемой задачи

Для выделения комбинаций маркеров (ключевых слов) был произведен анализ базы данных из 1366490 сообщений и отобраны 829 сообщений клиентов, уличенных ранее в рассылке СПАМ-сообщений.

Выделенные сообщения сформировали обучающую выборку (тестовый набор), в которой были выделены пять основных групп (Таблица Е.1): пропаганда запрещенных веществ; недобросовестная реклама; мошенничество; угрозы и оскорбления; не несущие признаков нежелательных сообщений. По каждой группе определена возможная рекомендация модуля поддержки принятия решения, в частности, блокировка или нет отправки сообщения (изменения его статуса) и блокировка или нет отправителя.

Таблица Е.1. Основные группы нежелательных сообщений

№ гр.	Наименование группы	Всего	Отобрано	Реакция системы	
				Новый статус сообщения	Новый статус пользователя (индекс)
1	Пропаганда запрещенных веществ	346	253	Ошибка	Заблокирован (1)
2	Недобросовестная реклама	12	8	На модерации	Активен (0)
3	Мошенничество	236	80	Ошибка	Заблокирован (1)
4	Угрозы и оскорбления	185	146	На модерации	Заблокирован (1)
5	Не несущие признаков нежелательных сообщений	50	0	-	-
Итого:		829	487		

Выделенные концепты были использованы при построении платформо-независимых моделей, в том числе в форме таблицы решений, содержащей

информацию о 487 уникальных наборах (Рис. Е.3). Таблица была разработана путем автоматизированного анализа базы данных сообщений, структура таблицы формируется путем перечисления свойств всех концептов, где каждое свойство - это столбец таблицы.

	B	C	D	E	F	G	H	I	J
1	InputData::Keyword	InputData::Keyword	InputData::Keyword	InputData::Keyword	InputData::	InputData::	#OutPut::MesStatus	#Out	#OutPut::FiredRule
2	pharmbar.com	pharmbar.net					ошибка	1	rule 1-10
3	левые	железного	зеленого	крыльца	ступеньки	уголка	ошибка	1	rule 1-100
4	видно	внизу	даже	салфетке	стороны		ошибка	1	rule 1-101
5	киви	лежит	мегафон	номер			ошибка	1	rule 1-102
6	валлет	сумма	виза	оплат	услуг	киви	ошибка	1	rule 1-103
7	давай	киви	мегафоновский	номер			ошибка	1	rule 1-104
8	готовый	киви	клад	конечно	мегафон		ошибка	1	rule 1-105
9	изза	меня	проблеммы	седня	тебя	филя	ошибка	1	rule 1-106
10	дублируются	сверить	клады	сохраняй	стопроцентные		ошибка	1	rule 1-107
11	городом	клады	остались				ошибка	1	rule 1-108
12	березой	кичигино	остановка	руль			ошибка	1	rule 1-109
13	клады	курительные	легал	пробы	смеси-1п-500		ошибка	1	rule 1-11
14	легал	наличии					ошибка	1	rule 1-110
15	меня	пока	тихо	южнике			ошибка	1	rule 1-111
16	готовые	докидывай	остались	токо	южнике		ошибка	1	rule 1-112
17	качеству	надо	привыкли	чему			ошибка	1	rule 1-113

Рис. Е.3. Фрагмент таблицы решений с комбинациями маркеров (ключевых слов)

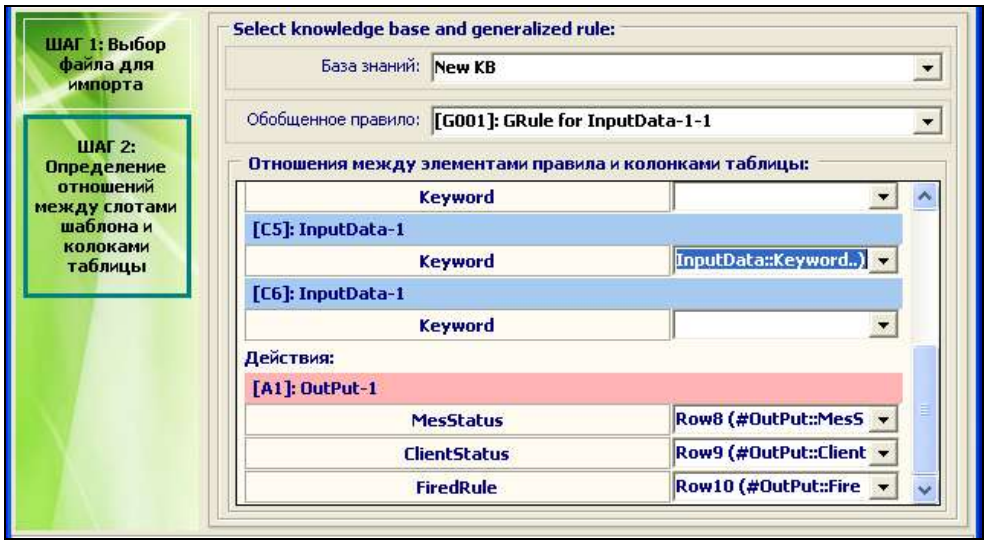


Рис. Е.4. Фрагмент экранной формы PKBD: импорт таблицы решений

Построение платформо-зависимых моделей осуществлялось с использованием PKBD путем импорта разработанной таблицы решений (Рис.Е.4) с последующим их уточнением в виде RVML (Рис.Е.5), включая приоритет правил и значения «по умолчанию».

Посредством генератора PKBD было синтезировано 6871 строк PHP кода, описывающих 487 правило. При этом каждое логическое правило было представлено в форме условного оператора, в условии которого анализируется вхождение определенных ключевых слов в отправляемое сообщение. Фрагмент полученного кода приведен далее.

Приоритеты правил были учтены путем сортировки атомарных условных операторов в рамках вычислительного блока (функции), таким образом, в случае

активации правила происходила остановка вывода и выход из блока, при этом возвращался результат последнего активированного правила.

В дальнейшем сгенерированный код был синтерирован в платформу «СМС-Органайзер» (Рис.Д.6).

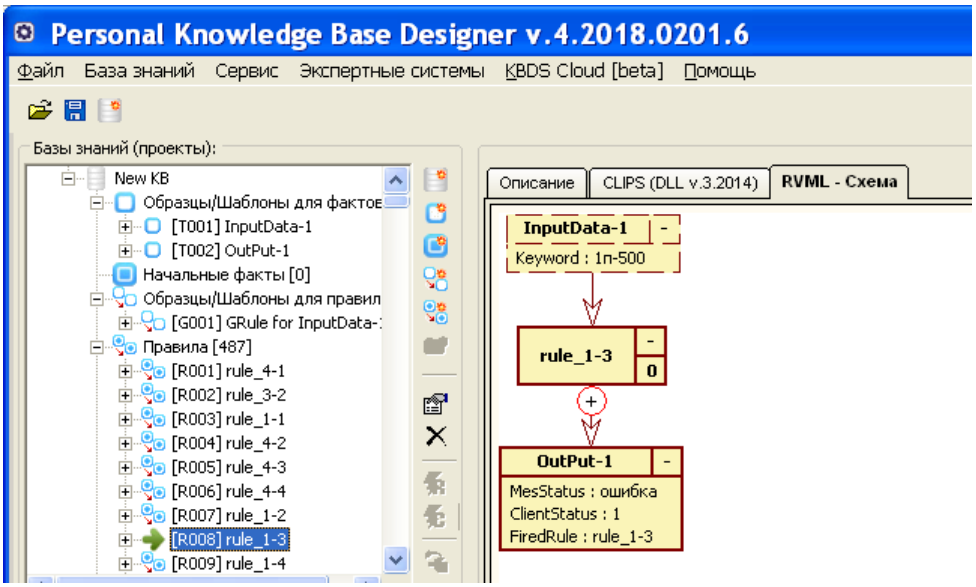


Рис. Е.5. Фрагмент экранной формы PKBD: представление правила в виде диаграмм RVML

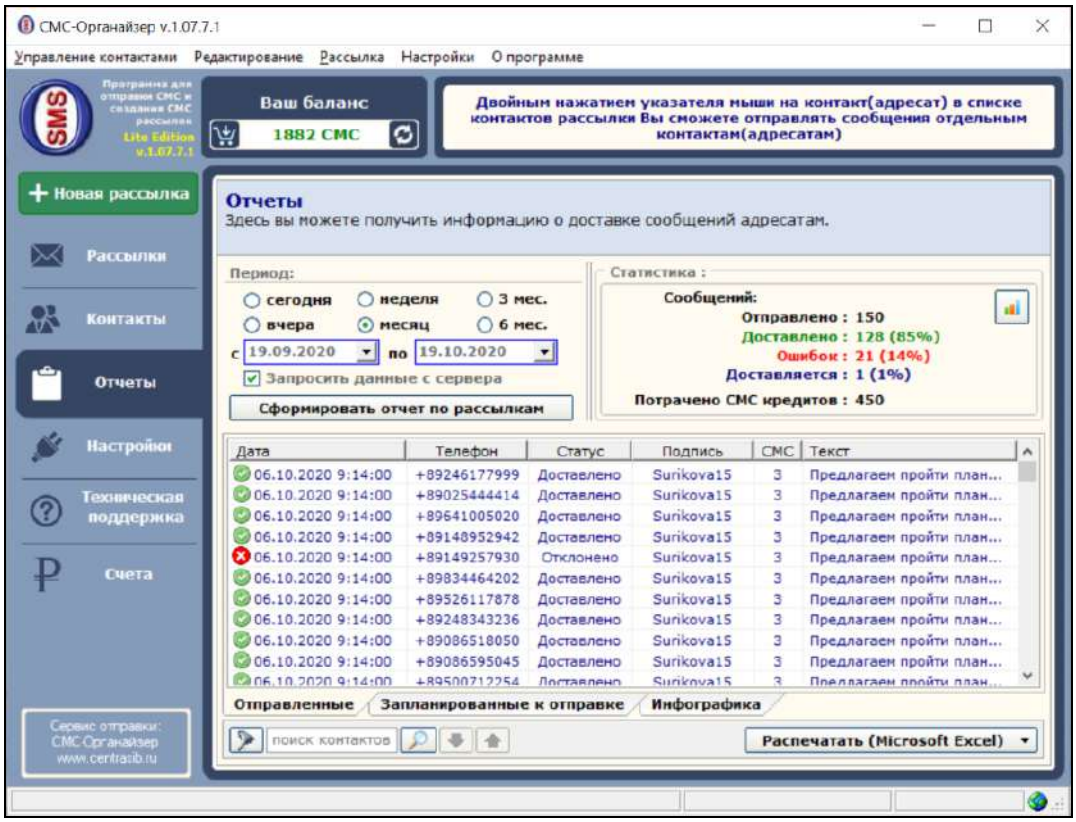


Рис. Д.6. Фрагмент экранной формы клиентской части приложения «СМС-Органайзер»

Тестирование разработанного модуля «Детектор» осуществлялось с использованием выделенной ранее базы сообщений. В частности, после модификации база знаний модуля составила 498 правил, которые обеспечили обнаружение 653 нежелательных сообщений из тестового набора, таким образом, точность (accuracy) составила 0,83. Листинг программного кода модуля составил 7030 строк. Среднее время выполнения (быстродействие) составило 0,00026 с. В результате проверки подсистемой «Детектор» набора из 1366490 сообщений было выявлено 1145 СПАМ-сообщений и 25 клиентов (отправителей), нарушивших положения 38-ФЗ «О рекламе», но которые ранее не были обнаружены.

По результатам применения технологии при решении данной задачи получен акт внедрения программы для ЭВМ (см. Приложение А).

Фрагмент листинга модуля обнаружения нежелательных сообщений:

```
<?php
//*************** exported from PKBD
// version: 4.2018.0201.6
// knowledge base:
// info:

//****************** classes
class InputData_1{
 var $Keyword;
 function Init(){
  $this->Keyword = "";
 }
}
class OutPut_1{
 var $MesStatus;
 var $ClientStatus;
 var $FiredRule;
 function Init(){
  $this->MesStatus = "";
  $this->ClientStatus = "";
  $this->FiredRule = "";
 }
}
//******** Initialization (facts)
$InputData_1_ = new InputData_1;
$InputData_1_->Init();
$OutPut_1_ = new OutPut_1;
$OutPut_1_->Init();

//**************** rules
function RunKB($InputData_1_){
//rule_4-1
if (
((strpos($InputData_1_->Keyword,
"гулящая") !== false))
 and
((strpos($InputData_1_->Keyword,
"упала") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-1";
}
//rule_3-2
if (
((strpos($InputData_1_->Keyword, "8800-
500-01-01") !== false))
 and
((strpos($InputData_1_->Keyword,
"справка") !== false))
){
 $OutPut_1_->MesStatus = "Ошибка";
 $OutPut_1_->ClientStatus = "1";
 $OutPut_1_->FiredRule = "rule_3-2";
}
//rule_1-1
/*if (
((strpos($InputData_1_->Keyword, "токо")
!== false))
){
 $OutPut_1_->MesStatus = "Ошибка";
 $OutPut_1_->ClientStatus = "1";
 $OutPut_1_->FiredRule = "rule_1-1";
}*/
//rule_4-2
if (
((strpos($InputData_1_->Keyword,
"маменькин") !== false))
 and
((strpos($InputData_1_->Keyword,
"молодой") !== false))
 and
((strpos($InputData_1_->Keyword,
"ноющий") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-2";
}
//rule_4-3
if (
((strpos($InputData_1_->Keyword,
"блудная") !== false))
 and
((strpos($InputData_1_->Keyword,
"нагулялась") !== false))
 and
```

```
((strpos($InputData_1_->Keyword, "овца")
!== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-3";
}
//rule_4-4
if (
((strpos($InputData_1_->Keyword, "баба-
мужик") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-4";
}
//rule_1-2
if (
((strpos($InputData_1_->Keyword,
"1п+1п=2п") !== false))
){
 $OutPut_1_->MesStatus = "Ошибка";
 $OutPut_1_->ClientStatus = "1";
 $OutPut_1_->FiredRule = "rule_1-2";
}
//rule_1-3
if (
((strpos($InputData_1_->Keyword, "1п-
500") !== false))
){
 $OutPut_1_->MesStatus = "Ошибка";
 $OutPut_1_->ClientStatus = "1";
 $OutPut_1_->FiredRule = "rule_1-3";
}
//rule_1-4
if (
((strpos($InputData_1_->Keyword,
"89320112866") !== false))
){
 $OutPut_1_->MesStatus = "Ошибка";
 $OutPut_1_->ClientStatus = "1";
 $OutPut_1_->FiredRule = "rule_1-4";
}
//rule_1-5
if (
((strpos($InputData_1_->Keyword, "ложи")
!== false))
 and
((strpos($InputData_1_->Keyword, "сюда")
!== false))
){
 $OutPut_1_->MesStatus = "Ошибка";
 $OutPut_1_->ClientStatus = "1";
 $OutPut_1_->FiredRule = "rule_1-5";
}
//rule_1-6
if (
((strpos($InputData_1_->Keyword,
"9320112866") !== false))
){
 $OutPut_1_->MesStatus = "Ошибка";
 $OutPut_1_->ClientStatus = "1";
 $OutPut_1_->FiredRule = "rule_1-6";
}
//rule_4-5
if (
((strpos($InputData_1_->Keyword,
"стремотой") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-5";
}
//rule_4-6
/*if (
((strpos($InputData_1_->Keyword,
"развод") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-6";
}*/
//rule_4-7
if (
((strpos($InputData_1_->Keyword,
"сменить") !== false))
 and
((strpos($InputData_1_->Keyword,
"теперь") !== false))
 and
((strpos($InputData_1_->Keyword,
"фамилию") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-7";
}
//rule_4-8
if (
((strpos($InputData_1_->Keyword, "трэш")
!== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-8";
}
//rule_4-9
if (
((strpos($InputData_1_->Keyword,
"отстойный") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-9";
}
//rule_4-10
if (
((strpos($InputData_1_->Keyword,
"скинется") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-10";
}
//rule_4-11
if (
((strpos($InputData_1_->Keyword,
"мозгов") !== false))
 and
((strpos($InputData_1_->Keyword,
"навозных") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-11";
}
//rule_4-12
if (
((strpos($InputData_1_->Keyword,
"актёришка") !== false))
```

```
 and
((strpos($InputData_1_->Keyword,
"аплодисментами") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-12";
}
//rule_4-13
if (
((strpos($InputData_1_->Keyword,
"злючая") !== false))
 and
((strpos($InputData_1_->Keyword,
"курить") !== false))
 and
((strpos($InputData_1_->Keyword,
"мамашка") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-13";
}
//rule_4-14
if (
((strpos($InputData_1_->Keyword,
"висят") !== false))
 and
((strpos($InputData_1_->Keyword,
"когда") !== false))
 and
((strpos($InputData_1_->Keyword,
"нравится") !== false))
 and
((strpos($InputData_1_->Keyword,
"спаниеля") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-14";
}
//rule_4-16
if (
((strpos($InputData_1_->Keyword,
"агрессией") !== false))
and
((strpos($InputData_1_->Keyword,
"позорной") !== false))
 and
((strpos($InputData_1_->Keyword,
"шизофренической") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-16";
}
//rule_4-17
if (
((strpos($InputData_1_->Keyword,
"душат") !== false))
 and
((strpos($InputData_1_->Keyword,
"послал") !== false))
 and
((strpos($InputData_1_->Keyword,
"посылают") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-17";
}
//rule_4-18
if (
((strpos($InputData_1_->Keyword,
"брошенка") !== false))
 and
((strpos($InputData_1_->Keyword,
"разведёнка") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-18";
}
//rule_4-19
if (
((strpos($InputData_1_->Keyword,
"жесть") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-19";
}
//rule_4-23
if (
((strpos($InputData_1_->Keyword,
"мамочка-ш") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-23";
}
//rule_4-25
if (
((strpos($InputData_1_->Keyword,
"бессовестная") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-25";
}
//rule_4-28
if (
((strpos($InputData_1_->Keyword,
"блочишь") !== false))
 and
((strpos($InputData_1_->Keyword,
"женихов") !== false))
 and
((strpos($InputData_1_->Keyword,
"подыскиваем") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-28";
}
//rule_4-29
if (
((strpos($InputData_1_->Keyword, "дать")
!== false))
 and
((strpos($InputData_1_->Keyword,
"женщине") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-29";
}
//rule_4-30
if (
```

```
((strpos($InputData_1_->Keyword,
"гонется") !== false))
 and
((strpos($InputData_1_->Keyword,
"деньгами") !== false))
 and
((strpos($InputData_1_->Keyword,
"старикашке") !== false))
 and
((strpos($InputData_1_->Keyword,
"чужими") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-30";
}
//rule_4-32
if (
((strpos($InputData_1_->Keyword,
"мужик") !== false))
 and
((strpos($InputData_1_->Keyword,
"непорядочной") !== false))
 and
((strpos($InputData_1_->Keyword,
"развестись") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-32";
}
//rule_4-33
if (
((strpos($InputData_1_->Keyword,
"обострение") !== false))
 and
((strpos($InputData_1_->Keyword,
"признаки") !== false))
 and
((strpos($InputData_1_->Keyword,
"ревности") !== false))
 and
((strpos($InputData_1_->Keyword,
"шизофрении") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-33";
}
//rule_4-34
if (
((strpos($InputData_1_->Keyword,
"камера") !== false))
 and
((strpos($InputData_1_->Keyword,
"лысая") !== false))
 and
((strpos($InputData_1_->Keyword,
"скрытая") !== false))
 and
((strpos($InputData_1_->Keyword,
"снимала") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-34";
}
//rule_4-36
if (
((strpos($InputData_1_->Keyword,
"брошенка") !== false))
 and
((strpos($InputData_1_->Keyword,
"разведёнкой") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-36";
}
//rule_4-37
if (
((strpos($InputData_1_->Keyword,
"брошенки") !== false))
 and
((strpos($InputData_1_->Keyword,
"любовница") !== false))
 and
((strpos($InputData_1_->Keyword,
"позорище") !== false))
 and
((strpos($InputData_1_->Keyword,
"тёлка") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-37";
}
//rule_4-38
if (
((strpos($InputData_1_->Keyword,
"бывший") !== false))
 and
((strpos($InputData_1_->Keyword,
"забудь") !== false))
 and
((strpos($InputData_1_->Keyword,
"опрокинул") !== false))
 and
((strpos($InputData_1_->Keyword,
"разводе") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-38";
}
//rule_4-39
if (
((strpos($InputData_1_->Keyword,
"брошенка") !== false))
 and
((strpos($InputData_1_->Keyword,
"никчёмности") !== false))
 and
((strpos($InputData_1_->Keyword,
"отвращения") !== false))
 and
((strpos($InputData_1_->Keyword,
"разведёнка") !== false))
){
 $OutPut_1_->MesStatus = "На модерации";
 $OutPut_1_->ClientStatus = "0";
 $OutPut_1_->FiredRule = "rule_4-39";
}
//…
?>
```

