ЗАДАНИЕ НА ДИПЛОМНОЕ ПРОЕКТИРОВАНИЕ

РЕФЕРАТ

Квалификационная работа на степень бакалавра на тему «Использование лингвистических переменных в РДО-моделях» содержит 57 страниц машинописного текста формата А4, 28 илюстраций. При выполнении данной работы было использовано 10 различных источников.

Ключевые слова:

Лингвистические переменные, нечеткая логика, нечеткое управление, имитационная модель, язык РДО.

Текст реферата:

В данной дипломной работе приводится вариант использования аппарата нечеткой логики в среде имитационного моделирования РДО. Показана возможность применения лингвистических переменных в качестве параметров ресурсов и в продукционных правилах моделей языка РДО при имитационном моделировании. Целью работы является осуществление концепции использования нечеткой логики в интеллектуальной среде имитационного моделирования РДО. Эффективность применения нечеткой логики в сложных системах очень высока, а область применения таких систем абсолютно разная – в автомобильной, аэрокосмической и транспортной промышленности, в области изделий бытовой техники, в сфере финансов, анализа и принятия управленческих решений и многих других. Результаты подтверждают обоснованность применения нечеткой логики в имитационных моделях при решении задач управления, значительно упрощая и сокращая центральный блок принятия решений, а при решении сложных и громоздких задач вычисления точных воздействий подменяется простой и гибкой стратегией.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, СИМВОЛОВ И СПЕЦИАЛЬНЫХ ТЕРМИНОВ С ИХ ОПРЕДЕЛЕНИЕМ

ИМ - Имитационная модель.

ЛП - Лингвистическая переменная.

СДС - Сложная дискретная система.

ОТК - отдел технического контроля

ВВЕДЕНИЕ

Математическая теория нечетких множеств (fuzzy sets) и нечеткая логика (fuzzy logic) являются обобщениями классической теории множеств и классической формальной логики. Данные понятия были впервые предложены американским ученым Лотфи Заде (Lotfi Zadeh) в 1965 г. Основной причиной появления новой теории стало наличие нечетких и приближенных рассуждений при описании человеком процессов, систем, объектов.

Прежде чем нечеткий подход к моделированию сложных систем получил признание во всем мире, прошло не одно десятилетие с момента зарождения теории нечетких множеств. И на этом пути развития нечетких систем принято выделять три периода.

Первый период (конец 60-х–начало 70 гг.) характеризуется развитием теоретического аппарата нечетких множеств (Л. Заде, Э. Мамдани, Беллман). Во втором периоде (70–80-е годы) появляются первые практические результаты в области нечеткого управления сложными техническими системами (парогенератор с нечетким управлением). Одновременно стало уделяться внимание вопросам построения экспертных систем, построенных на нечеткой логике, разработке нечетких контроллеров. Нечеткие экспертные системы для поддержки принятия решений находят широкое применение в медицине и экономике. Наконец, в третьем периоде, который длится с конца 80-х годов и продолжается в настоящее время, появляются пакеты программ для построения нечетких экспертных систем, а области применения нечеткой логики заметно расширяются. Она применяется в автомобильной, аэрокосмической и транспортной промышленности, в области изделий бытовой техники, в сфере финансов, анализа и принятия управленческих решений и многих других.

Триумфальное шествие нечеткой логики по миру началось после доказательства в конце 80-х Бартоломеем Коско знаменитой теоремы FAT (Fuzzy Approximation Theorem). В бизнесе и финансах нечеткая логика получила признание после того как в 1988 году экспертная система на основе нечетких правил для прогнозирования финансовых индикаторов единственная предсказала биржевой крах. И количество успешных фаззи-применений в настоящее время исчисляется тысячами.

Гибридизация методов интеллектуальной обработки информации – девиз, под которым прошли 90-е годы у западных и американских исследователей. В результате объединения нескольких технологий искусственного интеллекта появился специальный термин – "мягкие вычисления" (soft computing), который ввел Л. Заде в 1994 году. В настоящее время мягкие вычисления объединяют такие области как: нечеткая логика, искусственные нейронные сети, вероятностные рассуждения и эволюционные алгоритмы. Они дополняют друг друга и используются в различных комбинациях для создания гибридных интеллектуальных систем.

Влияние нечеткой логики оказалось, пожалуй, самым обширным. Подобно тому, как нечеткие множества расширили рамки классической математическую теорию множеств, нечеткая логика "вторглась" практически в большинство методов Data Mining, наделив их новой функциональностью.

В данной работе представлена подсистема, связывающая аппарат нечеткой логики и систему имитационного моделирования РДО. Подсистема, интегрированная в систему РДО, осуществляет работу с нечеткими переменных в качестве параметров ресурсов модели, написанной на языке РДО. В процессе проектирования данной подсистемы были разработаны синтаксические правила описания нечетких параметров ресурсов модели, написанной на языке РДО, а также алгоритмы доступа к значениям нечетких параметров ресурса и работы нечеткого продукционного правила в модели РДО.

1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

На концептуальном уровне СДС можно представить как множество некоторых ресурсов, взаимодействующих между собой. И каждый ресурс описывается множеством его параметров, и в модели СДС ресурс имеет уникальное имя, отличающее его от других ресурсов. В общем случае значениями различных параметров могут быть как значения математических функций, так и слова естественного языка. Если для значений параметра ресурса используются нечеткие лингвистические значения, то СДС относится к нечетким системам, а множество значений параметра ресурса описывается нечетким множеством, к которым применяется аппарат нечеткой логики.

1.1. Нечеткие множества и лингвистические переменные

При описании объектов и явлений с помощью нечетких множеств используется понятие нечеткой и лингвистической переменных.

Нечеткая переменная характеризуется тройкой <a, X, A>, где

- a - имя переменной,
- X - универсальное множество (область определения a),
- A - нечеткое множество на X, описывающее ограничение (то есть m A(x)) на значение нечеткой переменной a.

Лингвистической переменной называется набор <b ,T,X,G,M>, где

- b - имя лингвистической переменной;
- T - множество его значений (терм-множество), представляющие имена нечетких переменных, областью определения, которых является множество X. Множество T называется базовым терм-множеством лингвистической переменной;
- G - синтаксическая процедура, позволяющая оперировать элементами терм-множества T, в частности, генерировать новые термы (значения). Множество ТИG(T), где G(T) - множество сгенерированных термов, называется расширенным терм-множеством лингвистической переменной;

- M - семантическая процедура, позволяющая преобразовать новое значение лингвистической переменной, образованной процедурой G, в нечеткую переменную, то есть сформировать соответствующее нечеткое множество.

Во избежание большого количества символов:

- символ b используют как для названия самой переменной, так и для всех его значений;
- для обозначения нечеткого множества и его названия пользуются одним символом, например, терм "молодой", является значением лингвистической переменной b = "возраст", и одновременно нечетким множеством M ("молодой").

Присваивание нескольких значений символам предполагает, что контекст допускает неопределенности.

Для каждого терма из терм-множества T задается функция принадлежности, определенная на всем интервале X. Функция принадлежности может задаваться графически, либо аналитически.

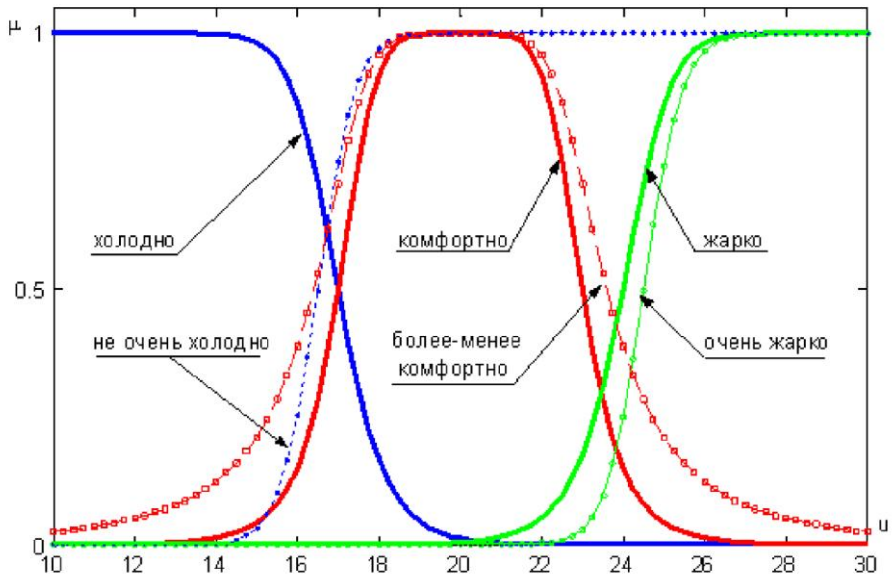


Рис. 1. Определение функций принадлежности для лингвистических переменных.

Существует свыше десятка типовых форм кривых для задания функций принадлежности. Наибольшее распространение получили: треугольная, трапецеидальная и гауссова функции принадлежности.

Треугольная функция принадлежности определяется тройкой чисел (a,b,c), и ее значение в точке x вычисляется согласно выражению:

$$MF(x)=\begin{cases}1-\dfrac{b-x}{b-a}, & a\le x\le b\\ 1-\dfrac{x-c}{c-b}, & b\le x\le c\\ 0, \text{ в остальных случаях.}\end{cases}$$

При (b-a)=(c-b) имеем случай симметричной треугольной функции принадлежности, которая может быть однозначно задана двумя параметрами из тройки (a,b,c).

Аналогично для задания трапецеидальной функции принадлежности необходима четверка чисел (a,b,c,d):

$$MF(x)=\begin{cases}1-\dfrac{b-x}{b-a}, & a\le x\le b\\ 1, & b\le x\le c\\ 1-\dfrac{x-c}{d-c}, & c\le x\le d\\ 0, \text{ в остальных случаях.}\end{cases}$$

При (b-a)=(d-c) трапецеидальная функция принадлежности принимает симметричный вид.

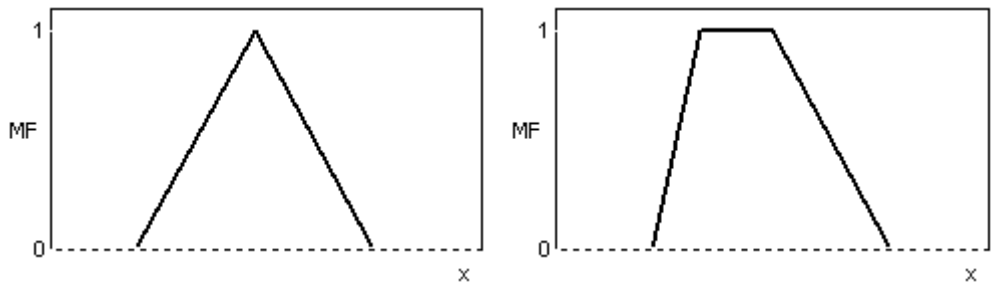


Рис. 2. Типовые кусочно-линейные функции принадлежности.

Функция принадлежности гауссова типа описывается формулой

$$MF(x)=\exp\left[-\left(\frac{x-c}{\sigma}\right)^2\right]$$

и оперирует двумя параметрами. Параметр *c* обозначает центр нечеткого множества, а параметр отвечает за крутизну функции.

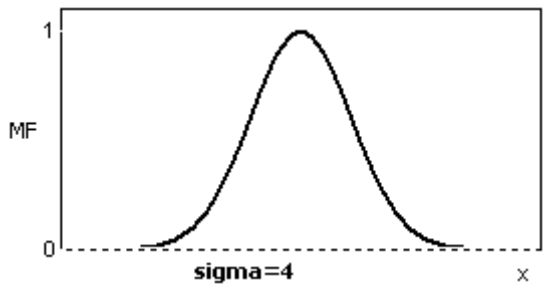


Рис. 3. Гауссова функция принадлежности.

Совокупность функций принадлежности для каждого терма из базового терм-множества T обычно изображаются вместе на одном графике.

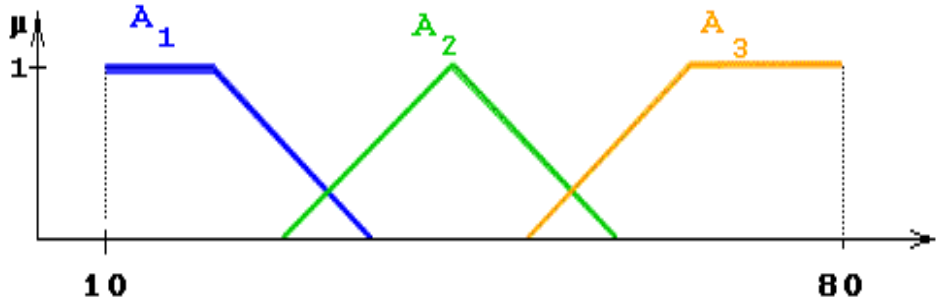


Рис. 4. "маленькая толщина" = A1 , "средняя толщина"= A2, " большая толщина"= A3.

1.2. Операции с нечеткими множествами. Нечеткая логика.

Базовыми операциями над нечеткими множествами являются *пересечение*, *объединение* и *отрицание* нечетких множеств. Кроме того, для объединения двух нечетких множеств имеется оператор максимума, а для пересечения двух нечетких множеств оператор минимума. Эти операторы совпадают с обычными (четкими) объединением и пересечением, только рассматриваются степени принадлежности 0 и 1.

Как пример: Пусть *A* нечеткий интервал от 5 до 8 и *B* нечеткое число *около* 4, как показано на рисунке.

