Uniwersytet Ekonomiczny w Katowicach

Studia Ekonomiczne. Zeszyty Naukowe
Uniwersytetu Ekonomicznego w Katowicach
ISSN 2083-8611 Nr 308 · 2016

Informatyka i Ekonometria 8

Vasyl Martsenyuk

Akademia Techniczno-Humanistyczna w Bielsku-Białej
Wydział Budowy Maszyn i Informatyki
Katedra Informatyki i Automatyki
vmartsenyuk@ath.bielsko.pl

Andriy Semenets

Państwowy Uniwersytet Medyczny w Tarnopolu
Katedra Informatyki Medycznej
semteacher@tdmu.edu.ua

SYSTEM ELEKTRONICZNYCH ZAPISÓW MEDYCZNYCH DLA WSPOMAGANIA DECYZJI Z WYKORZYSTANIEM GOOGLE APPLICATION ENGINE (GAE)

Streszczenie: W artykule określono aktualność użycia i alternatywne podejścia dla wdrożenia medycznego systemu informatycznego (MeIS) w dziedzinie opieki zdrowotnej. Omówiono możliwości wdrożenia systemu wspomagania decyzji (DSS) w diagnostyce powikłania ciąży. Przedstawiono wyniki opracowania platformy DSS jako modułu (wtyczki) dla rozpowszechnionego publicznie MeIS o otwartym kodzie OpenEMR. Opracowano model informacyjny bazy danych DSS. Zrealizowano dialogowy komponent DSS z użyciem API zawartym w MeIS OpenEMR. Opracowano moduł administracyjny DSS z użyciem frameworku Yii2 w technologii PHP. Przedstawiono podejścia do realizacji algorytmu procesu podejmowania decyzji w postaci odrębnej usługi środkami Google App Engine.

Słowa kluczowe: system zapisów medycznych, wspomaganie decyzji medycznych, drzewo decyzji, Google Application Engine.

Wprowadzenie

Ważnym problemem w informatyzacji dziedziny opieki zdrowotnej Ukrainy jest szerokie wdrożenie medycznych systemów informatycznych (MeIS – *Medical Information System*). Szybki rozwój w zakresie technologii informacyjnych umożliwia zwiększenie jakości zapewniania opieki zdrowotnej ludności przez zabezpieczenie personelu placówek służby zdrowia technicznymi środkami dla wydajnego przetwarzania informacji klinicznej [Aminpour, 2014]. Wydajność

opieki zdrowotnej istotnie zależy od obecności pełnej i wierzytelnej informacji zarówno o bieżącym stanu zdrowia pacjenta, jak i o przeszłych sytuacjach klinicznych dotyczących pacjenta. Kształtowanie i wsparcie elektronicznych zapisów medycznych (EMR – *Electronic Medical Record*) są jednym z konceptualnych kierunków wdrożenia współczesnych technologii informacyjnych w medycynie [Aminpour, 2014; Fritz, 2015; Global healthcare…].

Rynek światowy MeIS wykazuje dynamiczny wzrost już od wielu lat [Aminpour, 2014; Global healthcare…, Wikipedia; Pahl, 2015]. Jednak wyżej wymienione MeIS bardzo często są komercyjnym oprogramowaniem (np. eClinicalWorks, PlatinumEMR, MedicsDocAssistant EHR, Kareo EHR tp.) z wyraźnymi wadami, wśród których wymienia się:

– wysoki koszt, który z kolei wpływa na cenę usług medycznych;
– zamknięty kod źródłowy, będący własnością kompanii deweloperskiej;
– zależność od kompanii deweloperskiej w przypadku pytań serwisowej obsługi MeIS.

Równolegle z rynkiem komercyjnego oprogramowania aktywnie rozwija się branża aplikacji w dziedzinie opieki zdrowotnej na podstawie rozpowszechnianego publicznie oprogramowania o otwartym kodzie [Reynolds, 2011; Aminpour, 2014]. Na przykład Wikipedia przedstawia listę systemów MeIS o otwartym kodzie, zawierającym więcej niż 20 systemów [Wikipedia]. Szeroko stosuje się takie systemy MeIS o otwartym kodzie jako WorldVistA [www 1], OpenEMR [www 2] i OpenMRS [www 3] [Aminpour, 2014; Fritz, 2015]. Pozytywne cechy MeIS o otwartym kodzie to [Wikipedia]:

– bezpłatny dostęp zarówno samego oprogramowania MeIS, jak i dodatkowych komponentów, takich jak oprogramowanie serwera baz danych oraz web serwera;
– szerokie funkcjonalne możliwości, często wcale nie gorsze od komercyjnych MeIS;
– niezależność od architektury – większość MeIS o otwartym kodzie są integrowanymi przez Internet aplikacjami i mogą pracować na dowolnej platformie, włącznie z współczesnymi urządzeniami mobilnymi;
– dostępność kodów źródłowych oraz interfejsów programowych dla tworzenia własnych aplikacji, na przykład aplikacji dla wspomagania decyzji.

