Metody i narzędzia wizualnego projektowania reguł decyzyjnych

Krzysztof Kluza[1], Grzegorz J. Nalepa[1,2]

[1] Katedra Automatyki, Akademia Górniczo-Hutnicza.
E-mail: kluza@agh.edu.pl, gjn@agh.edu.pl
[2] Instytut Fizyki, Uniwersytet Jana Kochanowskiego.

Streszczenie. Artykuł omawia wybrane metody modelowania reguł w języku UML (URML, PRR) oraz podejścia adaptujące technologie regułowe do celów biznesowych (*Business Process Modeling* i *Business Rules*). Pokrótce prezentuje projektowanie reguł w projekcie HeKatE (metodologia ARD i XTT). Główna część dotyczy autorskiego podejścia do modelowania rozpatrywanych w HeKatE reguł XTT przy pomocy reprezentacji opartej o język UML.

Słowa kluczowe: systemy regułowe, projektowanie wizualne, UML

1 Wprowadzenie

Regułowe systemy ekspertowe [1,2] stanowią istotną grupę systemów inteligentnych w Sztucznej Inteligencji [3]. Jest tak ponieważ reguły są intuicyjną metodą reprezentacji wiedzy [4], a eksperci z łatwością potrafią sformułować swoją wiedzę w ten sposób. Ponadto, w dłuższym czasie, systemy ekspertowe są tańszym rozwiązaniem, ze względu na cenę wiedzy ekspertów. Cechą systemów ekspertowych jest również to, że przechowywana wiedza jest oddzielona od metod wnioskowania, co daje możliwość stosowania różnych metod w oparciu o tę samą wiedzę ekspercką, a bazę wiedzy można modyfikować nie naruszając przy tym metod wnioskowania.

W praktyce, projektowanie systemów regułowych napotyka na szereg problemów związanych między innymi z akwizycją wiedzy oraz jej wizualną reprezentacją. Inne problemy dotyczą między innymi formalnej analizy takich systemów.

W ostatnich latach można zaobserwować renesans technologii regułowych głównie za sprawą użycia reguł do modelowania logiki biznesowej w inżynierii oprogramowania [5]. Istotne podejścia to między innymi *Business Rules* i *Business Process Modeling*. Podejścia te starają się zaadaptować istniejące metody reprezentacji, np. standaryzowany przez OMG (*Object Management Group*) język UML (*Unified*

Modeling Language) [6], lub też opracować własne. Wśród nowo opracowanych rozwiązań, na uwagę zasługuje notacja do modelowania procesów biznesowych OMG BPMN (*Business Process Modeling Notation*) [7].

Projektowanie baz wiedzy o dużej liczbie reguł jest procesem złożonym. Wizualna reprezentacja takich systemów pozwala poradzić sobie z tą złożonością. Rozpatrywanym w artykule problemem ogólnym jest optymalna (ze względu na proces projektowania) reprezentacja reguł w takich systemach. Natomiast rozpatrywanym problemem szczegółowym jest reprezentacja reguł dla metodologii XTT (*EXtended Tabular Trees*) w języku UML. Taka reprezentacja ma posłużyć integracji metod inżynierii wiedzy z metodami inżynierii oprogramowania.

W artykule omówione są praktyczne zagadnienia i problemy związane z użyciem tych metod do projektowanie reguł i procesów biznesowych. Omówione jest również pokrótce projektowanie złożonych systemów regułowych w projekcie HeKatE (*Hybrid Knowledge Engineering Project*). Celem artykułu jest między innymi zaprezentowanie autorskiego podejścia do modelowania rozpatrywanych w HeKatE reguł XTT przy pomocy reprezentacji opartej o język UML, które stanowi rozwiązanie wspomnianego problemu szczegółowego.

2 Modelowanie reguł w UML

2.1 Użycie podstawowych diagramów

Obecnie do modelowania systemów używa się najczęściej notacji graficznych. Językiem służącym do komunikacji pomiędzy analitykami a klientami biznesowymi może być BPMN, natomiast między analitykami a programistami zazwyczaj używany jest UML.

W języku UML wyróżnić można dwie klasy diagramów: statyczne (ukazujące budowę i strukturę systemu np. diagramy klas) oraz dynamiczne (ukazujące zachowanie systemu np. diagramy aktywności). Pomimo dużej złożoności języka UML, żaden z diagramów nie był stworzony z myślą o projektowaniu zorientowanym regułowo.