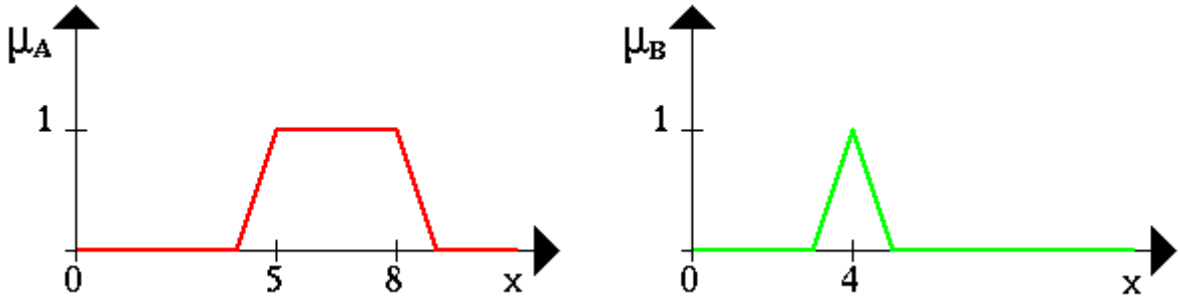


Рис. 5. Функции принадлежности для двух термов

Следующий пример иллюстрирует нечеткое множество *между* 5 *и* 8 **И (AND)** *около* 4 (синяя линия).

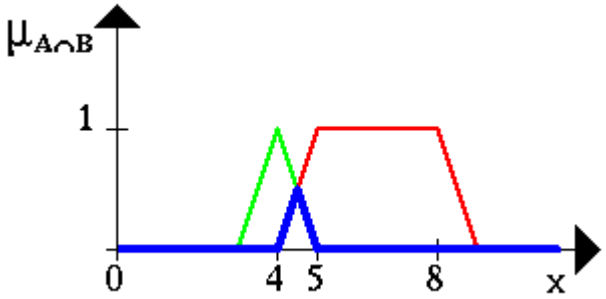


Рис. 6. Пересечение двух нечетких множеств.

Нечеткое множество *между* 5 *и* 8 **ИЛИ (OR)** *около* 4 показано на следующем рисунке (снова синяя линия).

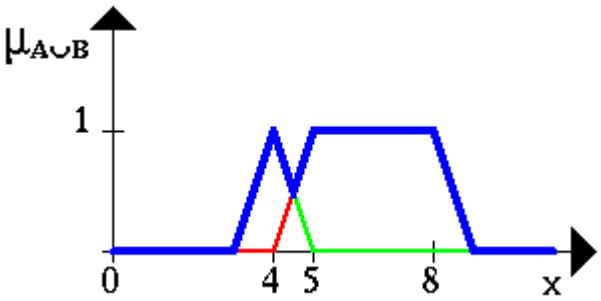


Рис. 7. Объединение двух нечетких множеств

Следующий рисунок иллюстрирует операцию отрицания. Синяя линия - это **ОТРИЦАНИЕ** нечеткого множества *A*.

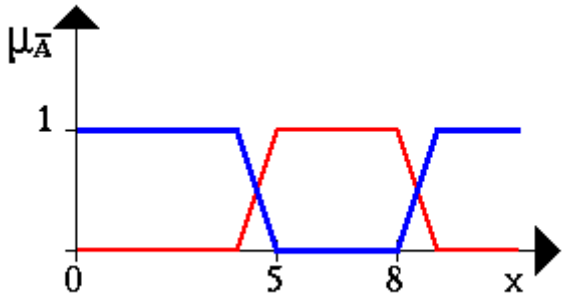


Рис. 8. Отрицание нечеткого множества.

1.3. Фазификация и дефазификация нечетких переменных

Основой для проведения операции нечеткого логического вывода является база правил, содержащая нечеткие высказывания в форме "Если-то" и функции принадлежности для соответствующих лингвистических термов. При этом должны соблюдаться следующие условия:

Существует хотя бы одно правило для каждого лингвистического терма выходной переменной.

Для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки (левая часть правила).

В противном случае имеет место неполная база нечетких правил.

В общем случае механизм логического вывода включает четыре этапа: введение нечеткости (фазификация), нечеткий вывод, композиция и приведение к четкости, или дефазификация (см. рисунок 9).

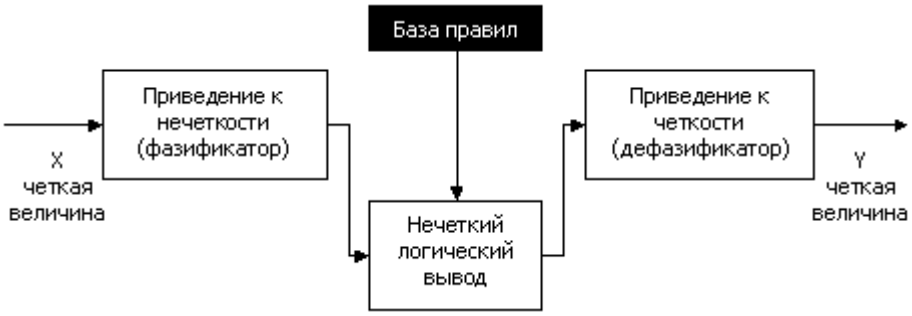


Рис. 9. Система нечеткого логического вывода.

Алгоритмы нечеткого вывода различаются главным образом видом используемых правил, логических операций и разновидностью метода дефазификации. Разработаны модели нечеткого вывода Мамдани, Сугено, Ларсена, Цукамото.

Рассмотрим подробнее нечеткий вывод на примере механизма Мамдани (Mamdani). Это наиболее распространенный способ логического вывода в нечетких системах. В нем используется минимаксная композиция нечетких множеств. Данный механизм включает в себя следующую последовательность действий.

Процедура фазификации: определяются степени истинности, т.е. значения функций принадлежности для левых частей каждого правила (предпосылок). Для базы правил с m правилами обозначим степени истинности как $A_{ik}(x_k)$, i=1..m, k=1..n.

Нечеткий вывод. Сначала определяются уровни "отсечения" для левой части каждого из правил:

$$alfa_i = \min_i (A_{ik}(x_k))$$

Далее находятся "усеченные" функции принадлежности:

$$B_i^*(y) = \min_i (alfa_i , B_i(y))$$

Композиция, или объединение полученных усеченных функций, для чего используется максимальная композиция нечетких множеств:

$$MF(y) = \max_i (B_i^*(y))$$

где MF(y) – функция принадлежности итогового нечеткого множества.

Дефазификация, или приведение к четкости. Существует несколько методов дефазификации. Например, метод среднего центра, или центроидный метод:

$$MF(y) = \max_i (B_i^*(y)).$$

Геометрический смысл такого значения – центр тяжести для кривой MF(y). Рисунок 6 графически показывает процесс нечеткого вывода по Мамдани для двух входных переменных и двух нечетких правил R1 и R2.

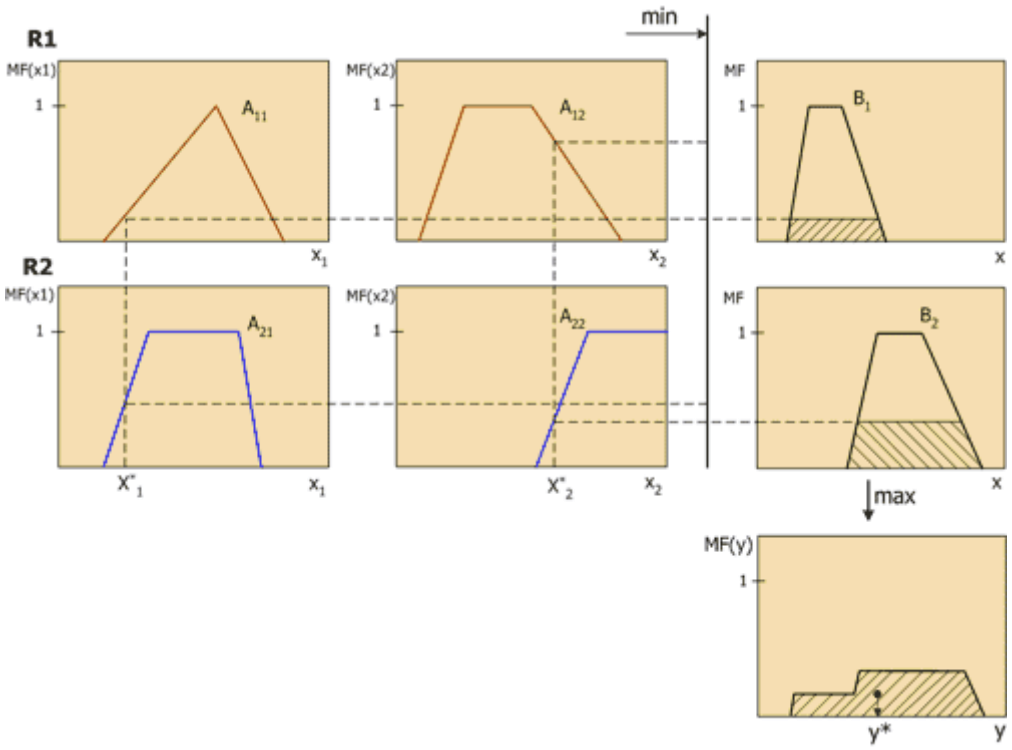


Рис. 10. Схема нечеткого вывода по Мамдани.

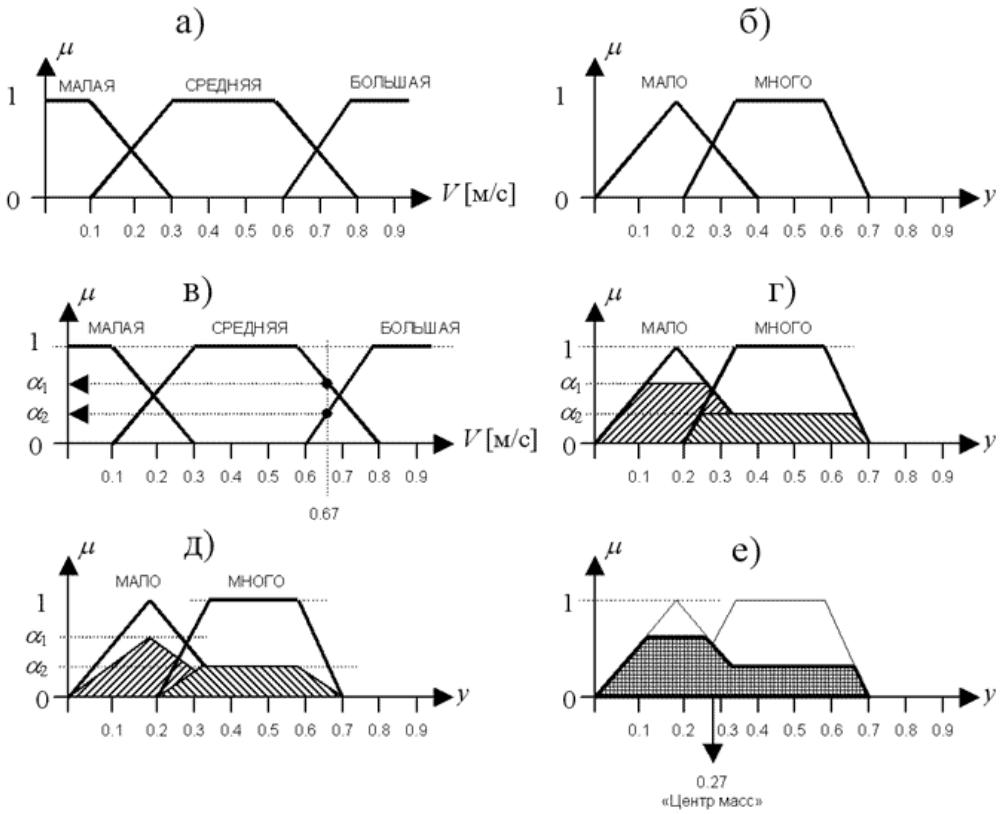


Рис. 11. Принцип нечеткого логического вывода: а) Функции принадлежности по входной переменной; б) Функции принадлежности по выходной переменной; в) Процесс фаззификации; г) Процесс формирования выходных множеств по методу MAX-MIN; д) Процесс формирования выходных множеств по методу MAX-DOT; е) Процесс дефаззификации

1.4. Применение нечетких систем и нечеткого управления

Наиболее важное приложение теории нечетких множеств - контроллеры нечеткой логики. Их функционирование немного отличается от работы обычных контроллеров; для описания системы используются знания экспертов вместо дифференциальных уравнений. Эти знания могут быть выражены естественным образом с помощью *лингвистических переменных*, которые описываются нечеткими множествами.

Общая структура микроконтроллера, использующего нечеткую логику, показана на рис.1. Она содержит:

- блок фазификации;
- базу знаний;
- блок решений;
- блок дефазификации.

Блок фазификации преобразует четкие величины, измеренные на выходе объекта управления, в нечеткие величины, которые описаны лингвистическими переменными в базе знаний.

Блок решений использует нечеткие продукционные (ЕСЛИ - ТО) правила, заложенные в базу знаний, для преобразования нечетких входных данных в необходимые управляющие влияния, которые также носят нечеткий характер.

Блок дефазификации превращает нечеткие данные с выхода блока решений в четкую величину, которая используется для управления объектом.

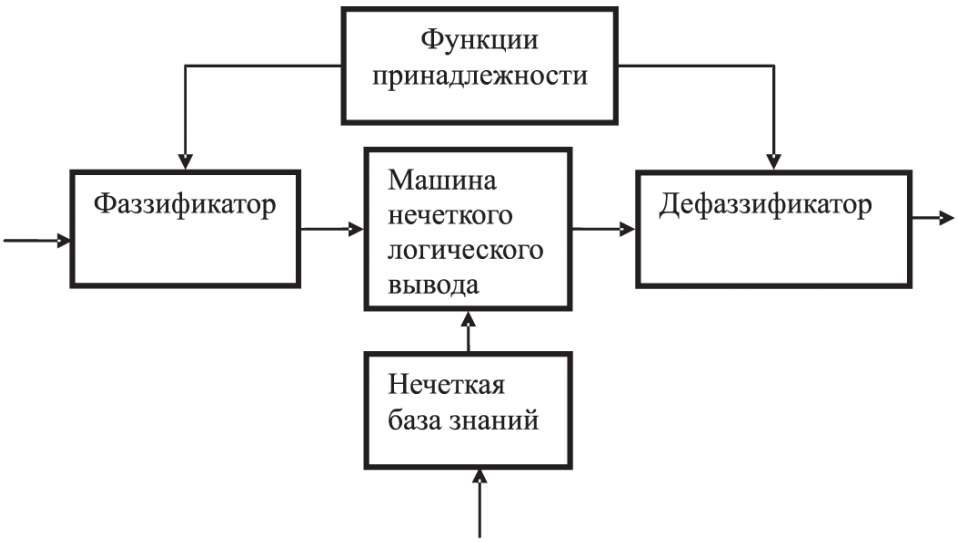


Рис. 12. Общая структура нечеткого микроконтроллера

В качестве примера известных микроконтроллеров, использующих нечеткую логику можно назвать 68HC11, 68HC12 фирмы Motorola, MCS-96 фирмы Intel, а также некоторые другие.

