# HQEd - wizualne narzędzie wspierające projektowanie systemów ekspertowych opartych o reprezentację XTT

Krzysztof Kaczor [1], Grzegorz J. Nalepa [1]

[1] Katedra Automatki, Akademia Górniczo-Hutnicza.
E-mail: kk@agh.edu.pl, gjn@agh.edu.pl

**Streszczenie.** XTT jest metodą wspierającą projektowanie systemów ekspertowych opartych na regułach. Model logiki działania systemu zaprojektowany z wykorzystaniem XTT nie jest płaski, lecz posiada swoją strukturę co powoduje, że jest bardziej czytelny i zrozumiały. W związku z tym strukturalizacja, którą wspiera metoda, czyni z niej bardzo wydajne narzędzie do projektowania dużych systemów. Omawiana metoda umożliwia kontrolę poprawności i jakości tworzonego systemu, efektem czego model jest logicznie pełny, jednoznaczny i spójny. HQEd jest narzędziem projektowym umożliwiającym praktyczne zastosowanie XTT do tworzenia systemów regułowych. W narzędziu położono nacisk na możliwość przyrostowego tworzenia modelu reguł, z równoczesnym zachowaniem jakości modelu. Poniższy artykuł przedstawia możliwości edytora, oraz na konkretnym przykładzie systemu regułowego prezentuje przebieg procesu projektowania.

**Słowa kluczowe:** projektowanie, systemy ekspertowe, XTT

## 1 Wprowadzenie

Sztuczna Inteligencja [1] rozwija metody konstruowania systemów inteligentnych. Systemy takie opierają się najczęściej na deklaratywnych metodach reprezentacji i przetwarzania wiedzy [2]. Jedną z metod reprezentacji są reguły decyzyjne. Systemy oparte o tą metodę nazywamy regułowymi systemami ekspertowymi [3].

Regułowe systemy ekspertowe są bardzo popularne ze względu na prosty i intuicyjny sposób reprezentacji wiedzy. Obecnie, systemy ekspertowe znalazły praktyczne zastosowanie w wielu dziedzinach. Rozwiązywane przez nie problemy są często bardzo złożone ze względu na ilość potrzebnej wiedzy i skomplikowane zależności pomiędzy reprezentowanymi faktami. Projekt i implementacja systemu regułowego, który będzie wspomagał rozwiązywanie takiego problemu, są bardzo trudne ponieważ, w przypadku złożonej dziedziny problemu, rozmiar bazy wiedzy jest bardzo duży. Zbyt duża liczba reguł sprawia, że tworzony model systemu staje się mało czytelny. To z kolei, sprzyja powstawaniu trudnych do wykrycia i naprawy błędów. Obecne narzędzia implementacyjne nie umożliwiają hierarchizacji wprowadzanych reguł. Tworzony w nich model bazy wiedzy jest płaski, a co za tym idzie trudny w zarządzaniu. Często nie posiadają one też mechanizmów kontroli poprawności i jakości tworzonego modelu. Wszystko to powoduje, że implementacja dużych systemów regułowych jest trudna i uciążliwa.

Metoda XTT (*eXtended Tabular Trees*) [9] umożliwia wizualizację procesu modelowania systemów regułowych. Poprzez wizualizację oraz dobry opis formalny rozwiązuje powyższe problemy. Reguły są grupowane w tabele decyzyjne i prezentowane w formie graficznej. Taka reprezentacja powoduje powstawanie usystematyzowanego modelu, który jest bardziej czytelny. Dobry opis formalny metody umożliwia pełną formalną analizę modelu, jeszcze w fazie tworzenia. Dzięki temu, naprawa błędów modelu staje się łatwiejsza [8].

HQEd (*Heakte Qt Editor*) [7] jest narzędziem wykorzystującym zalety metody XTT. Narzędzie umożliwia stopniowe tworzenie spójnego modelu XTT, który jest na bieżąco kontrolowany, a wszystkie wykryte błędy są sygnalizowane. Do komunikacji z użytkownikiem wykorzystuje przyjazny graficzny interfejs.

