Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Adam Fuksa
Nr albumu: 209240

Enlil – zaawansowany edytor tekstów starosumeryjskich

Praca licencjacka
na kierunku INFORMATYKA
w zakresie OPROGRAMOWANIA I METOD INFORMATYKI

Praca wykonana pod kierunkiem
dr Jacka Chrząszcza
Instytut Informatyki

Wrzesień 2005

Oświadczenie kierującego pracą

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem
i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny
z obowiązującymi przepisami.
Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.
Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączona wersją elektroniczną.

Data

Streszczenie

W pracy przedstawiono program Enlil - zaawansowany edytor tekstów wspomagający pracę historyka - badacza starożytnego Sumeru. Program powstał na potrzeby specjalisty z Instytutu Historii Uniwersytetu Warszawskiego dr Marka Stępnia. Autorami programu są: Jakub Bogaczewicz, Konrad Durnoga, Adam Fuksa, Jędrzej Fulara i Rafał Wawrzyńczyk (zespół studentów UW działający w ramach zajęć z Zespołowego Projektu Programistycznego prowadzonych przez dr Jacka Chrząszcza).

Słowa kluczowe

tabliczki sumeryjskie, XML, wyszukiwanie, obiekty syntaktyczne, obiekty językowe, obiekty semantyczne

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna (wg ACM Computing Classification System)

H. Information Systems
H.3 INFORMATION STORAGE AND RETRIEVAL
H.3.1 Content Analysis and Indexing
H.3.3 Information Search and Retrieval

Spis treści

Wprowadzenie 7

1. Od glinianej tabliczki do drzewa XML – zarys problematyki związanej z analizą tekstów starosumeryjskich 9
2. Enlil – charakterystyka programu 11
    2.1 Charakterystyka ogólna 11
    2.2 Dekompozycja logiczna systemu 12
3. Dokumentacja użytkowa. 14
4. Enlil – podział pracy 15
5. Podsumowanie 19

A. Podział na pakiety 20
B. Przykładowy przepływ (otwarcie tabliczki za pośrednictwem drzewa grup) 22
C. Zawartość płyty 24

Bibliografia.....25

Wprowadzenie

Około 4 tysięcy lat temu ludzie z terenów dzisiejszego Iraku dokumentowali różne aspekty swojego życia za pomocą pisma klinowego. Nośnikiem takiej informacji były gliniane tabliczki. Aby stawiać hipotezy na temat cywilizacji starożytnego Sumeru, naukowiec – historyk musi porównywać informacje pochodzące z setek i tysięcy tekstów zapisanych na takich właśnie tabliczkach.

Stosowane przez historyków metody przechowywania transkrypcji starosumeryjskich tekstów (najczęściej w postaci „płaskich" plików tekstowych) w praktyce uniemożliwiały efektywną pracę. Problemem było nie tylko wyszukanie określonego znaku czy ciągu znaków, ale i zatracenie pierwotnej struktury przechowywanego tekstu (w oryginale dzielił się on na tabliczki, warstwy, kolumny i wiersze).

Potrzeba wsparcia pracy historyka napotykającego na tego rodzaju problemy stanowiła bodziec do nawiązania współpracy pomiędzy Wydziałem Matematyki, Informatyki i Mechaniki a Instytutem Historii Uniwersytetu Warszawskiego. Efektem tej współpracy jest między innymi powstanie programu Enlil.

Praca składa się z pięciu rozdziałów. W rozdziale 1 przedstawiono zarys problematyki związanej z gromadzeniem i analizowaniem tekstów starosumeryjskich. Kolejne części pracy poświęcone są programowi Enlil: rozdział 2 jest prezentacją tej aplikacji, rozdział 3 opisuje sposób jej użytkowania, a 4 – wkład autora pracy do projektu. Rozdział 5 zawiera podsumowanie. Rozdziały 1 – 3 oraz 5 są wspólne dla wszystkich członków zespołu.