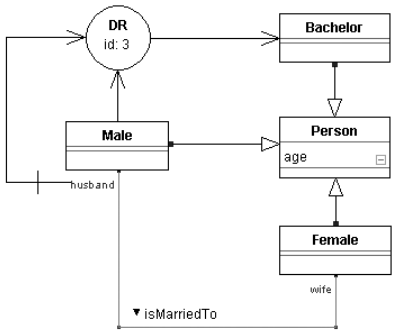
Przy użyciu języka OCL (*Object Constraint Language*) [8], który stanowi dodatek do języka UML, można co prawda sformułować pewne reguły, jednak będą one zapisane w notacji tekstowej bez wizualnej reprezentacji. Tymczasem przy systemach złożonych z wielu reguł, w szczególności posiadających wspólne warunki lub konkluzje, bardziej czytelne wydaje się użycie graficznej notacji ich zapisu.

2.2 URML

URML (*UML-Based Rule Modeling Language*) [9], czyli Ujednolicony Język Modelowania Reguł stworzony przez REWERSE Working Group I1 (*http://rewerse.net/I1*), pozwala na wizualne modelowanie reguł przy użyciu

rozszerzonego diagramu klas UML. Diagram klas poszerzono o graficzny symbol reguły oraz asocjacje z nią związane.

Na Rys. 1 przedstawiono przykładową regułę: *A bachelor is a male that is not a husband* (Kawaler to osoba płci męskiej, która nie jest mężem) [9].



Rysunek 1. Przykład diagramu URML [9]

Graficznie regułę reprezentuje okrąg wraz z identyfikatorem reguły oraz etykietą określającą rodzaj reguły (derywowana, produkcji, reakcji). Strzałki dochodzące do reguły reprezentują warunki (zbiór przesłanek), natomiast wychodzące reprezentują akcje (zestaw konkluzji). Warunki definiują sytuację, w której należy podjąć akcję lub kiedy konkluzja jest spełniona. Ponadto element do którego odnosi się warunek może być ograniczony odpowiednim wyrażeniem (tak by np. w warunku brane były pod uwagę tylko określone jego instancje), a przekreślenie strzałki warunku przy jej początku oznacza negację warunku (wtedy wymagany jest dodatkowy warunek pozytywny, który zawiera elementy, które mogą być objęte tą negacją) [9].

2.3 PRR

Grupa OMG (*Object Management Group*) zaproponowała własny standard reprezentacji reguł produkcji – PRR (*Production Rule Representation*) [10], który daje możliwość reprezentacji reguł produkcji dla modelu UML. Reguła produkcji jest zdefiniowana jako instrukcja logiki programowalnej, która określa wykonanie jednej lub więcej akcji (co skutkuje zmianą stanu systemu), w przypadku gdy zachodzą określone warunki i jest postaci: **if** [condition] **then** [action-list].

Póki co, PRR dostarcza metamodel dla reguł produkcji, nie określa natomiast składni reguł. Specyfikacja z 2007 roku w wersji Beta 1 wprowadza co prawda PRR OCL, czyli bazującą na OCL składnię definiowania reguł, nie czyni jej jednak obligatoryjną, a przykłady podawane są zarówno w PRR OCL, jak i w języku reguł produkcji składnią przypominającym język Java.

2.4 Problemy i ograniczenia

Choć URML daje możliwość zapisu różnych rodzajów reguł, jego wadą jest rozszerzenie składni języka UML. Tym samym zarówno zwiększa on możliwość niespójności języka, jak i sprawia, że zdecydowana większość narzędzi do modelowania w języku UML nie będzie nadawała się do modelowania zorientowanego regułowo w URML.

Brak wspólnego porozumienia co do jednolitej składni PRR może wydawać się mankamentem specyfikacji PRR, jednak ponieważ różne składnie mają wspólny metamodel, a narzędzia powinny umożliwiać serializację do XMI, nie powinno być problemu z translacją reguł produkcji pomiędzy różnymi składniami.

3 Modelowanie procesów biznesowych

3.1 Elementy podejścia

Proces biznesowy to ciąg zadań prowadzący do uzyskania określonych rezultatów, a pojedyncze zadanie ma na celu przekształcenie określonego stanu wejściowego w wyjściowy wg określonej reguły. Istnieje wiele różnych notacji umożliwiających modelowanie procesów biznesowych, najprostsze z nich to schematy blokowe (*Flow Charts*), diagramy przepływu danych (*Data Flow Diagrams*), czy diagram Gantta (*Gantt chart*). Nie są one jednak dedykowane wyłącznie do modelowania procesów biznesowych, przez co są zbyt ogólne i mało precyzyjne, a ponadto nie umożliwiają pokazania powiązań pomiędzy procesami.