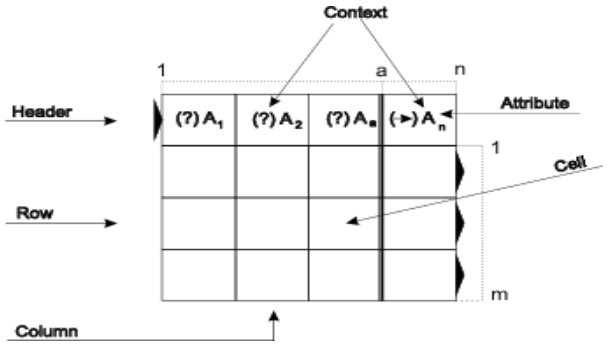
Po krótkim wprowadzeniu do metody XTT w części 2, artykuł przedstawia najważniejsze informacje na temat narzędzia HQEd. W częściach 3 i 4 omówione są zagadnienia projektu i implementacji. Część 5 opisuje najważniejsze funkcje narzędzia, oraz pokazuje ich praktyczne zastosowanie podczas modelowania konkretnego systemu. Artykuł zakończony jest częścią 6, która podsumowuje obecny stan prac nad narzędziem, oraz wskazuje dalszy ich kierunek.

## 2 Metoda XTT

Metoda projektowania XTT (*eXtended Tabular Trees*) [9] jest rozwijana w ramach projektu HeKatE (hekate.ia.agh.edu.pl) realizowanym na Akademii-Górniczo-Hutniczej w Krakowie. Jej główną zaletą jest to, że umożliwia prezentację procesu modelowania w sposób graficzny. Taki sposób reprezentacji wprowadza przejrzystość modelu nawet w przypadku dużej liczby reguł. Co więcej, wymusza usystematyzowanie i ustrukturyzowanie modelu, co w dużym stopniu przyczynia się do zwiększenia jego czytelności. Pozwala to na wyeliminowanie przez projektanta wielu anomalii modelu jeszcze w fazie projektu. Metoda XTT jest sformalizowana, co pozwala na analizę formalną modelu. Model może zostać sprawdzony pod kątem spójności, redukowalności, determinizmu, czy zupełności. Weryfikacja może być prowadzana na bieżąco, co pozwala na szybkie wykrywanie i korekcję błędów [8].

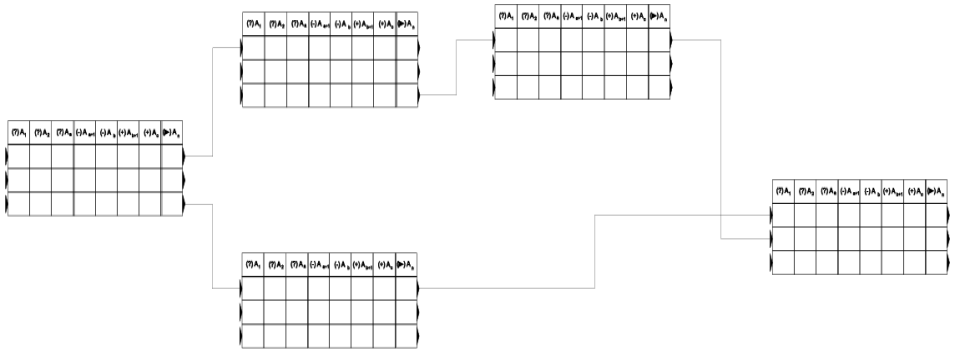
Mechanizm analizy poprawności modelu opiera się na formalnym opisie stanu systemu za pomocą atrybutów [10]. Każdy atrybut posiada swój typ, wartość oraz dziedzinę. Dzięki ścisłej typizacji i dokładnemu określeniu dziedziny atrybutu, możemy określać czy dana wartość atrybutu jest poprawna. Można również sprawdzać stopień pokrycia iloczynu kartezjańskiego dziedzin atrybutów w części warunkowej tabeli. W metodzie XTT jednym z ważniejszych założeń jest to, że przyjmujemy zawsze skończoną dziedzinę atrybutu. Metoda wprowadza pojęcie uogólnionego atrybutu przyjmującego wartości zbiorowe. Proces wnioskowania przy pomocy formuł opartych o atrybuty jest sformalizowany przy pomocy logiki ALSV(FD) [11].