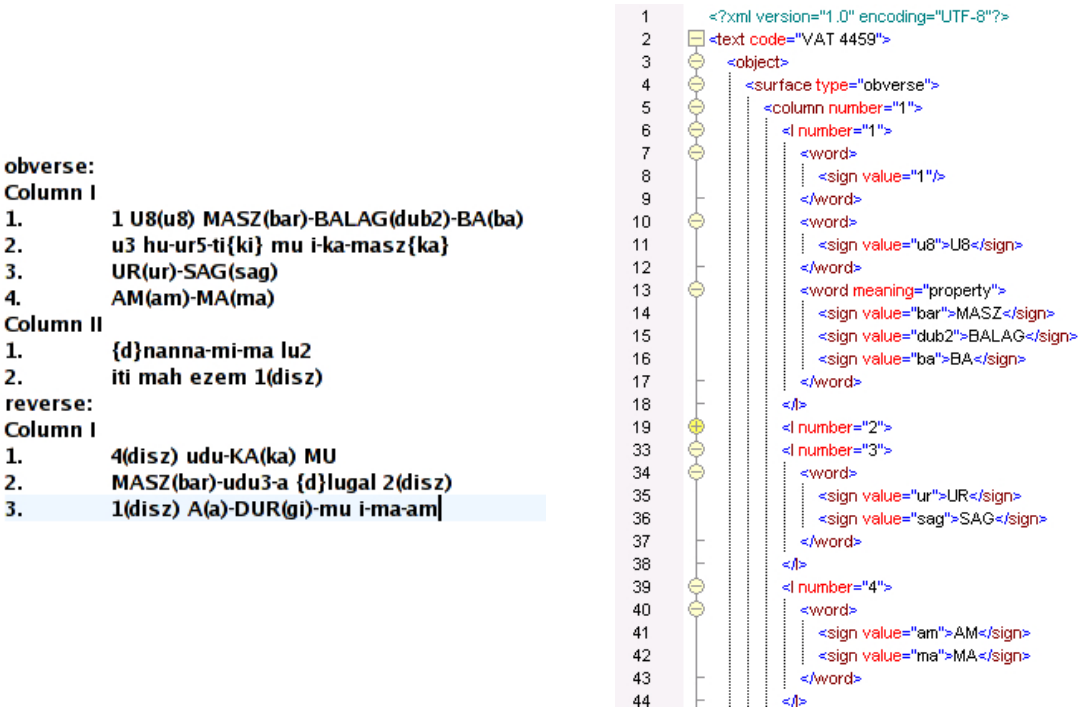
Rozdział 1
Od glinianej tabliczki do drzewa XML – zarys problematyki związanej z analizą tekstów starosumeryjskich

Teksty zapisane na sumeryjskich tabliczkach mają strukturę hierarchiczną: *tekst* może składać się z wielu lub jednej *tabliczki*, ta posiada powierzchnie (w tym boczne), a te z kolei dzielą się na *kolumny*. Kolumna składa się najczęściej z kilku lub kilkunastu *wierszy*; w wierszach zapisywane są *znaki*. Obiekty te określić można mianem *syntaktycznych*. *Obiektami językowymi* nazwiemy ciągi znaków uznane za słowa. Wyróżnienie obiektów językowych jest jednym z kluczowych elementów pracy historyka-sumerologa, prowadzącym do identyfikacji *obiektów semantycznych* – znaczeń słów. Na ich podstawie stawiane są hipotezy badawcze dotyczące życia gospodarczego, wierzeń, poziomu wiedzy, itd., jakimi cechowała się kultura starożytnych Sumerów.

Jak dotąd brak jednolitego standardu kodowania informacji uzyskanych w wyniku transkrypcji i analizy tekstów pochodzących z tabliczek[1]. Powstało wiele różnych sposobów notacji tych samych atrybutów tekstu (np. znaków zatartych, niejednoznacznych, itd.), a same transkrypcje przechowywane są najczęściej w postaci „płaskich" plików tekstowych. Ten sposób archiwizacji danych jest bardzo nieefektywny: utrudnia wyszukanie nie tylko żądanego słowa czy frazy, ale i całej tabliczki (zbiory przechowywane przez historyków liczą po kilka tysięcy tekstów). „Płaska" struktura plików powoduje również przenikanie się informacji z różnych poziomów tekstu; bardziej odpowiednim sposobem zapisu wydaje się być taki, który odzwierciedlałby oryginalną strukturę tekstów np. w postaci drzewa.

Taki właśnie charakter ma format zapisu wykorzystany przez twórców programu Enlil, a mianowicie drzewo XML. Różnicę między oboma sposobami archiwizacji danych przedstawiono obrazowo na rys. 1.

[1] Zaawansowane prace nad przechowywaniem i analizą tekstów sumeryjskich prowadzi m.in. University of Pennsylvania (projekt CDLI).



Rysunek 1. Tekst „płaski” a drzewo XML.