3.2 OMG BPMN

Graficzna notacja do modelowania procesów biznesowych BPMN (*Business Process Modeling Notation*) powstała z inicjatywy organizacji BPMI (*Business Process Modeling Initiative*) wyłonionej z OMG. Oferuje ona jeden diagram BPD (*Business Process Diagram*), który ma uwidaczniać logikę biznesową procesu i umożliwiać modelowanie procesów biznesowych o różnym poziomie złożoności.

BPMN posiada skończony i jednoznacznie zdefiniowany zbiór elementów graficznych, takich jak tory i pasy, połączenia (przebiegi procesu, powiązania, przesyłanie wiadomości), artefakty (służące do grupowania lub opisu innych elementów) oraz elementy aktywne: zdarzenia, zadania i bramki logiczne [7].

3.3 Perspektywy rozwoju

Mimo że notacja BPMN posługuje się tylko jednym rodzajem diagramu, dzięki szerokiej gamie elementów aktywnych umożliwia opis nawet skomplikowanych procesów biznesowych, a sformalizowanie notacji pozwala na automatyczne

wygenerowanie kodu BPEL4WS (*Business Process Execution Language for Web Services*) służącego do automatycznego tworzenia aplikacji.

Programy takie jak Corel iGrafx Process pozwalają na symulację przebiegu procesu i wykrycie błędów. Symulacja taka daje możliwość manualnej optymalizacji procesu np. ze względu na czas obsługi, czy koszty. Perspektywiczna jest automatyczna optymalizacja przebiegu procesu, jednak musi ona uwzględnić dodatkowe ograniczenia związane z organizacją firmy, a także z dostępnością zasobów. Niestety po takiej optymalizacji diagram BPMN może nie być czytelny i intuicyjny.

3.4 Ograniczenia metody

Notacja BPMN jest ograniczona do modelowania wyłącznie procesów biznesowych i modeluje ona jedynie przepływ sterowania. Biorąc jednak pod uwagę jej zastosowanie ograniczenie to nie stanowi wady notacji.

W pracy [11] zwraca się także uwagę na to, że na gruncie procesów biznesowych notacja ta nie wspiera opisu dynamicznych grup oraz hierarchii użytkowników.

4 Podejście *Business Rules*

4.1 Koncepcja

Reguły biznesowe (*Business Rules)* definiują lub ograniczają pewne aspekty działań biznesowych. Pojedyncze reguły opisujące te same aspekty zazwyczaj grupuje się w zbiory reguł. Business Rules Group wyróżnia cztery kategorie reguł biznesowych [12]:

- definicje terminów biznesowych (w celu uwspólnienia języka używanego w regułach),
- fakty, które odnoszą się do relacji pomiędzy terminami biznesowymi (mogą być specyfikowane zarówno w języku naturalnym, jak i modelowane graficznie),
- ograniczenia zachowań (służą zarówno do ograniczania aktualizacji danych, jak i nie dopuszczania do wykonania określonej akcji),
- derywowane (określają w jaki sposób wiedza w jednej formie może być przekształcona w inną wiedzę, w innej formie).

4.2 Narzędzia i metody

Niektóre reguły biznesowe mogą być wprost reprezentowane na diagramie UML (np. regułę *każdy uczeń jest przypisany do dokładnie jednej klasy szkolnej* można zaprezentować jako asocjację pomiędzy klasami *uczeń* a *klasa szkolna* z oznaczoną krotnością) lub w prosty sposób wyrażone za pomocą wyrażenia OCL (np. *wiek ucznia nie może być niższy niż 6 lat*).

Istnieje wiele narzędzi związanych z *Business Rules*. W [13] Nalepa dostępne rozwiązania narzędziowe dzieli na kilka kategorii:

- shelle i silniki wnioskujące (*Business Rules Engines*) np. Jess,
- języki oparte na XMLu wspierające reprezentację BR i umożliwiające wymianę pomiędzy różnymi systemami np. RuleML,
- dedykowane metody reprezentacji – narzędzia do wizualnego modelowania reguł biznesowych np. opisany w pkt. 2.2 URML i narzędzie go wspierające – Strelka,
- rozwiązania zintegrowane (Business Rules Management Systems) np. JRules.

