УДК 004.89

Формирование баз знаний продукционного типа на основе UML-моделей

Н.О. Дородных, А.Ю. Юрин
Институт динамики и теории управления им. В.М. Матросова СО РАН (ИДСТУ СО РАН)
Иркутский национальный исследовательский технический университет (ИрНИТУ)
tualatin32@mail.ru, iskander@icc.ru

Дородных Н.О., Юрин А.Ю. Формирование баз знаний продукционного типа на основе UML-моделей. *В работе рассмотрено программное средство (Personal Knowledge Base Designer, PKBD) автоматизирующее разработку баз знаний продукционного типа. В качестве целевого языка представления знаний используется CLIPS (C Language Integrated Production System). Данное средство позволяет интегрироваться с CASE-средствами IBM Rational Rose и StarUML, в части импорта UML-моделей и генерации на их основе кода баз знаний в формате CLIPS. Для промежуточного представления и хранения извлекаемых логических правил (продукций) используется специальная обобщенная продукционная модель, которая позволяет абстрагироваться от особенностей определенных языков представления знаний. Средство ориентировано на непрограммирующего специалиста и обладает расширяемой архитектурой, за счет подключения программных модулей, в виде динамических библиотек, поддерживающих генерацию кода на других языках представления знаний. Тестирование разработанных баз знаний осуществляется путем подключения и запуска машин вывода. Приведены предварительные результаты тестирования (апробации) программного средства при решении различных учебных задач.*

Ключевые слова: *программная система, базы знаний, концептуальная модель, генерация кода, CLIPS, UML.*

Введение

Сложность и трудоемкость процесса разработки экспертных систем (ЭС) обусловлена, главным образом, сложностью и трудоемкостью этапа разработки баз знаний (БЗ), который включает задачи по формализации предметных знаний и их описанию на определенном языке представления знаний (ЯПЗ) [1]. Повышение эффективности решения данных задач, путем их автоматизации, обусловливает необходимость разработки специализированных программных средств. Подобные программные средства в виде специализированных редакторов БЗ (например, Visual Expert System Designer, Expert System Designer, ES-Builder и др.), позволяют повысить эффективность процесса разработки за счет использования визуального моделирования, шаблонов представления знаний, автоматизации процесса верификации и генерации кода БЗ.

Одним из распространенных источников знаний являются концептуальные модели, создаваемые в процессе проектирования программного обеспечения, анализа и моделирования предметной области. Данные модели, представленные в форме диаграмм (DFD, IDEF0, IDEF5, UML и др.) (см. рис. 1), являясь результатом этапов идентификации и концептуализации, в большинстве случаев не преобразовываются (или преобразование сильно ограничено) в программные коды БЗ.

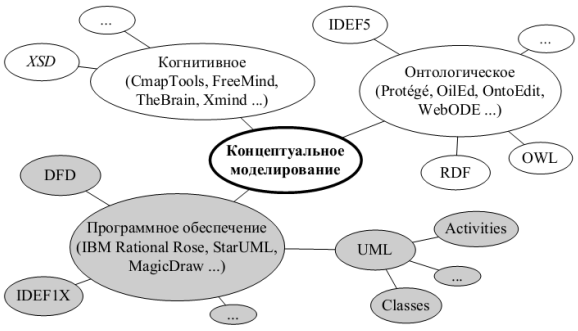

Рисунок 1 – Концепт-карта стандартов концептуального моделирования.

При наличии большого количества CASE-средств, обеспечивающих построение диаграмм (в частности, UML: IBM Rational Rose, StarUML, Enterprise Architect и др.) и синтез каркасных программных кодов на языках программирования общего назначения, не выявлено ни одного средства, генерирующего программные коды для БЗ ЭС. В тоже время в наиболее распространенных системах программирования продукционных БЗ (например, Exsys Corvid, ClipsWin и др.) отсутствует возможность

интеграции с CASE-средствами в части импорта концептуальных моделей. Что, в свою очередь, обуславливает постановку задачи создания специализированного алгоритмического и программного обеспечения, обеспечивающего анализ концептуальных моделей и синтез программных кодов БЗ продукционного типа.