Schemat danych („Schema”) została opracowana przez Jana Posiadałę – studenta wydziału MIMUW pełniącego rolę pośrednika między klientem a wykonawcami aplikacji. Stanowi ona – obok wyników współpracy z zespołem zajmującym się opracowaniem Enlila – część jego pracy magisterskiej. Jan Posiadała, oprócz specyfikacji wymagań klienta, dostarczył autorom systemu tabliczki sumeryjskie w zaproponowanym przez siebie formacie.

Rozdział 2
Enlil - charakterystyka programu

2.1 Charakterystyka ogólna

Aplikację Enlil napisano w języku Java w wersji 1.5. Współpracuje ona z niekomercyjną XML-ową bazą danych eXist. Podczas prac wykorzystywano środowisko programistyczne Eclipse 3.0.1.

Przypadki użycia obsługiwane przez program Enlil można podzielić na następujące grupy:

- przypadki związane z bazą danych tekstów – dodanie, usunięcie lub zmiana nazwy tabliczki, itp.
- operacje edytorskie – edycja tekstu tabliczki, dodanie i usunięcie obiektu syntaktycznego, itp.
- operacje na obiektach językowych – zaznaczenie zbioru obiektów syntaktycznych jako obiektu językowego, odznaczenie tego obiektu, itp.
- operacje na grupach – przypadki użycia dotyczące przechowywania tabliczek w zdefiniowanych przez użytkownika zbiorach (*grupach*), zaprojektowanych na wzór struktury katalogów
- wyszukiwanie – zarówno informacji z poziomu syntaktycznego, jak i semantycznego (tabliczek z danym ciągiem znaków lub ciągiem wartości)
- wizualizacja – przypadki użycia dotyczące sposobu wyświetlania danych zawartych na konkretnej tabliczce (m.in. trzy tryby wyświetlania danych: *tryb syntaktyczny*, umożliwiający edycję obiektów syntaktycznych, tryb *semantyczny*, pozwalający na zarządzanie obiektami językowymi i semantycznymi: wyróżnianie słów, grup znaków

o ustalonym znaczeniu, itp., oraz tryb *podglądu* prezentujący tabliczkę w standardowy „płaski” sposób).

2.2 Dekompozycja logiczna systemu

Enlil składa się z trzech zasadniczych warstw:

- interfejsu
- modelu danych
- bazy danych.

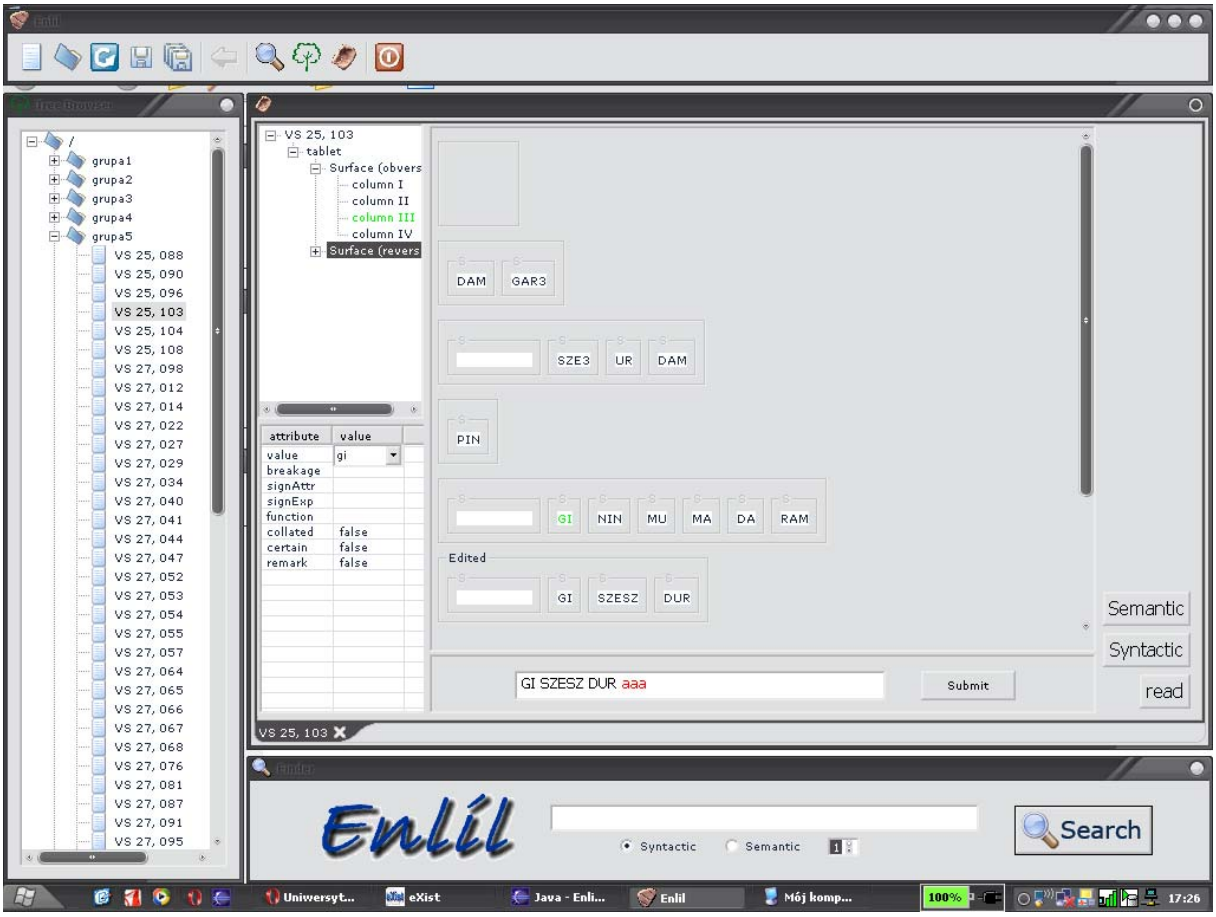
Warstwa interfejsu odpowiada za prezentację wyników pracy programu i umożliwia komunikację z użytkownikiem. Interfejs użytkownika stanowią cztery autonomiczne okna (patrz rys. 2):