4.3 Ograniczenia podejścia

Obecnie reguły biznesowe mogą być wyrażone na różnym poziomie formalizacji. Począwszy od języka nieformalnego, poprzez bardziej rygorystyczny w zastosowaniach technicznych, takich jak np. budowa mostów, kończąc na zdaniach logicznych. Różne podejścia mogą stosować różny poziom, co uniemożliwi translację reguł pomiędzy różnymi narzędziami. Istotne jest zatem dostosowanie zapisu reguł do standardu np. PRR, co umożliwi przetwarzanie regułowe niezależnie od wybranego narzędzia.

Inne ograniczenie narzuca sam proces zbierania reguł. Znaczącą innowacją będzie wprowadzenie automatycznego narzędzia do odkrywania i gromadzenia reguł.

5 Modelowanie reguł w HeKatE

5.1 Proces projektowania

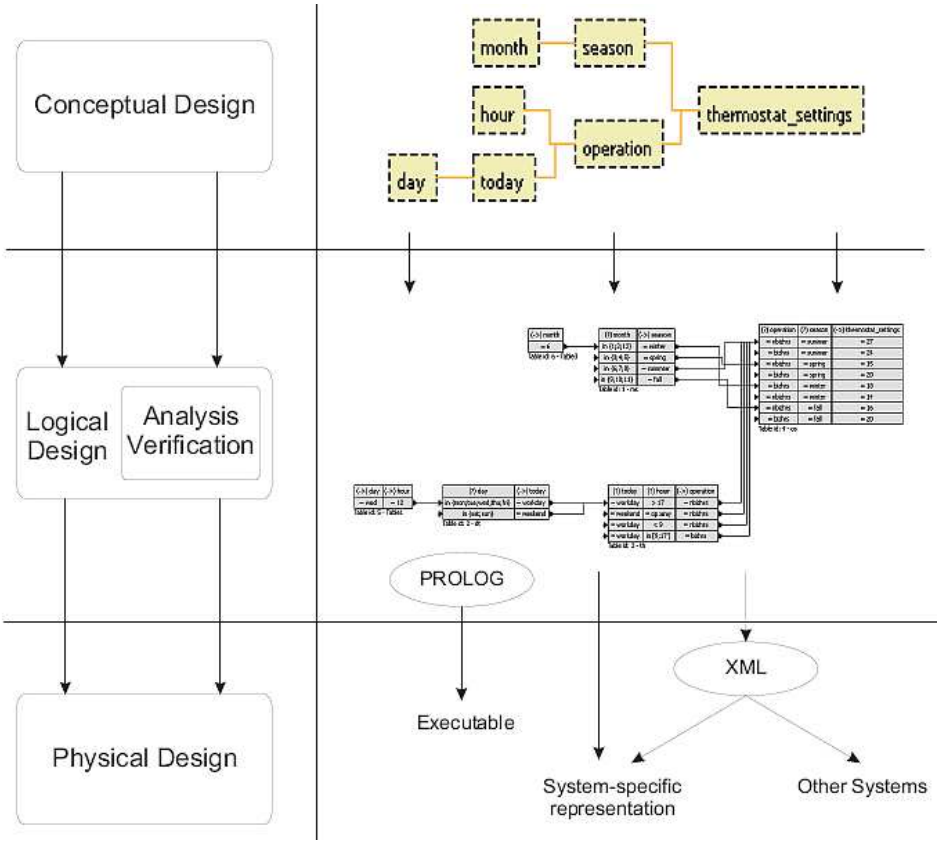
W pracy [14] został przedstawiony proces projektowania XTT (*EXtended Tabular Trees)* obejmujący następujące etapy:

Etap 1. **modelowania konceptualnego** (*conceptual design*), w którym identyfikuje się system atrybutów i ich związki funkcjonalne. W fazie tej korzysta się z metodologii modelowania ARD, która umożliwia określenie funkcjonalnych zależności między atrybutami systemu za pomocą ich graficznych reprezentacji. Przy użyciu tego modelu można zaprojektować logiczną strukturę XTT.

Etap 2. **projektowania logicznego** (logical design), w którym struktura systemu jest reprezentowana jako struktura hierarchiczna XTT. W fazie tej struktura systemu może być poddana analizie, weryfikacji, a także optymalizacji (przy użyciu Prologu).