Widoczne stają się perspektywy użycia rozpowszechnianych publicznie MeIS w krajach rozwijających się. W szczegółach ta kwestia została omówiona w badaniach F. Aminpoura [2014], F. Fritza [2015, C.J. Reynoldsa [2011] i innych autorów.

Wdrożenie MeIS nie musi ograniczać się tylko do wykorzystania oprogramowania dla elektronicznych zapisów medycznych. Stosowanie w codziennej działalności współczesnego lekarza klinicznych systemów wspomagania decyzji (CDSS – *Clinical Decision Support System*) jest koniecznym warunkiem poprawy jakości dostarczania opieki zdrowotnej. W badaniach [Jaspers, 2011; Brigth, 2012; Roshanov, 2013] potwierdzono pozytywny związek między procesem wdrożenia DSS a poprawą jakości dostawy opieki zdrowotnej. Przewagi stosowania DSS w placówkach służby zdrowia krajów rozwijających się przedstawiono w pracy: [Esmaeilzadeh, 2015]. Konieczność integracji MeIS różnego rodzaju, w pierwszej kolejności MeIS, DSS i MeIS elektronicznych zapisów medycznych, została omówiona w pracy: [Goldspiel, 2014]. Naukowcy reprezentujący Katedrę Informatyki Medycznej Uniwersytetu Medycznego w Tarnopolu w ciągu ostatnich lat opracowują zarówno zasady teoretyczne, jak i środki programowe DSS [Martsenyuk, Andrushchak, Gvozdetska, 2015].

Jednym z możliwych kierunków zastosowania aplikacji DSS jest diagnostyka powikłania ciąży i dzięki temu uniknięcie porodów przedwczesnych. Metody stosowania zarówno MeIS, DSS w szczególności, jak i MeIS w elektronicznych zapisach medycznych w ogóle, w celu wczesnej diagnostyki powikłania ciąży w położnictwie i ginekologii, zanalizowane zostały w pracach wielu autorów [Edelman, 2014; Martirosyan et al., 2014; Pahl, 2015].

Celem niniejszej pracy jest przedstawienie doświadczenia autorów według opracowania modułu (wtyczki) dla rozpowszechnianego publicznie MeIS w elektronicznych zapisach medycznych OpenEMR, wspierającego funkcjonalne możliwości platformy DSS z opracowaniem drzewa decyzji, który został zrealizowany przez specjalnie opracowaną aplikację dla platformy GAE.

1. Realizacja algorytmu procesu podejmowania decyzji

Współdziałanie DSS z usługą GAE Decision Tree jest związane z użyciem dodatkowej biblioteki yii2 – curl [www 4], umożliwiającej stworzenie zapytań HTTPRequest oraz ich przetwarzanie. W metodzie tej otrzymuje się dane z bieżącej ankiety, które zostają przekazane na serwer wykonujący usługę GAE Decision Tree w celu przetwarzania tychże danych i uzyskania wniosku diagnostycznego.

Z punktu widzenia matematyki zadanie indukcji drzewa decyzji formułuje się w taki sposób. Mamy zbiór *D*, który zawiera N zestawów danych edukacyjnych. Ponadto każdy *i*-y zestaw $(A_1^i, A_2^i, ..., A_p^i, C^i)$ zawiera w sobie dane

wejściowe – atrybuty $A_1,...,A_p$ oraz dane wyjściowe – atrybuty klasy C. Atrybuty $A_1,...,A_p$ mogą przyjmować wartości zarówno liczebne, jak i kategorialne. Atrybut klasy C przyjmuje jedno z K wartości dyskretnych: $C \in \{1,...,K\}$. Celem jest prognozowanie za pomocą drzewa decyzji wartości atrybutów klasy C na podstawie wartości atrybutów $A_1,...,A_p$. Ponadto trzeba zmaksymalizować precyzyjność prognozowania atrybuty klasy, tj. $P\{C = c\}$ na węzłach terminalnych dla dowolnego $c \in \{1,...,K\}$. Algorytmy indukcji drzew decyzji automatycznie rozbijają na węzłach wartości liczebnych atrybutów A_i na dwa okresy: $A_i \le x_i$ i $A_i > x_i$, oraz atrybutów kategorialnych A_j na dwa podzbiory: $A_j \in S_j$, $A_j \notin S_j$. Rozbicie liczebnych atrybutów jest oparte z reguły na miarach na podstawie entropii albo w indeksie Gini. Proces rozbicia rekursywnie powtarza się, dopóki nie będzie obserwowało się polepszania precyzji prognozowania. Ostatni krok włącza usuwanie węzłów dla unikania overfitingu modelu. W wyniku musimy otrzymać zbiór reguł, które idą od korzenia do każdego terminalnego węzła, zawierają nierówności dla atrybutów liczebnych i warunki włączenia dla atrybutów kategorialnych.

Celem niniejszej pracy jest opracowanie metody indukcji drzewa decyzji z możliwością realizacji programowej dla klinicznego systemu wspomagania decyzji.