Таким образом, требуется разработать программную систему, обеспечивающую:

поддержку процесса проектирования БЗ продукционного типа;

интеграцию с CASE-средствами в части импорта концептуальных моделей в форме диаграмм классов UML;

синтез и проверку программных кодов на CLIPS (Language Integrated Production System) [2];

поддержку RVML (Rule Visual Modelling Language) [3] – графической нотации для моделирования логических правил.

В качестве CASE-средства для интеграции предлагается использовать IBM Rational Rose [4].

Применение подобной программной системы позволит исключить ошибки, возникающие при ручном программировании, и обеспечит возможность быстрого прототипирования БЗ. Подробному описанию алгоритмического обеспечения, особенностям преобразования концептуальных моделей и краткому описанию RVML посвящены статьи [5, 6], поэтому в данной работе уделим основное внимание описанию программного средства и оценке эффективности его применения путем апробации на учебных (тестовых) примерах.

Предлагаемое решение

В результате решения поставленной задачи была разработана программная система – Personal Knowledge Base Designer, которая представляет собой специализированный редактор для разработки и тестирования продукционных БЗ ЭС, в частности, для CLIPS [7].

Основными функциями редактора являются:

создание элементов продукционных БЗ (шаблонов фактов и правил, а также фактов и правил) непрограммирующим пользователем, благодаря использованию либо набора подпрограмм-мастеров, либо предварительно подготовленных шаблонов фактов и правил;

поддержка авторской графической нотации – RVML [3] для визуального представления продукций. Что позволяет более наглядно отобразить причинно-следственные отношения и учесть некоторые особенности CLIPS (например, важность правил и коэффициенты уверенности);

интеграция с CASE-средствами IBM Rational Rose [4] и StarUML, в части импорта концептуальных моделей (диаграмм классов UML), которые могут быть использованы для генерации элементов БЗ;

интеграция с CLIPS [2], в части синтеза отчуждаемого кода БЗ и его тестирования, благодаря включению в состав модулей программной системы машины вывода CLIPS;

формирование специализированных отчетов.

С целью реализации требований и функций, разработана архитектура [7], включающая следующие основные модули (см. рис. 2):

управления базами знаний – обеспечивает загрузку и сохранение БЗ в формате EKB (XML-подобный формат программной системы для хранения знаний);

управления метамоделью – обеспечивает внутреннее представление продукционной модели знаний, которое не зависит от определенного языка программирования БЗ, а также манипулирование (создание, редактирование, удаление) элементами этой модели;

управления модулями поддержки языков программирования – обеспечивает подключение и отключение модулей ЯПЗ, а также доступ к их функциям;

интеграции с концептуальными моделями – обеспечивает загрузку элементов из концептуальных моделей (диаграмм классов UML), построенных в CASE-средствах IBM Rational Rose и StarUML;

управления машинами вывода – обеспечивает использование машин вывода (в виде динамических библиотек) для тестирования БЗ, включая объяснение полученных результатов;

редактор RVML – обеспечивает представление элементов БЗ в виде графических примитивов, расширяющих UML;

графический пользовательский интерфейс – обеспечивает доступ к перечисленным функциям.

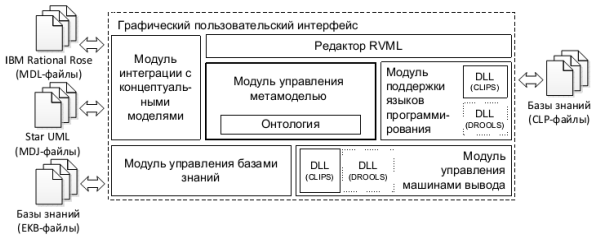

Рисунок 2 – Архитектура программной системы.

Разработанный редактор обладает простым и интуитивно понятным графическим интерфейсом, который представлен основным рабочим пространством и набором программ-мастеров, которые подставляют собой последовательность экранных форм, сегментирующих и упорядочивающих процессы ввода и редактирования элементов БЗ.

В частности, при создании шаблона факта пользователю последовательно предлагается задать: имя шаблона (используется для отображения в редакторе), описание и свойства (слоты). Подобные программы-мастера также применяются для создания и редактирования шаблонов правил.

На основе данных шаблонов, пользователь может вводить конкретные факты и правила.

Пример применения

Редактор позволяет быстро создавать и тестировать продукционные БЗ, в том числе, на основе анализа и преобразование концептуальных моделей, в частности, диаграмм классов UML.