Najważniejszymi elementami reprezentacji graficznej XTT są tabele (Rys. 1) składające się z wierszy reprezentujących pojedyncze reguły.




**Rysunek 1.** Schemat tabeli XTT.

Każdy wiersz (a więc i tabela) składa się z dwóch części, które odpowiadają odpowiednio części warunkowej reguły i części decyzyjnej. Na diagramie XTT separatorem pomiędzy tymi częściami jest podwójna pionowa linia w tabeli. Do każdej kolumny tabeli jest przypisany jeden atrybut określony w nagłówku kolumny. Dzięki temu cała tabela zawiera tylko te reguły które posiadają ten sam prototyp, tzn. posiadają takie same atrybuty w części warunkowej i decyzyjnej. Taka budowa tabeli pozwala na grupowanie "podobnych" reguł, co sprawia że tworzony model jest ustrukturalizowany.



**Rysunek 2.** Schemat drzewa XTT.

Cały diagram XTT stanowi struktura składająca się z tabel i połączeń między nimi (Rys. 2). Połączenia definiują kolejność analizowania tabel i reguł przez mechanizm wnioskujący. Każda tabela jest analizowana od góry do dołu. Napotkana reguła może zostać wykonana jeżeli jej część warunkowa jest spełniona. Następnie mechanizm wnioskujący przechodzi do kolejnej reguły w tabeli, lub podąża za połączeniem do kolejnej tabeli. Celem metody jest ułatwienie projektowania systemów regułowych poprzez graficzną reprezentację wiedzy oraz analizę tworzonego modelu. Kolejnym krokiem projektowania jest wygenerowanie fizycznej implementacji systemu. Jest ona generowana automatycznie na podstawie translacji diagramu XTT do metareprezentacji

w języku Prolog [10]. Wygenerowany kod może być kompilowany i uruchamiany przy pomocy metaintepretera.

Metoda prototypowania ARD [6] umożliwia projektowanie systemu na jeszcze wyższym niż XTT, konceptualnym poziomie. Jest to metoda hierarchiczna, gdzie zaczynając od najbardziej ogólnego opisu systemu, poprzez kolejne etapy, definiujemy coraz dokładniej jakie atrybuty opisują nasz system i jakie są między nimi zależności. Na podstawie otrzymanych zależności można automatycznie wygenerować szkielet diagramu XTT zawierający nagłówki tabel wraz z uzupełnionymi atrybutami i wstępną mapą połączeń między tabelami.

## 3 Projekt narzędzia HQEd

MVC (ang. *Model – View – Controller*) jest wzorcem projektowym używanym w inżynierii oprogramowania. Wprowadza podział aplikacji na trzy niezależne warstwy/komponenty:

- *Warstwa modelu* odpowiada za wewnętrzną prezentację danych przetwarzanych przez aplikację.
- *Warstwa widoku* odpowiada za szeroko rozumianą prezentację danych.
- *Warstwa kontrolera* pośredniczy pomiędzy dwoma pozostałymi warstwami, przekazując komunikaty i zdarzenia przez nie generowane.

Podział aplikacji na wymienione moduły wprowadza wiele ułatwień w jej tworzeniu, rozwoju i utrzymaniu. Warstwy mogą być tworzone równocześnie i niezależnie od innych co przekłada się na skrócenie czasu projektowania. Możliwa jest aktualizacja pojedynczej warstwy bez zbędnej ingerencji w pozostałe.