2. Metoda indukcji drzewa decyzji

W omawianej metodzie za podstawę wzięto poniższą rekursywną procedurę pracy [Han, Kamber, 2001].

Generacja *drzewa decyzji*

Wejściowe dane: D – zbiór zestawów edukacyjnych danych $(A_1^i, A_2^i,..., A_p^i, C^i)$.

Wyjściowe dane: drzewo decyzji.

Metoda:

1. Stworzyć węzeł *N*.
2. Jeśli wszystkie zestawy w *D* należą do wspólnej klasy *C*, wtedy zwrócić węzeł N jako liść z nazwą klasy *C*.
3. Jeśli lista atrybutów (a więc i *D*) jest pusta, wtedy zwrócić węzeł *N* jako liść z nazwą najbardziej rozpowszechnionej klasy w *D*.
4. Stosować *Algorytm doboru atrybutu* z listy atrybutów i dla zbioru *D* w celu odzyskania „najlepszego” atrybutu podziału.

5. Wycofać atrybut podziału z listy atrybutów.
6. Dla każdego warunku podziału *j* dla atrybutu podziału rozważyć D_j – zbiór zestawów z *D*, które spełniają warunek podziału *j*.
7. Jeśli D_j jest pusty, wtedy dołączyć do węzła *N* liść pod tytułem najbardziej rozpowszechnionej klasy w D_j, w innym wypadku dołączyć do *N* węzeł, jaki otrzymamy przez rekursywne wywołanie metody *Generacja drzewa decyzji* z danymi wejściowymi D_j oraz listą atrybutów.
8. Koniec pętli kroku 6.
9. Zwrócić węzeł *N*.

Dla *Algorytmu doboru atrybutu* na *j*-m kroku rekursji jest określony taki wskaźnik informacyjny:

$$Gain(A_i) = Info(D_j) - Info_{A_i}(D_j) \tag{1}$$

Oto

$$Info(D_j) = -\sum_{k=1}^{K} p_k^j \log_2(p_k^j) \tag{2}$$

– informacja potrzebna do klasyfikacji zestawu $(A_1, A_2, ..., A_p)$ w D_j,

$$Info_{A_i}(D_j) = \sum_{l=1}^{K_i} \frac{\#(D_j^l)}{\#(D_j)} Info(D_l) \tag{3}$$

– informacja potrzebna do klasyfikacji $(A_1, A_2, ..., A_p)$ w D_j po podziale D_j na podzbiory D_j^l, odpowiednio do wartości atrybutu A_i.

W (2) prawdopodobieństwo tego, że dowolny zestaw z D_j należy do zbioru C_{k,D_j}, ocenia się jak $p_k^j = \frac{\#(C_{k,D_j})}{\#(D_j)}$, gdzie C_{k,D_j} – zbiór zestawów z D_j, dla których atrybut klasy $C = k$. W tym $\#(\bullet)$ oznacza liczbę elementów w zbiorze.

W (3) $\frac{\#(D_j^l)}{\#(D_j)}$ jest ocena prawdopodobieństwa tego, że dowolny zestaw z D_j należy do zbioru D_j^l, gdzie D_j^l – zbiór zestawów z D_j, dla których atrybut $A_i = a_i^l$. W tym atrybut $A_i \in \{a_i^1, a_i^2, ..., a_i^{K_i}\}$.

Wynika z tego, że $Gain(A_i)$ ocenia zmniejszenie informacji koniecznej do klasyfikacji dowolnego zestawu danych w D_j na podstawie wiadomej wartości atrybutu A_i. W takim razie z istniejących atrybutów na każdym węźle drzewa decyzji dla warunku podziału trzeba odbierać atrybut A_{i*} z największą wartością $Gain(A_{i*})$. Wskutek takiego wyboru dla zakończenia procesu klasyfikacji zestawu danych w D_j będzie potrzebne najmniej informacji.

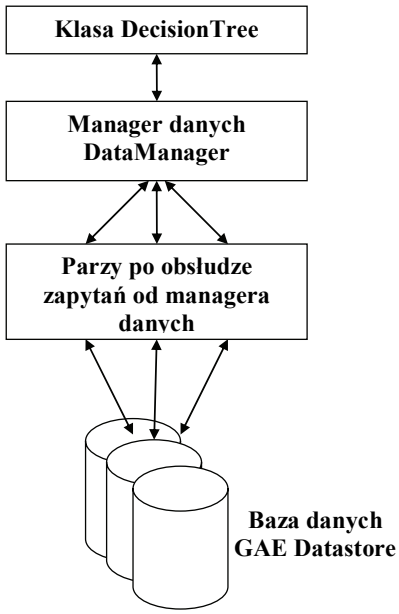
Metodę zrealizowano w postaci usługi GAE w języku programowania Java. Bazę danych edukacyjnych rozwijano w Google Datastore, która jest bazą danych niekorzystającą z SQL. Dla współdziałania z Datastore wykorzystuje się środki dla budowy zapytań w frameworku Objectify. Na rysunku 1 przedstawiono model konceptualny usługi GAE. W klasie DecisionTree bezpośrednio zrealizowano metodę indukcji drzewa decyzji. W klasę DataManager wchodzą wywołania od DecisionTree w celu wykonania zapytań do bazy danych Datastore według otrzymania danych edukacyjnych.