Рассмотрим простейший пример (использовался в учебном процессе) разработки ЭС для поддержки принятия решений при решении задач прогнозирования лесных пожаров. Основные понятия и отношения предметной области могут быть описаны в виде концептуальной модели (Рис. 3), анализ которой при импортировании в редактор позволяет сформировать описание шаблонов фактов и правил, используемых в дальнейшем для создания конкретных фактов и правил. При этом каждый элемент БЗ имеет графическое представление в виде RVML-схемы (Рис.4) на основе которого синтезируются программный код CLIPS.

Рисунок 3 – Пример UML-модели.

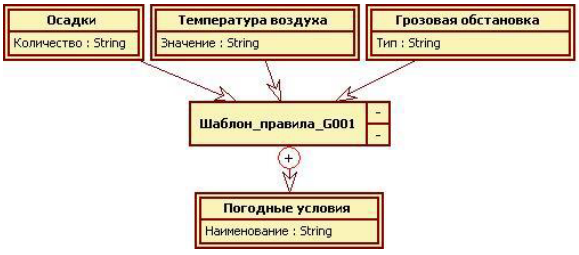

Рисунок 4 – Пример RVML-схемы.

Для проведения тестирования разработанной БЗ необходимо ввести начальные факты и активировать машину вывода (Рис. 5). В результате подобной проверки возможно ознакомиться с активированными правилами и изменениями в рабочей памяти, например, добавленными фактами.

Проверенная экспертом БЗ может быть сохранена в формате CLIPS и использована в сторонних приложениях.

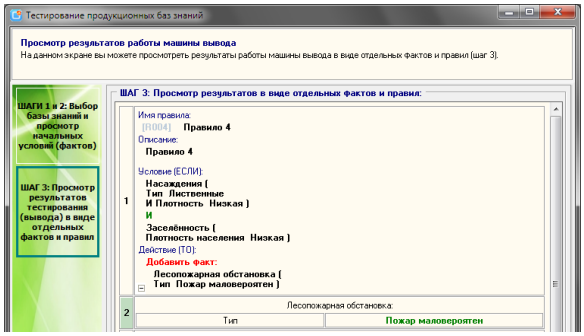

Рисунок 5 – Форма результатов проверки созданной базы знаний.

Оценка эффективности

Разработанной алгоритмическое и программное обеспечение проходило апробацию на базе Иркутского национального исследовательского технического университета (ИрНИТУ). В апробации приняло участие 60 студентов Института кибернетики им. Е.И. Попова изучающих курсы «CASE-средства», «Инструментальные средства информационных технологий» и «Технологии программирования», группы АСУз-10, АСУбз-11, АСУб-12, ЭВМбзс-12. Студенты обладали знаниями в области проектирования программного обеспечения, UML и искусственного интеллекта.

Основная задача апробации состояла в том, чтобы на примере решения учебных задач оценить трудоемкость разработки БЗ в формате CLIPS с использованием разработанного редактора и без него (путем применения других средств).

Таким образом, студентам было предложено разработать статические ЭС для решения задач диагностирования или прогнозирования в определенной предметной области (в зависимости от варианта задания, Табл. 1) тремя различными способами:

С1 – путем построения диаграмм классов UML в IBM Rational Rose, с последующим их импортом в Personal Knowledge Base Designer (PKBD) и доработкой полученных структур;

С2 – путем построения диаграмм классов UML в IBM Rational Rose, с последующим их использованием в других средствах разработки БЗ CLIPS;

С3 – путем использования других средствах разработки БЗ CLIPS без построения каких либо диаграмм.

В качестве «другого средства» для разработки БЗ CLIPS выбрана среда разработки –

ClipsWin [8].

При этом для обеспечения возможности неоднократного повторения процесса решения задач и их временной компактности выполнения были введены ограничения на варианты заданий:

число предметных сущностей: 5-10;

число свойств предметных сущностей: до 3;

число связей между сущностями: 5-10;

число причинно-следственных связей: 3-4;

число экземпляров причинно-следственных связей (возможных правил): 10-15.