Приложение Ж
Прототип базы знаний ИС «АвиаТехПом»

Предлагаемые методы и средства применялись при прототипировании базы знаний интеллектуальной системы поддержки принятия решений технического персонала при поиске и устранении неисправностей воздушного судна (ИС «АвиаТехПом») [53, 95, 199, 308] на примере системы электроснабжения Сухой Суперджет. Задача решалась во взаимодействии с сотрудниками Иркутского филиала МГТУ ГА.

Диагностика неисправностей воздушных судов продолжает оставаться актуальной проблемой, требующей для своего оперативного решения применения технологий семантического веба (в частности, онтологий [79, 308]), а также методов искусственного интеллекта в форме продукционных ИС [171, 199]. Данные технологии обеспечивают не только согласованное на качественном уровне и формализованное представление предметных знаний экспертов, но и поиск решений благодаря применению дедуктивного и индуктивного вывода.

Основой для формирования баз знаний в рамках решаемой задачи являются сценарии работ по локализации и ликвидации неисправностей. Подобные сценарии представляют собой последовательности работ, формирующих технологическую карту. Существует ряд способов формирования подобных сценариев для дальнейшей их реализации в экспертных системах:

- описание последовательности работ в онтологии в рамках каждой неисправности путем ввода отдельных типов отношений, используя средства онтологического моделирования (например, Protégé), с последующим ручным или полуавтоматическим кодированием;

- формирование матрицы смежности работ внешними средствами с последующим ручным кодированием;

- прямое программирование конструкций на языке общего назначения или специализированном языке представления знаний;

- описание сценариев с помощью аппарата деревьев событий во внешних средствах с последующим преобразованием в код.

Последний способ является наиболее предпочтительным с точки зрения специалистов предметников [19], которые в общем случае не владеют навыками программирования, однако знакомы с общесистемными и специализированными формализмами и нотациями, такими как графы, деревья событий и отказов.

Применение данного способа в контексте решения задачи потребовало разработки подхода обеспечивающего не только описание сценариев, но синтез программных кодов (Рис. Ж.1).

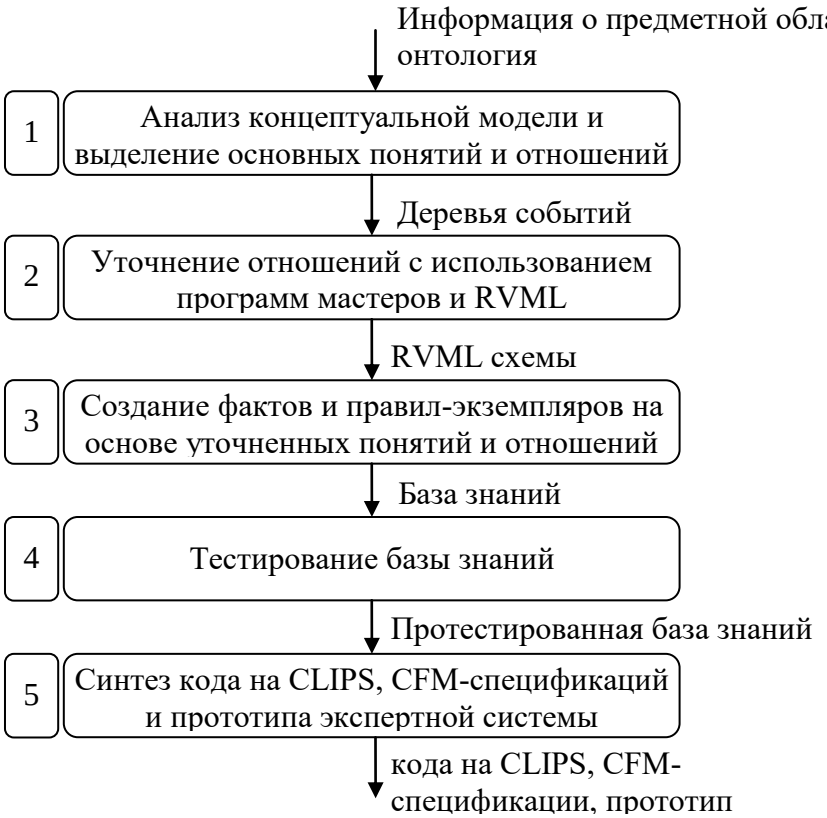


Рис.Ж.1. Схема разработки базы знаний с использованием деревьев событий

Разработанный подход предполагает формирование онтологии диагностики неисправности воздушного судна; формирование деревьев событий, описывающих последовательности работ по устранению конкретных неисправностей; построение баз знаний продукционного типа. При этом используются программные средства, интегрированные «по данным»: Protégé, Extended Event Tree Editor (EETE) [287] и Personal Knowledge Base Designer (PKBD) [367]. Интеграция «по данным» реализуется путем модельных трансформаций.

Определение неисправностей воздушного судна

Несмотря на актуальность задачи диагностики гражданских воздушных судов и существования автоматизированных диагностических систем, таких как AirNav компании AirBus для A319/A320/A321, A330, A340 и A380, создание подобных систем для отечественных самолетов остается нерешенной задачей. Необходимо отметить, что исследования в данной области ведутся и можно выделить как принципиальные решения концептуального уровня [314, 334],

которые не привязаны к конкретному судну, так и конкретные работы, направленные на оцифровку технической документации и ее интеграцию с бортовой системой технического обслуживания (БСТО) [343]. При этом практически отсутствуют работы направленные на использования методов искусственного интеллекта для поддержки поиска, локализации и ликвидации неисправностей.

Среди популярных методов решения задачи исследования можно выделить онтологии, как способ концептуализации и согласования знаний [79, 253] и продукционные экспертные системы [15, 171], которые до сих пор привлекают предметников своей наглядной простотой представления знаний и прозрачностью принимаемых решений.

Онтологическое моделирование

В рамках данного проекта онтологии рассматривались как способ унификации (согласования) представлений предметников и разработчиков программного обеспечения. Для формирования онтологии был осуществлен анализ документации: руководства по технической эксплуатации (РЭ) и руководства по поиску и устранению неисправностей (РУН) воздушного судна. При этом РЭ описывает информацию по обслуживанию, замене, регулировке, осмотру и проверке оборудования систем самолета, выполняемых на перроне или в ангаре технического обслуживания. РЭ также содержит информацию об осмотрах и техническом обслуживании конструкции планера самолета и содержит описание процедур планового технического обслуживания самолета. В свою очередь РУН перечисляет возможные отказы и неисправности, последовательности действий и работ по их выявлению и устранению по каждой системе воздушного судна.

В рамках текущего исследования был проанализирован раздел 24, связанный с электроснабжением Сухой Суперджет (RRJ-95), где рассматриваются неисправности 7 подсистем.

В качестве средства концептуализации использовано полуполярное программное средство Protégé, которое позволило структурировать основные понятия и отношения в форме онтологии диагностики неисправности воздушного судна. Техническое диагностирование (диагностирование технического состояния) – процесс определения технического состояния изделия с определенной точностью, результатом которого является заключение о техническом состоянии объекта с

указанием при необходимости места, вида и причин дефектов [215]. Данный процесс является частью процесса технической эксплуатации рассматриваемого объекта, в котором можно особо выделить процесс технического обслуживания и ремонта.

Определим онтологию, которая использовалась для технического диагностирования, как отношения между понятиями: объект диагностирования, где протекают процессы, обусловленные функционированием, и процесс технического диагностирования, представляющий собой последовательность работ, направленных на решение частных задач процесса (Рис.Ж.2).

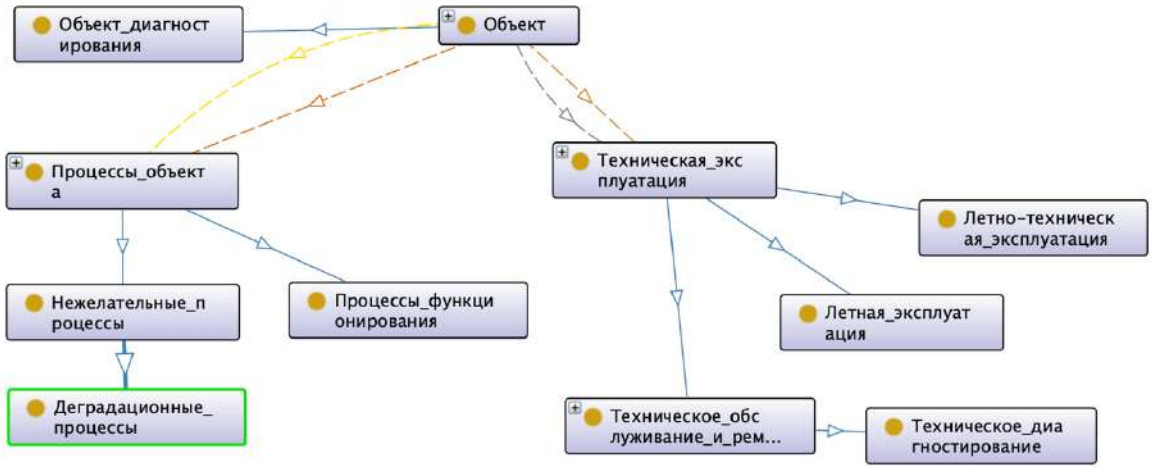


Рис. Ж.2. Отношения между понятиями объекта и процесса технического диагностирования

При этом разработанная онтология включает онтологию объекта, онтологию работ, разделяющуюся на онтологии работ технической эксплуатации, работ по техническому обслуживанию, работ по поиску и устранению неисправностей.

Онтология объекта может быть создана на основе шаблона [300] онтологии, отражающего структуру: система – подсистема – под-подсистема, что аналогично структуре объекта онтологии технического обслуживания механических систем [215]: механическая система – сборочная единица – деталь (Рис.Ж.3).

Рис. Ж.3. Понятия, отражающие структуру исследуемого объекта

Для систематизации информации об объекте исследования необходимо использовать иерархию типа «общее-частное» для описания иерархии систем, подсистем и под-подсистем по их назначению. Данный аспект реализации онтологии показан на примере системы электроснабжения (Рис.Ж.4-Ж.5).

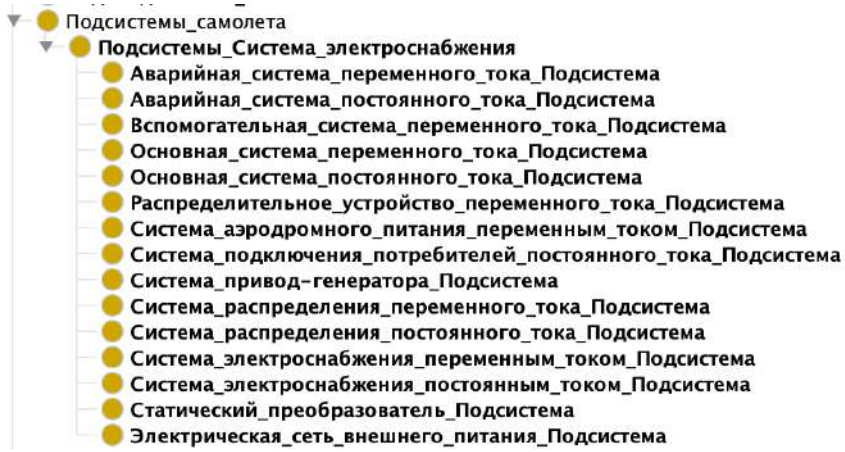


Рис. Ж.4. Понятия, отражающие перечень подсистем системы электроснабжения

Рис. Ж.5. Понятия, отражающие перечень под-подсистем системы электроснабжения

На основе иерархии «общее-частное» была описана иерархия «часть – целое» на основе структурного шаблона, описанного выше, и соответствующих объектных свойств: имеет подсистемы, имеет под-подсистемы (Рис.Ж.6).

Онтология работ также описывает иерархию «общее – частное» и «часть – целое». В первом случае использованы различные основания классификации: по назначению (Рис.Ж.7), по функциональному коду (Рис.Ж.8), классификация работ согласно структуре объекта (Рис.Ж.9), виду работ с точки зрения выполняемых процессов: технической эксплуатации, техническое обслуживание, ремонт и устранение неисправностей.