Na rysunku 2 przedstawiono schemat klas dla schroniska GAE Datastore. Baza danych zawiera w sobie klasy encji: Problem, przeznaczony do zachowania informacji o konkretnym zadaniu (na przykład prognozowanie przebiegu porodów); Attribute – do zachowania opisowej informacji o atrybutach zestawów edukacyjnych; Tuple – do zachowania zestawu edukacyjnego; CategorizedData – do wartości atrybutów, włączonych do zestawów edukacyjnych; Class AttributeValue – do wartości atrybutu klasy w zestawie edukacyjnym. Deklaracje klas encji Datastore przedstawiono poniżej.

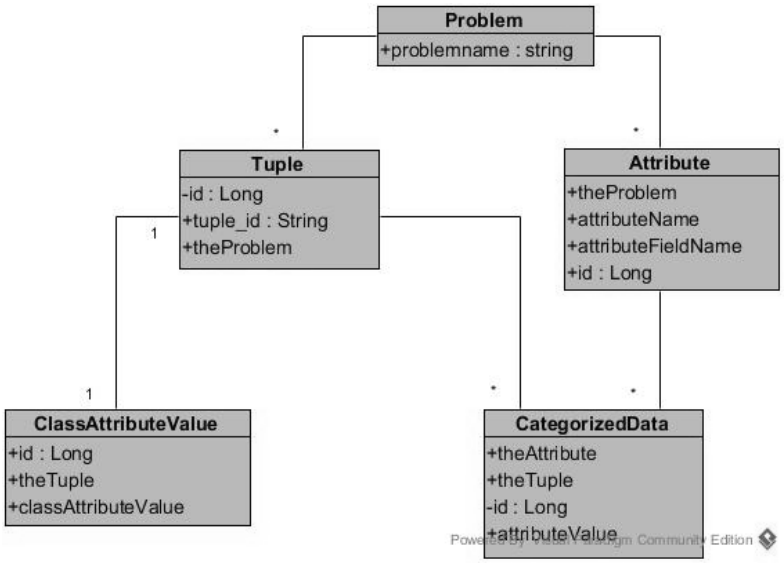
```
@Entity
public class Problem {
 @Id public String problemname;
}

@Entity
public class Attribute {
 @Parent com.googlecode.objectify.Ref<Problem> theProblem;
 @Id public Long id;
 public String attributeName;
 @Index public String attributeFieldName;
 @Index public Date date;
}
```

```
@Entity
public class Tuple {
 @Parent Ref<Problem> theProblem;
 @Index public String tuple_id;
 @Id public Long id;
}

@Entity
public class CategorizedData {
 @Load public Ref<Attribute> theAttribute;
 @Id Long id;
 @Parent public Key<Tuple> theTuple;
 public String attributeValue;
}

@Entity
public class ClassAttributeValue {
 @Id public Long id;
 @Parent public Key<Tuple> theTuple;
 public String classAttributeValue;
}
```

Programowe klasy projektu włączono do pakietu com.google.gwt.decision_tree.model. Wchodzą do niego beans-klasy Attribute, Attribute_for_list oraz CategorisedData do przetwarzania danych odpowiednich tabeli. Zapytanie na podstawie frameworka Objectify w celu otrzymania odpowiednich danych, włączając różne informacyjne wskaźniki, zrealizowano w klasie AttributeListPeer.



Rys. 1. Model konceptualny usługi GAE indukcji drzewa decyzji



Rys. 2. Schemat klas dla zachowania zestawów edukacyjnych w GAE Datastore

Klasa DecisionTree ma dwa elementy klasy: m_dataManager – manager danych oraz m_htAttribute_list – hash-tabela z listą atrybutów. Hash-tabelę z listą atrybutów (w metodach klasy DecisionTree występuje pod nazwą htAttri-

bute_list) tworzy się dla każdego węzła drzewa decyzji. Ma ona dwie funkcje – oprócz listy włączonych dla bieżącego węzła atrybutów może zachowywać warunki podziału (*splitting conditions*), które przeszły do danego węzła od węzłów-rodziców. Każdy węzeł drzewa decyzji jest obiektem klasy TextInBoxWithUser Object. W jakości obiektu każdy węzeł zachowuje obiekt klasy NodeObject, której deklarację zamieszczono poniżej:

```
class NodeObject {
    Attribute attribute;
    Hashtable htAttribute_list;
    String splitting_criterion;
    String sLabel;
    public String toString() {
        if (splitting_criterion.matches("")) { return sLabel; }
        else return "if '" + splitting_criterion + "' then '" + sLabel + "'"; }
    }
```

Attribute oznacza atrybut, który zwraca metoda Attribute_selection_ method, splitting_criterion to warunek podziału, który jest zwracany od rodzicielskiego węzła, sLabel – napis na węźle. Hash-tabela htAttribute_list jest wykorzystana do budowy zestawów danych edukacyjnych D_j dla każdego z węzłów i ma taką strukturę:

Tabela 1. Struktura Hash-tabeli htAttribute_list

Type of key	int
Type of object	Attribute_for_list
Structure of object	Attribute attribute; Hashtable htSplitting_outcomes; String splitting_criterion; boolean included

Tu included jest boolowską zmienną sterującą przynależności atrybutu attribute do listy atrybutów bieżącego węzła. Można stwierdzić, że kiedy included = true, to węzeł z nazwą attribute jest dla bieżącego węzła filialnym (na pewnym niższym poziomie hierarchii). W przypadku, kiedy atrybut attribute nie należy do listy atrybutów dla bieżącego węzła (included = false), węzeł z nazwą attribute jest rodzicielski (na pewnym poziomie hierarchii), a w zmiennej splitting_ criterion zachowuje się warunek podziału, któremu podporządkowuje się dany węzeł pod względem rodzicielskiego węzła attribute.

Hash-tabela htSplitting_outcomes zawiera wszystkie ewentualne skutki (warunki podziału) według atrybutu attribute.

Metoda Generate_decision_tree jest bezpośrednią realizacją metody indukcji drzewa decyzji. Deklaracja metody jest w postaci:

> private void Generate_decision_tree (Hashtable htAttribute_list, TextInBoxWithUserObject tbSubroot, String splitting_criterion, String tree_type).

Jako argumenty metoda ta wykorzystuje główny węzeł drzewa oraz listę związanych z nim atrybutów htAttribute_list i warunek podziału splitting_ criterion. Drzewo decyzji buduje się wskutek rekursywnego wywołania metody Generate_decision_tree.

W celu wizualizacji przedstawienia drzewa wykorzystano pakiet org.abego. treelayout. Rozwiązano problem użycia tego pakietu dla platformy GAE. Ponadto drzewo decyzji wizualizuje się na stronie na podstawie formatu SVG wspieranego większością współczesnych przeglądarek WWW.

3. Realizacja procesu opracowania platformy systemu wspomagania decyzji w ramach MIS OpenEMR

3.1. Środki OpenEMR dla opracowania wtyczek

MeIS OpenEMR zawiera szereg programowych środków, które umożliwiają użytkownikom systemu opracowywać własne wtyczki (pluginy) aplikacji:

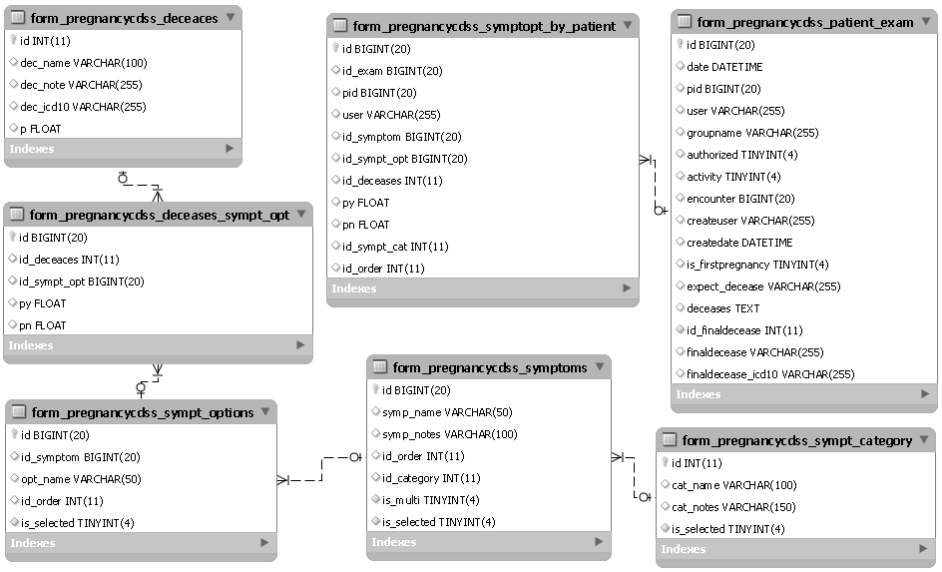
1. Redaktor interaktywny dla stworzenia szablonów form w celu rejestracji danych przeglądu pacjenta – Layout Based Visit Forms [www 5].
2. Redaktor wizualny w celu tworzenia szablonów przetwarzania zapisów medycznych – Nation Notes [www 6].
3. Zestaw interfejsów programowych (API) dla stworzenia użytkowników modułów (pluginów) na języku programowania PHP [www 7].

Środki 1 i 2 są proste w użyciu i nie potrzebują wiedzy z zakresu języków programowania. Jednak nie zawierają one środków koniecznych dla realizacji funkcjonalności DSS. Zatem dla opracowania modułu (wtyczki) realizującej funkcjonalne możliwości platformy DSS autorzy wykorzystali programowy interfejs dla opracowania form (komponentów dialogowych) OpenEMR [www 8]. Ponadto konieczna jest znajomość języków programowania HTML, PHP i SQL.