Все системы с нечеткой логикой функционируют по одному принципу: показания измерительных приборов: фазифицируются (превращаются в нечеткий формат), обрабатываются, дефазифицируются и в виде обычных сигналов подаются на исполнительные устройства.

Использование нечеткого управления рекомендуется...

- для очень сложных процессов, когда не существует простой математической модели, описывающей систему.
- для нелинейных процессов высоких порядков.
- если должна производиться обработка (лингвистически сформулированных) экспертных знаний

Использование нечеткого управления не рекомендуется, если...

- приемлемый результат может быть получен с помощью общей теории управления
- уже существует формализованная и адекватная математическая модель
- проблема не разрешима

Что касается отечественного рынка коммерческих систем на основе нечеткой логики, то его формирование началось в середине 1995 года. Популярными являются следующие пакеты:

CubiCalc 2.0 RTC - одна из мощных коммерческих экспертных систем на основе нечеткой логики, позволяющая создавать собственные прикладные экспертные системы ;

CubiQuick - дешевая "университетская" версия пакета CubiCalc ;

RuleMaker - программа автоматического извлечения нечетких правил из входных данных ;

FuziCalc - электронная таблица с нечеткими полями, позволяющая делать быстрые оценки при неточных данных без накопления погрешности;

OWL - пакет, содержащий исходные тексты всех известных видов нейронных сетей, нечеткой ассоциативной памяти и т.д.

Основными потребителями нечеткой логики на рынке СНГ являются банкиры и финансисты, а также специалисты в области политического и экономического анализа. Они используют CubiCalc для создания моделей разных экономических, политических, биржевых ситуаций. Что же касается пакета FuziCalc, то он занял свое место на компьютерах больших банкиров и специалистов по чрезвычайным ситуациям - то есть тех, для кого важна скорость проведения расчетов в условиях неполноты и неточности входной информации. Однако можно с уверенностью сказать, что эпоха расцвета прикладного использования нечеткой логики на отечественном рынке еще впереди.

Сегодня элементы нечеткой логики можно найти в десятках промышленных изделий - от систем управления электропоездами и боевыми вертолетами до пылесосов и стиральных машин. Без применения нечеткой логики немыслимы современные ситуационные центры руководителей западных стран, где принимаются ключевые политические решения и моделируются разные кризисные ситуации. Одним из впечатляющих примеров масштабного применения нечеткой логики стало комплексное моделирование системы здравоохранения и социального обеспечения Великобритании (National Health Service - NHS), которое впервые позволило точно оценить и оптимизировать затраты на социальные нужды.

Не обошли средства нечеткой логики и программные системы, обслуживающих большой бизнес. Первыми, разумеется, были финансисты, задачи которых требуют ежедневного принятия правильных решений в сложных условиях непредвиденного рынка. Первый год использования системы Fuji Bank принес банку в среднем $770000 на месяц (и это только официально объявленная прибыль!).

Вслед за финансистами, обеспокоенные успехами японцев и потерей стратегической инициативы, когнитивными нечеткими схемами заинтересовались промышленные гиганты США. Motorola, General Electric, Otis Elevator, Pacific Gas & Electric, Ford и другие в начале 90-х начали инвестировать в разработку изделий, использующих нечеткую логику. Имея солидную финансовую "поддержку", фирмы, специализирующиеся на нечеткой логике, получили возможность

адаптировать свои разработки для широкого круга применений. "Оружие элиты" вышло на массовый рынок.

Среди лидеров нового рынка выделяется американская компания Hyper Logic, основанная в 1987 году Фредом Уоткинсом (Fred Watkins). Сначала компания специализировалась на нейронных сетях, однако в скором времени целиком сконцентрировалась на нечеткой логике. Недавно вышла на рынок вторая версия пакета CubiCalc фирмы HyperLogic, которая является одной из мощнейших экспертных систем на основе нечеткой логики. Пакет содержит интерактивную оболочку для разработки нечетких экспертных систем и систем управления, а также run-time модуль, позволяющий оформлять созданные пользователем системы в виде отдельных программ.

Кроме Hyper Logic среди "патриархов" нечеткой логики можно назвать фирмы IntelligenceWare, InfraLogic, Aptronix. Всего же на мировом рынке представлено более 100 пакетов, которые так или иначе используют нечеткую логику. В трех десятках СУБД реализована функция нечеткого поиска. Собственные программы на основе нечеткой логики анонсировали такие гиганты как IBM, Oracle и другие.

На принципах нечеткой логики создан и один из российских программных продуктов - известный пакет "Бизнес-прогноз". Назначение этого пакета - оценка рисков и потенциальной прибыльности разных бизнес-планов, инвестиционных проектов и просто идей относительно развития бизнеса. "Ведя" пользователя по сценарию его замысла, программа задает ряд вопросов, которые допускают как точные количественные ответы, так и приближенные качественные оценки - типа "маловероятно", "степень риска высокая" и т.д. Обобщив всю полученную информацию в виде одной схемы бизнес-проекта, программа не только выносит окончательный вердикт о рискованности проекта и ожидаемых прибылей, но и указывает критические точки и слабые места в авторском замысле. От аналогичных иностранных пакетов "Бизнес-прогноз" отличается простотой, дешевизной и, разумеется, русскоязычным интерфейсом. Впрочем, программа "Бизнес-прогноз" - лишь первая ласточка, за которой неминуемо появятся новые разработки ученых СНГ.

Ниже приведены несколько примеров того, как реально применяется нечеткая логика:

- Упрощенное управление роботами
  (Hirota, Fuji Electric, Toshiba, Omron)
- Наведение телекамер при трансляции спортивных событий
  (Omron)
- Предотвращение нежелательных температурных флуктуаций в системах кондиционирования воздуха
  (Mitsubishi, Sharp)
- Эффективное и стабильное управление автомобильными двигателями
  (Nissan)
- Управление экономичной скоростью автомобилей
  (Nissan, Subaru)
- Улучшение эффективности и оптимизация промышленных систем управления
  (Aptronix, Omron, Meiden, Sha, Micom, Mitsubishi, Nisshin-Denki, Oku-Electronics)
- Позиционирование приводов в производстве полупроводников wafer-steppers
  (Canon)
- Системы прогнозирования землетрясений
  (Inst. of Seismology Bureau of Metrology, Japan)
- Распознавание движения изображения в видеокамерах
  (Canon, Minolta)
- Автоматическое управление двигателем пылесосов с автоматическим определением типа поверхности и степени засоренности
  (Matsushita)
- Управление освещенностью в камкодерах
  (Sanyo)
- Компенсация вибраций в камкодерах
  (Matsushita)
- Однокнопочное управление стиральными машинами
  (Matsushita, Hitatchi)
- Распознавание рукописных текстов, объектов, голоса
  (CSK, Hitachi, Hosai Univ., Ricoh)

- Вспомагательные средства полета вертолетов
  (Sugeno)
- САПР производственных процессов
  (Aptronix, Harima, Ishikawajima-OC Engeneering)
- Управление скоростью линий и температурой при производстве стали
  (Kawasaki Steel, New-Nippon Steel, NKK)
- Управление метрополитенами для повышения удобства вождения, точности остановки и экономии энергии
  (Hitachi)
- Оптимизация потребления бензина в автомобилях
  (NOK, Nippon Denki Tools)
- Повышение чувствительности и эффективности управления лифтами
  (Fujitec, Hitachi, Toshiba)
- Повышение безопасности ядерных реакторов
  (Hitachi, Bernard, Nuclear Fuel div.)

1.5. Основы РДО-метода

Имитационное моделирование (ИМ) на ЭВМ находит широкое применение при исследовании и управлении сложными дискретными системами (СДС) и процессами, в них протекающими. К таким системам можно отнести экономические и производственные объекты, морские порты, аэропорты, комплексы перекачки нефти и газа, ирригационные системы, программное обеспечение сложных систем управления, вычислительные сети и многие другие. Широкое использование ИМ объясняется тем, что размерность решаемых задач и неформализуемость сложных систем не позволяют использовать строгие методы оптимизации. Так выделяют, например, следующие проблемы в исследовании операций, которые не могут быть решены сейчас и в обозримом будущем без ИМ:

1. Формирование инвестиционной политики при перспективном планировании.
2. Выбор средств обслуживания (или оборудования) при текущем планировании.

3. Разработка планов с обратной информационной связью и операционных предписаний.

Эти классы задач определяются тем, что при их решении необходимо одновременно учитывать факторы неопределенности, динамическую взаимную обусловленность текущих решений и последующих событий, комплексную взаимозависимость между управляемыми переменными исследуемой системы, а часто и строго дискретную и четко определенную последовательность интервалов времени. Указанные особенности свойственны всем сложным системам.

Проведение имитационного эксперимента позволяет:

1. Сделать выводы о поведении СДС и ее особенностях:
    - без ее построения, если это проектируемая система;
    - без вмешательства в ее функционирование, если это действующая система, проведение экспериментов над которой или слишком дорого, или небезопасно;
    - без ее разрушения, если цель эксперимента состоит в определении пределов воздействия на систему.
2. Синтезировать и исследовать стратегии управления.
3. Прогнозировать и планировать функционирование системы в будущем.
4. Обучать и тренировать управленческий персонал и т.д.

ИМ является эффективным, но и не лишенным недостатков, методом. Трудности использования ИМ, связаны с обеспечением адекватности описания системы, интерпретацией результатов, обеспечением стохастической сходимости процесса моделирования, решением проблемы размерности и т.п. К проблемам применения ИМ следует отнести также и большую трудоемкость данного метода.

Интеллектуальное ИМ, характеризующиеся возможностью использования методов искусственного интеллекта и прежде всего знаний, при принятии решений в процессе имитации, при управлении имитационным экспериментом, при реализации интерфейса пользователя, создании информационных банков ИМ, использовании нечетких данных, снимает часть проблем использования ИМ.

В основе системы РДО – «Ресурсы, Действия, Операции» – лежат следующие положения:

- Все элементы сложной дискретной системы (СДС) представлены как ресурсы, описываемые некоторыми параметрами.
- Состояние ресурса определяется вектором значений всех его параметров; состояние СДС – значением всех параметров всех ресурсов.
- Процесс, протекающий в СДС, описывается как последовательность целенаправленных действий и нерегулярных событий, изменяющих определенным образом состояния ресурсов; действия ограничены во времени двумя событиями: событиями начала и конца.
- Нерегулярные события описывают изменение состояния СДС, непредсказуемые в рамках продукционной модели системы (влияние внешних по отношению к СДС факторов либо факторов, внутренних по отношению к ресурсам СДС). Моменты наступления нерегулярных событий случайны.
- Действия описываются операциями, которые представляют собой модифицированные продукционные правила, учитывающие временные связи. Операция описывает предусловия, которым должно удовлетворять состояние участвующих в операции ресурсов, и правила изменения ресурсов в начале и конце соответствующего действия.

При выполнении работ, связанных с созданием и использованием ИМ в среде РДО, пользователь оперирует следующими основными понятиями:

Модель - совокупность объектов РДО-языка, описывающих какой-то реальный объект, собираемые в процессе имитации показатели, кадры анимации и графические элементы, используемые при анимации, результаты трассировки.

Прогон - это единая неделимая точка имитационного эксперимента. Он характеризуется совокупностью объектов, представляющих собой исходные данные и результаты, полученные при запуске имитатора с этими исходными данными.

Проект - один или более прогонов, объединенных какой-либо общей целью. Например, это может быть совокупность прогонов, которые направлены на исследование одного конкретного объекта или выполнение одного контракта на имитационные исследования по одному или нескольким объектам.

Объект - совокупность информации, предназначенной для определенных целей и имеющая смысл для имитационной программы. Состав объектов обусловлен РДО-методом, определяющим парадигму представления СДС на языке РДО. Подробнее эти объекты рассмотрены последующих разделах, здесь же приведем лишь их перечень.

Объектами исходных данных являются:

- типы ресурсов (с расширением .rtp);
- ресурсы (с расширением .rss);
- образцы операций (с расширением .pat);
- операции (с расширением .opr);
- точки принятия решений (с расширением .dpt);
- константы, функции и последовательности (с расширением .fun);
- кадры анимации (с расширением .frm);
- требуемая статистика (с расширением .pmd);
- прогон (с расширением .smr).

Объекты, создаваемые РДО-имитатором при выполнении прогона:

- результаты (с расширением .pmv);
- трассировка (с расширением .trc).

Модель, написанная на языке РДО, можно разделить на базу данных и базу знаний. В первую входит описание типов ресурсов и самих ресурсов, которые определяют структуру базы данных и ее начальное состояние соответственно. База знаний – это процедурная часть модели, которая состоит из образцов операций, а также операций, описывающих знания о моделируемой системе или предметной области.

Сейчас в РДО существует 4 типа образцов: нерегулярное событие (irregular_event), продукционное правило (rule), операция (operation) и клавиатурная операция (keyboard). Остановимся подробнее на образцах типа rule и operation, представляющих продукционное правило и модифицированное продукционное правило соответственно.