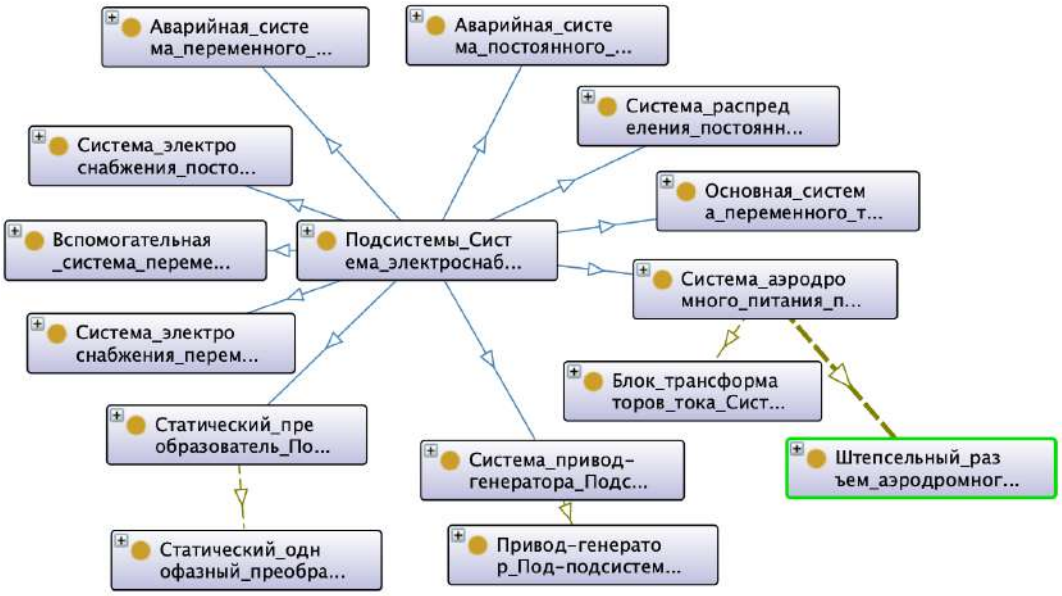


Рис. Ж.6. Фрагмент онтологии, отражающей систему электроснабжения

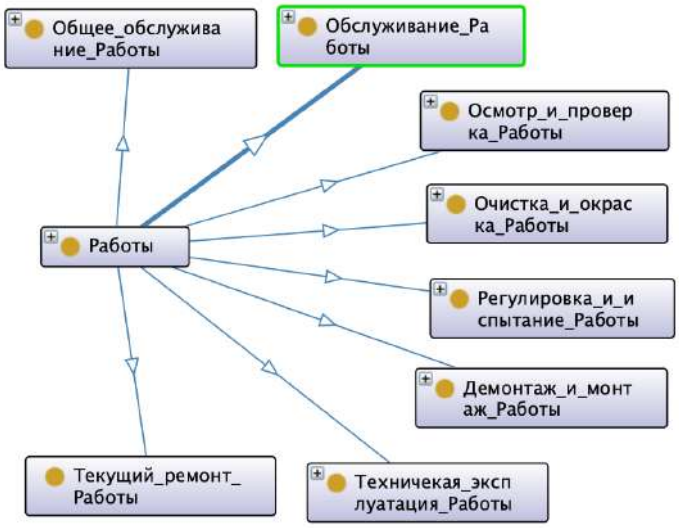


Рис.Ж.7. Классификация работ по назначению

Рис.Ж.8. Классификация работ по функциональному коду

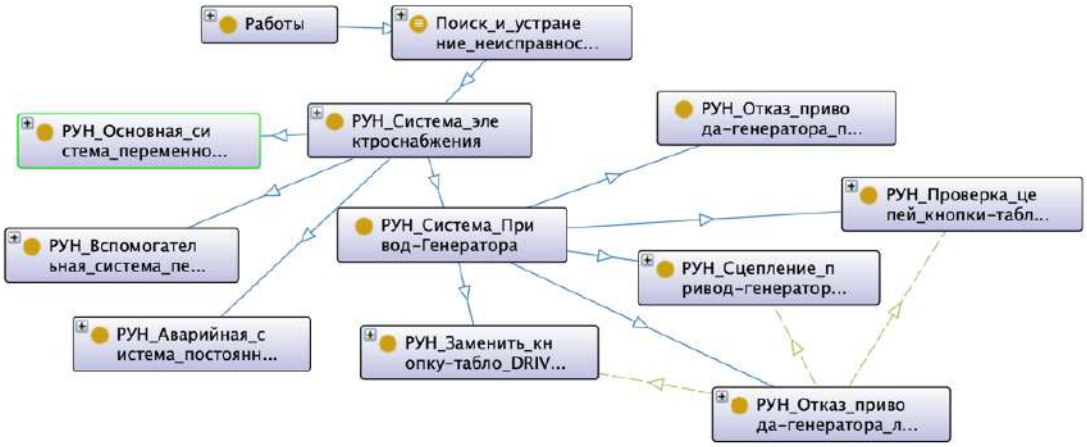


Рис.Ж.9. Фрагмент классификации работ по устранению неисправностей (РУН) для системы электроснабжения

Второй тип отношений позволил описать структуру работ, т.е. какая из работ является частью другой работы (Рис.Ж.10). Работа по устранению неисправностей содержит работы, группируемые по этапам процесса выполнения работы: подготовительный этап, подтверждение неисправности, поиск и устранение неисправности, подтверждение устранения неисправности и заключительный этап. Данные работы описаны в виде иерархий «общее - частное» и «часть целое». Последний тип отношений описан с помощью объектных свойств: имеет подготовительные работы, имеет заключительные работы, имеет работы по подтверждению, поиску и устранению неисправностей и др.

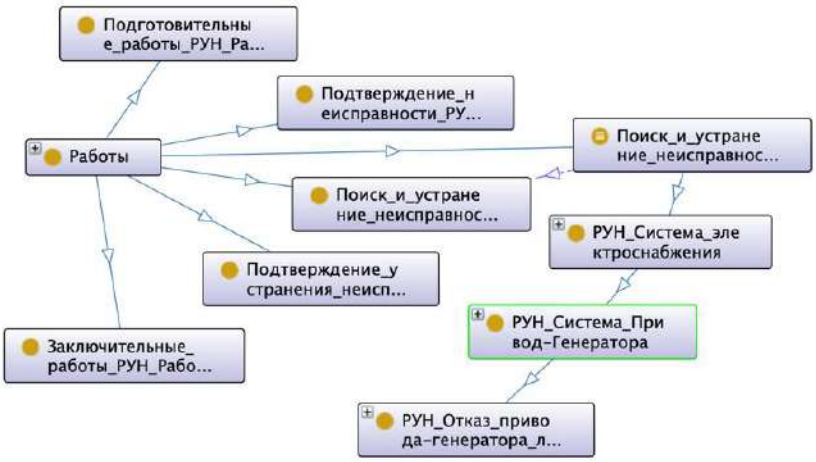


Рис. Ж.10. Структура понятий информации о работе с детализацией одной из РУН подсистемы привод-генератора

Дальнейшее описание детализирует различные свойства работ и их отношения. В частности, работы по поиску и устранению неисправностей включают: работы подготовительные, подтверждающие неисправность и др. (Рис.Ж.11), номер работы, перечень признаков неисправностей, возможные причины отказа (Рис.Ж.12) и др.

В настоящее время онтология содержит 355 понятий и 1066 аксиом.

Онтология использована в качестве исходной модели автоматизированного создания продукционной базы знаний для осуществления поддержки принятия решений при формировании последовательности работ по поиску и устранению неисправностей.

Создание базы знаний на основе модельных трансформаций

Рассмотрим процесс создания сегмента базы знаний с использованием разработанной онтологии, которая в контексте решаемой задачи может быть рассмотрена как часть вычислительно-независимой модели.

Другая часть вычислительно-независимой модели состоит из деревьев событий, целью использования которых является построение сценариев или последовательностей работ. Как показала практика, решение данной задачи средствами Protégé является достаточно трудоемким и не очевидным (не естественным). В связи с этим было решено обеспечить преобразование (трансформацию) части результатов онтологического моделирования, в частности, информации о работах и неисправностях в формат Extended Event Tree Editor (EETE) [287]. EETE представляет собой специализированный редактор для построения классических деревьев событий и отказов, а также расширенного их варианта с использованием понятия «Механизм» и тематических уровней [19, 257].

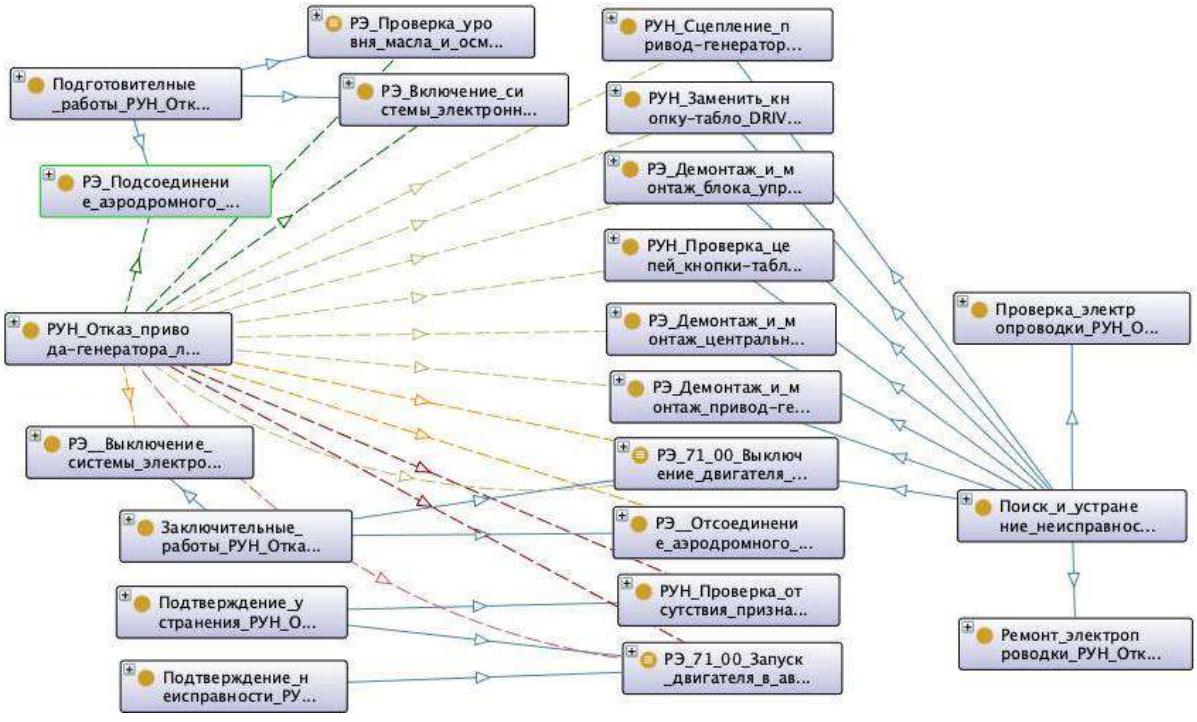


Рис. Ж.11. Детализированная структура работы «Отказ привода-генератора левого двигателя или расцеплены привод-генератор и коробка приводов левого двигателя»

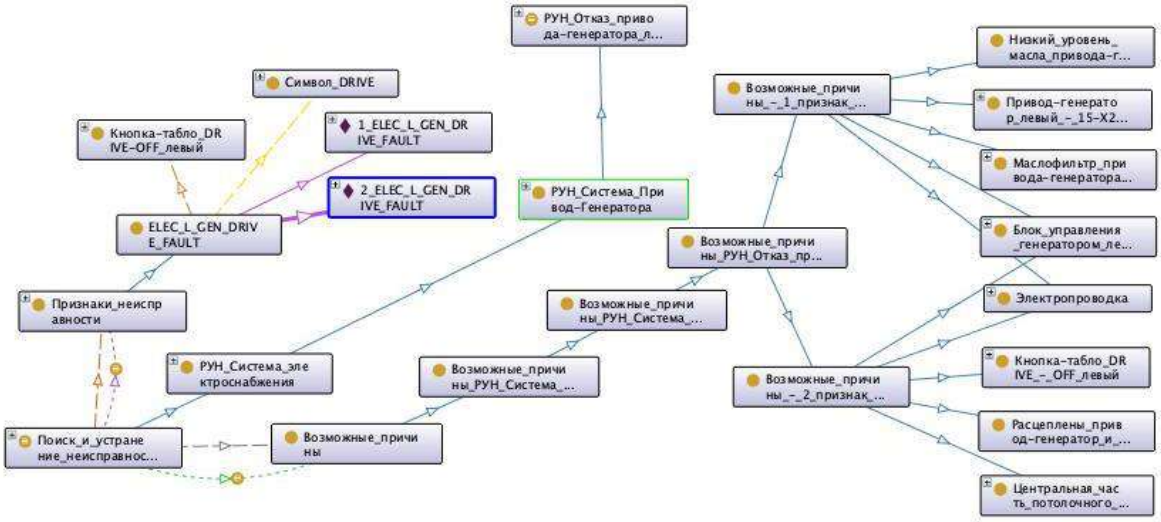
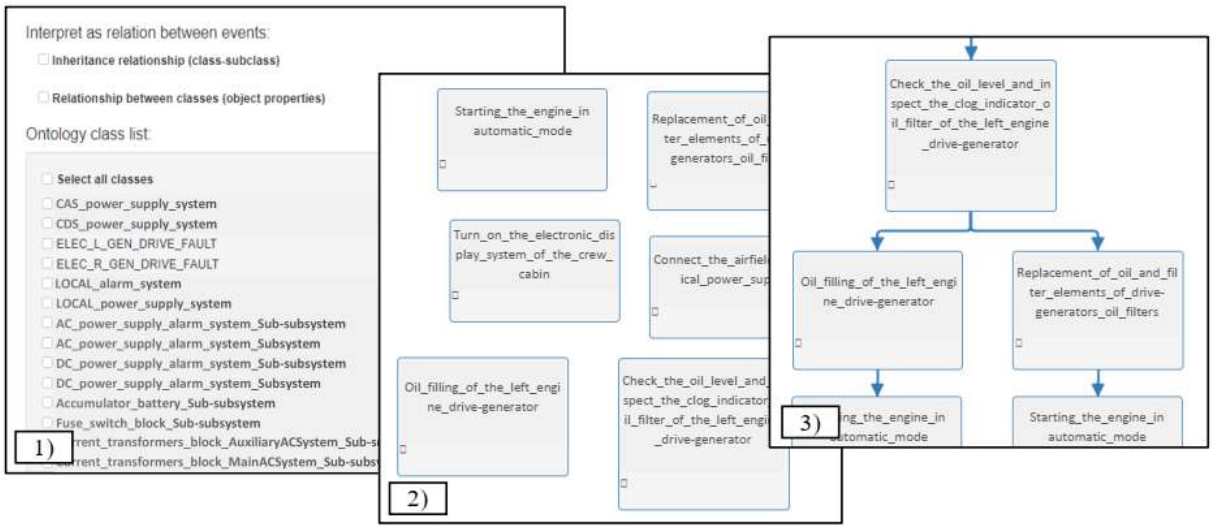


Рис. Ж.12. Фрагмент свойств работы «Отказ привода-генератора левого двигателя или расцеплены привод-генератор и коробка приводов левого двигателя»

В результате трансформации в EETE формируется набор доступных событий онтологии (работ и неисправностей), манипулируя которыми в интерактивном режиме происходит построение сценариев (Рис.Ж.13). Важным аспектом успешного построения подобных сценариев является наличие свойства «Условие» в описании работ, обеспечивающего их возможную семантическую «склейку». Результатом данного этапа является формирование древовидных структур, описывающих технологические карты для определенных неисправностей.



РисЖ.13. Импортирование онтологии в редактор деревьев событий: 1) Выбор понятий и способа интерпретации отношений; 2) Импортированные понятия преобразованы в события; 3) Уточненное дерево событий

Все преобразования между форматами выполнены в соответствии с концепцией модельных трансформаций модельно-ориентированного подхода [41]. Для реализации подобных трансформаций используются как специализированные языки (например, ATL, Epsolon, TMRL [46]), так и языки общего назначения. В

данной работе трансформации реализованы с помощью языков общего назначения PHP и Object Pascal. В упрощенном виде они могут быть представлены в виде таблицы соответствий (Таблица Ж.1).

Таблица Ж.1. Основные соответствия между элементами форматов OWL, EETE и EKB (PKBD)

OWL	EETE	PKBD
owl:Ontology	Diagram	KowledgeBase
owl:Class	Event	Template
owl:ObjectProperty	Operator	
owl:ObjectProperty / rdfs:domain	Operator (lhs)	Condition
owl:ObjectProperty / rdfs:range	Operator (rhs)	Action
owl:DatatypeProperty	Parameter	Slot
owl:DatatypeProperty / rdfs:domain	Parameter (class)	Slot (name)
owl:DatatypeProperty / rdfs:range	Parameter (value)	Slot (value)

На Рис. Ж.14 представлен фрагмент дерева событий и соответствующий ему XML-подобный код.