3.2. Opracowanie informacyjnego modelu wtyczki DSS

Autorzy zastosowali metodę modelowania procesu podejmowania decyzji diagnostycznych na podstawie realizacji algorytmu budowy „drzewa decyzji", z użyciem technologii Data Mining, jak przedstawiono w pracy [Andrushchak, Gvozdetska, Martsenyuk, 2015]. Takie podejście umożliwia zwiększenie prawdopodobieństwa końcowej decyzji diagnostycznej.

Autorzy podjęli decyzję o opracowaniu uniwersalnej wtyczki realizującej funkcjonalne możliwości platformy DSS w zestawie rozpowszechnianego publicznie MIS OpenEMR. Ponadto wewnętrzne przedstawienie modelu informacyjnego było zmodernizowane i adaptowane dla wykorzystania bazy danych MIS OpenEMR sterowanej przez MySQL (rys. 3).



Rys. 3. Struktura tabel bazy danych DSS diagnostyki powikłania ciąży zawartych w OpenEMR (MySQL)

Przetwarzanie danych pacjenta w bazie danych zrealizowano z użyciem podejścia MVC (MVC, *Model – View – Controller*). Ponadto dla realizacji procesu ankietowania pacjenta opracowano takie klasy modelu:

– SymptCategory_Model.class.php – model kategorii symptomów;
– Symptoms_Model.class.php – model zestawu symptomów;
– SymptOptions_Model.class.php – model obecności symptomów;

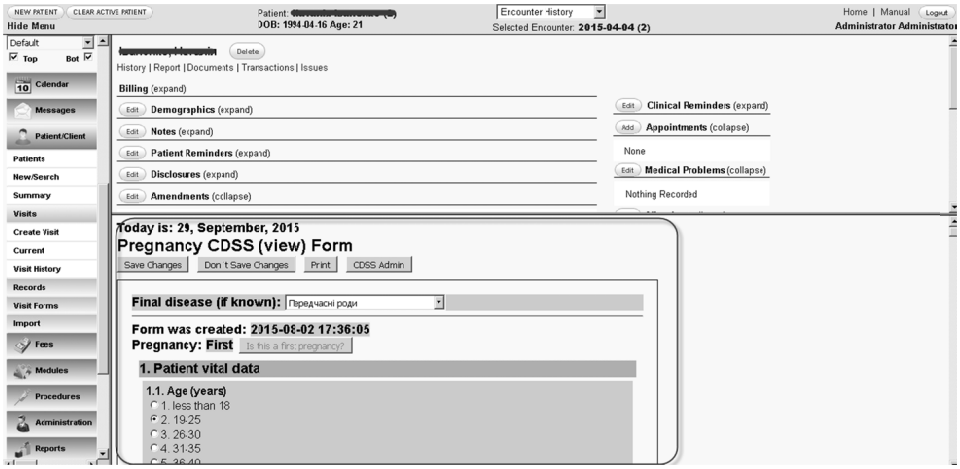
– DeceasesSymptOpt_Model.class.php – model prawdopodobieństwa diagnozy w zależności od obecności symptomu;
– SymptByPatient_Model.class.php – model wyników ankietowania pacjenta.

Według reguł MIS OpenEMR dla opracowania wtyczek użytkowników [www 9] współdziałanie z bazą danych następuje z użyciem biblioteki ADODB [www 10].

3.3. Realizacja komponentu dialogowego wtyczki CDSS

Przy opracowaniu wtyczki użytkowników dla MIS OpenEMR mamy możliwość stosować większość współczesnych podejść do oprogramowania, włącznie z technologiami HTML5, CSS3, AJAX (używając środków JQuery). Dla DSS diagnostyki powikłania ciąży, zawartego w MIS OpenEMR, opracowano główną formę ankiety realizującej komponent dialogowy DSS oraz szereg form-sprawozdań. Wymienione formy, w ramach aplikacji na podstawie podejścia MVC, zrealizowano przez takie przedstawienia:

– SymptByPatient2_Form.html – główna forma ankiety (rys. 4);
– SymptByPatient_Form2Report.class.php – klasa przedstawienia danych DSS diagnostyki powikłania ciąży w formie pacjenta w MIS OpenEMR (rys. 4);
– SymptByPatient_Form2Print.class.php – klasa przedstawienia protokołu ankietowania pacjenta i wniosku diagnostycznego.