Etap 3. **projektowania fizycznego** (physical design), w którym dla modelu XTT generowany jest kod np. w Prologu, który może być skompilowany, wykonany i poprawiony.

Rys. 2 przedstawia cały proces projektowania XTT dla problemu termostatu (system ustawiania temperatury dla biura). Szczegółowy opis problemu systemu termostatu można znaleźć w [3].



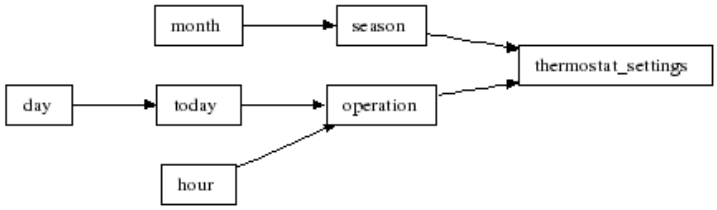
Rysunek 2. Fazy zintegrowanego procesu projektowania XTT [17]

5.2 ARD

Metoda ARD (*Attribute Relationship Diagram*) została zaprezentowana w [15, 16]. Ma ona za zadanie określić funkcjonalne zależności między elementami (własnościami) systemu. Własność jest opisana przy użyciu jednego (prosta) lub więcej atrybutów (złożona). ARD wspiera dwa rodzaje atrybutów: konceptualne (abstrakcyjne, ogólne pojęcie, które następnie jest specyfikowane i uściślane) i fizyczne (konkretne dobrze zdefiniowane właściwości obiektów, atomiczne aspekty systemu) oraz dwa rodzaje transformacji: *finalizacja* i *podział*. Atrybuty konceptualne w trakcie procesu projektowania są *finalizowane* w możliwie największą liczbę atrybutów fizycznych. Natomiast *podział* rozdziela złożone własności systemu na kilka atrybutów i ustala zależności funkcjonalne pomiędzy nimi.

Rys. 3 prezentuje diagram ARD po procesie projektowania – wszystkie atrybuty są atrybutami fizycznymi i zostały między nimi określone zależności funkcjonalne. Jego interpretacja jest następująca: nastawienia termostatu (*thermostat settings*) zależą od

pory roku (*season*) oraz godzin roboczych (*operations hours*). Godziny robocze wyznaczane są na podstawie godziny (*hour*) oraz dnia roboczego (*today*), a dzień roboczy wyznaczany jest na podstawie dnia tygodnia (*day*).

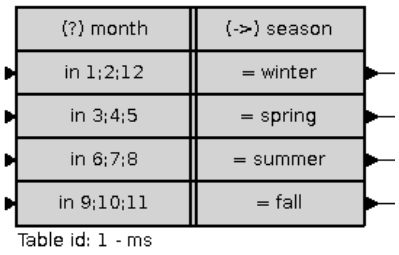


Rysunek 3. Przykładowy diagram ARD [16]

5.3 XTT

XTT [15] jest formalizmem reprezentacji wiedzy regułowej pozwalającym na ustrukturyzowanie bazy reguł poprzez wprowadzenie tabel grupujących reguły posiadające te same atrybuty oraz łączy pomiędzy tabelami, pozwalających na sterowanie wnioskowaniem między tabelami.

Na Rys. 4 pokazano przykładową tabelę XTT, natomiast Rys. 2 w środkowej części przedstawia grupę połączonych tabel.



Rysunek 4. Przykładowa tablica XTT [17]

5.4 Narzędzia

Narzędziem CASE opracowanym na potrzeby metodologii XTT jest edytor HQEd (*Hekate Qt Editor*) [17], który wspiera bazujące na ARD i XTT metody graficznego projektowania, a także umożliwia weryfikację formalnych właściwości projektowanego systemu regułowego. Ponieważ specyfikacja logiczna jest bezpośrednio tłumaczona do reprezentacji bazującej na języku Prolog, zapewnia to możliwość uruchomienia prototypu.