Рис. Ж.14. Фрагмент дерева работ по устранению неисправности и соответствующий ей XML-подобный код (EETE)

На текущий момент практически отсутствуют средства обеспечивающие генерацию кодов баз знаний из древовидных семантических структур. В связи с этим была обеспечена трансформация построенных деревьев при помощи модуля расширения PKBD.ExTree в структуры базы знаний. PKBD обеспечило уточнение полученных структур в соответствии с формализмом логических правил, а также их тестирование благодаря встроенному интерпретатору. По результатам отладки возможна кодогенрация на CLIPS (C Language Integrated Production System), DROOLS и PHP (Hypertext Preprocessor) для дальнейшей интеграции и использования.

База знаний для диагностики неисправностей системы электроснабжения RRJ-95

В качестве примера рассмотрим построение фрагмента базы знаний для формирования последовательности работ по устранению неисправности «ELEC APU GEN FAULT».

Итак, как было отмечено выше, на основе анализа нормативных документов была сформирована иерархическая онтология, содержащая, в том числе, информацию о неисправностях, их проявлениях и работах по их устранению.

Внешним признаком неисправности является сообщение на дисплее параметров двигателей и аварийных сообщений «ELEC APU GEN FAULT» (символ APU GEN желтого цвета), возможные причины: блок трансформаторов тока (TBD); блок управления генератором ВСУ и наземным питанием (4-X242); генератор ВСУ(1-X242); кнопка табло APU GEN (пульт электроснабжения); центральная часть потолочного пульта (5-F311); электропроводка.

Из работ, связанных с неисправностью, формируются классы онтологии. Далее, используя возможности импорта в EETE переносится информация о работах, что позволяет сформировать их последовательности (деревья) (Рис. Ж.14). Построенные деревья импортируются в PKBD с созданием шаблона факта «Работа», шаблона правила, описывающего связь работ, и конкретные правила с информацией о связях. После уточнения симпортированной информации осуществлено тестирование фрагмента и синтезирован программный код (Рис. Ж.15), который при необходимости может быть дополнен.

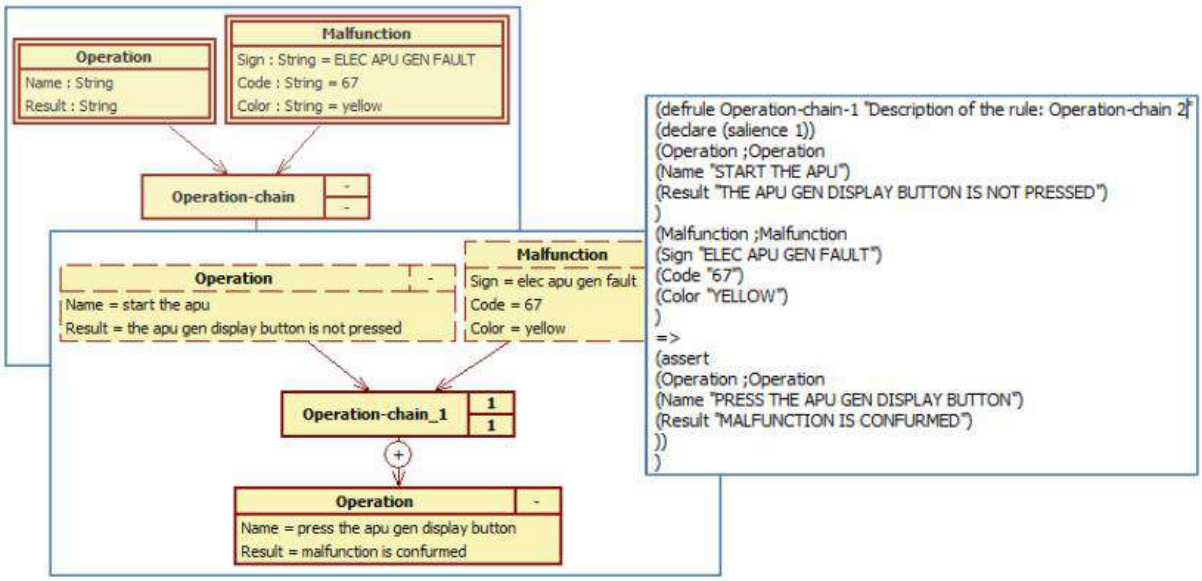


Рис. Ж.15. Примеры шаблона правила в нотации RVML и соотвествующего ему сгенерированного CLIPS кода (PKBD)

На Рис. Ж16-Ж.17 представлены примеры экранных форм ИС ППР «АвиаТехПом».

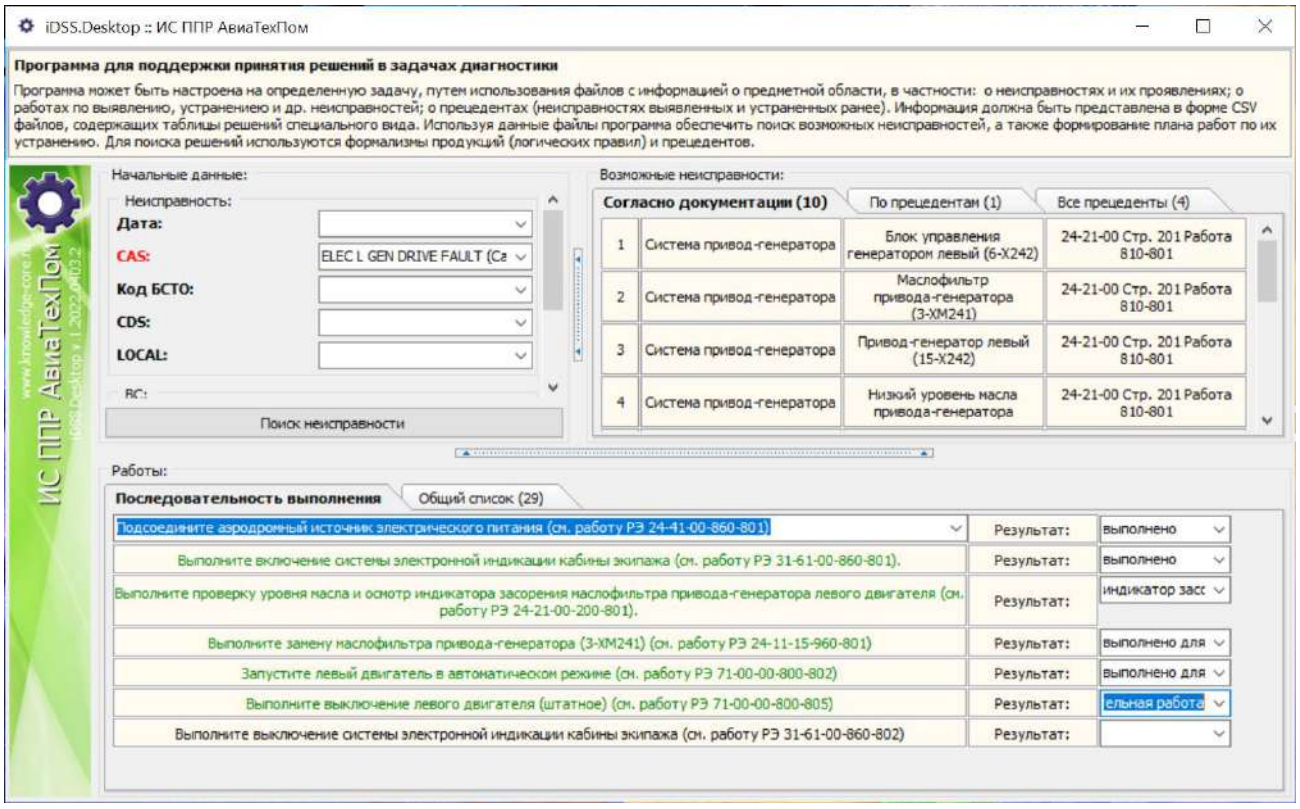


Рис. Ж.16. ИС ППР «АвиаТехПом»: Результаты поиска возможных неисправностей и список работ для их устранения

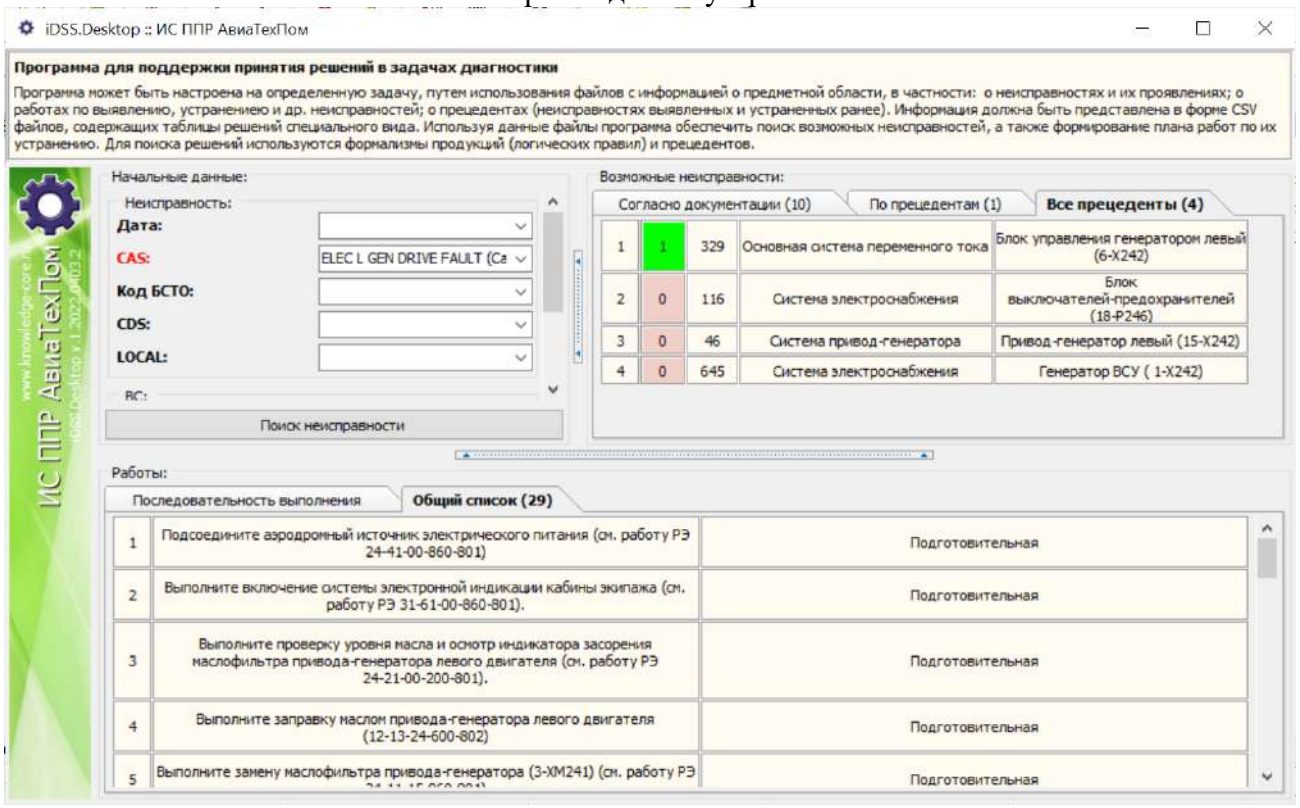


Рис. Ж.17. ИС ППР «АвиаТехПом»: Список прецедентов и полный сртсок работ

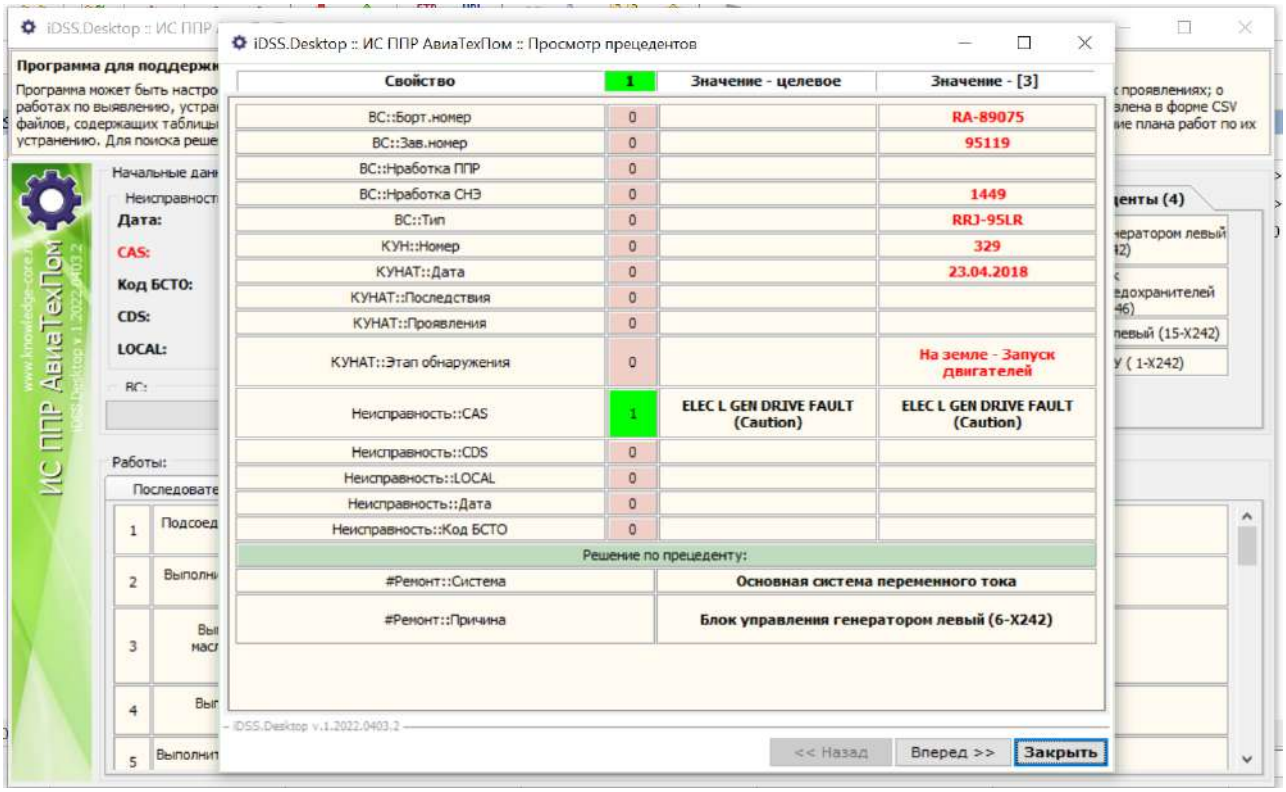


Рис. Ж.17. ИС ППР «АвиаТехПом»: Просмотр прецедентов

По результатам применения технологии при решении данной задачи получен акт внедрения программы для ЭВМ (см. Приложение А).

Приложение З
База знаний сервиса для прогнозирования риска природных пожаров

Одной из решаемых задач при помощи разработанных методов и средств являлась задача разработки интеллектуальных модулей в форме WPS сервисов для поддержки прогнозирования риска природных лесных пожаров [200, 202], что являлось частью работ по проекту Министерства науки и высшего образования Российской федерации грант № 075-15-2020-787 «Фундаментальные основы, методы и технологии цифрового мониторинга и прогнозирования экологической обстановки Байкальской природной территории» [28].

При решении данной задачи рассматривались два способа формирования оценок риска пожароопасности в лесных кварталах:

- основанный на статистическом анализе информации о пожарах за предыдущий период, с учетом определенного лесного квартала и временного интервала. Предполагает статистическую обработку больших объемов данных, получаемые оценки не зависят от условий запроса;
- основанный на методах искусственного интеллекта, в частности, продукционных экспертных системах. Предполагает не только статистическую обработку больших объемов данных, но и концептуальное моделирование, интеллектуальный анализ данных с целью поиска закономерностей и их дальнейшую формализацию в виде логических правил. При этом появляется возможность получения оценок, учитывающих условия запросов.

Для реализации этих способов решались следующие задачи:

- сбор, очистка и анализ данных о лесных пожарах и выявлением эвристик, позволяющих идентифицировать территориальный риск пожароопасности леса, который в данной работе будет рассматриваться как риск пожароопасности лесного квартала/дачи/лесничества;
- разработка базы знаний для анализа и прогнозирования риска (опасности) лесного пожара на основе информации о классе пожароопасности лесов, метеоусловий и других факторов.

Процесс решения второй задачи соответствует методу и методике (см. 4.1) и может быть представлен в виде следующей схемы (Рис.З.1).