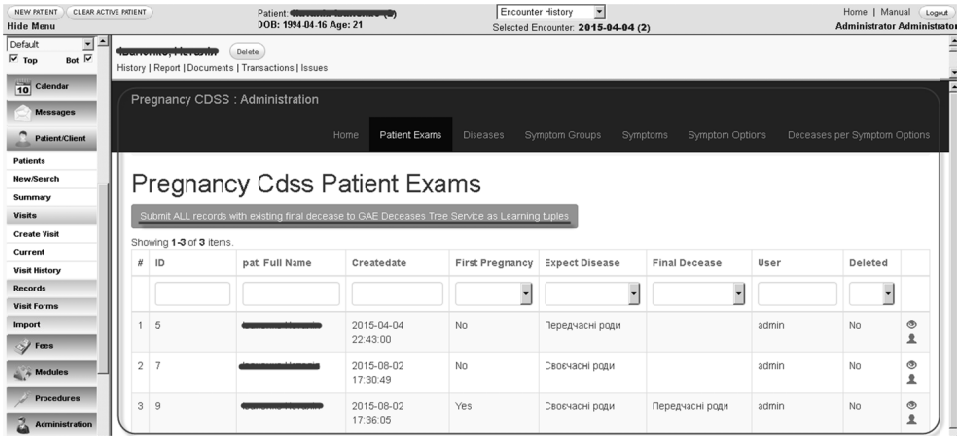
Rys. 4. Komponent dialogowy DSS diagnostyki powikłania ciąży w MeIS OpenEMR

Proces ustawienia formy ankiety i form-sprawozdań w ramach aplikacji na podstawie podejścia MVC dokonano z użyciem kontrolera PatientExam_Form_Controller, realizującego takie metody:

- new_action () i view_action ($form_idexam) – umożliwia stworzenie nowej i przedstawienie istniejącej formy dialogowej ankietowania pacjenta (rys. 3);
- report_action ($form_idexam) – umożliwia przedstawienie danych DSS w formie pacjenta w MIS OpenEMR za pomocą klasy przedstawienia SymptByPatient_Form2Report.class.php;
- print_action ($form_idexam) – umożliwia przedstawienie protokołu ankietowania pacjenta i wniosku diagnostycznego DSS w MIS OpenEMR za pomocą klasy przedstawienia SymptByPatient_Form2Print.class.php.

3.4. Realizacja środków zarządzania platformą DSS

Przy opracowaniu DSS autorzy przewidzieli możliwość jej użycia do rozwiązania rozmaitych zadań diagnostycznych w oparciu o stosowanie algorytmów podejmowania decyzji. W tym celu opracowano specjalny środek zarządzania platformą DSS w postaci osobnego modułu administracyjnego, również zintegrowanego z MeIS OpenEMR (rys. 5).



Rys. 5. Zintegrowany środek zarządzania platformą DSS diagnostyki powikłania ciąży w MeIS OpenEMR

Wymieniony moduł zawiera w sobie środki do:

- zmiany listy możliwych wniosków diagnostycznych;
- redagowania elementów ankiety pacjenta;

– modyfikacji statystycznych parametrów między diagnostycznymi wskaźnikami a wnioskami.

Przy opracowaniu administracyjnego modułu DSS zastosowano Yii 2 – współczesnego frameworka w języku PHP [www 11]. Dla przyśpieszenia procesu opracowania oprogramowania autorzy stosowali środki automatycznej generacji kodu źródłowego w Yii 2 – Gii [www 12]. Ten moduł pozwala istotnie zaoszczędzić czas przy realizacji typowych operacji CRUD (Create/Read/Update/Delete) dla tabel bazy danych. Kod zgenerowany przez Gii odpowiada standardom opracowania w języku PHP, dzięki czemu deweloper może łatwo dodawać konieczny funkcjonał.

Współdziałanie modułu administrowania DSS z usługą GAE Decision Tree zorganizowano również z użyciem wymienionej dodatkowej biblioteki yii2-curl, umożliwiającej stworzenie zapytań HTTPRequest i ich przetwarzanie.

Podsumowanie

Stosowanie w codziennej praktycznej działalności współczesnego lekarza klinicznych DSS jest koniecznym warunkiem zapewnienia jakości dostawy opieki zdrowotnej. Dzięki użyciu podejścia MVC dla opracowania wtyczki DSS i współczesnych frameworków do opracowania odrębnego modułu administrowania wtyczki autorzy otrzymali programową platformę DSS dla MeIS OpenEMR.

Perspektywę przyszłych badań stanowi adaptacja opracowanej platformy DSS w celu rozstrzygnięcia diagnostycznych zadań przez przetwarzanie informacji zachowywanej w MeIS OpenEMR.

Literatura

Aminpour F., Sadoughi F., Ahamdi M. (2014), *Utilization of Open Source Electronic Health Record Around the World: A Systematic Review*, "Journal of Research in Medical sciences – The Official Journal of Isfahan University of Medical Sciences", Vol. 19, No. 1, s. 57-64.

Andrushchak I.Y., Gvozdetska I.S., Martsenyuk V.P. (2015), *Qualitative Analysis of the Antineoplastic Immunity System on the Basis of a Decision Tree*, "Cybernetics and Systems Analysis", Vol. 51, No. 3, s. 461-470, DOI: 10.1007/s10559-015-9737-6.

Bright T.J., Wong A., Dhurjati R. (2012), *Effect of Clinical Decision-Support Systems: A Systematic Review*, "Annals of Internal Medicine", Vol. 157, No. 1, s. 29-43.