- menu
- okno główne, do którego zadań należą:
    - edycja informacji z warstwy syntaktycznej
    - edycja informacji z warstwy semantycznej
    - wyświetlenie „klasycznego” widoku tabliczki
    - prezentacja wyników wyszukiwania
- okno wyszukiwania, pozwalające na wprowadzenie zapytań odpowiadających potrzebom użytkownika
- drzewo grup, umożliwiające sprawną nawigację po całym utrzymywanym zbiorze tekstów.

Interfejs zrealizowano w nowoczesnej technologii SWT (Standard Widget Toolkit), która zapewnia:

- lepszą wydajność niż inne dostępne biblioteki UI
- natywną grafikę danego systemu operacyjnego, dzięki czemu wygląd interfejsu jest bardziej „przyjazny” niż w alternatywnych standardach AWT i Swing.

Warstwa modelu danych zarządza wewnętrzną reprezentacją danych aplikacji. Należą tutaj klasy implementujące strukturę tekstów, mechanizm odpowiadający za zwalnianie pamięci, sylabariusz (mechanizm przypisujący nazwom znaków ich wartości i odwrotnie), itp.



Rysunek 2. Enlil – interfejs użytkownika.

Warstwa bazy danych odpowiada za komunikację aplikacji z bazą danych, parsuje wyniki zapytań do wewnętrznej reprezentacji javowej, itp. Dostęp do XML-owej bazy eXist odbywa się za pośrednictwem zapytań XQuery i XUpdate.

Wykaz pakietów składających się na aplikację Enlil wraz z ich krótkimi opisami zawiera Załącznik A.

Rozdział 3
Dokumentacja użytkowa

Aplikacja Enlil współpracuje ze wszystkimi platformami sprzętowymi i systemami operacyjnymi dla których dostępna jest wirtualna maszyna Javy JSE 5.0.

Aby rozpocząć pracę w systemie Enlil, należy:

- uruchomić bazę danych eXist
- uruchomić dostarczony skrypt o nazwie *Enlil.bat*

W celu otwarcia tabliczki należy dwukrotnie kliknąć jej nazwę w drzewie grup. Otworzona tabliczka pokazuje się na nowej zakładce w głównym oknie programu w trybie podglądu. Celem przejścia do trybu edycji należy wcisnąć przycisk *Edit*. Wszystkie operacje edycyjne wywołuje się z poprzez wybranie odpowiedniej akcji z menu kontekstowego dla zaznaczonego obiektu (grupy obiektów).

Aby wyszukać ciągi znaków lub ich znaczeń należy wpisać je w oknie wyszukiwania,
a następnie wcisnąć przycisk *Search*. Jako wynik wyszukiwania w głównym oknie programu zostanie wyświetlona zakładka z listą tabliczek zawierających poszukiwane obiekty. Kliknięcie w jedną z nazw tabliczek otwiera odpowiednią tabliczkę.