В качестве исходных данных использовалась информация о термальных

точках (пикселах горения), выявленных по спутниковым данным с использованием программного комплекса FireProc, созданного в ИСЗФ СО РАН. Данные термальные точки в работе рассматриваются как возможные лесные пожары на Байкальской природной территории. Данные охватывают период с 2017-2020 гг. и включают более 45 000 записей с информацией о координатах термальных точек, дате обнаружения и площади распространения. Для решения задач определения причин возникновения пожаров и прогнозирования их динамики в рамках работы осуществлен сбор и подготовка данных о погодных условиях, а также информация об инфраструктуре (дороги, населенные пункты), лесничеств, лесных участков и дач, типе растительности и классах пожароопасности лесов.

Рис. 3.1. Схема разработки базы знаний с использованием PKBD

Прогнозирование риска лесных пожаров

Тематика автоматизированного формирования баз знаний для интеллектуальной поддержки принятия решений при прогнозировании риска (опасности) пожаров в лесных кварталах в научной литературе представлена достаточно слабо. Можно отдельно выделить работы, связанные с модельными трансформациями и применением модельно-ориентированного подхода, а также отдельно работы, направленные на прогнозирование риска (опасности) лесных пожаров.

Первая группа работ более подробно рассмотрена в [27, 184, 192].

Во второй группе можно выделить несколько работ, иллюстрирующих основные направления исследований:

- выявление факторов, влияющих на пожароопасность лесов [127, 248], в том числе: высота, уклон, топографический индекс влажности, аспект, расстояние от городских районов, среднегодовая температура, землепользование, расстояние от дорог, среднегодовое количество осадков, расстояние до реки, температура воздуха (среднесуточная и максимальная), даты перехода средних суточных температур через пороговые пределы, даты наступления и схода устойчивого снежного покрова, относительная влажность (среднесуточная и минимальная), дефицит влажности воздуха, число дней с относительной влажностью ≤ 30 % в один из сроков наблюдения за определенный период, годовой режим выпадения атмосферных осадков, число дней с дождем, индекс сухости, ветровой режим, число дней с грозой и др.;
- совершенствование шкалы оценки классов пожарной опасности лесов в зависимости от условий погоды [301] с целью введения новых факторов, в частности, показателей влажности [338], либо учета региональных особенностей [359];
- применение существующих методик для оценки пожаропасности в различных регионах [347].

В контексте данного исследования приведенные работы были использованы для анализа предметной области и выявления факторов, влияющих на оценку пожароопасности лесов.

Создание прототипа базы знаний сервиса

Важными и вычислительно сложными задачами, связанными с подготовкой данных для разработки баз знаний, являются следующие:

- группировка (агрегирование) информации о пожарах с определением продолжительности, минимальной и максимальной площади определенного возгорания;
- выделение пожаров, расположенных в границах промышленных зон, населенных пунктов и зон добычи полезных ископаемых, не являющихся природными пожарами.

- определение статистики пожаров для определенных классов пожароопасности на основе лесных планов лесничеств Иркутской области (с учетом структуры лесничество/участки(дачи)/кварталы);
- определение набора независимых факторов, влияющих на риск пожароопасности лесного участка;
- расчет значений факторов влияющих на риск пожароопасности лесного участка и их преобразование к интервальному или нечеткому виду;
- определение риска (опасности) лесного пожара по текущим значениям комплекса факторов через определенный класс пожароопасности и его статистику

Более подробно данные задачи будут рассмотрены в других работах. В данном разделе далее рассмотрен процесс построения баз знаний на основе анализа таблиц решений, считая, что предобработка данных уже выполнена. При этом используется специализированная форма таблиц решений, рассмотренная выше.

В ходе применения технологии была построена концептуальная модель предметной области, описывающая факторы, влияющие на класс пожарной опасности лесного участка и риск (вероятность) возникновения пожара. Фрагмент модели показан на Рис. 3.2.

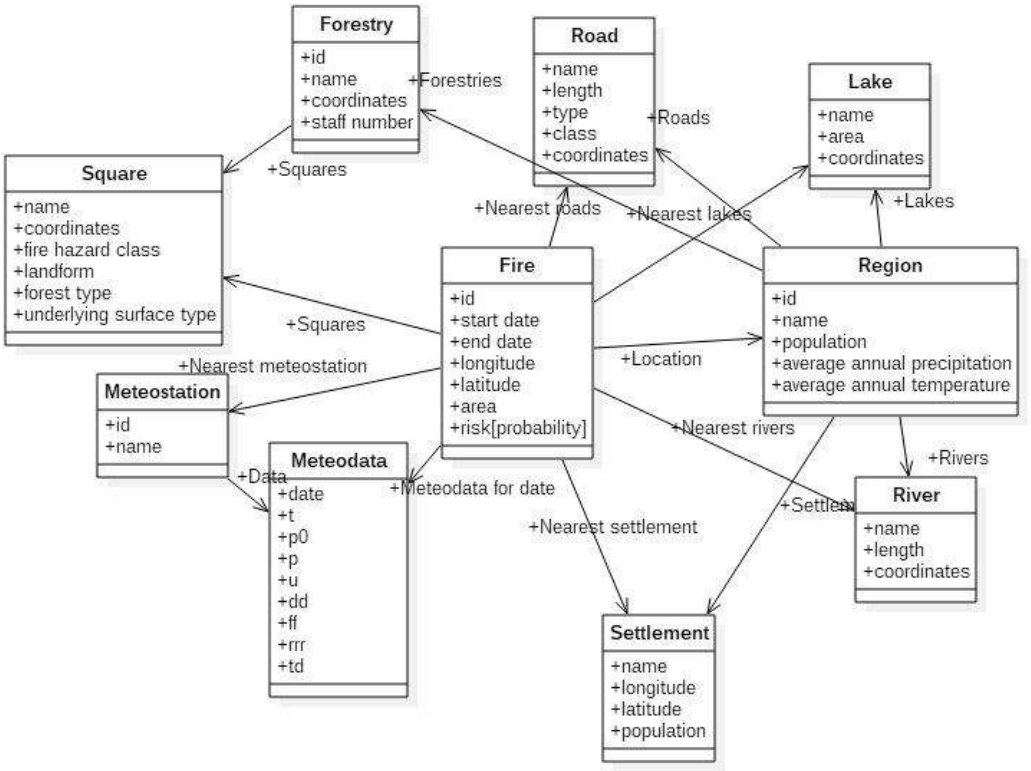


Рис. 3.2. Фрагмент концептуальной модели предметной области

Далее были разработаны таблицы решений, описывающие структурный аспект предметной области. Эти таблицы содержат информацию о сочетаниях признаков, описывающих класс пожарной опасности лесного участка и риск (опасность) лесного пожара.

В частности, следующая структура таблиц (заголовки) была определена для определения класса пожарной опасности:

```
Road::distance_to_car_road, Road::distance_to_railway,
River::distance_to_river, Lake::distance_to_lake, Meteodata::rrr,
Meteodata::ff, Meteodata::u, Meteodata::t,
Settlement::distance_to_settlement, Settlement::population,
Region::population, Region::average_annual_temperature, Season::name,
Forestry::staff_number, Square::landform, Square::forest_type,
Square::underlying_surface_type, #Square::name,
#Square::fire_hazard_class.
```

Для определения риска (вероятности) пожара используется следующая структура таблицы (заголовки):

```
Square::name, Square::fire_hazard_class, Season::name,
#Fire::risk[probability].
```

Таблица решений с промежуточными данными представлена на Рис.3.3. Затем с помощью модуля расширения PKBD.DnTable таблицы решений были преобразованы и представлены в виде логических правил, которые были уточнены в форме диаграмм RVML (Рис.3.4) с дальнейшей генерацией программных кодов на CLIPS и PHP.

Данные результаты будут в дальнейшем использованы при программной реализации сервиса.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	id	Fire::id	Fire::date	Fire::lon	Fire::lat	Fire::poly	Fire::ar	Road::di	Road::di	River::dist	Lake::dist	Meteostation::id	Meteosta	Mete	Me	Meteodat	Meteodat	Square::name	Square::fire_
2	70	23	01.04.2017 8:02	106.806	57.9885	0106000020	2.065	0.270	0.244	0.029	0.350	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 94],	['3']
3	337	23	02.04.2017 11:24	106.811	57.9808	0106000029	9.693	0.267	0.240	0.026	0.348	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 94],	['3', '2']
4	387	23	03.04.2017 7:39	106.813	57.9837	0106000020	11.663	0.267	0.240	0.026	0.348	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 94],	['3', '2']
5	540	23	04.04.2017 5:47	106.807	57.9816	0106000021	7.781	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
6	615	23	05.04.2017 21:10	106.807	57.9816	0106000021	7.781	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
7	651	23	06.04.2017 7:05	106.807	57.9816	0106000021	7.781	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
8	670	23	09.04.2017 1:49	106.805	57.98	0106000021	9.808	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
9	739	23	10.04.2017 9:51	106.805	57.981	0106000021	0.830	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
10	839	23	11.04.2017 21:42	106.807	57.9804	0106000022	2.300	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
11	855	23	12.04.2017 1:14	106.807	57.9804	0106000022	2.300	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
12	932	23	13.04.2017 7:25	106.807	57.9804	0106000022	2.300	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
13	1050	23	14.04.2017 21:07	106.807	57.9804	0106000022	2.300	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
14	1085	23	15.04.2017 7:02	106.807	57.9804	0106000022	2.323	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
15	1369	23	16.04.2017 20:44	106.807	57.9804	0106000022	2.323	0.264	0.239	0.023	0.345	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
16	1412	23	17.04.2017 11:51	106.805	57.9783	0106000022	4.940	0.263	0.238	0.022	0.343	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
17	1495	23	18.04.2017 11:40	106.804	57.9786	0106000022	5.312	0.263	0.238	0.022	0.343	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
18	1544	23	19.04.2017 11:28	106.804	57.9786	0106000022	5.312	0.263	0.238	0.022	0.343	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']
19	1590	23	20.04.2017 11:16	106.804	57.9786	0106000022	5.312	0.263	0.238	0.022	0.343	30229	ВЕРХНЕМАРКОВ			30.0	20.0	['Ust-Kutskoe', 93],	['3', '2']

Рис. 3.3. Фрагмент таблицы решений с промежуточными данными

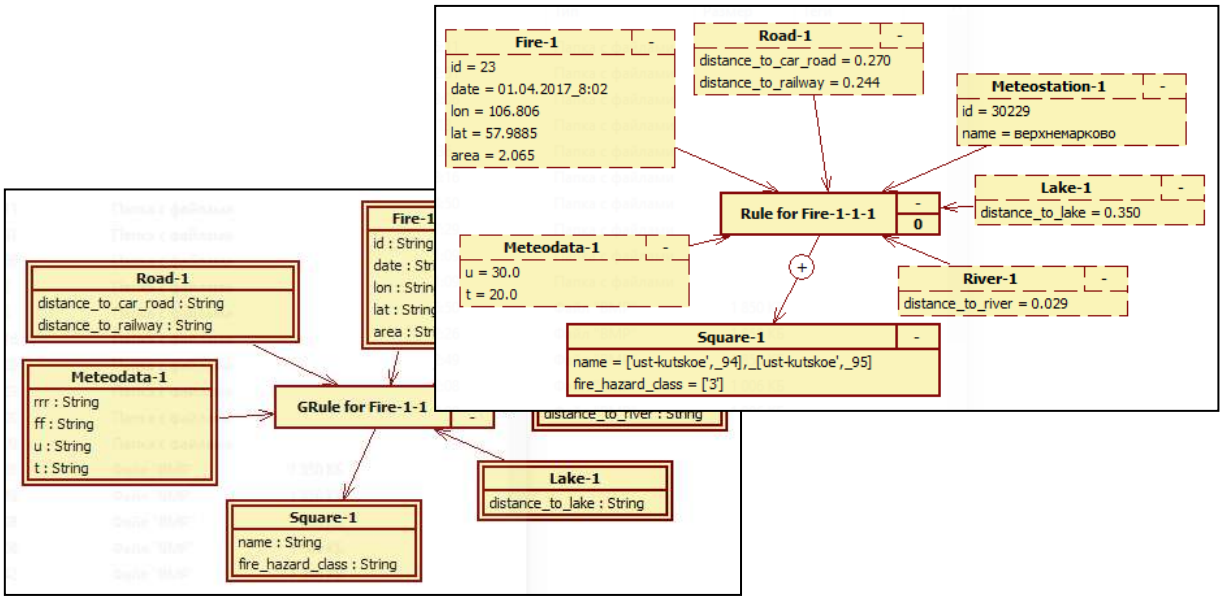


Рис. 3.4. Примеры шаблона правил и конкретного в форме диаграмм RVML

Приложение И
Разработка программных компонентов-конверторов концептуальных моделей

Апробация методов и средства создания программ трансформации и компонентов-конверторов осуществлено на примере трансформаций диаграмм классов UML, концепт-карт, деревьев событий.

Создание компонента для трансформации диаграмм классов UML

Процесс создания компонента [198, 266, 267] соответствует методам (см. 5.1 и 5.2). На начальном этапе определяется тип компонента, определяющий используемые целевые метамодели (для примера выбраны продукции), анализатор и генератор. Вся разработка ведется в KBDS при этом программное средство информирует пользователя о результатах его действий посредством сообщений.

В текущем примере в качестве исходных использованы UML диаграммы классов и соответствующая метамодель была получена в результате импорта исходной диаграммы в формате XML (Рис. И.1). В результате импорта были извлечены узлы XML-структуры, которые сформировали набор элементов метамодели; атрибуту узлов – свойства элементов; вложенность узлов – связи.

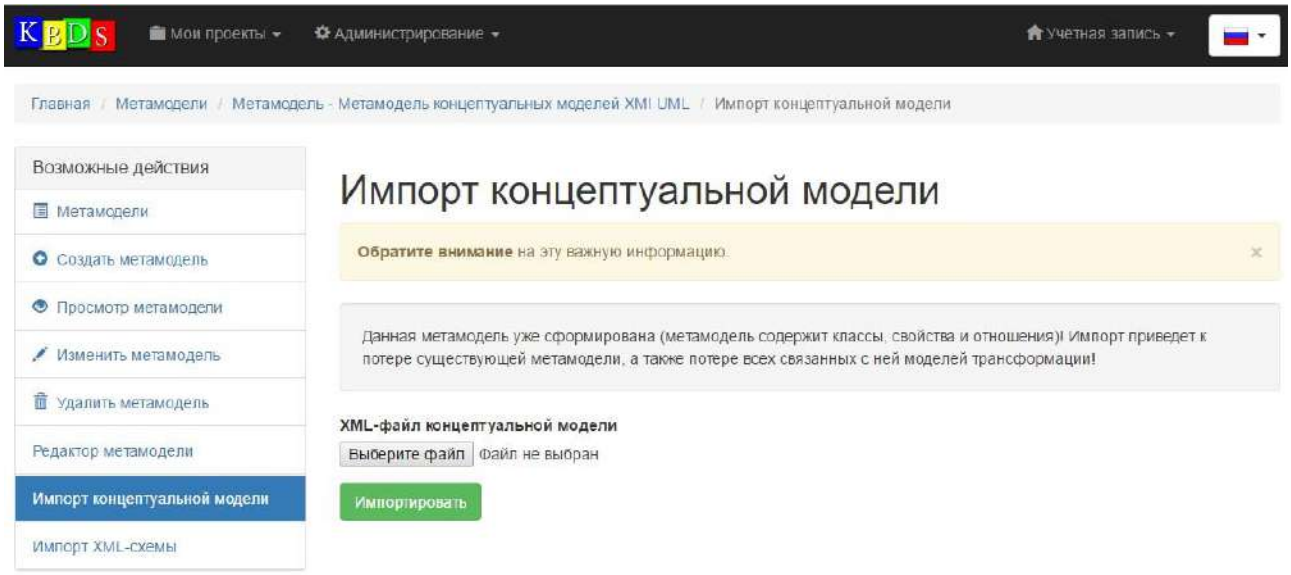


Рис. И.1. Фрагмент формы KBDS: Импорт концептуальной модели

Полученная метамодель была уточнена в редакторе метамоделей (Рис. И.2), на Рисунке И.3 представлена итоговая дополненная метамодель для данной задачи.

Далее, были описаны трансформации в специальном редакторе (Рис. И.4) путем установления соответствий между элементами исходной и целевой метамоделей.