Narzędzie HQed zostało zaprojektowane z wykorzystaniem MVC. Kod źródłowy programu został podzielony na trzy główne moduły:

- *Model* – odpowiada za wewnętrzną reprezentację modelu ARD i XTT.
- *Widok* – odpowiada za prezentację *modelu* użytkownikowi oraz za cały interfejs komunikacji z aplikacją.
- *Kontroler* – przechwytuje i zarządza zdarzeniami generowanymi przez warstwy *modelu* i *widoku*.

Narzędzie powstało zgodnie z wymogami przewidzianymi dla aplikacji wspierającej projektowanie metodą XTT. Wszystkie wymagania zostały podzielone ze względu na funkcjonalność ogólną aplikacji, względem *modelu*, względem *widoku* oraz względem interfejsu użytkownika.

Zgodnie z wymogami interfejsu użytkownika, aplikacja posiada graficzny interfejs, który został stworzony pod kątem intuicyjności i łatwości obsługi narzędzia. Wszystkie operacje realizowane są przy pomocy specjalnych okien dialogowych. Każde okno zostało wyposażone w komponenty umożliwiające wyświetlanie i wprowadzenie danych. Zastosowanie odpowiednich komponentów ułatwiło ograniczenie "swobody" użytkownika podczas wprowadzania danych. Dane wprowadzone w jednym miejscu wpływają na rodzaj i ilość dostępnych funkcjonalności na kolejnych etapach edycji.

Głównym zadaniem narzędzia jest wizualne wsparcie projektowania. Narzędzie HQEd zapewnia pełne wsparcie dla tworzenia i edycji modelu XTT. Dane

przechowywane jako model są prezentowane użytkownikowi w sposób graficzny. Narzędzie zostało wyposażone w interaktywny sposób prezentacji. Użytkownik może wykonywać szereg operacji za pomocą menu oraz bezpośrednio na diagramie.

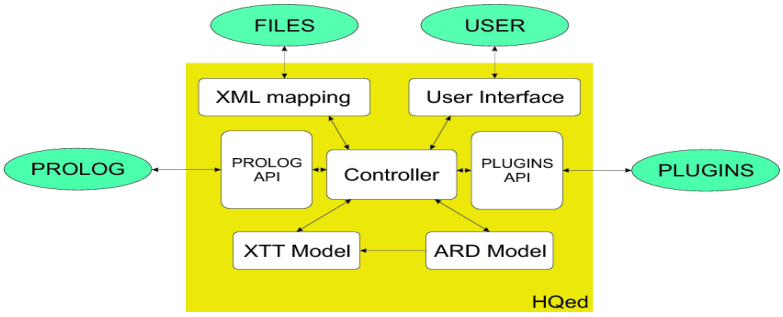
## 4 Architektura HQEd

Kształt architektury narzędzia jest wynikową zastosowania wzorca MVC oraz wymagań postawionych aplikacji. Schemat poglądowy architektury jest pokazany na Rys. 3.

*Controller* jest warstwą centralną, która pośredniczy w wymianie danych pomiędzy pozostałymi warstwami. Jest to warstwa odpowiadająca warstwie *kontrolera* z MVC. Jej zadaniem jest odbieranie zdarzeń generowanych przez pozostałe warstwy i przekazywanie ich dalej. Warstwa *modelu* składa się z dwóch podwarstw odpowiedzialnych za wewnętrzną reprezentację danych modelu ARD – *ARD Model*, oraz modelu XTT – *XTT Model*. Pozostałe warstwy służą jako warstwy prezentujące dane przechowywane w warstwie *modelu* i wchodzą w skład warstwy *widok* z MVC.
Sposób prezentacji danych zależy od warstwy:

- User Interface – odpowiada za graficzną prezentację danych użytkownikowi.
- XML mapping – jest odpowiedzialna za reprezentację plikową. Narzędzie umożliwia zapis i odczyt do/z plików. Dane są zapisywane w formacie HML (*Heakte Markup Language*) będącym językiem bazującym na XML. HML umożliwia przechowywanie zarówno danych o modelu ARD jak również XTT.
- Plugins API – odpowiada za komunikację pomiędzy narzędziem a zewnętrznymi wtyczkami. Komunikacja jest realizowana przy pomocy specjalnego API, które umożliwia odczyt i wykonywanie operacji na modelu programom zewnętrznym.
- Prolog API – wykorzystuje API środowiska SWI-Prolog dla C++ do komunikacji z HeaRT.