В отличие от обычных продукционных правил, имеющих вид ЕСЛИ (условие) ТО (действие), и представляющих собой логическую взаимосвязь действий для описания поведения системы без учета времени, модифицированные продукционные правила имеют длительность выполнения. Такие модифицированные продукционные правила имеют следующий вид:

ЕСЛИ (условие) ТО1 (событие 1) ЖДАТЬ (временной интервал) ТО2 (событие 2)

В РДО описание операции имеет следующий формат:
$Pattern <имя_образца> : <тип_образца> [<признак_трассировки>]
[**$Parameters** <описание_параметров_образца>]
$Relevant_resources <описание_релевантных_ресурсов_образца>
[<способ_выбора>]
$Time = <выражение_времени>
$Body <тело_образца>
$End

Здесь в теле образца описывается выбор релевантных ресурсов, участвующих в операции, и действия над ними. Для этого указывается имя релевантного ресурса и правило его использования, описываемое для продукционного правила в виде:

Choice from <логическое_выражение> [<способ_выбора>]
[Convert_rule <конвертор>]

Для операции (модифицированного продукционного правила):
Choice from <логическое_выражение> [<способ_выбора>]

[**Convert_begin** <конвертор>]

[**Convert_end** <конвертор>]

В этой записи в блоке Choice_from содержится часть продукционного правила ЕСЛИ (условие), а в конверторах Convert_rule, Convert_begin и Convert_end – события, изменяющие состояние релевантного ресурса. Более детально синтаксис описан в [1].

1.6. Преимущества использования нечетких систем в моделировании

Преимущества использования нечетких систем (fuzzy-систем) можно охарактеризовать как:

- Возможность оперировать нечеткими входными данными: например, непрерывно изменяющиеся во времени значения (динамические задачи), значения, которые невозможно задать однозначно (результаты статистических опросов, рекламные компании и т.д.);
- Возможность нечеткой формализации критериев оценки и сравнения: оперирование критериями "большинство", "возможно", преимущественно" и т.д.;
- Возможность проведения качественных оценок как входных данных, так и выходных результатов: вы оперируете не только значениями данных, но и их степенью достоверности (не путать с вероятностью!) и ее распределением;
- Возможность проведения быстрого моделирования сложных динамических систем и их сравнительный анализ с заданной степенью точности: оперируя принципами поведения системы, описанными fuzzy-методами, вы во-первых, не тратите много времени на выяснение точных значений переменных и составление описывающих уравнений, во-вторых, можете оценить разные варианты выходных значений.

2. КОНСТРУКТОРСКАЯ ЧАСТЬ

2.1. Концептуальное проектирование

2.1.1. Общие сведения о программном комплексе RAO-studio

Программный комплекс **RAO-studio** предназначен для разработки и отладки имитационных моделей на языке РДО. Основные цели данного комплекса - обеспечение пользователя легким в обращении, но достаточно мощным средством разработки текстов моделей на языке РДО, обладающим большинством функций по работе с текстами программ, характерных для сред программирования, а также средствами проведения и обработки результатов имитационных экспериментов.

Объектами исходных данных в системе RAO-Studio являются:

- типы ресурсов (с расширением .rtp);
- ресурсы (с расширением .rss);
- образцы операций (с расширением .pat);
- операции (с расширением .opr);
- точки принятия решений (с расширением .dpt);
- константы, функции и последовательности (с расширением .fun);
- кадры анимации (с расширением .frm);
- требуемая статистика (с расширением .pmd);
- прогон (с расширением .smr).

Типы ресурсов определяют структуру глобальной базы данных программы (модели) и их описывают в отдельном объекте (имеет расширение .rtp).

Описание каждого типа ресурса имеет следующий формат:

$Resource_type <имя_типа> : <вид_ресурсов>
$Parameters
<описание_параметра> { <описание_параметра> }
$End

Образцы составляют совместно с операциями процедурную часть программы на РДО-языке. Они представляют собой знания о функционировании моделируемой системы (знания о предметной области), записанные в виде модифицированных продукционных правил в соответствии с синтаксисом языка. Описание всех образцов содержится в объекте образцов (с расширением .pat). Описание образца имеет следующий формат:

$Pattern <имя_образца> : <тип_образца> [<признак_трассировки>]
[$Parameters <описание_параметров_образца>]
$Relevant_resources <описание_релевантных_ресурсов_образца>
[<способ_выбора>]
$Time = <выражение_времени>
$Body <тело_образца>
$End

Тело образца имеет следующий формат:
<имя_релевантного_ресурса> <правило_использования>
{ <имя_релевантного_ресурса> <правило_использования> }

для образца типа rule:

<предусловие> [Convert_rule <конвертор>]
предусловие
Предусловие записывают в следующем формате:
Choice from <логическое_выражение> [<способ_выбора>]
или
Choice NoCheck [<способ_выбора>]

2.1.2. Функциональные диаграммы моделирования системы В РДО

Диаграмма IDEF0 должна описывать процесс и последовательность моделирования системы в среде РДО.

На диаграмме отмечены главные этапы моделирования производственной системы. На вход процесса моделирования подаётся исходный текст модели на

языке РДО, который компилируется системой во внутренние объекты. Затем внутренние объекты участвуют в прогоне ИМ, после чего формируются и выводятся результаты, полученные при прогоне.

Компиляция исходного текста модели осуществляется в четыре этапа. Первый этап – это лексический анализ кода модели, после которого распознанные лексемы в процессе грамматического анализа преобразуются во внутренние объекты системы РДО.

Диаграмма приведена в Приложении 2, 3, 4.

2.1.3. Разработка синтаксических правил описания нечетких параметров ресурса

Синтаксические диаграммы для описания нечеткого типа ресурса нечеткой лингвистической в системе РДО можно представить так, как на рисунке 1.

Рис 1. Описание нечеткого типа ресурса

Ключевое слово $Fuzzy_Parameters обозначает начало перечисления нечетких параметров ресурса, которые в свою очередь могут содержать в себе набор термов для каждого параметра. Заканчивается описание нечетких параметров ключевым словом $End.

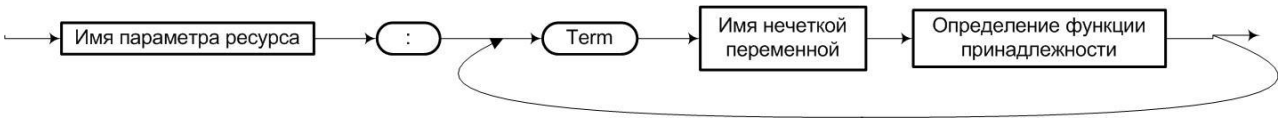


Рис. 2. Описание нечеткого параметра ресурса

На рис. 2. изображена синтаксическая диаграмма для описания нечеткого параметра ресурса. Имя параметра ресурса – простое имя в РДО, ключесвое слово Term обозначает следующий за ним имя нечеткой переменной, которое также имеет тип простого имени в РДО, за которым следует определение функции принадлежности.

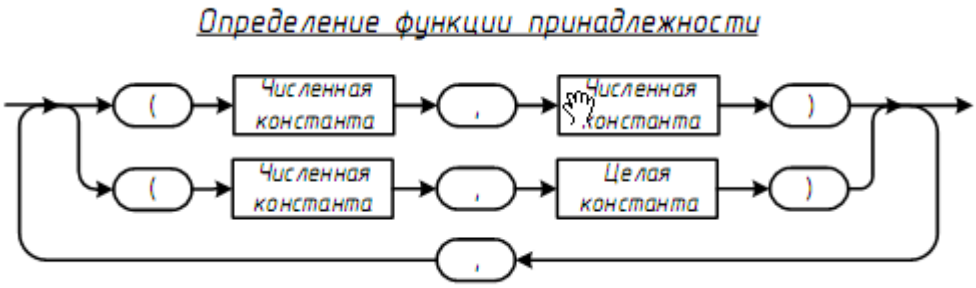


Рис. 3. Определение функции принадлежности.

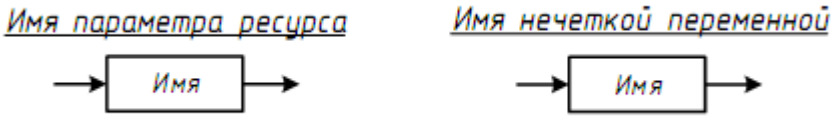


Рис. 4. Имя параметра ресурса и имя нечеткой переменной.

Определение функции принадлежности осуществляется путем перечисления точек – вершин ломанной кривой функции. Точки должны быть дейстыительными или целочисленными.

Имя параметра ресурса и имя нечеткой переменной имеет тип простого имени системы РДО.

2.2. Технический этап

2.2.1. Разработка алгоритма доступа к значению нечеткой переменной

Процесс фазификации в нечеткую переменную происходит следующим образом: сначала на вход поступает четкое базовое значение, затем для каждого лингвистического значения рассчитывается и присваивается значение функции принадлежности. Этот процесс проводится для всех лингвистических значений каждого нечеткого параметра ресурса.

Алгоритм определения значения функции принадлежности для одного нечеткого параметра ресурса имеет следующую структуру:

Рис. 5. Алгоритм определения значения функции принадлежности

Полностью алгоритм фазификации нечеткой переменной представлен в приложении 6.

2.2.2. Разработка алгоритма работы нечеткого продукционного правила

Процесс работы нечеткого вывода на основе продукционных правил осуществляется в несколько этапов. Для каждого продукционного правила производится расчет степени принадлежности для заключения на основе значений функций принадлежности предпосылок, и операций, их связывающих.

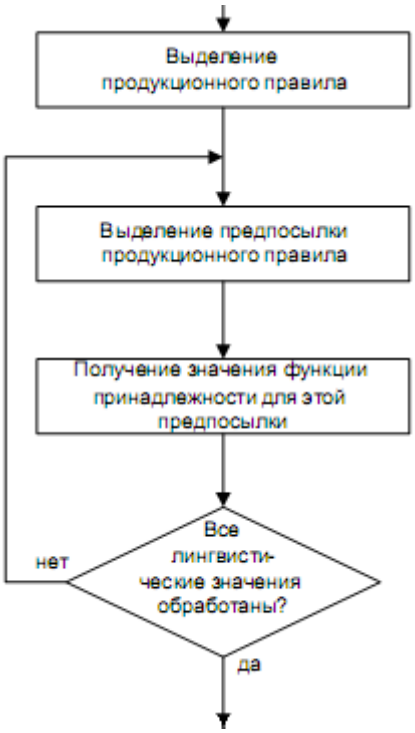


Рис. 6. Алгоритм определения значения степени принадлежности для предпосылки продукционного правила

Полностью алгоритм определения значения степени принадлежности для предпосылки продукционного правила представлен в приложении 7.

2.3. Рабочий этап

2.3.1. Разработка диаграмм классов

Для реализации подсистемы, интегрированной в систему РДО, работающей с нечеткими переменных в качестве параметров ресурсов модели, написанной на языке РДО в систему были введены новые классы, отвечающие за хранение и операции с нечеткими переменными. Эти классы организованы в виде древовидной структуры и объединены классом системы RDORTPParamType.

Класс RDORTPFuzzyMembershiftPoint предназначен для хранения каждой точик ломанной кривой, задающей график функции принадлежности для лингвистической переменной. Он содержит два вещественных значения этой точки а также функции для их возврата в вышестоящий класс.

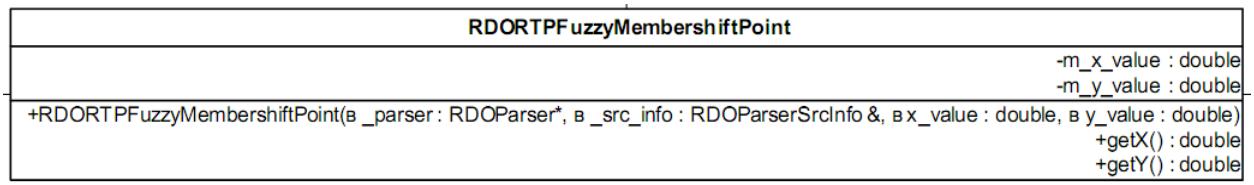


Рис. 7

Класс RDORTPFuzzyMembershiftFun объединяет все точки ломанной кривой рафика функции принадлежности для одной лингвистической переменной (объекты класса RDORTPFuzzyMembershiftPoint). Его параметры – это вектор m_points – вектор точек RDORTPFuzzyMembershiftPoint*, и вещественная переменная m_value, которая содержит значение, рассчитанной по графику функции принадлежности. За эту операцию отвечает функция getVal(). Также класс имеет функцию добавления add() точки графика.

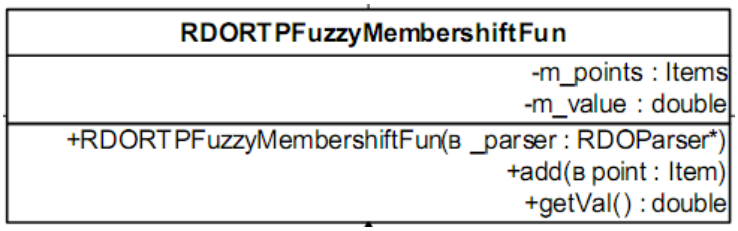


Рис. 8

Класс RDORTPFuzzyTerm соответствует терму лингвистической переменной. Он хранит в себе имя терма, значение функции принадлежности этого терма и связан указателем с объектом RDORTPFuzzyMembershiftFun, содержащим точки ломанной кривой рафика функции принадлежности для одной лингвистической переменной.

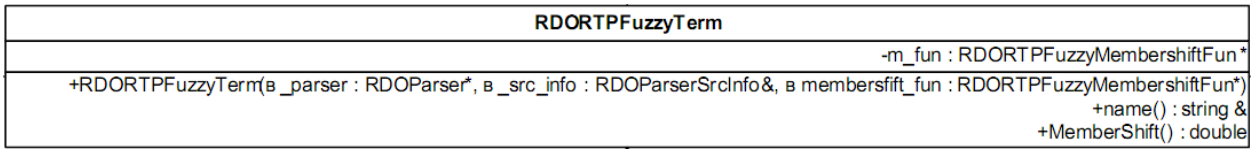


Рис. 9

Класс RDORTPFuzzyTermsSet необходим для объединения всех термов одного параметра в один вектор m_terms, а также для добавления нового терма с помощью функции add().