6 Propozycja reprezentacji UML dla XTT

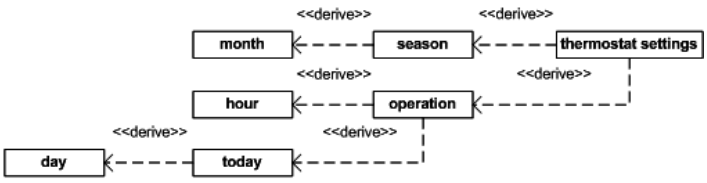
6.1 Koncepcja translacji

Choć zarówno ARD, jak i XTT posiadają swoje graficzne notacje, są one jednak wspierane jedynie przez dedykowane narzędzie HQEd. Obecnie najpopularniejszym językiem modelowania systemów jest UML, stąd inicjatywa, by zaproponować reprezentację ARD i XTT w języku UML [18]. W ten sposób dowolne narzędzie wspierające język UML i umożliwiające serializację do XMI (*XML Metadata Interchange*), będzie mogło wspierać model ARD i XTT.

Następnie arkusz XSLT (*XSL Transformations*), realizujący algorytm translacji UML do XTT, przekształci wygenerowany kod XMI do kodu HML (*Hekate Markup Language*) , który może być wykonany przy użyciu parsera HML w Prologu.

6.2 Reprezentacja ARD

Kluczowym elementem dla reprezentacji ARD na diagramie UML było określenie rodzaju zależności zachodzącej pomiędzy elementami diagramu. Stereotyp *«derive»* odnoszący się do zależności, określa związek pochodzenia między elementami. Związek ten specyfikuje, że element może być wyznaczony (obliczony) na podstawie innego elementu. W ten sposób reprezentacja ARD na Rys. 5 wydaje się naturalna w stosunku do diagramu ARD (na Rys. 3).



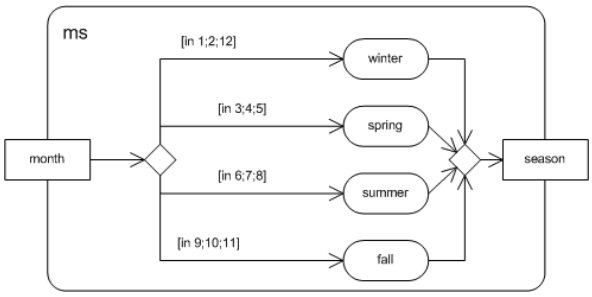
Rysunek 5. Przykładowy diagram UML dla diagramu ARD

6.3 Reprezentacja XTT

Co prawda żaden z diagramów UML nie oddaje w pełni przetwarzania regułowego, jednak w związku z tym, że reguły w pojedynczej tabeli XTT wykonywane są sekwencyjnie, rozsądnym pomysłem było użycie diagramów UML ukazujących nie strukturę systemu, ale jego zachowanie (dynamikę), jak diagramy aktywności.

Po translacji tabeli XTT na diagram aktywności UML atrybuty tabeli XTT reprezentowane są w postaci parametrów aktywności, natomiast akcje nadające wartość atrybutom wyjściowym tabeli XTT reprezentowane są przez akcje na diagramie UML. Przepływy z warunkami dozoru prowadzone od parametrów wejściowych do akcji reprezentują część wejściową tabeli XTT, zaś przepływy od akcji do parametrów

wyjściowych część wyjściową tabeli XTT. Przykładowy diagram UML dla tablicy XTT przedstawionej na Rys. 4 ukazuje Rys. 6.



Rysunek 6. Przykładowy diagram UML dla tablicy *ms* XTT

6.4 Ewaluacja

Reprezentacja UML dla XTT pokazuje zarówno strukturę zależności prezentowaną w ARD, jak i odzwierciedla proces sprawdzania i przetwarzania reguł w tabelach XTT. W przeciwieństwie języka URML proponowana reprezentacja nie rozszerza notacji języka UML, lecz wykorzystuje jego własną notację.

Niektóre z wykorzystanych elementów języka UML pojawiły się dopiero w wersji UML 2.0 np. parametry aktywności. Niestety ze względu na bogatą strukturę języka większość programów nie implementuje całej dostępnej w specyfikacji struktury języka UML, co uniemożliwia swobodne przenoszenie modeli pomiędzy różnymi programami. Dodatkowo wersje XMI znacząco różnią się między sobą, co utrudnia stworzenie jednego translatora z reprezentacji UML do XTT.

7 Podsumowanie

W artykule omówione zostały dotychczasowe metody modelowania reguł w języku UML, a także podejścia adaptujące technologie regułowe do celów biznesowych, takie jak *Business Process Modeling* i *Business Rules*. Pokrótce omówione zostało projektowanie reguł w projekcie HaKatE.