Ważnym elementem implementacji narzędzia była jego niezależność od platformy. Stworzenie pojedynczego pliku wykonywalnego, który mógłby być uruchamiany na wszystkich platformach jest skomplikowane. Dużo łatwiejszym rozwiązaniem jest napisanie uniwersalnego kodu, który może być skompilowany dla każdej platformy. Kod źródłowy HQEd został stworzony przy pomocy biblioteki Qt (www.trolltech.com). Biblioteka Qt jest środowiskiem programistycznym dedykowanym dla języka C++. Charakteryzuje się w pełni obiektową architekturą. Jej głównym składnikiem są klasy wspomagające projektowanie graficznego interfejsu użytkownika. Jednak najważniejszym powodem dla którego Qt zostało wykorzystane przy tworzeniu narzędzia jest fakt, że umożliwia tworzenie uniwersalnego kodu, który bez żadnych modyfikacji można kompilować na różnych platformach. Biblioteka umożliwia tworzenie kodu, który jest kompilowalny dla platform X11, Windows, MacOS, oraz tzw. systemów wbudowanych opartych na GNU/Linuxie.




**Rysunek 3.** Architektura narzędzia.

# 5 Modelowanie reguł XTT w HQed

Tworzenie diagramu XTT w HQed polega na definiowaniu typów, atrybutów, tabel, reguł i połączeń. Wszystkie te czynności są konieczne do powstania kompletnego modelu. Modelowanie należy zacząć od czynności podstawowych takich jak definiowanie typów atrybutów a następnie samych atrybutów. Dopiero po ich zdefiniowaniu narzędzie pozwala na tworzenie tabel. Najpierw definiujemy schemat tabeli, czyli określamy atrybuty części warunkowej i części decyzyjnej, a następnie przechodzimy do precyzowania reguł. W XTT reguła jest reprezentowana poprzez wiersz tabeli. Każdy wiersz umożliwia utworzenie połączenia, które reprezentują ścieżkę wnioskowania. Najlepszą praktyką tworzenia diagramu jest wykonywanie operacji w następującej kolejności: definiowanie typów, atrybutów, schematu tabeli, definiowanie wierszy w tabelach (reguł), definiowanie połączeń. Narzędzie częściowo wymusza tę kolejność. Na przykład: nie można zdefiniować tabeli, jeżeli nie ma zdefiniowanego żadnego atrybutu.

Definiowanie atrybutu należy rozpocząć od zdefiniowania dziedziny. Jest to możliwe dzięki oknu dialogowemu *Type Editor*, które umożliwia zdefiniowanie nazwy typu, pozwala na wprowadzenie dodatkowego opisu typu, umożliwia zdefiniowanie typu na którym nowy typ będzie bazował, pozwala na określenie zbioru dozwolonych wartości.

Po zdefiniowaniu typu, można przystąpić do dalszego definiowania atrybutu. Okno edytora atrybutów umożliwia dodanie oraz edycję atrybutu, w tym: umożliwia zdefiniowanie nazwy atrybutu, pozwala na wprowadzenie skróconej nazwy atrybutu oraz opis atrybutu, określa czy dany atrybut jest atrybutem stanu, wejściowym czy wyjściowym, pozwala na określenie typu atrybutu (może istnieć wiele atrybutów o tym samym typie), pozwala zdefiniować wielowartościowość atrybutu, tzn. czy atrybut będzie mógł przyjmować wartość zbiorową czy też nie.