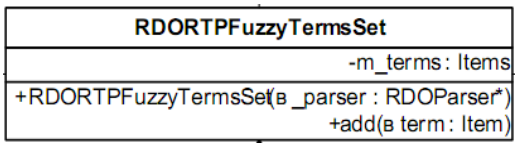


Рис. 10

Класс RDORTPFuzzyParam содержит данные одного нечеткого параметра, а именно его имя, и класс RDORTPFuzzyTermsSet, черег который передаются все термы параметра, а также их имена и значения функций принадлежности

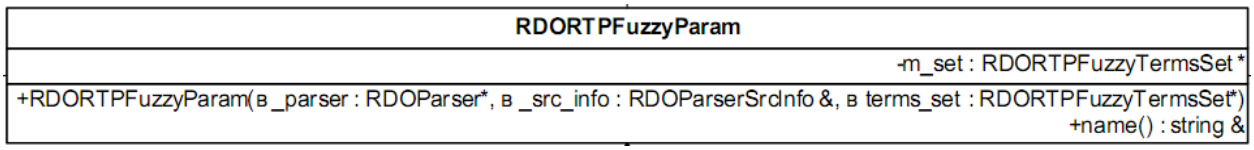


Рис. 11

Таким образом в системе РДО формируется еще один тип параметров для работы с нечеткими переменными. Полностью диаграмма классов представлена в приложении 5.

2.3.2. Программная реализация подсистемы для работы с нечеткими переменными

Подсистема для работы с нечеткими переменными в качестве параметров ресурсов модели, написанной на языке РДО, состоит из нескольких блоков. Первый отвечает за лексический анализ исходного текста, написанного на языке РДО. Сам анализ ведется с помощью программы flex.exe, на вход которой поступает распознаваемый текстовый файл и набор лексем и правил ля их выделения.

Для описания нечетких параметров были заведены следующие ключевые слова:

```
$Fuzzy_Parameters   return(RDO_Fuzzy_Parameters);
$fuzzy_parameters   return(RDO_Fuzzy_Parameters);
$Fuzzy_parameters   return(RDO_Fuzzy_Parameters);
$fuzzy_Parameters   return(RDO_Fuzzy_Parameters);
Term                return(RDO_Fuzzy_Term);
term                return(RDO_Fuzzy_Term);
```

RDO_Fuzzy_Term – лексема, возвращаемая программой flex.exe и поступающая на вход грамматическому анализатору.

В качестве грамматического анализатора применяется подпрограмма bison.exe. Для этой программы был написан файл грамматики, содержащий грамматические структуры, определенные синтаксическими диаграммами описания типа ресурса с нечеткими параметрами. В результате работы грамматического анализатора bison создаются внутренние объеты системы, описанные выше диаграммой классов, которые участвуют в прогоне имитационной модели. Файл грамматики для системы bison представлен в приложении 9.

3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

3.1. Описание тестового примера

В качестве тестового примера была выбрана модель работы отдела технического контроля на предприятии. В отдел поступают детали определенного типа. У детали контролируется шероховатость и квалитет установочной поверхности. Задача отдела технического контроля состоит в том, чтобы принять решение о годности/негодности (исправимый или неисправимый брак) детали. При этом модель должна работать таким образом, чтобы ошибка принятия решения была минимальна, т.е. чтобы детали, которые действительно оказались бы негодными при дальнейшем использовании, браковались отделом ОТК.

3.2. Тестовый пример

На языке РДО модель будет выглядеть следующим образом:

Типы ресурсов системы:

$Resource_type Детали_1 : **temporary**

$Parameters

Состояние : (Поступила, Идёт_контроль, Контроль_закончен, Заключение_сделано)

$Fuzzy_Parameters

Шероховатость : **real** [0.0..80.0]

Term Низкая (0.0, 1.0), (0.32, 1.0), (0.63, 0.0)

Term Удовлетворительная (0.32, 0.0), (0.63, 1.0), (1.25, 1.0), (1.8, 0.0)

Term Неудовлетворительная (1.25, 0.0), (1.8, 1.0), (80.0, 1.0)

Квалитет : **real** [0.0..18.0]

Term Высокий (0.0, 1.0), (8.0, 1.0), (9.0, 0.0)

Term Удовлетворительный (8.0, 0.0), (9.0, 1.0), (11.0, 0.0)

Term Неудовлетворительный (9.0, 0.0), (11.0, 1.0), (19.0, 1.0)

Заключение : **real** [0.0..10.0]

Term Годная (6.0, 0.0), (8.0, 1.0), (10.0, 0.0)

Term Исправимый_брак (2.0, 0.0), (4.0, 1.0), (6.0, 1.0), (8.0, 0.0)

Term Неисправимый_брак (0.0, 1.0), (2.0, 1.0), (4.0, 0.0)

$End

Для лингвистической переменной «шероховатость» зададим следующий график функций принадлежности:

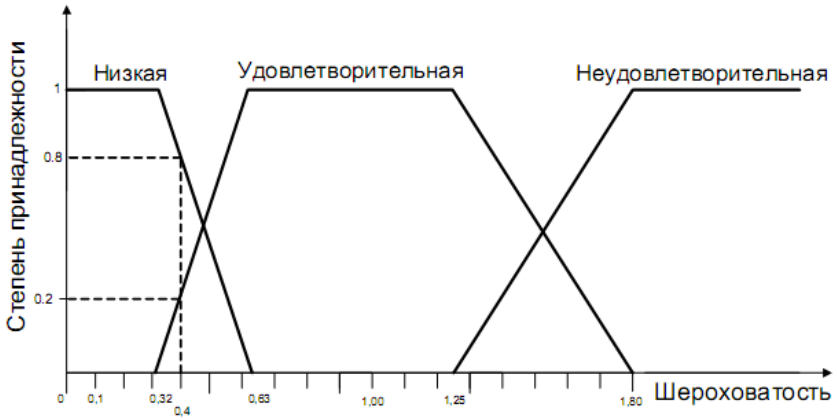


Рис. 1

Лингвистическая переменная «квалитет» определяется графиком функций принадлежности, представленным на рис.

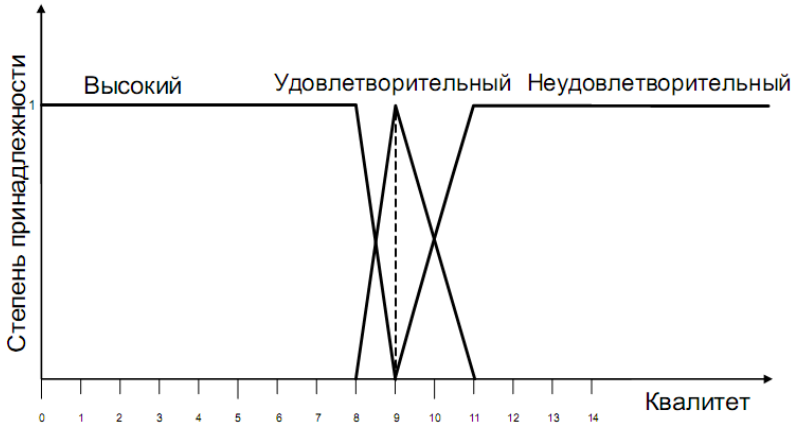


Рис. 2

И для переменной «заключение» зададим следующий график функции принадлежости:

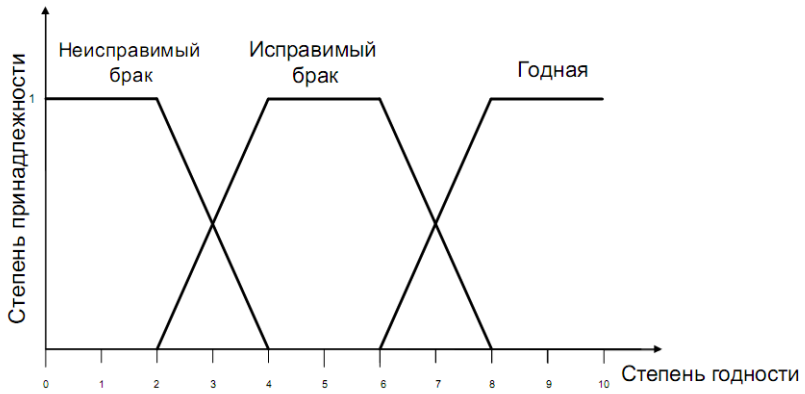


Рис. 3

Образцы операций:

$Fuzzy_Pattern Образец_Заключения : **rule**

$Relevant_resources

Деталь_1 : Детали_1 **Keep**

$Body

Вал

Choice from Деталь_1.Состояние = Контроль_закончен **AND**

(Деталь_1.Шероховатость = Неудовлетворительная **Or** Деталь_1.Квалитет = Неудовлетворительный)

Convert_rule

Состояние **set** Заключение_сделано

Заключение **set** Неисправимый_брак

Choice from Деталь_1.Состояние = Контроль_закончен **AND**

((Деталь_1.Шероховатость = Удовлетворительная **AND** Деталь_1.Квалитет = Высокий) **Or**

(Деталь_1.Шероховатость = Удовлетворительная **AND** Деталь_1.Квалитет = Удовлетворительный))

Convert_rule

Состояние **set** Заключение_сделано

Заключение **set** Исправимый_брак

Choice from Деталь_1.Состояние = Контроль_закончен **AND**

((Деталь_1.Шероховатость = Низкая **AND** Деталь_1.Квалитет = Высокий) **Or**

(Деталь_1.Шероховатость = Низкая **AND** Деталь_1.Квалитет = Удовлетворительный))

Convert_rule

Состояние **set** Заключение_сделано

Заключение **set** Годная

$End

3.3. Результаты тестирования

В процессе моделирования в ОТК будут поступать заготовки деталей, которым по заданному закону будут генерироваться значения шероховатости и квалитета. Блок принятия решения, основанный на нечетких продукционных правилах будет производить фазификацию нечеткой переменной, выполнит работу продукционных правил в соответствии с MAX/MIN правилами для конъюнкции и дизъюнкции, и затем согласно методам импликации, агрегации и дефазификации рассчитает выходную переменную – заключение.

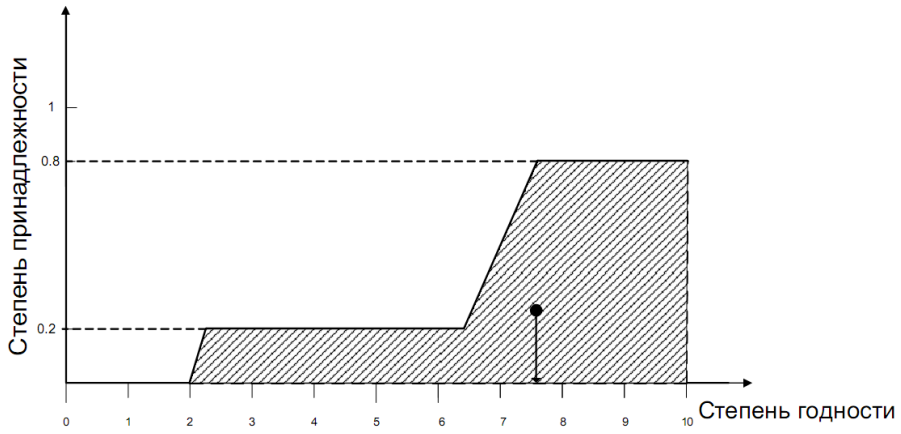


Рис. 4

В данном проекте была создана подсистемы, интегрированной в систему РДО, для работы с нечеткими переменных в качестве параметров ресурсов модели, написанной на языке РДО. Работа этой подсистемы была отлажена и проверена путем прогона в отоладчике Visual Studio 2005 (см. рис. 5 и приложение 8.), и проверено, что подпрограмма лексического и грамматического анализа исходного текста РДО-модели верно распознают разработанную для описания нечеткого параметра синтаксическую структуру, и все объекты, содержащие нечеткие параметры ресурсов создаются абсолютно правильно.