Следует отметить, что первые 4 пункта представляют собой ограничения, налагаемые на этап концептуального моделирования (ограничения на элементы диаграммы классов UML). Последний пункт относится к этапу программирования (реализации) кода БЗ в среде разработки (ограничение на количество возможных правил в БЗ).

Таблица 1. Описание решаемых задач.

№	Предмет. сущности (ед.)	Связи (ед.)	Прич.-след. связи (ед.)	Правила (ед.)
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

Для оценки трудоемкости использовался временной критерий (затраты времени на выполнение отдельных этапов разработки ЭС). Оценка осуществлялась на следующих этапах [1, 9]:

Концептуализация:

формулировка базовых концепции и отношений между ними, включая: характеристику различных видов используемых данных, анализ информационных потоков и лежащих в их основе структур в предметной области в терминах, причинно-следственных связей, отношений частное/целое, постоянное/временное и т.п.;

построение концептуальной модели.

Формализация:

перевод ключевых понятий и отношений на некоторый формальный ЯПЗ;

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.

Реализация:

преобразование формализованных знаний в работающую программу (код БЗ).

Основным результатом этапов концептуализации и формализации являлась концептуальная модель предметной области, представленная в форме диаграмм классов UML. Основной результат этапа реализации – синтаксически корректный программный код БЗ, проверенный на адекватность и непротиворечивость.

Результаты оценки временных затрат представлены в таблице 2, 3 и на рисунке 7. При этом выделены минимальные и максимальные процентные значения относительной разницы между С1 и С3, С1 и С2.

Отметим ряд особенностей способов:

С2: данный способ показал самые большие временные показатели, обусловленные тем, что построенные модели вручную переносились в среду разработки ClipsWin, так как у данного средства отсутствует поддержка возможности автоматической кодогенерации БЗ на основе созданных концептуальных моделей. При этом следует отметить, что другие программные средства, позволяющие синтезировать код БЗ в формате CLIPS (например, [10]), не удалось применить для этой задачи;

С3: Функциональные ограничения ClipsWin в части редактирования водимого программного кода обусловили применение дополнительного текстового редактора (Programmer’s Notepad) при выполнении этапа кодирования. В частности, сначала осуществлялось описание кода БЗ во внешнем текстовом редакторе (используя возможности копирования и вставки отдельных блоков кода), а затем полученный код импортировался в ClipsWin, где осуществлялась проверка синтаксиса. На практике данная схема позволила снизить в 1,5 раза время на создание БЗ.

Анализ эффективности предлагаемого метода по временному критерию показал, что эффективность разработки баз знаний методом С1 может быть повышена, в среднем на 60.3% по сравнению с С2 и на 48.2% по сравнению с С3 за счет автоматической кодогенерации на основе визуальных моделей, что в свою очередь позволяет:

Таблица 2. Результаты оценки временных затрат.

№	Моделирование, мин.	PKBD, мин.	C1, мин.	C2, мин.	C3, мин.
1	10,89	7,2	18,09	41,29	30,4
2	8,36	7,1	15,46	32,86	24,5
3	8,58	8,3	16,88	36,46	27,88
4	9,36	5,83	15,19	26,82	17,46
5	11,25	5,52	16,77	64,41	53,16
6	10,78	4,6	15,38	43,8	33,02
7	6,6	15,82	22,42	68	61,4
8	10,95	7,56	18,51	57,23	46,28
9	7,37	7,2	14,57	54,71	47,34
10	12,58	6,6	19,18	42,7	30,12
11	8,69	5,5	14,19	38,01	29,32
12	8,36	6	14,36	45,22	36,86
13	10,64	7,42	18,06	44,31	33,67
14	10,66	10,23	20,89	50,57	39,91
15	10,01	7,56	17,57	55,5	45,49
16	10,92	8,96	19,88	49,42	38,5
17	8,14	8,36	16,5	45,59	37,45
18	11,55	18	29,55	47,43	35,88
19	11,85	5,2	17,05	43,28	31,43
20	9,12	7,44	16,56	40,95	31,83

Таблица 3. Результаты оценки временных затрат.