Po zdefiniowaniu wszystkich atrybutów opisujących system, kolejnym krokiem jest zdefiniowanie tabel i reguł. W pierwszej kolejności należy zdefiniować schemat tabeli, czyli określić które atrybuty znajdują się w części warunkowej a które w części decyzyjnej. Do tego celu służy okno dialogowe *Table editor* , które dodatkowo pozwala na zdefiniowanie nazwy, opisu i typu tabeli.

Ostatnim etapem tworzenia tabeli jest zdefiniowanie reguł jako wierszy w tabeli. Klikając prawym przyciskiem myszki na nowo utworzony schemat tabeli i wybierając polecenie *Cell editor* pojawi się okno w którym można określić ilość wierszy tabeli, oraz zdefiniować zawartość każdej komórki. Ostatnim krokiem jest zdefiniowanie połączeń pomiędzy tabelami. Może ono zostać zrealizowane przy pomocy okna dialogowego *Connection editor*. Jednak znacznie szybszym i wygodniejszym sposobem definiowania połączenia jest wykorzystanie techniki *drag&drop*. Każdy wiersz zakończony jest specjalnym znacznikiem, który może zostać złapany, przeciągnięty i upuszczony nad tabelą do której połączenie chcemy zdefiniować.

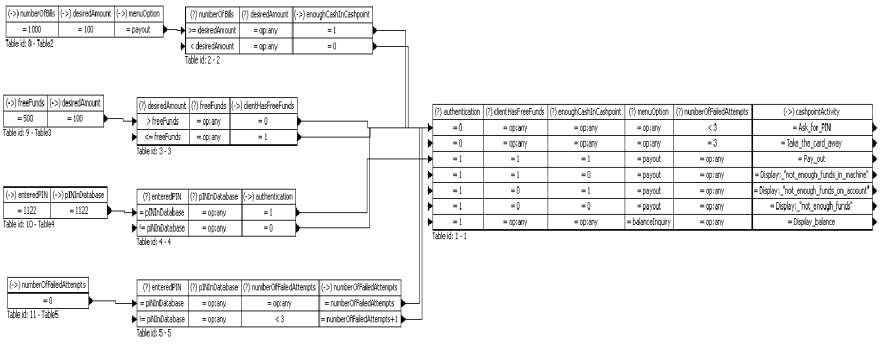
# 6 Przykład modelowania bankomatu

Poniższy przykład pokazuje proces modelowania zachowań bankomatu w typowych przypadkach użycia: wypłata gotówki, zapytanie o saldo konta. Każda z operacji wymaga podania poprawnego kodu PIN. Użytkownik ma trzy próby podania poprawnego kodu. Po trzech nieudanych próbach karta zostaje zwrócona użytkownikowi. W przypadku kiedy użytkownik chce wypłacić gotówkę, bankomat sprawdza czy wprowadzona kwota nie przekracza wolnych funduszy użytkownika, jak również zasobów bankomatu. Atrybutami wejściowymi do systemu są: PIN oraz akcje użytkownika. Wyjściami z systemu mogą być: gotówka, informacje wyświetlane użytkownikowi. Przy pomocy atrybutów definiujemy zbiór reguł rządzących zachowaniem bankomatu:

```
Rule: 1 if numberOfBills is greater or equal to desiredAmount then
enoughCashInCashpoint is true
Rule: 2: if numberOfBills is less than desiredAmount then
enoughCashInCashpoint is false
Rule: 3: if desiredAmount is greater or equal to freeFunds
thenclientHasFreeFunds is true
Rule: 4: if desiredAmount is less than freeFunds then clientHasFreeFunds is
false
Rule: 5: if enteredPIN is pINInDatabase then authentication is true
Rule: 6: if enteredPIN is pINInDatabase then authentication is false and
numberOfFailedAttempts is numberOfFailedAttempts + 1
Rule: 7: ifauthentication is false and numberOfFailedAttempts is less than
3 thenaction: 'ask_for_pin'
Rule: 8: ifauthentication is false and numberOfFailedAttempts is 3
thenaction: 'take_away_card'
Rule: 9: ifauthentication is true and clientHasFreeFunds is true and
enoughCashInCashpoint is true and menuOption is 'payout' thenaction:
'payout'
Rule: 10: ifauthentication is true and clientHasFreeFunds is true and
enoughCashInCashpoint is false and menuOption is 'payout' thenaction: 'not
enough funds in machine'
```