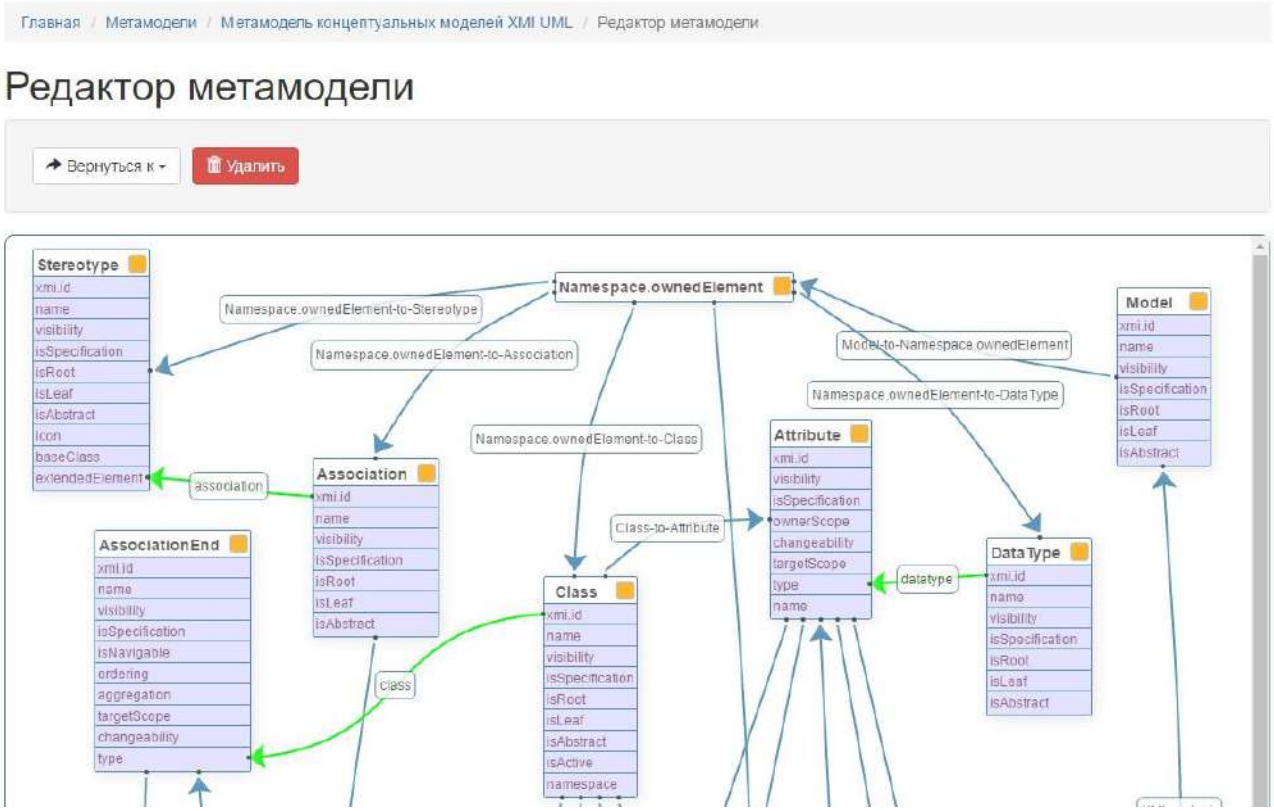


Рис. И.2. Фрагмент формы KBDS: Редактор метамоделей

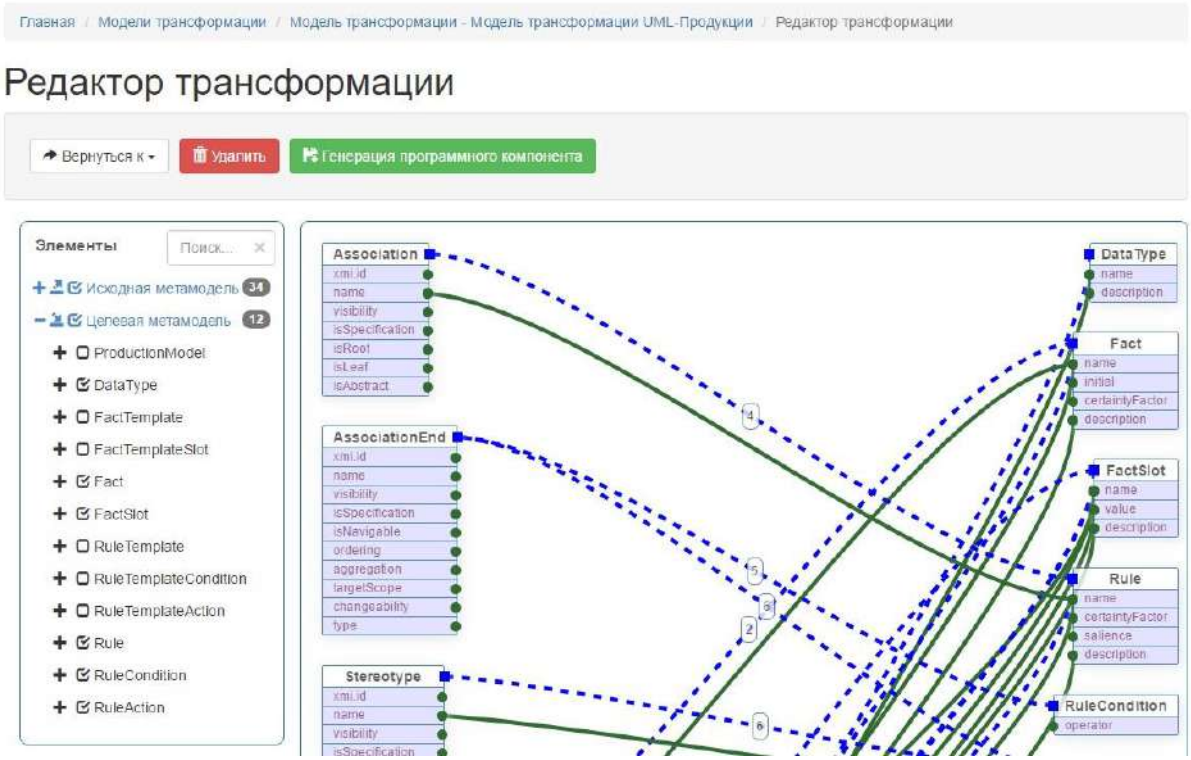


Рис. И.4. Фрагмент формы KBDS: Редактор трансформаций (описаны трансформации диаграмм классов UML в модель продукций)

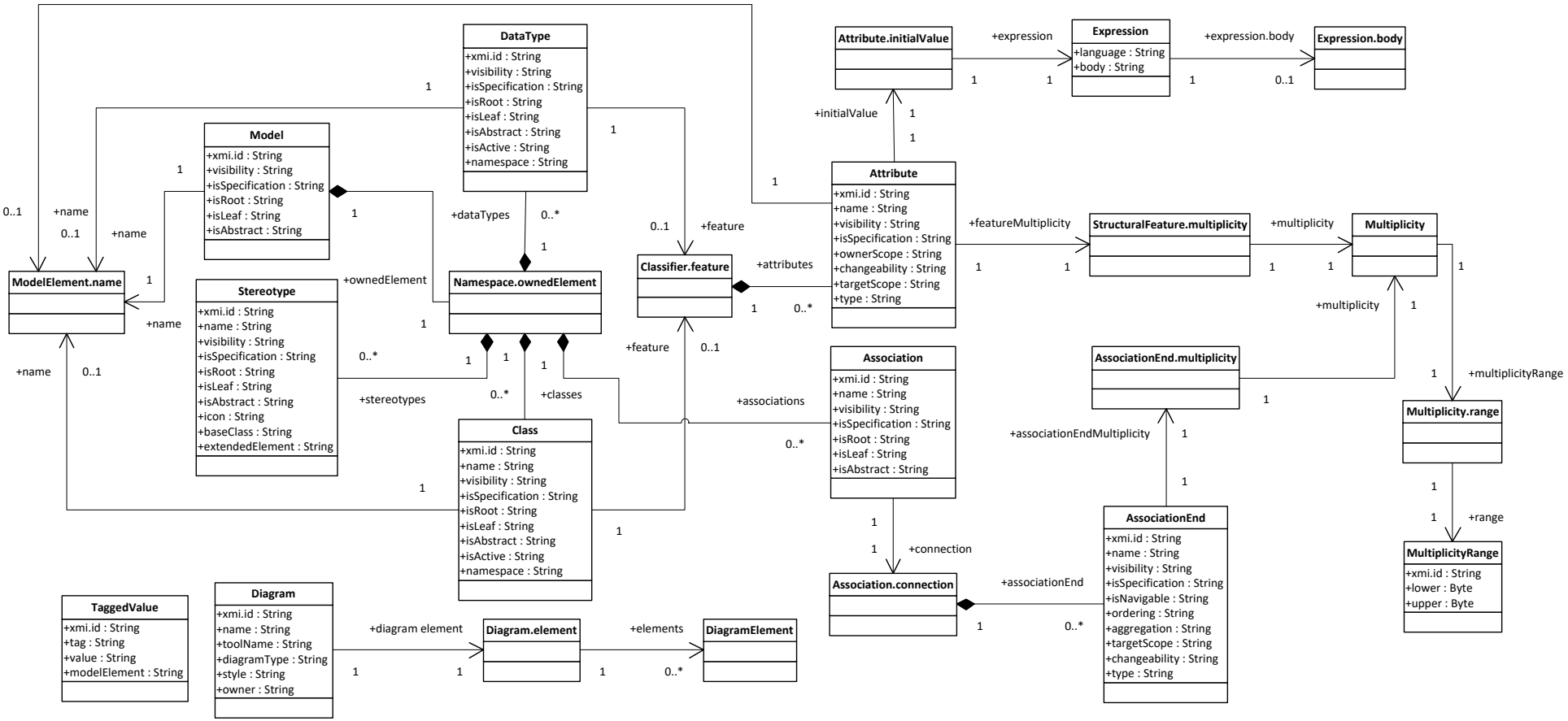


Рис. И.3. Итоговая метамодель UML диаграммы классов

При задании соответствий указываются приоритеты (очередность) выполнения того или иного правила трансформации. В данном примере модель трансформации содержит 6 правил, из них: одно простое бинарное и пять сложных.

Пример простого правила приведен на Рисунке И.6, фиксирует взаимно-однозначное (эквивалентное) соответствие между элементами.

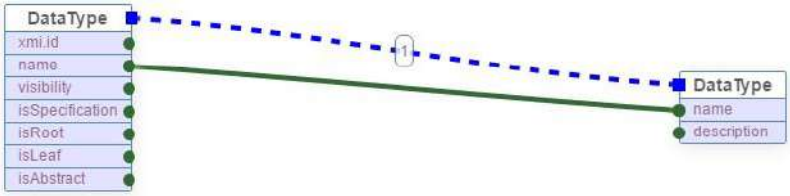


Рис. И.5. Визуальное отображение простого бинарного правила трансформации: соответствие типов данных

Примеры сложных правил приведены на рисунках И.6 и И. 7.

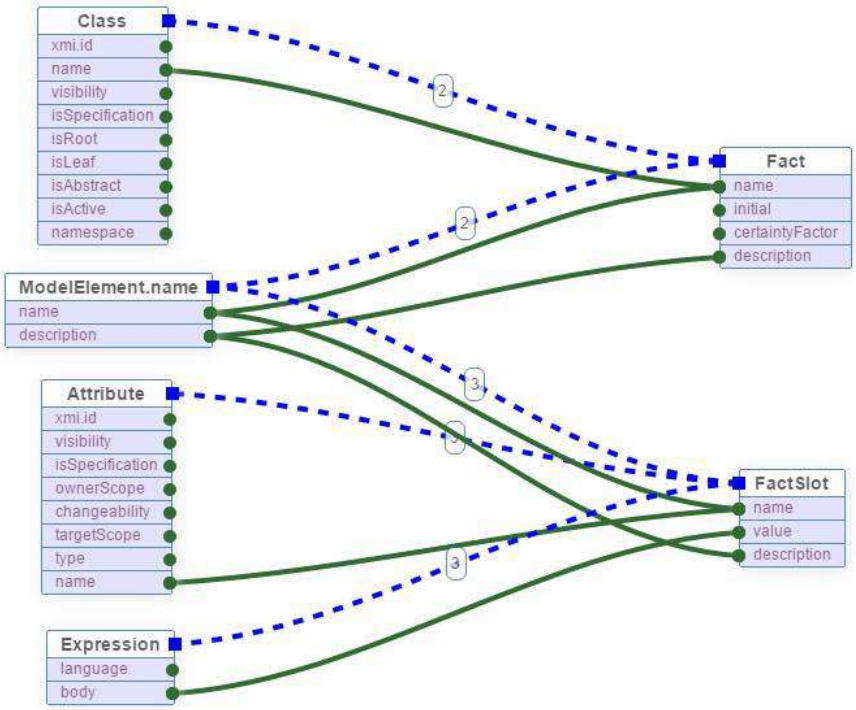


Рис. И.6. Визуальное отображение сложного правила: ModelElement.name может соответствовать двум элементам: именем класса или его атрибута

По завершению описания трансформаций был синтезирован TMRL код программы трансформации (Рис. И.8), который в дальнейшем был уточнен.

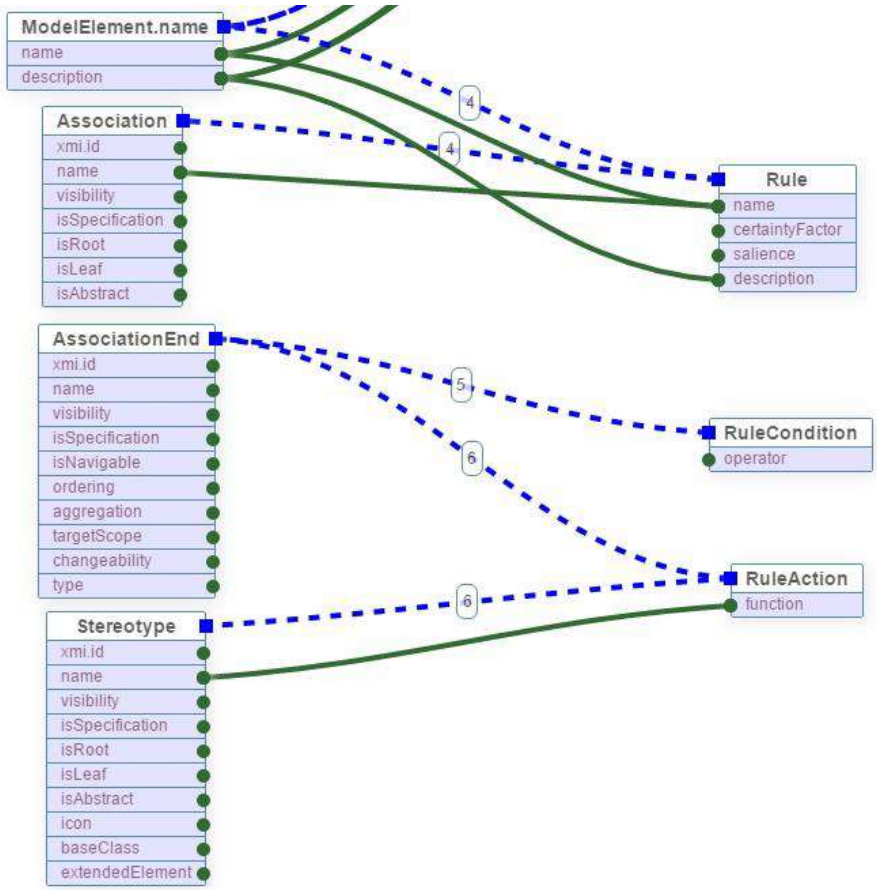


Рис. И.7. Визуальное отображение сложного правила: соответствие ассоциация-правило, часть ассоциации-условие и -действие