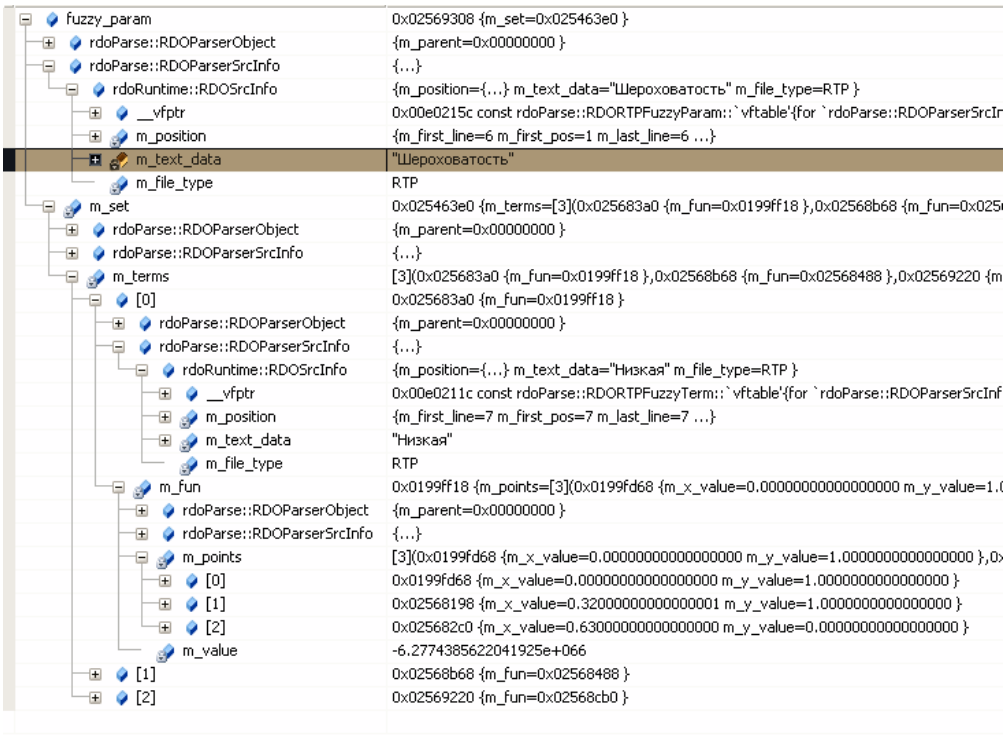

fuzzy_param
0x02569308 {m_set=0x025463e0 }
rdoParse::RDOParserObject
{m_parent=0x00000000 }
rdoParse::RDOParserSrcInfo
{...}
rdoRuntime::RDOSrcInfo
{m_position={...} m_text_data="Шероховатость" m_file_type=RTP }
__vfptr
0x00e0215c const rdoParse::RDORTPFuzzyParam::`vftable'{for `rdoParse::RDOParserSrcIr
m_position
{m_first_line=6 m_first_pos=1 m_last_line=6 ...}
m_text_data
"Шероховатость"
m_file_type
RTP
m_set
0x025463e0 {m_terms=[3](0x025683a0 {m_fun=0x0199ff18 },0x02568b68 {m_fun=0x025
rdoParse::RDOParserObject
{m_parent=0x00000000 }
rdoParse::RDOParserSrcInfo
{...}
m_terms
[3](0x025683a0 {m_fun=0x0199ff18 },0x02568b68 {m_fun=0x02568488 },0x02569220 {m
[0]
0x025683a0 {m_fun=0x0199ff18 }
rdoParse::RDOParserObject
{m_parent=0x00000000 }
rdoParse::RDOParserSrcInfo
{...}
rdoRuntime::RDOSrcInfo
{m_position={...} m_text_data="Низкая" m_file_type=RTP }
__vfptr
0x00e0211c const rdoParse::RDORTPFuzzyTerm::`vftable'{for `rdoParse::RDOParserSrcInf
m_position
{m_first_line=7 m_first_pos=7 m_last_line=7 ...}
m_text_data
"Низкая"
m_file_type
RTP
m_fun
0x0199ff18 {m_points=[3](0x0199fd68 {m_x_value=0.00000000000000000 m_y_value=1.(
rdoParse::RDOParserObject
{m_parent=0x00000000 }
rdoParse::RDOParserSrcInfo
{...}
m_points
[3](0x0199fd68 {m_x_value=0.00000000000000000 m_y_value=1.0000000000000000 },0x
[0]
0x0199fd68 {m_x_value=0.00000000000000000 m_y_value=1.0000000000000000 }
[1]
0x02568198 {m_x_value=0.32000000000000001 m_y_value=1.0000000000000000 }
[2]
0x025682c0 {m_x_value=0.63000000000000000 m_y_value=0.00000000000000000 }
m_value
-6.2774385622041925e+066
[1]
0x02568b68 {m_fun=0x02568488 }
[2]
0x02569220 {m_fun=0x02568cb0 }

Рис. 5

ЗАКЛЮЧЕНИЕ

В данном дипломном проекте была разработана подсистема использования нечетких переменных в качестве параметров ресурсов модели, написанной на языке РДО. В рамках работы были разработаны синтаксические правила описания нечетких параметров ресурсов модели, написанной на языке РДО, детально разработаны правила и алгоритмы функционирования подсистем для работы с нечеткими переменными, интегрированными в систему РДО - алгоритмы доступа к значениям нечетких параметров ресурса и работы нечеткого продукционного правила в модели РДО. Также была разработана подсистема компиляции нечетких параметров ресурсов языка РДО.

Использование лингвистических переменных позволяет приблизиться к описанию имитационных моделей на естественном языке пользователя, а механизм нечеткого вывода может позволить более адекватно описать процессы обслуживания и принимаемые при этом решения, нежели правила классической четкой логики.

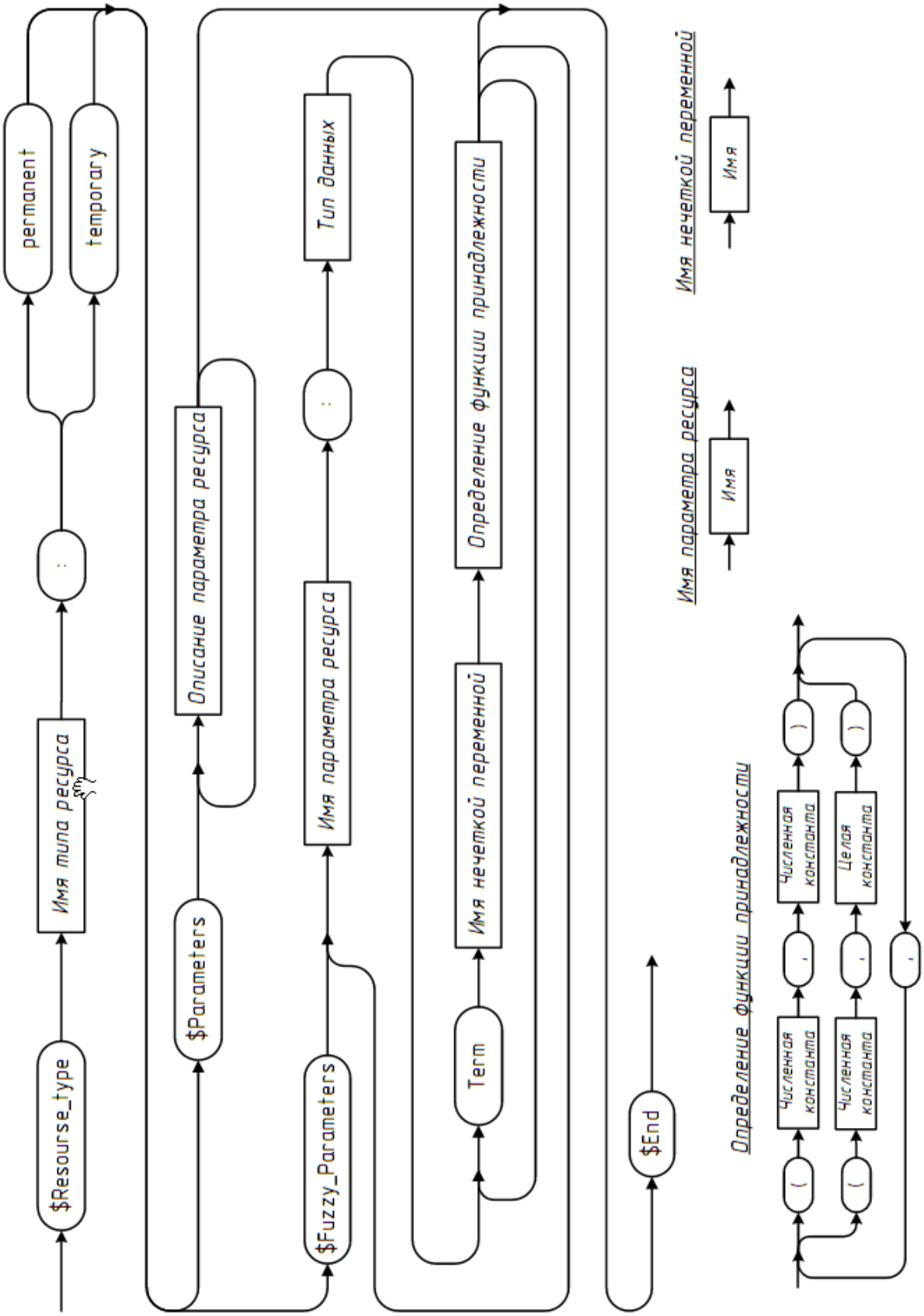
СПИСОК ЛИТЕРАТУРЫ

1. Емельянов В.В., Ясиновский С.И. Введение в интеллектуальное имитационное моделирование сложных дискретных систем и процессов. Язык РДО. – М.: АНВИК, 1998. – 427 с.
2. Ярушкина, Н.Г. Основы теории нечетких игибридных систем: Учеб. Пособие. – М.: Финансы и статистика, 2004. – 320 с.
3. Горнев В. Ф., Грибанов Н.Г., Овсянников М.В., Методические указания к дипломному проектированию для студентов кафедры РК-9 «Компьютерные системы автоматизации производства». Учебное пособие. - М.: Каф. РК9 МГТУ им. Н.Э.Баумана, 2004, 41с. (Файл: 04В10Метод ДП Горнев В.Ф., Грибанов Н.Г., Овсянников М.В.).
4. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание: Общие требования и правила составления [Текст]. – Взамен ГОСТ 7.1-84; введ. 2004-07-01. – М.: Изд-во стандартов, 2004. – 48 с.
5. ГОСТ 7.82-2001. Библиографическая запись. Библиографическое описание электронных ресурсов: Общие требования и правила составления [Текст].. – Введ. 2002-07-01. – М.: Изд-во стандартов, 2002. – 23 с.
6. Единое окно доступа к образовательным ресурсам. [Электронный ресурс]: дипломное проектировнаие. – Электрон. дан. – [Б.м.], [200-]. – Режим доступа: http://window.edu.ru/window_catalog/pdf2txt?p_id=2717&p_page=4. – Загл. с экрана.
7. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений: Пер. с англ. М.: Мир, 1976.
8. Манжай И. С., Урусов А. В. Интеграция лингвистических переменных в модели РДО//Четвертая Международная научно-практическая конференция "Интегрированные модели и мягкие вычисления в искусственном интеллекте". Сб. научных трудов, 2007.
9. Бьерн Страуструп. Язык программирования С++. Специальное издание. 3 издание Бином, 2004 г. 1104 стр.
10. Манжай И. С., Урусов А. В. Использование лингвистических переменных при описании образцов операций в РДО-моделях. Четвертая Международная научно-

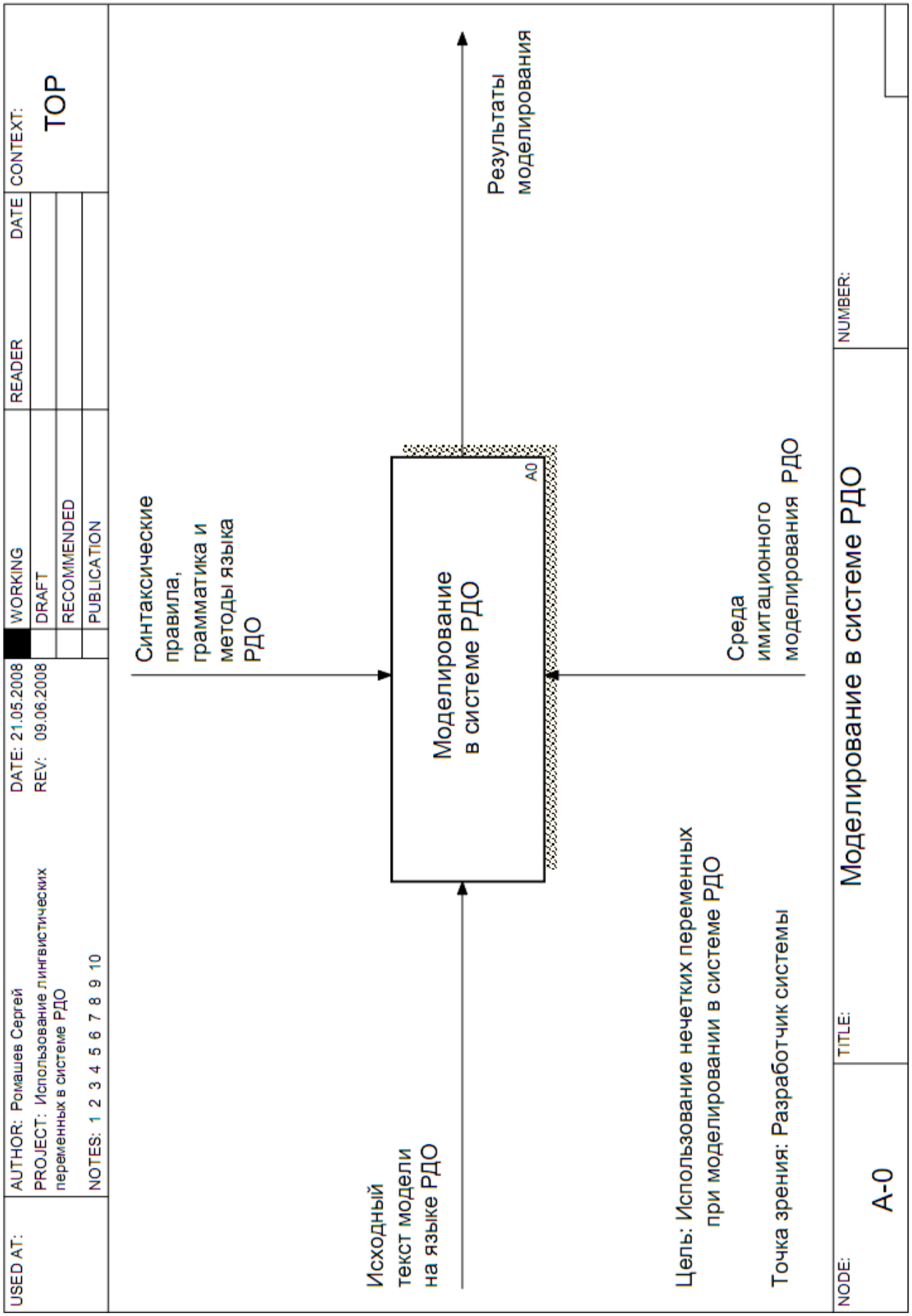
практическая конференция "Интегрированные модели и мягкие вычисления в искусственном интеллекте". Сб. научных трудов, 2007.