**Rule: 11: ifauthentication is true and clientHasFreeFunds is false and enoughCashInCashpoint is true and menuOption is 'payout' thenaction: 'not enough funds on accound'**
**Rule: 12: if authentication is true and clientHasFreeFunds is false and enoughCashInCashpoint is false and menuOption is 'payout' thenaction: 'not enough funds'**
**Rule: 13: if authentication is true and menuOption is 'balanceInquiry' then action: 'display_balance'**

Aby zamodelować opisany system w HQEd należy najpierw zdefiniować typy atrybutów. Po zdefiniowaniu wszystkich typów można przejść do definiowania atrybutów. Po zakończeniu definiowania atrybutów, można przejść do tworzenia tabel. Edytor kolumn umożliwia zdefiniowanie atrybutu związanego z daną kolumną, oraz określenie w której części tabeli/reguły (warunkowej, decyzyjnej) ma zostać umieszczony. Kolejnym etapem jest dodawanie wierszy i definiowanie komórek. Ostatnim etapem modelowania jest zdefiniowanie połączeń pomiędzy tabelami. Połączenie można utworzyć przeciągając znacznik końca wiersza na tabelę docelową. Podczas wykonywania wszystkich opisanych czynności, narzędzie cały czas monitoruje poprawność powstającego modelu. O wszystkich nieprawidłowościach wykrytych podczas modelowania użytkownik jest natychmiast informowany. Na Rys. 4 pokazany jest w.w. przykład zamodelowany w HQEd.



**Rysunek 4.** Przykład bankomatu w XTT.

# 7 Ewaluacja podejścia

Podejście projektowania w XTT różni się od innych znanych podejść takich jak np. CLIPS. W systemie CLIPS model składa się z reguł, faktów i obiektów. Są to „luźne" elementy modelu w którym nie ma zdefiniowanych zależności pomiędzy nimi. Zdarzenie polegające na dodaniu, usunięciu bądź modyfikacji dowolnego faktu powoduje że system na nowo testuje części warunkowe wszystkich reguł i uruchamia wszystkie te dla których ten warunek jest spełniony. Model XTT składa się z atrybutów

(reprezentujących fakty), reguł, tabel i połączeń pomiędzy nimi. Zmiana wartości dwolnego atrybutu nie determinuje analizy wszystkich reguł, a jedynie ściśle określonych poprzez strukturę modelu (rodział 2).

Edytor HQEd może stanowić samodzielne narzędzie projektowania. Jednak aby w pełni wykorzystać możliwości jakie daje nam metoda XTT, konieczne jest połączenie edytora z serwerem HeaRT (*Heakte RunTime*). HeaRT jest narzędziem rozwijanym równolegle z edytorem w ramach projektu. Jest to serwer logiki napisany w Prologu. Jego zadaniem jest min. wykrywanie anomali, pojawiających się w tworzonym modelu, na poziomie logicznym. Gotowe algorytmy są w stanie sprawdzić model pod kątem redukowalności, determinizmu, czy zupełności. Dwukierunkowa komunikacja serwera z edytorem umożliwia bieżącą analizę modelu w trakcie projektowania, oraz informowanie projektanta o pojawiających się problemach. Sam edytor także przeprowadza analizę, która polega na wykrywaniu błędów syntaktycznych w wyrażeniach w komórkach. Dodatkowo sprawdzany jest także stopień wykorzystania zdefiniowanych atrybutów, osiągalność reguł, poprawne użycie atrybutu w tabeli, oraz gdzie to możliwe, przynależność wartości do dziedziny.