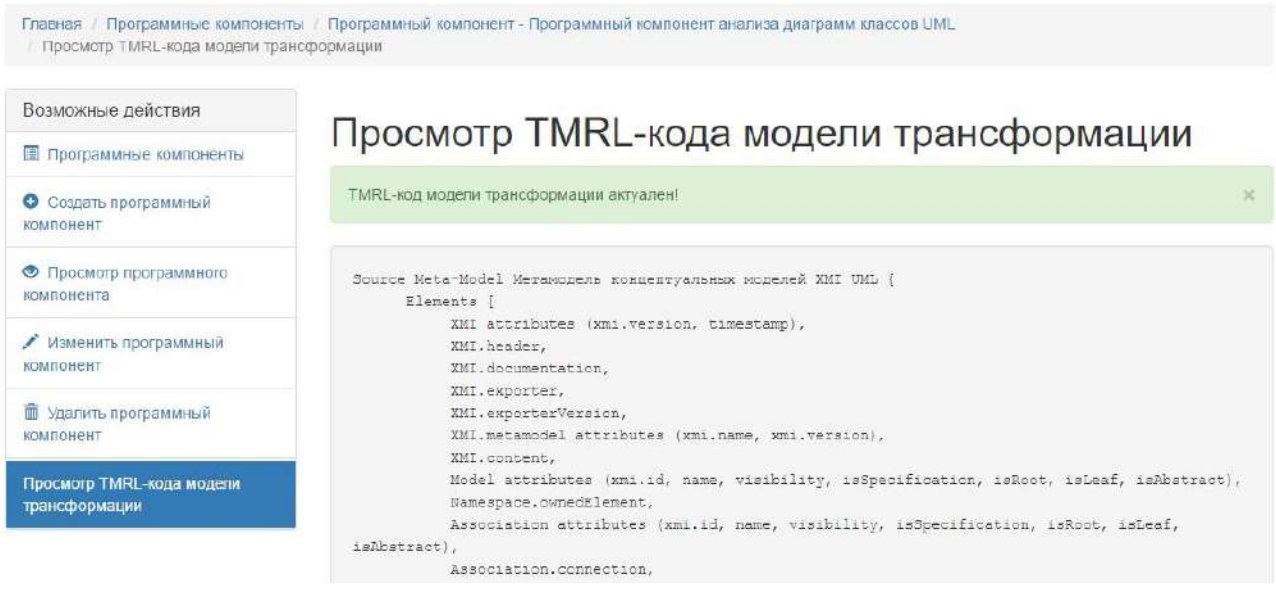


Рис. И.8. Фрагмент формы KBDS: Просмотра TMRL-кода

Ниже приведен полученный код TMRL программы для преобразования UML диаграмм классов в продукции:

```
Source Meta-Model Метамодель концептуальных моделей XMI UML {
Elements [
XMI attributes (xmi.version, timestamp),
XMI.header,
```

```
XMI.documentation,
XMI.exporter,
XMI.exporterVersion,
XMI.metamodel attributes (xmi.name, xmi.version),
XMI.content,
Model attributes (xmi.id, name, visibility, isSpecification, isRoot, isLeaf,
isAbstract),
Namespace.ownedElement,
Association attributes (xmi.id, name, visibility, isSpecification, isRoot, isLeaf,
isAbstract),
Association.connection,
AssociationEnd attributes (xmi.id, name, visibility, isSpecification, isNavigable,
ordering, aggregation, targetScope, changeability, type),
AssociationEnd.multiplicity,
Multiplicity,
Multiplicity.range,
MultiplicityRange attributes (xmi.id, lower, upper),
Stereotype attributes (xmi.id, name, visibility, isSpecification, isRoot, isLeaf,
isAbstract, icon, baseClass, extendedElement),
Class attributes (xmi.id, name, visibility, isSpecification, isRoot, isLeaf,
isAbstract, isActive, namespace),
Attribute attributes (xmi.id, visibility, isSpecification, ownerScope, changeability,
targetScope, type, name),
ModelElement.name attributes (name, description),
StructuralFeature,
Classifier.feature,
StructuralFeature.multiplicity,
Attribute.initialValue,
Expression attributes (language, body),
DataType attributes (xmi.id, name, visibility, isSpecification, isRoot, isLeaf,
isAbstract),
Expression.body,
TaggedValue attributes (xmi.id, tag, value, modelElement),
Diagram attributes (xmi.id, name, toolName, diagramType, style, owner),
Diagram.element,
DiagramElement attributes (xmi.id, geometry, style, subject)
]
Relationships [
XMI is associated with XMI.header,
XMI.header is associated with XMI.documentation,
XMI.documentation is associated with XMI.exporter,
XMI.documentation is associated with XMI.exporterVersion,
XMI.header is associated with XMI.metamodel,
XMI is associated with XMI.content,
XMI.content is associated with Model,
Model is associated with Namespace.ownedElement,
Namespace.ownedElement is associated with Association,
Association is associated with Association.connection,
Association.connection is associated with AssociationEnd,
AssociationEnd is associated with AssociationEnd.multiplicity,
AssociationEnd.multiplicity is associated with Multiplicity,
Multiplicity is associated with Multiplicity.range,
Multiplicity.range is associated with MultiplicityRange,
Namespace.ownedElement is associated with Stereotype,
Namespace.ownedElement is associated with Class,
Class is associated with Attribute,
Attribute is associated with ModelElement.name,
Attribute is associated with StructuralFeature,
StructuralFeature is associated with ModelElement.name,
Class is associated with ModelElement.name,
Class is associated with Classifier.feature,
Classifier.feature is associated with Attribute,
Attribute is associated with StructuralFeature.multiplicity,
StructuralFeature.multiplicity is associated with Multiplicity,
Attribute is associated with Attribute.initialValue,
Attribute.initialValue is associated with Expression,
Namespace.ownedElement is associated with DataType,
Expression is associated with Expression.body,
Classifier.feature is associated with StructuralFeature.multiplicity,
XMI.content is associated with TaggedValue,
```

```
XMI.content is associated with Diagram,
Diagram is associated with Diagram.element,
Diagram.element is associated with DiagramElement,
DataType (xmi.id) is Attribute (type),
Class (xmi.id) is AssociationEnd (type),
Association (xmi.id) is Stereotype (extendedElement)
]
}
Target Meta-Model Метамодель продукций {
Elements [
ProductionModel attributes (name, description),
DataType attributes (name, description),
FactTemplate attributes (name, description),
FactTemplateSlot attributes (name, defaultValue, description),
Fact attributes (name, initial, certaintyFactor, description),
FactSlot attributes (name, value, description),
RuleTemplate attributes (name, salience, description),
RuleTemplateCondition attributes (operator),
RuleTemplateAction attributes (function),
Rule attributes (name, certaintyFactor, salience, description),
RuleCondition attributes (operator),
RuleAction attributes (function)
]
Relationships [
ProductionModel is associated with FactTemplate,
ProductionModel is associated with Fact,
ProductionModel is associated with RuleTemplate,
ProductionModel is associated with Rule,
FactTemplate is associated with FactTemplateSlot,
DataType is associated with FactTemplateSlot,
Fact is associated with FactSlot,
DataType is associated with FactSlot,
FactTemplate is associated with Fact,
RuleTemplate is associated with Rule,
RuleTemplate is associated with RuleTemplateCondition,
RuleTemplate is associated with RuleTemplateAction,
FactTemplate is associated with RuleTemplateCondition,
FactTemplate is associated with RuleTemplateAction,
Rule is associated with RuleCondition,
Rule is associated with RuleAction,
Fact is associated with RuleCondition,
Fact is associated with RuleAction
]
}
Transformation Метамодель концептуальных моделей XMI UML to Метамодель продукций {
Rule DataType to DataType priority 1 [
DataType(name) is DataType(name)
]
Rule (Class, ModelElement.name) to Fact priority 2 [
Fact(name) is Class(name) or ModelElement.name(name)
Fact(description) is ModelElement.name(description)
]
Rule (Expression, Attribute, ModelElement.name) to FactSlot priority 3 [
FactSlot(value) is Expression(body)
FactSlot(name) is Attribute(name) or ModelElement.name(name)
FactSlot(description) is ModelElement.name(description)
]
Rule (Association, ModelElement.name) to Rule priority 4 [
Rule(name) is Association(name) or ModelElement.name(name)
Rule(description) is ModelElement.name(description)
]
Rule (AssociationEnd, MultiplicityRange) to RuleCondition priority 5 [
RuleCondition(name) is AssociationEnd
RuleCondition(operator) is "AND" [
if (MultiplicityRange(lower) is "1") and (MultiplicityRange(upper) is "-1")
] or "OR" [
if (MultiplicityRange(lower) is "0") and (MultiplicityRange(upper) is "1")
] or "NOT" [
if (MultiplicityRange(lower) is "0") and (MultiplicityRange(upper) is "0")
]
```

```
]
Rule (AssociationEnd, Stereotype) to RuleAction priority 6 [
RuleAction(name) is AssociationEnd
RuleAction(function) is Stereotype(name)
]
       }
```

Синтезированный код использован в новом компоненте, обеспечивающем трансформацию UML диаграмм классов в модель продукций (преобразование T_{CM-UM}). Для генерации кода для целевой БЗ, например, на CLIPS предлагается использовать встроенный в KBDS генератор.

Время разработки компонента в KBDS – 41 мин., включая следующие действия:

1) Разработку метамодели для исходной концептуальной модели – 10 мин.

2) Выбор целевой метамодели БЗ (создание проекта нового программного компонента и определение его типа) – 4 мин.

3) Разработку программы трансформации – 27 мин., в том числе:

- автоматическое выделение элементов метамоделей – 0 мин.;
- автоматическое построение предварительной модели трансформации (без правил трансформации) – 0 мин.;
- описание соответствий между элементами метамоделей – 14 мин.;
- генерация кода на TMRL – 0 мин.;
- уточнение TMRL кода – 13 мин.

4) Создание программного компонента-конвертора – осуществляется автоматически по результатм предыдущих действий.

Данная задача была решена другим способом – путем прямого программирования специализированного модуля в среде программирования Delphi. Время разработки модуля – 5 ч. 12 мин., включая следующие действия:

- анализ структуры исходного MDL-файла с целью выделения языковых конструкций, соответствующих элементам UML диаграмм классов – 35 мин.;
- программирование – 3 ч. 14 мин.;
- тестирование и отладка – 1 ч. 23 мин.

Отличие времени разработки компонента в KBDS и модуля в Delphi составила: 4 ч. 1 мин. или в 7 раз. При этом необходимо выделить, что KBDS

обеспечивает вовлечение в процесс разработки конверторов конечных пользователей с максимальной автоматизацией отдельных этапов и использования принципов визуального программирования.

Создание компонента для трансформации концепт-карт XTM

Процесс создания компонента анализа концепт-карт XTM [277] в KBDS соответствует процессу, подробно описанными в предыдущем подразделе. Поэтому приведем основные этапы и затраченное время:

- разработка метамодели для исходной концептуальной модели – 4 мин. (Рис. И.9);
- выбор целевой метамодели БЗ (создание проекта нового программного компонента и определение его типа) – 2 мин.;
- описание соответствий между элементами метамоделей – 15 мин. (Рис. И.11);
- генерация кода на TMRL – 0 мин.;
- уточнение TMRL кода – 5 мин.;
- создание программного компонента-конвертора – осуществляется автоматически по результатам предыдущих действий.

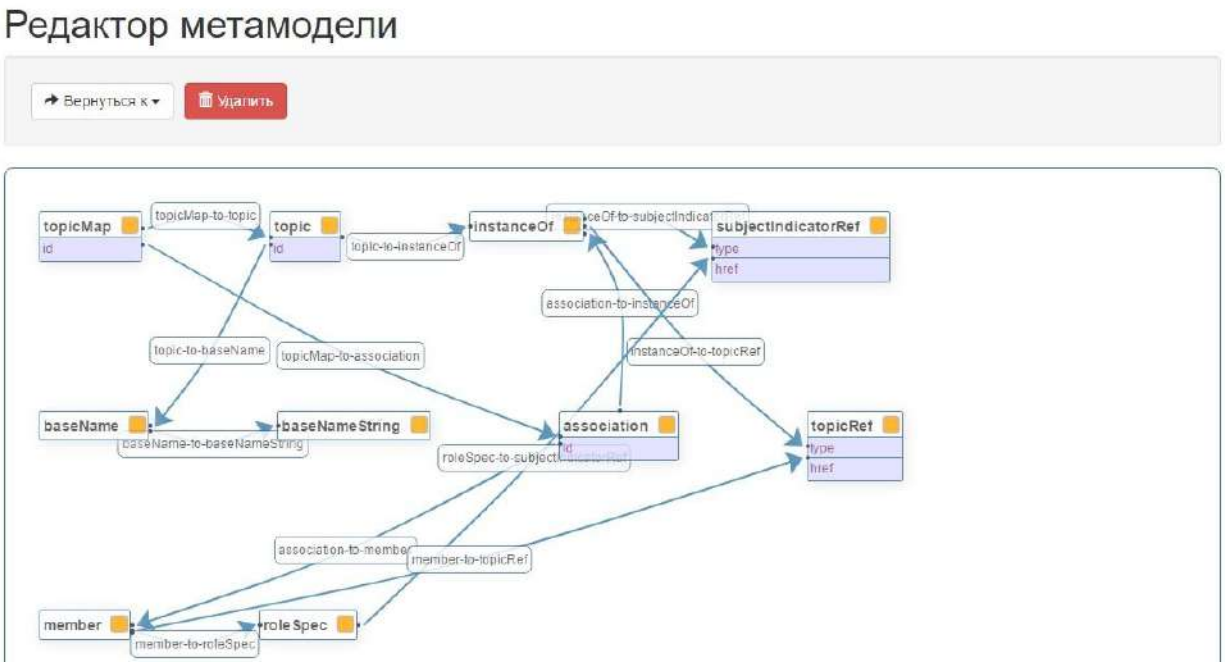


Рис. И.9. Фрагмент формы KBDS: Редактор метамоделей (метамодель концепт-карт XTM)

Время создания данного программного компонента составило 26 мин.

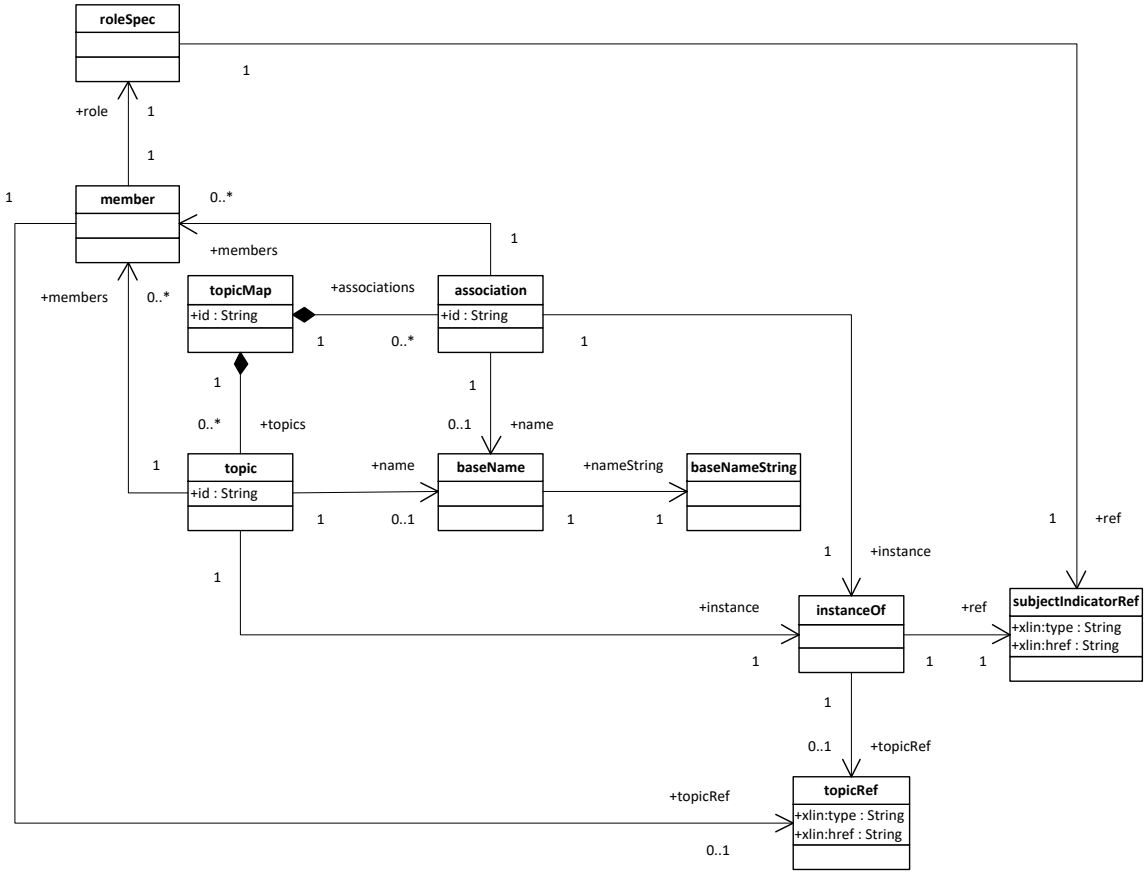


Рис. И.10. Итоговая метамодель исходной концепт-карты

Необходимо подчеркнуть, что так же, как и при создании предыдущего компонента, отдельные действия были автоматизированы. Наиболее сложным оказался этап описания соответствий (12 мин.). При этом на уточнение TMRL-кода потребовало только 5 мин., ввиду отсутствия необходимости описания сложных правил.