Edelman E.A., Lin B.K., Doksum T. (2014), *Evaluation of a Novel Electronic Genetic Screening and Clinical Decision Support Tool in Prenatal Clinical Settings*, "Maternal and Child Health Journal", Vol. 18, No. 5, s. 1233-1245.

Esmaeilzadeh P., Sambasivan M., Kumar N., Nezakati H. (2015), *Adoption of Clinical Decision Support Systems in a Developing Country: Antecedents and Outcomes of Physician's threat to Perceived Professional Autonomy*, "International Journal of Medical Informatics", Vol. 84, Issue 8, s. 548-560.

Fritz F., Tilahun B., Dugas M. (2015), *Success Criteria for Electronic Medical Record Implementations in Low-Resource Settings: A Systematic Review*, "Journal of the American Medical Informatics Association", Vol. 22, No. 2, s. 479-488.

Global healthcare it market analysis and segment forecasts to 2020 – healthcare it industry, outlook, size, application, product, share, growth prospects, key opportunities, dynamics, trends, analysis, healthcare it report – grand view research inc, http://www.grandviewresearch.com/industry-analysis/healthcare-it-market (dostęp: 3.06.2017).

Goldspiel B.R., Flegel W.A., DiPatrizio G. (2014), *Integrating Pharmacogenetic Information and Clinical Decision Support into the Electronic Health Record*, "Journal of the American Medical Informatics Association: JAMIA", Vol. 21, No. 3, s. 522-528.

Han J., Kamber M. (2001), *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco.

Jaspers M.W.M., Smeulers M., Vermeulen H., Peute L.W. (2011), *Effects of Clinical Decision-Support Systems on Practitioner Performance and Patient Outcomes: A Synthesis of High-Quality Systematic Review Findings*, "Journal of the American Medical Informatics Association: JAMIA", Vol. 18, No. 3, s. 327-334.

Martirosyan H., Frize M., Ong D.E., Gilchrist J. (2014), *A Decision-Support System for Expecting Mothers and Obstetricians*, 6th European Conference of the International Federation for Medical and Biological Engineering MBEC 2014, 7-11 September 2014, Dubrovnik, Croatia, Springer International Publishing, Vol. 45, s. 703-706.

Pahl C., Mehrbakhsh N., Zare M. (2015), *Role of Openehr as an Open Source Solution for the Regional Modelling of Patient Data in Obstetrics*, "Journal of Biomedical Informatics", Vol. 55, s. 174-187, https://doi.org/10.1016/j.jbi.2015.04.004.

Reynolds C.J., Wyatt J.C. (2011), *Open Source, Open Standards, and Health Care Information Systems*, "Journal of Medical Internet Research", Vol. 13, No. 1, s. 24.

Roshanov P.S., Fernandes N., Wilczynski J.M. (2013), *Features of Effective Computerised Clinical Decision Support Systems: Meta-Regression of 162 Randomised Trials*, "BMJ (Clinical Research ed.)", Vol. 346, February 14, s. 657.

Wikipedia, *List of Open-Source Healthcare Software*, http://en.wikipedia.org/wiki/List_of_open-source_healthcare_software#Electronic_health_or_medical_record (dostęp: 3.06.2017).

[www 1] http://worldvista.org/ (dostęp: 03.06.2017).

[www 2] http://www.open-emr.org/ (dostęp: 03.06.2017).

[www 3] http://openmrs.org/ (dostęp: 03.06.2017).

[www 4] http://www.yiiframework.com/extension/yii2 (dostęp: 03.06.2017).

[www 5] http://www.open-emr.org/wiki/index.php/LBV Forms (dostęp: 03.06.2017).

[www 6] http://www.open-emr.org/wiki/index.php/Nation Notes (dostęp: 03.06.2017).

[www 7] http://www.open-emr.org/wiki/index.php/The OpenEMR API (dostęp: 03.06.2017).

[www 8] http://www.open-emr.org/wiki/index.php/The Forms API (dostęp: 03.06.2017).

[www 9] http://www.open-emr.org/wiki/index.php/Development Policies (dostęp: 03.06. 2017).

[www 10] http://adodb.sourceforge.net/ (dostęp: 03.06.2017).

[www 11] http://www.yiiframework.com/ (dostęp: 03.06.2017).

[www 12] https://github.com/yiisoft/yii2-gii (dostęp: 03.06.2017).

SYSTEM OF ELECTRONIC MEDICAL RECORDS FOR DECISION SUPPORT USING GOOGLE APPLICATION ENGINE (GAE)

Summary: The article presents an actuality of usage and alternative approaches for application of medical information system (MeIS) in branch of healthcare. Possibilities of usage of decision support system (DSS) at diagnostics of pathology of pregnancy are analyzed. Outcomes of development of DSS platform as plugin for open-source MeIS OpenEMR are presented. Information model for database of DSS is developed. Dialog component of DSS using API of MeIS OpenEMR is implemented. Administration module of DSS is developed with help of PHP-framework Yii2. An approache for implementation of algorithm of decision making process in the form of separate service applying Google App Engine is presented.

Keywords: system of medical records, support of medical decisions, decision tree, Google Application Engine.