№	C3: ошибки програм., шт.	Относительная разница, % C1 и C3	Относительная разница, % C1 и C2
1	2	40,49	56,19
2	0	36,89	52,95
3	1	39,45	53,70
4	0	13,00	43,36
5	3	68,45	73,96
6	1	53,42	64,89
7	5	63,48	67,03
8	3	60,00	67,66
9	3	69,22	73,37
10	0	36,32	55,08
11	0	51,60	62,67
12	2	61,04	68,24
13	2	46,36	59,24
14	1	47,66	58,69
15	4	61,38	68,34
16	1	48,36	59,77
17	1	55,94	63,81
18	3	17,64	37,70
19	1	45,75	60,61
20	2	47,97	59,56
Итоговое сред. значение относ. разницы:		48,2	60,3

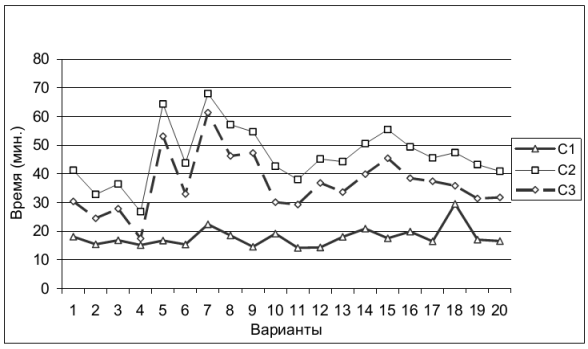

Рисунок 7 – Результаты оценки затрат времени на разработку баз знаний.

эффективно использовать результаты этапов концептуализации и формализации в форме диаграмм классов UML, рассматривая последние не как статические графические артефакты, а как основу для формирования программного кода в соответствии с идеологией модельно-управляемого подхода (Model Driven Architecture) [11];

снизить риск ошибок проектирования, за счет возможности быстрого прототипирования БЗ и получения их программного кода;

исключить ошибки программирования, за счет автоматического отображения элементов концептуальной модели в языковые конструкции CLIPS.

Заключение

Концептуальные модели остаются важным источником знаний при разработке БЗ ЭС, который в настоящее время достаточно ограниченно используется. Данные ограничения обусловлены недостатками существующего алгоритмического и программного обеспечения, что, в свою очередь, обуславливает необходимость создания специализированных программных систем в форме редакторов БЗ, способных интегрироваться с различными CASE-средствами в части импорта концептуальных моделей.

Примером подобного редактора является Personal Knowledge Base Designer [12], особенностями которого являются:

ориентация на непрограммирующего специалиста (свойство реализовано с помощью набора подпрограмм-мастеров, обеспечивающих описание знаний в виде продукций).

поддержка авторской графической нотации – Rule Visual Modeling Language (RVML) [3];

интеграция с CASE-средствами IBM Rational Rose [4] и StarUML, в части импорта и экспорта концептуальных моделей (диаграмм классов UML).

Программная система используется в учебном процессе в Иркутском национальном исследовательском техническом университете

(ИрНИТУ) при выполнении лабораторных работ по курсам «CASE-средства», «Инструментальные средства информационных систем» и «Технологии программирования».

Апробация редактора показала значительное снижение времени разработки прототипов БЗ продукционных ЭС. Кроме того, использование концептуальных моделей и Personal Knowledge Base Designer позволяет:

уменьшить риск ошибок проектирования (быстрое прототипирование позволяет проверять модели);

восстанавливать элементы БЗ на основе анализа концептуальных моделей;

исключить ошибки программирования, благодаря автоматической генерации кодов.

Необходимо отметить, что концептуальные модели, используемые в качестве основы для формирования БЗ продукционных ЭС, должны содержать описание причинно-следственных отношений. В противном случае результаты анализа моделей (осуществляемого при импорте) будут лишены содержательного смысла. Кроме того, текущая версия редактора поддерживает ограниченный набор основных конструкций языка, достаточный для автоматического синтеза исполнимых и синтаксически корректных программ. В частности, не поддерживаются: глобальные переменные, мультислоты, функции и сложные условия.

В дальнейшем планируется расширить поддержку форматов концептуальных моделей и языков программирования БЗ.

Работа выполнена при частичной финансовой поддержке РФФИ (проекты № 15-07-05641, 16-37-00122).

Литература

1. Гаврилова Т. А., Кудрявцев Д. В., Муромцев Д. И. Инженерия знаний. Модели и методы. – СПб.: Лань, 2016. – 324 с.