## 8 Przyszłe prace

Głównym tematem artykułu jest zaprezentowanie narzędzia HQed pozwalającego na praktyczne zastosowanie metody XTT. Jest to aktualnie jedyne narzędzie które w tym stopniu wspiera proces projektowania. Został omówiony projekt i implementacja narzędzia oraz zaprezentowany krótki opis funkcjonalności wraz z przykładem modelowania prostego systemu regułowego.

Kolejnym krokiem rozwoju narzędzia jest implementacja interfejsów komunikacji z systemami zewnętrznymi takimi jak Prolog. Do zapewnienia komunikacji z językiem Prolog można wykorzystać API środowiska SWI-Prolog dla C++. Pomimo udostępnionego API podjęto jednak decyzję o wykorzystaniu sieciowego protokołu TCP/IP. Prostota tego rozwiązania powoduje, że z danego interfejsu będzie mogła korzystać znacznie szersza gama systemów, a wykorzystanie będzie znacznie łatwiejsze. Biblioteka Qt posiada szereg klas, które w szerokim zakresie wspierają ten sposób komunikacji. Również środowisko SWI posiada możliwość komunikacji na tym poziomie. Ten sposób komunikacji wprowadza wyraźną separację edytora od serwera logiki HeaRT (rozdział 7).

Dalszy rozwój narzędzia przewiduje wsparcie modelowania na poziomie ARD, oraz integrację modelu ARD i XTT. Docelowo narzędzie powinno tworzyć pełne środowisko wspierające wszystkie etapy projektowania HeKatE.

## Bibliografia


1. Russell S., Norvig P., (2003). *Artificial Intelligence: A Modern Approach. 2nd edition.* Prentice-Hall.
2. Frank van Harmelen, Lifschitz V., Porter B., (2007). *Handbook of Knowledge Representation*. Elsevier Science.
3. Liebowitz J., (1998). *The Handbook of Applied Expert Systems*. CRC Press.
4. Nalepa G. J., (2004). *Meta-Level Approach to Integrated Process of Design and Implementation of Rule-Based Systems*. Ph.D. Dissertation, AGH University of Science and Technology.
5. Negnevitsky M., (2002). *Artificial Intelligence. A Guide to Intelligent Systems*. Addison-Wesley.
6. Ligęza A., (2006). *Logical Foundations for Rule-Based Systems*. Berlin, Heidelberg: Springer-Verlag.
7. Kaczor K., Grzegorz J. Nalepa G. J, (2008). *Design and implementation of hqed, the visual editor for the XTT+ rule design method*. Technical Report CSLTR 02/2008, AGH University of Science and Technology.
8. Ligęza A, Nalepa G. J., (2005). *Visual design and on-line verification of tabular rule-based systems with XTT*. In Klaus P. Jantke, Klaus-Pete Fuhnrich, and Wolfgang S. Wittig, editors, Marktplatz Internet: Von e-a Learning bis e-Payment : 13. Leipziger Informatik-Tage, LIT 2005, Lecture Notes in Informatics (LNI), pages 303 312, Bonn, 2005. Gesellschaft fur Informatik.
9. Nalepa G. J., Ligęza A., (2005). *A graphical tabular model for rule-based logic programming and verification*. Systems Science, 31(2):89-95.
10. Nalepa G. J., Ligęza A., (2006). *Prolog-based analysis of tabular rule-based systems with the XTT approach*. In Geoffrey C. J. Sutcliffe and Randy G. Goebel, eds., FLAIRS 2006: proceedings of the nineteenth in-Interational Florida Artificial Intelligence Research Society conference : [Melbourne Beach, Florida, May 11-13, 2006], FLAIRS Menlo Park, AAAI Press.
11. Nalepa G. J., Ligęza A., (2008). *XTT+ rule design using the alsv(fd)*. In Adrian Giurca, Anastasia Analyti, and Gerd Wagner, editors, ECAI2008: 18th European Conference on Artificial Intelligence: 2nd East European Workshop on Rule-based applications, RuleApps2008: Patras, 22 July 2008, Patras, 2008. University of Patras.