ПРИЛОЖЕНИЯ

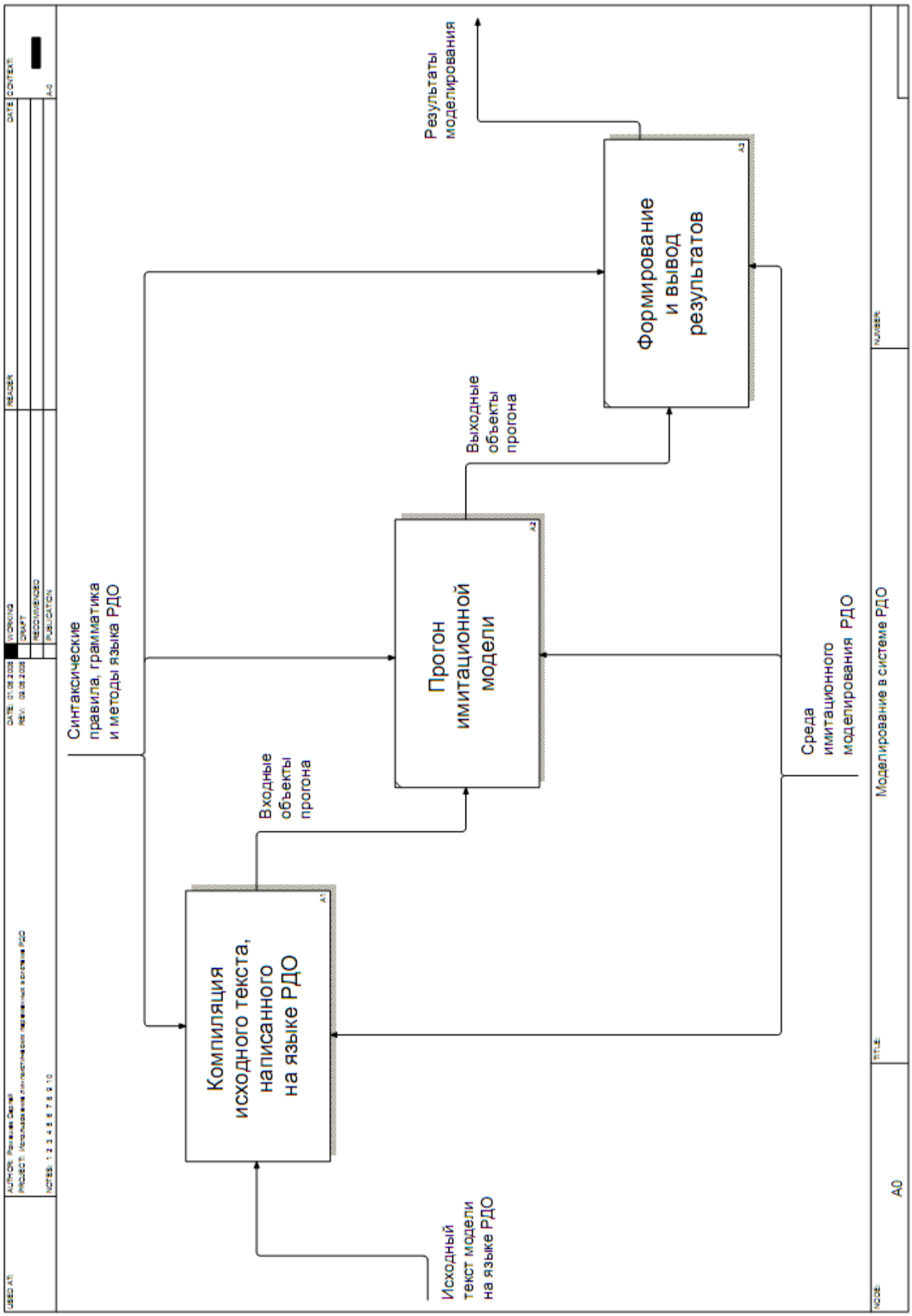
Приложение 1. Синтаксическая диаграмма описания типа ресурса.



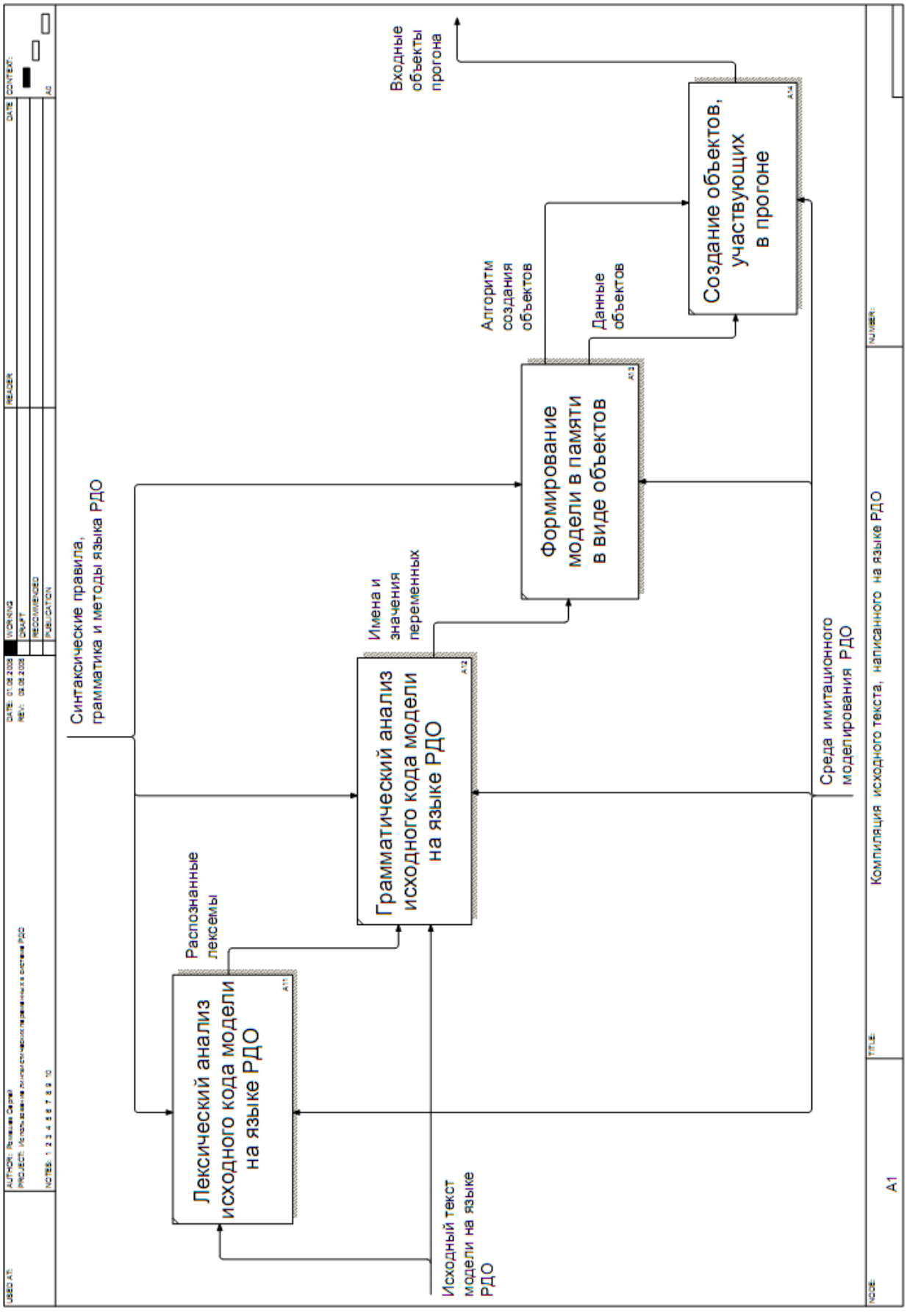
Приложение 2. Контекстная диаграмма А-0.



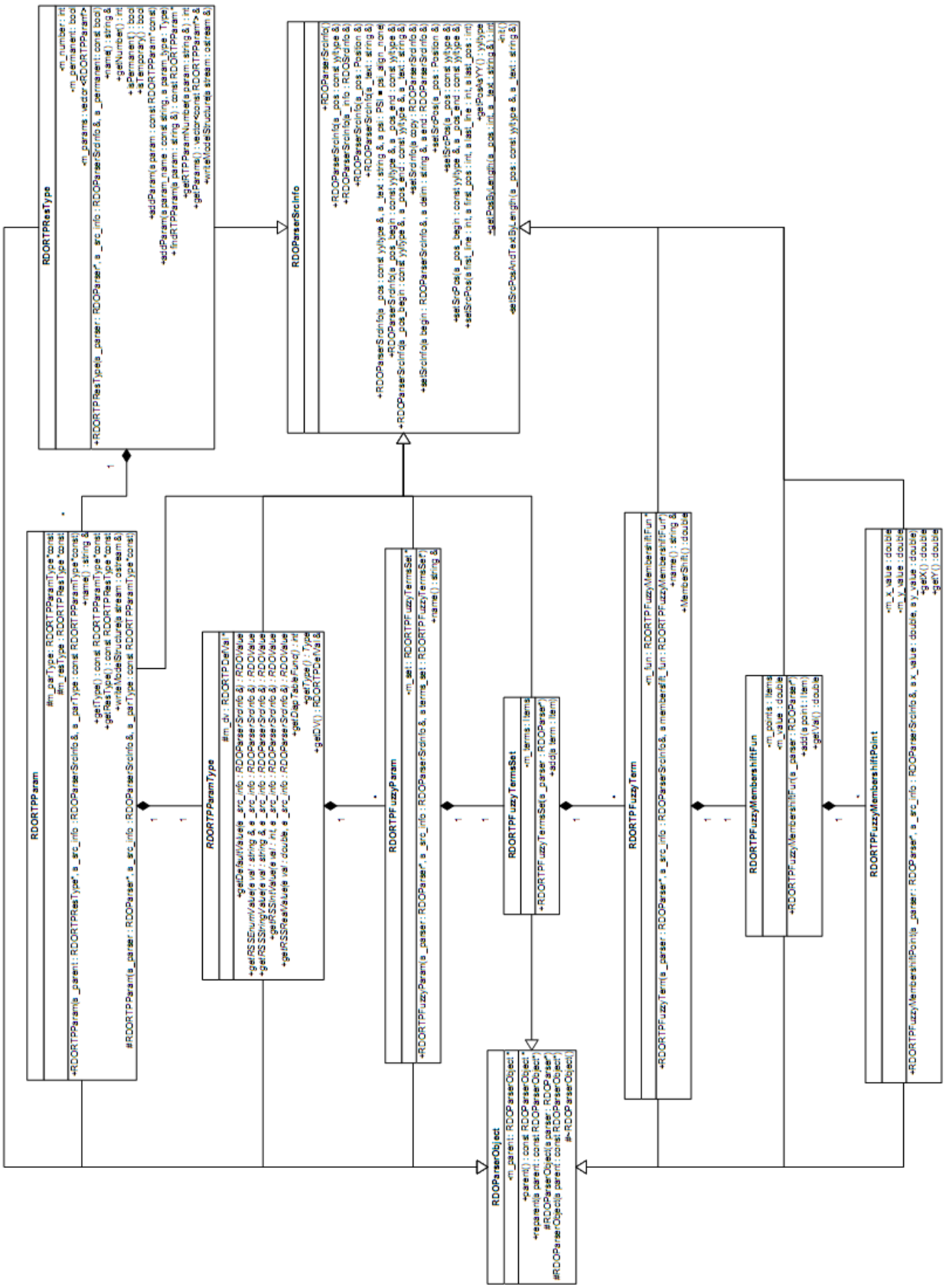
Приложение 3. Контекстная диаграмма A0. Моделирование в системе РДО.



Приложение 4. Диаграмма декомпозиции A1. Компиляция исходного текста, написанного на языке РДО.

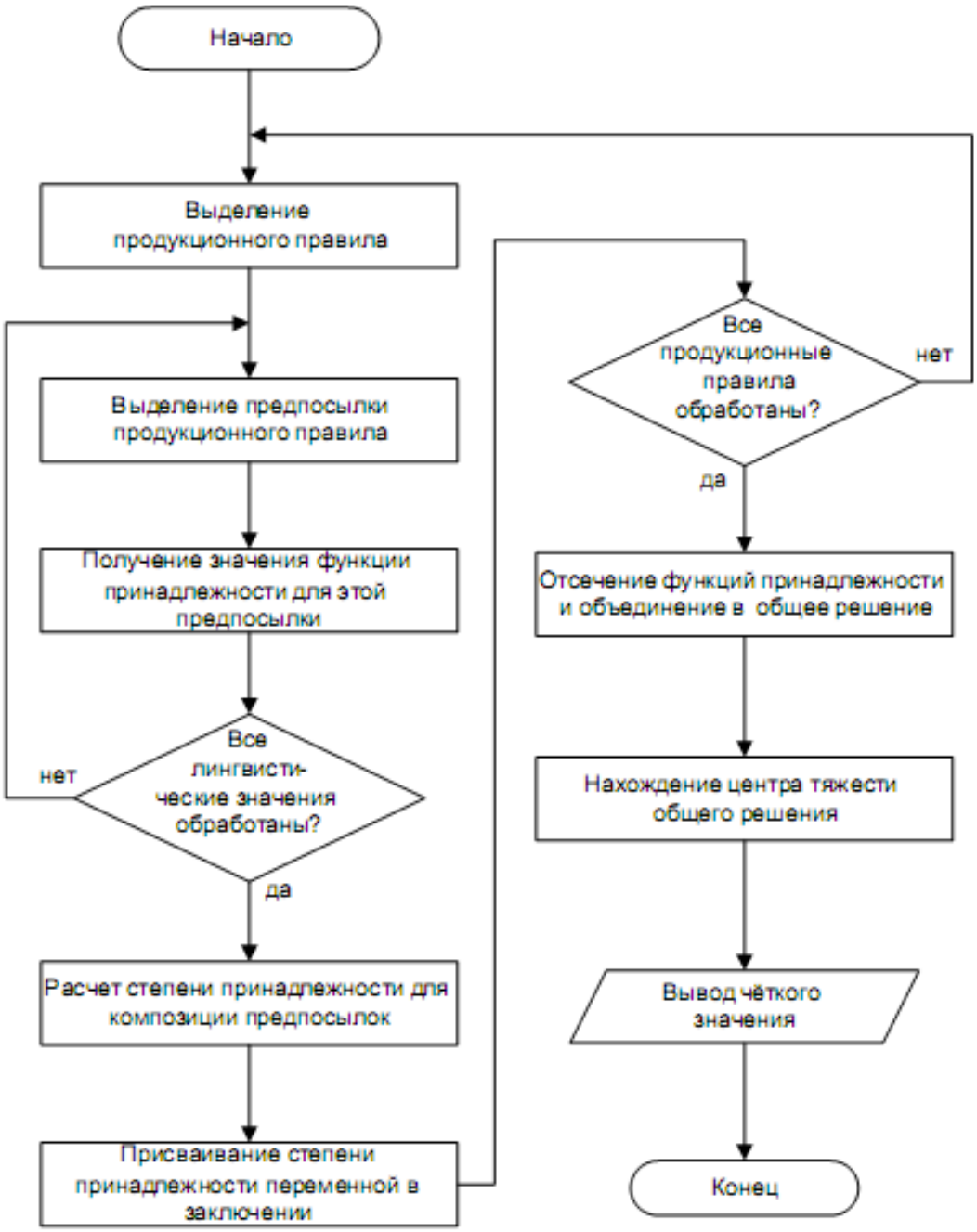


Приложение 5. Диаграмма классов.