2. Частиков А.П., Гаврилова Т.А., Белов Д.Л. Разработка экспертных систем среда CLIPS. СПб: БХВ-Петербург, 2003. – 393 с.

3. Юрин А.Ю. Нотация для проектирования баз знаний продукционных экспертных систем // Объектные системы. – 2016. – №12. – С.48-54.

4. IBM Rational Rose Enterprise. Режим доступа: http://www-03.ibm.com/software/products/ru/enterprise (дата обращения 27.10.2016).

5. Дородных Н.О., Юрин А.Ю. Использование диаграмм классов UML для формирования продукционных баз знаний // Программная инженерия. – 2015. – №4. – С.3-9.

6. Грищенко М.А., Дородных Н.О., Николайчук О.А., Юрин А.Ю. Применение модельно-управляемого подхода для создания продукционных экспертных систем и баз знаний // Искусственный интеллект и принятие решений. – 2016. – №2. – С. 16-29.

7. Юрин А.Ю., Грищенко М.А. Редактор баз знаний в формате CLIPS // Программные продукты и системы, 2012. – №4. – С. 83-87.

8. ClipsWin: CLIPS Rule Based Programming Language. Режим доступа: https://sourceforge.net/p/clipsrules/news/2008/01/clipswin-6241/ (дата обращения 27.10.2016).

10. Джексон П. Введение в экспертные системы. Пер. с англ. – М: Вильямс, 2001. – 624 с.

11. Meditskos G., Bassiliades N. CLIPS-OWL: A framework for providing object-oriented extensional ontology queries in a production rule engine // Data & Knowledge Engineering, Vol. 70, No. 7, 2011. – p. 661-681.

12. MDA Guide rev. 2.0 // OMG Document ormsc/2014-06-01. Режим доступа: http://www.omg.org/mda/specs.htm (дата обращения 27.10.2016).

13. Personal Knowledge Base Designer (PKBD). Режим доступа: http://www.knowledge-core.ru/index.php?p=pkbd (дата обращения 27.10.2016).

Дородных Н.О., Юрин А.Ю. Формирование баз знаний продукционного типа на основе UML-моделей. *В работе рассмотрено программное средство (Personal Knowledge Base Designer, PKBD) автоматизирующее разработку баз знаний продукционного типа. В качестве целевого языка представления знаний используется CLIPS (C Language Integrated Production System). Данное средство позволяет интегрироваться с CASE-средствами IBM Rational Rose и StarUML, в части импорта UML-моделей и генерации на их основе кода баз знаний в формате CLIPS. Для промежуточного представления и хранения извлекаемых логических правил (продукций) используется специальная обобщенная продукционная модель, которая позволяет абстрагироваться от особенностей определенных языков представления знаний. Средство ориентировано на непрограммирующего специалиста и обладает расширяемой архитектурой, за счет подключения программных модулей, в виде динамических библиотек, поддерживающих генерацию кода на других языках представления знаний. Тестирование разработанных баз знаний осуществляется путем подключения и запуска машин вывода. Приведены предварительные результаты тестирования (апробации) программного средства при решении различных учебных задач.*

Ключевые слова: *Программная система, базы знаний, концептуальная модель, генерация кода, CLIPS, UML.*

Dorodnykh N.O., Yurin A.Yu. Building the rule knowledge bases on the basis of UML models. *The paper considers software (Personal Knowledge Base Designer, PKBD) designed for development of rule knowledge bases. The C language production system (CLIPS) was selected as the targeted knowledge representation language. The software is integrated with CASE-tools (IBM Rational Rose and StarUML) in terms of UML-models import and code generation of knowledge bases in the CLIPS format. The special generalized rule-oriented model is used for the intermediate representation and storage of extracted logical rules. This model abstracts the specific features of knowledge representation languages. The software is focused on the non-programming specialists and has an extensible architecture that implemented the ability to connect the program modules (dynamic-link libraries) supporting the code generation in other knowledge representation languages. Testing developed knowledge bases is carried out by connecting and running the rule engines. The preliminary results of software approbation for solving a variety of learning tasks are presented.*

Keywords: *Software, knowledge bases, rules, conceptual model, code generation, CLIPS, UML.*

Статья поступила в редакцию 08.12.2016
Рекомендована к публикации д-ром техн. наук О.А.Николайчуком