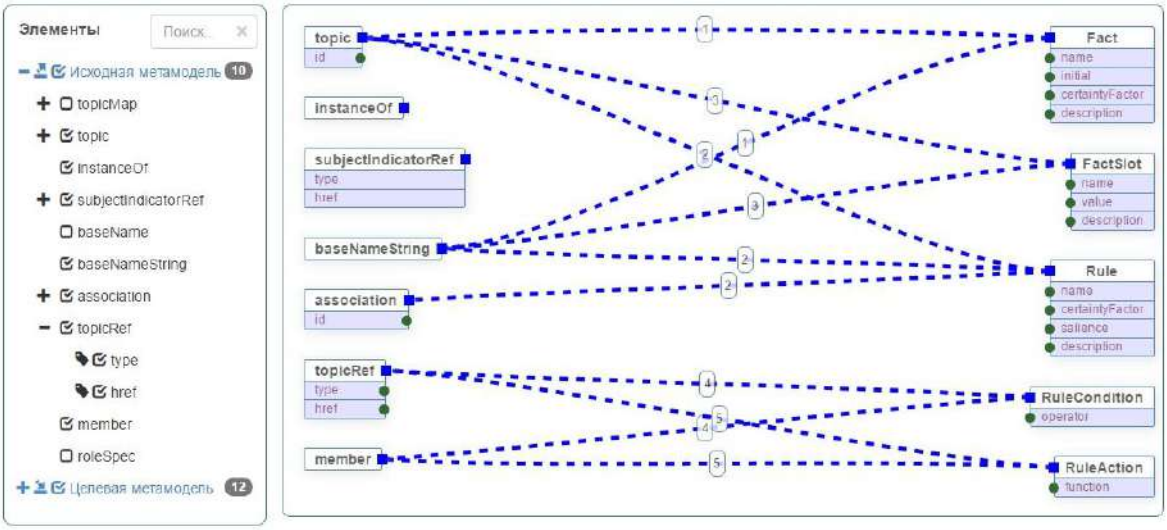


Рис. И.11. Фрагмент формы KBDS: Графический редактор трансформаций (описание соответствий элементов концепт-карт XTM и продукций)

Ниже приведен полученный код TMRL программы:

```
Source Meta-Model Метамодель концепт-карт CmapTools (XTM) {
Elements [
topicMap attributes (id),
topic attributes (id),
instanceOf,
subjectIndicatorRef attributes (type, href),
baseName,
baseNameString,
association attributes (id),
topicRef attributes (type, href),
member,
roleSpec
]
Relationships [
topicMap is associated with topic,
topic is associated with instanceOf,
instanceOf is associated with subjectIndicatorRef,
topic is associated with baseName,
baseName is associated with baseNameString,
topicMap is associated with association,
association is associated with instanceOf,
instanceOf is associated with topicRef,
association is associated with member,
member is associated with roleSpec,
roleSpec is associated with subjectIndicatorRef,
member is associated with topicRef
]
}
Target Meta-Model Метамодель продукций {
Elements [
ProductionModel attributes (name, description),
DataType attributes (name, description),
FactTemplate attributes (name, description),
FactTemplateSlot attributes (name, defaultValue, description),
Fact attributes (name, initial, certaintyFactor, description),
FactSlot attributes (name, value, description),
RuleTemplate attributes (name, salience, description),
RuleTemplateCondition attributes (operator),
RuleTemplateAction attributes (function),
Rule attributes (name, certaintyFactor, salience, description),
RuleCondition attributes (operator),
RuleAction attributes (function)
]
Relationships [
ProductionModel is associated with FactTemplate,
ProductionModel is associated with Fact,
ProductionModel is associated with RuleTemplate,
ProductionModel is associated with Rule,
FactTemplate is associated with FactTemplateSlot,
DataType is associated with FactTemplateSlot,
Fact is associated with FactSlot,
DataType is associated with FactSlot,
FactTemplate is associated with Fact,
RuleTemplate is associated with Rule,
RuleTemplate is associated with RuleTemplateCondition,
RuleTemplate is associated with RuleTemplateAction,
FactTemplate is associated with RuleTemplateCondition,
FactTemplate is associated with RuleTemplateAction,
Rule is associated with RuleCondition,
Rule is associated with RuleAction,
Fact is associated with RuleCondition,
Fact is associated with RuleAction
]
}
Transformation Метамодель концепт-карт CmapTools (XTM) to Метамодель продукций {
```

```
Rule (topic, baseNameString) to Fact priority 1 [
Fact(name) is topic or baseNameString
]
Rule (topic, baseNameString) to Rule priority 2 [
Rule(name) is topic or baseNameString
]
Rule (topic, baseNameString) to FactSlot priority 3 [
FactSlot(name) is topic or baseNameString
]
Rule (member, topicRef) to RuleCondition priority 4 [
RuleCondition(name) is member or topicRef
RuleCondition(operator) is "AND"
]
Rule (member, topicRef) to RuleAction priority 5 [
RuleAction(name) is member or topicRef
RuleCondition(function) is "assert"

]
}
```

Созданный компонент реализует преобразование концепт-карт (сериализованных в формате XTM) в модель продукций.

Данная задача также была решена путем прямого программирования специализированного модуля модуля для анализа концепт-карт из файлов IHMC CmapTools (CXL-формат)в среде программирования Delphi. Время разработки модуля – 3 ч. 30 мин., включая следующие действия:

- анализ структуры исходного CXL-файла, с целью выделения языковых конструкций, соответствующих элементам концепт-карт CmapTools – 30 мин.;
- программирование – 2 часа;
- тестирование и отладка – 1 час.

Отличие времени разработки компонента в KBDS и модуля в Delphi составила: 3 ч. 4 мин. или в 8 раз.

Создание компонента для трансформации деревьев событий

Процесс создания компонента анализа деревьев событий (ДС) [274] в KBDS соответствует процессу, подробно описанными в предыдущем подразделе. Поэтому приведем полученный код TMRL программы:

```
Source Meta-Model Метамодель ДС {
       Elements [
              EventTree attributes (caption, Graph_type_node, Func_type_node,
can_deleted, can_move, SGmode, SGCaption, Name),
              CauseEffectRelation attributes (Graph_type_relation, id, Name),
              Cause attributes (id, event, Name),
              Effect attributes (id, event, Name),
              InitialEvent attributes (caption, Func_type_node, id, can_deleted,
can_move, Graph_type_node, Name),
              Event attributes (Func_type_node, Graph_type_node, can_move, can_deleted,
id, caption, tag-ev, Name),
              Parameter attributes (id, caption, value)
       ]
       Relationships [
```

```
            EventTree is associated with CauseEffectRelation,
            CauseEffectRelation is associated with Cause,
            CauseEffectRelation is associated with Effect,
            EventTree is associated with InitialEvent,
            EventTree is associated with Event,
            Event is associated with Parameter,
            Event (id) is associated with Cause (event),
            Event (id) is associated with Effect (event),
            InitialEvent (id) is associated with Cause (event),
            InitialEvent (id) is associated with Effect (event)
      ]
}
Target Meta-Model Метамодель продукций {
      Elements [
            ProductionModel attributes (name, description),
            DataType attributes (name, description),
            FactTemplate attributes (name, description),
            FactTemplateSlot attributes (name, defaultValue, description),
            Fact attributes (name, initial, certaintyFactor, description),
            FactSlot attributes (name, value, description),
            RuleTemplate attributes (name, salience, description),
            RuleTemplateCondition attributes (operator),
            RuleTemplateAction attributes (function),
            Rule attributes (name, certaintyFactor, salience, description),
            RuleCondition attributes (operator),
            RuleAction attributes (function)
      ]
      Relationships [
            ProductionModel is associated with FactTemplate,
            ProductionModel is associated with Fact,
            ProductionModel is associated with RuleTemplate,
            ProductionModel is associated with Rule,
            FactTemplate is associated with FactTemplateSlot,
            DataType is associated with FactTemplateSlot,
            Fact is associated with FactSlot,
            DataType is associated with FactSlot,
            FactTemplate is associated with Fact,
            RuleTemplate is associated with Rule,
            RuleTemplate is associated with RuleTemplateCondition,
            RuleTemplate is associated with RuleTemplateAction,
            FactTemplate is associated with RuleTemplateCondition,
            FactTemplate is associated with RuleTemplateAction,
            Rule is associated with RuleCondition,
            Rule is associated with RuleAction,
            Fact is associated with RuleCondition,
            Fact is associated with RuleAction
      ]
}
Transformation Метамодель ДС to Метамодель продукций {
      Rule InitialEvent to Fact priority 1 [
            Fact(name) is InitialEvent(caption)
      ]
      Rule (Parameter, Event) to Fact priority 2 [
            Fact(name) is Event(caption)
            Fact(certaintyFactor) is Parameter(value) [
                  if (Parameter(caption) is "cf")
            ]
      ]
      Rule Parameter to FactSlot priority 3 [
            FactSlot(name) is Parameter(caption)
            FactSlot(value) is Parameter(value)
      ]
      Rule CauseEffectRelation to Rule priority 4 [
            Rule(name) is CauseEffectRelation
      ]
      Rule Cause to RuleCondition priority 5 [
            RuleCondition(name) is Cause
      ]
      Rule Effect to RuleAction priority 6 [
            RuleAction(name) is Effect
```

```
            RuleCondition(function) is "assert"
    ]
    }
```

Основные этапы процесса разработки и затраченное время::

- разработка метамодели для исходной концептуальной модели – 5 мин. (Рис. И.12);
- выбор целевой метамодели БЗ (создание проекта нового программного компонента и определение его типа) – 2 мин.;
- описание соответствий между элементами метамоделей – 12 мин. (Рис. И.13);
- генерация кода на TMRL – 0 мин.;
- уточнение TMRL кода – 10 мин.;
- создание программного компонента-конвертора – осуществляется автоматически по результатам предыдущих действий.

Время создания данного программного компонента составило 29 мин.

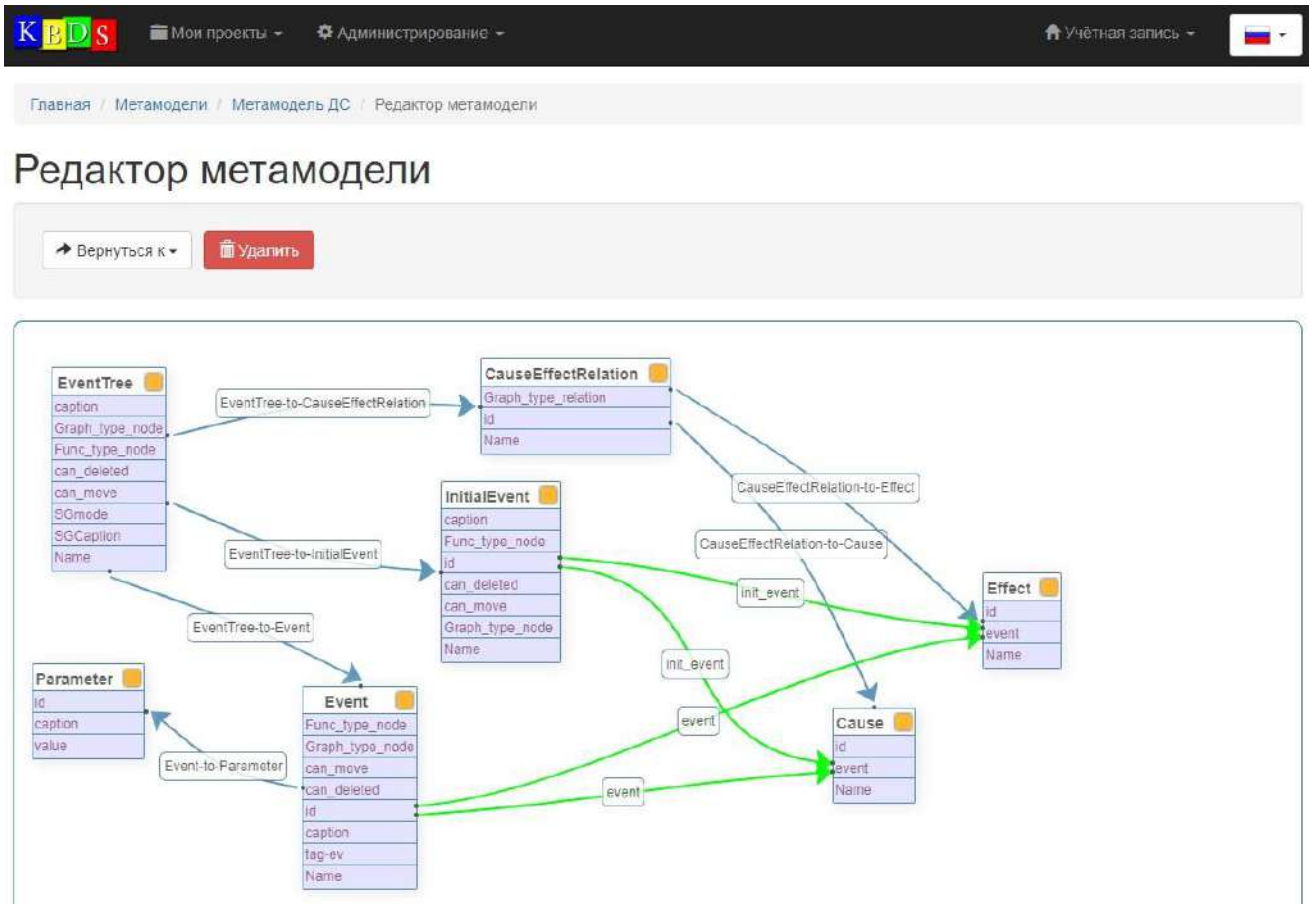


Рис. И.12. Фрагмент формы KBDS: Редактор метамоделей (метамодель ДС)

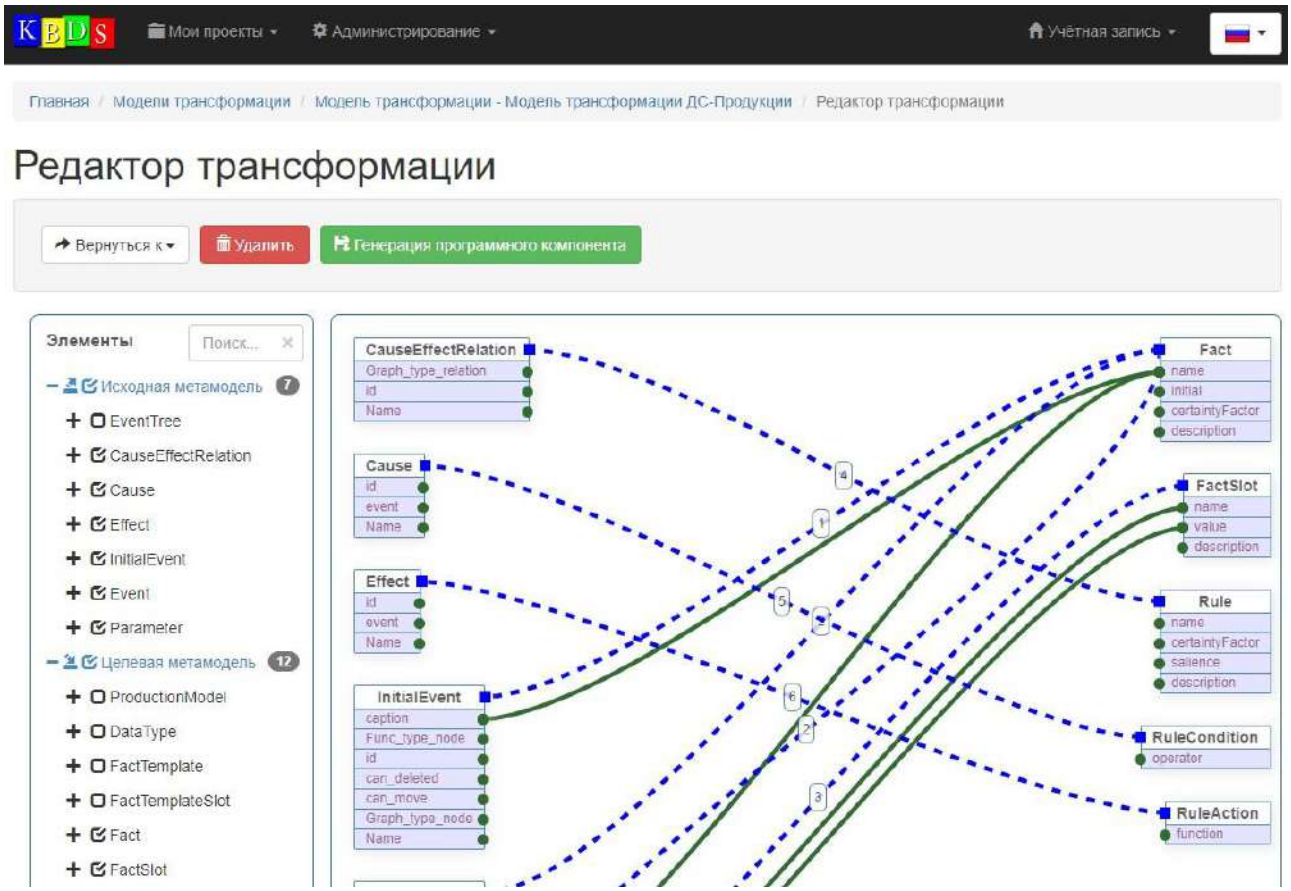


Рис. И.13. Фрагмент формы KBDS: Редактора трансформаций (описание соответствий элементов ДС и продукций)

Созданный компонент обеспечивает преобразование деревьев событий в модель продукций.