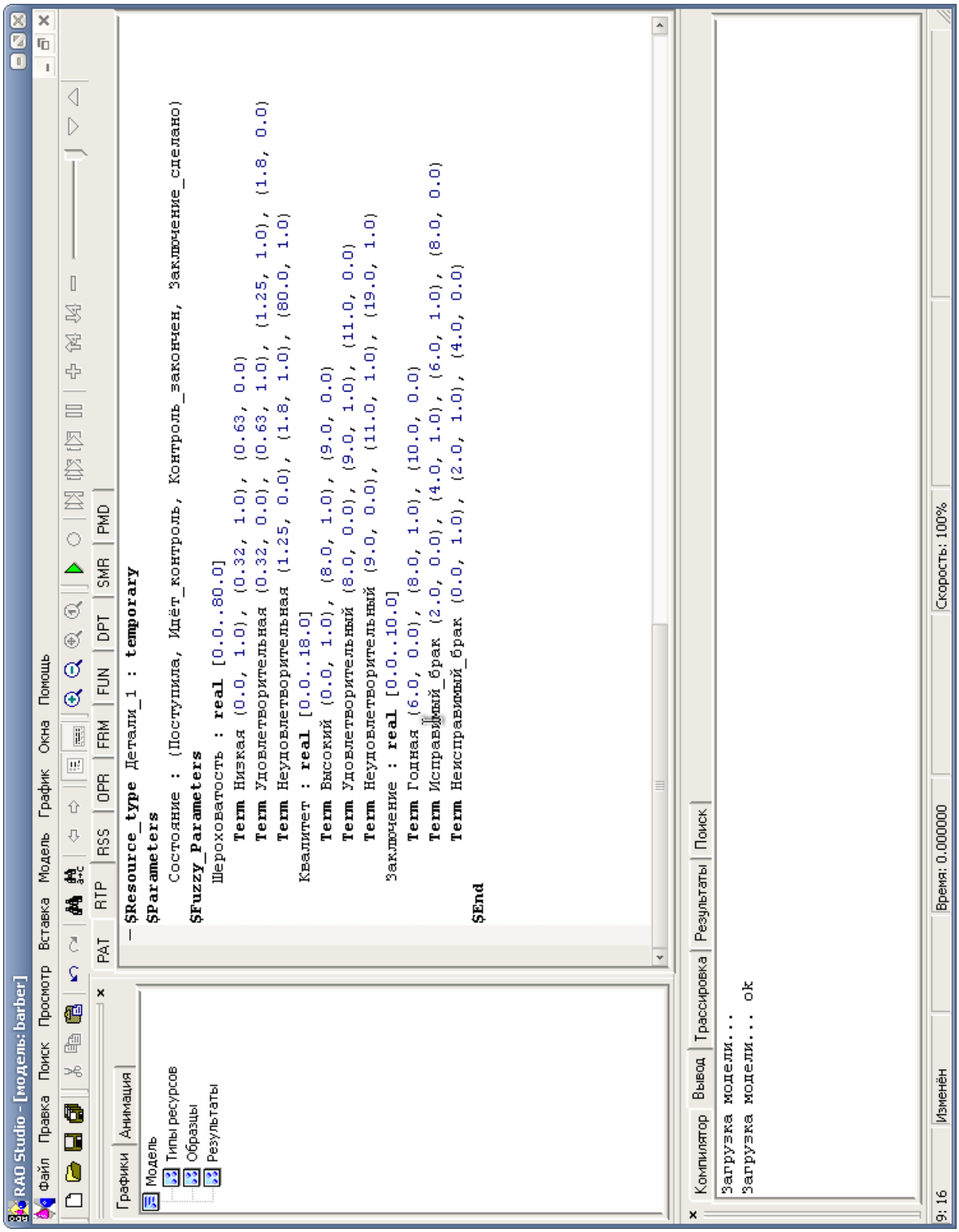


Приложение 6. Алгоритм фазификации нечетких переменных при описании параметров в РДО.

Приложение 7. Алгоритм работы нечеткого продукционного правила.



Приложение 8. Модель с нечеткими ресурсами в среде РДО.



Приложение 9. Входной файл грамматики для грамматического анализатора.

```
rtp_fuzzy_body:	/* empty */ {
			$$ = 0; // warning
		}
		| rtp_fuzzy_body rtp_fuzzy_param {
//			RDORTPFuzzyParam* fuzzy_param = reinterpret_cast<RDORTPFuzzyParam*>($2);
//			PARSER->getLastRTPResType()->addFuzzyParam( fuzzy_param );
			$$ = 1; // no warning
		};
rtp_fuzzy_param: RDO_IDENTIF_COLON fuzzy_param_type fuzzy_param_terms {
			RDOParserSrcInfo par_src_info(@1, *reinterpret_cast<std::string*>($1), RDOParserSrcInfo::psi_align_bytext);
//			RDORTPFuzzyParamType* fuzzy_parType = reinterpret_cast<RDORTPFuzzyParamType*>($2);
			RDORTPFuzzyTermsSet*   terms_set   = reinterpret_cast<RDORTPFuzzyTermsSet*>($3);
			RDORTPFuzzyParam* fuzzy_param = new RDORTPFuzzyParam( PARSER, par_src_info, terms_set );
			$$ = (int)fuzzy_param;
		}
		| RDO_IDENTIF_COLON fuzzy_param_type error {
			PARSER->error( @2, rdo::format( "Ожидаются термины нечеткого параметра" ) );
		}
		| RDO_IDENTIF_COLON error {
			if ( PARSER->lexer_loc_line() == @1.last_line ) {
				std::string str( reinterpret_cast<RDOLexer*>(lexer)->YYText() );
				PARSER->error( @2, rdo::format( "Неверный тип параметра: %s", str.c_str() ) );
			} else {
				PARSER->error( @1, "Ожидается тип параметра" );
			}
		}
		| error {
			PARSER->error( @1, "Неверное описание нечеткого параметра" );
		};
fuzzy_param_type: RDO_real param_real_diap param_real_default_val {
			RDORTPRealDiap* diap = reinterpret_cast<RDORTPRealDiap*>($2);
			RDORTPRealDefVal* dv = reinterpret_cast<RDORTPRealDefVal*>($3);
//			RDORTPRealParamType* rp = new RDORTPRealParamType( PARSER->getLastParsingObject(), diap, dv, RDOParserSrcInfo( @1, @3 ) );
//			$$ = (int)rp;
			$$ = 1
```

```
                    };

fuzzy_param_terms: /* empty */ {
                        RDORTPFuzzyTermsSet* terms_set = new
RDORTPFuzzyTermsSet( PARSER );
                        $$ = (int)terms_set;
                    }
                    | fuzzy_param_terms fuzzy_term {
                        RDORTPFuzzyTermsSet*   terms_set   =
reinterpret_cast<RDORTPFuzzyTermsSet*>($1);
                        RDORTPFuzzyTerm* term =
reinterpret_cast<RDORTPFuzzyTerm*>($2);
                        terms_set->add( term );
                        $$ = $1;
                    };

fuzzy_term: RDO_Fuzzy_Term RDO_IDENTIF fuzzy_membershift_fun {
                        RDOParserSrcInfo par_src_info(@2,
*reinterpret_cast<std::string*>($2), RDOParserSrcInfo::psi_align_bytext);
                        RDORTPFuzzyMembershiftFun* fuzzy_membershift_fun
= reinterpret_cast<RDORTPFuzzyMembershiftFun*>($3);
                        RDORTPFuzzyTerm* fuzzy_term = new
RDORTPFuzzyTerm( PARSER, par_src_info, fuzzy_membershift_fun );
//                      fuzzy_membershift_fun->reparent( fuzzy_term );
                        $$ = (int)fuzzy_term;
                    };

fuzzy_membershift_fun: /* empty */ {
                        RDORTPFuzzyMembershiftFun* fun = new
RDORTPFuzzyMembershiftFun( PARSER );
                        $$ = (int)fun;
                    }
                    | fuzzy_membershift_fun membershift_point {

                        RDORTPFuzzyMembershiftFun*   fun   =
reinterpret_cast<RDORTPFuzzyMembershiftFun*>($1);
                        RDORTPFuzzyMembershiftPoint* point =
reinterpret_cast<RDORTPFuzzyMembershiftPoint*>($2);
                        fun->add( point );
                        $$ = $1;
                        //Задание функции принадлежности точками -
вершинами ломанных кривых
                    };
membershift_point: '(' RDO_REAL_CONST ',' RDO_REAL_CONST ')' {

                        double x_value = *reinterpret_cast<double*>($2);
                        double y_value = *reinterpret_cast<double*>($4);
                        RDORTPFuzzyMembershiftPoint*
fuzzy_membershift_point = new RDORTPFuzzyMembershiftPoint( PARSER,
RDOParserSrcInfo( @1, @5 ), x_value, y_value);
                        $$ = (int)fuzzy_membershift_point;
                    }
```

```
                        | '(' RDO_REAL_CONST ',' RDO_REAL_CONST ')' ',' {

                                double x_value = *reinterpret_cast<double*>($2);
                                double y_value = *reinterpret_cast<double*>($4);
                                RDORTPFuzzyMembershiftPoint*
fuzzy_membershift_point = new RDORTPFuzzyMembershiftPoint( PARSER,
RDOParserSrcInfo( @1, @5 ), x_value, y_value);
                                $$ = (int)fuzzy_membershift_point;
                        }
                        | '(' RDO_REAL_CONST ',' RDO_INT_CONST ')' {

                                double x_value = *reinterpret_cast<double*>($2);
                                double y_value = $4;
                                RDORTPFuzzyMembershiftPoint*
fuzzy_membershift_point = new RDORTPFuzzyMembershiftPoint( PARSER,
RDOParserSrcInfo( @1, @5 ), x_value, y_value);
                                $$ = (int)fuzzy_membershift_point;
//                              RDORTPFuzzyMembershiftPoint*
fuzzy_membershift_point = new RDORTPFuzzyMembershiftPoint( PARSER, *((double *)$2),
*((double *)$4) );
//                              $$ = (int)(new RDORTPFuzzyMembershiftPoint(
PARSER, *((double *)$2), *((double *)$4)) );
                        }
                        | '(' RDO_REAL_CONST ',' RDO_INT_CONST ')' ',' {

                                double x_value = *reinterpret_cast<double*>($2);
                                double y_value = $4;
                                RDORTPFuzzyMembershiftPoint*
fuzzy_membershift_point = new RDORTPFuzzyMembershiftPoint( PARSER,
RDOParserSrcInfo( @1, @5 ), x_value, y_value);
                                $$ = (int)fuzzy_membershift_point;
                        };
```

СОДЕРЖАНИЕ

ЗАДАНИЕ НА ДИПЛОМНОЕ ПРОЕКТИРОВАНИЕ 2
РЕФЕРАТ 5
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, СИМВОЛОВ И СПЕЦИАЛЬНЫХ ТЕРМИНОВ С ИХ ОПРЕДЕЛЕНИЕМ 6
ВВЕДЕНИЕ 7
1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 9
1.1. Нечеткие множества и лингвистические переменные 9
1.2. Операции с нечеткими множествами. Нечеткая логика. 12
1.3. Фазификация и дефазификация нечетких переменных 14
1.4. Применение нечетких систем и нечеткого управления 17
1.5. Основы РДО-метода 22
1.6. Преимущества использования нечетких систем в моделировании 27
2. КОНСТРУКТОРСКАЯ ЧАСТЬ 28
2.1. Концептуальное проектирование 28
2.1.1. Общие сведения о программном комплексе RAO-studio 28
2.1.2. Функциональные диаграммы моделирования системы В РДО 29
2.1.3. Разработка синтаксических правил описания нечетких параметров ресурса 30
2.2. Технический этап 31
2.2.1. Разработка алгоритма доступа к значению нечеткой переменной 31
2.2.2. Разработка алгоритма работы нечеткого продукционного правила 32
2.3. Рабочий этап 33
2.3.1. Разработка диаграмм классов 33
2.3.2. Программная реализация подсистемы для работы с нечеткими переменными 35
3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ 37
3.1. Описание тестового примера 37
3.2. Тестовый пример 37
3.3. Результаты тестирования 40
ЗАКЛЮЧЕНИЕ 42
СПИСОК ЛИТЕРАТУРЫ 43

ПРИЛОЖЕНИЯ 45
Приложение 1. Синтаксическая диаграмма описания типа ресурса. 45
Приложение 2. Контекстная диаграмма А-0. 46
Приложение 3. Контекстная диаграмма А0. Моделирование в системе РДО. 47
Приложение 4. Диаграмма декомпозиции А1. Компиляция исходного текста, написанного на языке РДО. 48
Приложение 5. Диаграмма классов. 49
Приложение 6. Алгоритм фазификации нечетких переменных при описании параметров в РДО. 50
Приложение 7. Алгоритм работы нечеткого продукционного правила. 51
Приложение 8. Модель с нечеткими ресурсами в среде РДО. 52
Приложение 9. Входной файл грамматики для грамматического анализатора. 53