Istotą artykułu jest zaprezentowanie autorskiego podejścia do modelowania rozpatrywanych w HeKatE reguł XTT przy pomocy reprezentacji opartej o język UML. Proponowany model wykorzystuje zarówno diagramy statyczne (do modelowania struktury zależności pomiędzy atrybutami), jak i dynamiczne (ze względu na sekwencyjne sprawdzanie reguł w tabelach XTT).

Kolejnym rozpoczętym etapem projektu związanego z reprezentacją XTT i ARD w języku UML jest zbudowanie metamodelu dla zaproponowanej reprezentacji, który umożliwiłby weryfikację syntaktyczną modelu XTT i ARD. Następnie zweryfikowane diagramy będą mogły zostać serializowane do XMI i przy pomocy arkusza XSLT

przekształcone do kodu HML, który może być wykonany przy użyciu parsera HML w Prologu.

Wszelkie informacje i narzędzia związane z projektem HeKatE zostały udostępnione na stronie internetowej: *http://hekate.ia.agh.edu.pl/*. Dostępny jest edytor wspierający wizualne projektowanie XTT – HQEd, a także edytory wspierające wizualne projektowanie ARD – Varda i HJEd. Wkrótce zostaną udostępnione translatory XSLT.

Podziękowania

Praca naukowa finansowana ze środków na naukę w latach 2007-2009 jako projekt badawczy HeKatE.

Bibliografia

1. Bubnicki, Z., (1990). *Wstęp do systemów ekspertowych*. PWN, Warszawa.
2. Niederliński, A., (2008). *Rule- and Model-Based Expert Systems*. Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice.
3. Negnevitsky, M., (2002). *Artificial Intelligence. A Guide to Intelligent Systems*. Addison-Wesley.
4. Harmelen, F. van, Lifschitz, V., Porter, B., (2007). *Handbook of Knowledge Representation*. Elsevier Science.
5. Sommerville, I., (2004). *Software Engineering. 7th edn. International Computer Science*. Pearson Education Limited.
6. OMG, (2007). *Unified Modeling Language version 2.1.2. Infrastructure. Specification*, Object Management Group (formal/2007-11-03).
7. OMG, (2006). *Business Process Modeling Notation. Specification*, Object Management Group (dtc/06-02-01).
8. OMG, (2006). *Object Constraint Language version 2.0. Specification*, Object Management Group (formal/06-05-01).
9. REWERSE Working Group I1, (2006). *A UML-Based Rule Modeling Language*, http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=URML.
10. OMG, (2007). *Production Rule Representation version Beta 1. Specification*, Object Management Group (dtc/2007-11-04).
11. Piotrowski, M., (2007). *Business Process Modeling Notation. Notacja modelowania procesów biznesowych. Podstawy*. BTC.
12. BRG, (2000). *Defining Business Rules ~ What Are They Really?*, http://www.businessrulesgroup.org/first_paper/br01c1.htm#s1f.
13. Nalepa, G. J., (2007). Business Rules Design and Refinement using the XTT Approach. Paper presented at the *FLAIRS-20: Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference*. May 7-9, 2007, Key West, Florida, Menlo Park, California.

14. Nalepa, G. J., Wojnicki, I. (2007). A Proposal of Hybrid Knowledge Engineering and Refinement Approach. Paper presented at the *FLAIRS-20: Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference*. May 7-9, 2007, Key West, Florida, Menlo Park, California.
15. Ligęza, A., (2006). *Logical Foundations for Rule-Based Systems*. Berlin, Heidelberg: Springer-Verlag.
16. Nalepa, G. J., Wojnicki, I., (2008). Towards Formalization of ARD+ Conceptual Design and Refinement Method. Paper presented at the *FLAIRS-21: Proceedings of the twenty-first international Florida Artificial Intelligence Research Society Conference*. May 15-17, 2008, Coconut Grove, Florida, Menlo Park, California.
17. Kaczor, K., Nalepa, G. J., (2008). *Design and Implementation of HQED, the Visual Editor for the XTT+ Rule Design Method* (No. CSLTR02/2008). AGH University of Science and Technology.
18. Nalepa, G. J., Kluza, K. (2008). UML representation proposal for XTT rule design method. Paper presented at the *4th Workshop on Knowledge Engineering and Software Engineering (KESE2008)* at the 32nd German conference on Artificial Intelligence. September 23, 2008, Kaiserslautern, Germany.