Aplikację Enlil zawiera załączona płyta CD. Szczegółowy opis zawartości płyty zawiera załącznik C.

Rozdział 4
Enlil – podział pracy

Powstanie programu Enlil jest wynikiem wspólnej pracy zespołu Ninkashi, w skład którego wchodzą: Jakub Bogaczewicz, Konrad Durnoga, Adam Fuksa, Jędrzej Fulara i Rafał Wawrzyńczyk (kierownik zespołu). Podział pracy w ramach projektu był uzależniony od funkcji, jakie sprawowali poszczególni członkowie zespołu, a te z kolei – od ich predyspozycji, umiejętności i doświadczenia.

W trakcie pracy nad projektem, do obowiązków i zadań autora niniejszej pracy należały:

- utrzymywanie kontaktu z klientem

  Bezpośrednim klientem dla zespołu Ninkashi był Jan Posiadała -- student wydziału MIM UW, dla którego projekt Enlil był częścią jego pracy magisterskiej. System Enlil służy do pracy z tabliczkami napisanymi w języku sumeryjskim. Nie sposób tworzyć programów operujących na języku sumeryjskim bez choćby pobieżnej znajomości jego struktury, znaczenia itd. Dzięki wcześniejszym kontaktom z innym podobnym językiem aglutynacyjnym -- japońskim, mogłem pomagać innym członkom zespołu, a także klientowi w pełniejszym zrozumieniu i określeniu wymagań stawianych powstającemu systemowi.

- przygotowanie i przeprowadzenie szkolenia z systemu CVS

  System kontroli wersji to narzędzie praktycznie niezbędne, zwłaszcza grupie programistów. W pierwszym semestrze razem z Jakubem Bogacewiczem przygotowałem prezentację i szkolenie na temat podstaw pracy z programem CVS. CVS to klasyczny program, którego nazwa stała się synonimem dla tego typu systemów. Pokazaliśmy udogodnienia oferowanie przez CVS, ale też jego wady. Opowiedzieliśmy także o nowszych odpowiednikach CVSa, takich jak: Subversion, BitKeeper czy Darcs, częściowo pozbawionych wad wysłużonego CVSa. Technologie zbliżone do CVS są wszechobecne we współczesnym przemyśle i nauce, jednak większość naszych słuchaczy nie miała wcześniej z nimi styczności. Dlatego myślę, że nasza prezentacja jako jedna z nielicznych była naprawdę przydatna studentom.

- przygotowanie i przeprowadzenie szkolenia na temat technologii XML oraz XML Schema i XQuery

  XML i związane technologie są stosunkowo młodym i budzącym wiele kontrowersji wytworem przemysłu IT. XML był tematem drugiej prezentacji, którą przygotowałem razem z Jakubem Bogacewiczem. Staraliśmy się pokazać przykładowe zastosowania, wskazać na podobieństwa i różnice między XQuery a SQL. Powiedzieliśmy dlaczego wg. niektórych "XML is a giant step in no direction at all". Ta prezentacja przeznaczona była głównie dla grupy Ninkashi, ale ponieważ XML często wykorzystywany jest do przechowywania, przetwarzania i prezentacji wszelkich danych, mogła być ciekawa dla członków innych grup.

- analiza i dobór technologii używanych do tworzenia aplikacji

  Do zbudowania Enlila zespół wykorzystywał wiele różnych technologii. Część z nich została narzucona: język programowania Java, XML, XMLowa baza danych eXist. Wybór środowiska programistycznego był oczywisty -- Eclipse dzięki kolorowaniu składni, autouzupełnianiu kodu, kompilacji i wskazywaniu błędów jeszcze w trakcie pisania programu oraz integracji z systemem kontroli wersji oraz edytorem graficznego interfejsu użytkownika, pozwolił na znaczne przyśpieszenie prac programistycznych. Do stworzenia graficznego interfejsu użytkownika została wybrana biblioteka SWT. Biblioteki tej używają m.in. Eclipse oraz Azureus. SWT jest bardzo szybkie, używa natywnego środowiska graficznego systemu operacyjnego na którym jest uruchomiona aplikacja. Enlil do manipulowania XMLem używa biblioteki JDOM. JDOM nie jest tak szybki jak parsery SAX, DOM czy PULL, ale w Enlilu został użyty w miejscach gdzie szybkość wykonania nie była krytyczna. Zaletą JDOMa jest bardzo wygodny, obiektowy interfejs do konstruowania dokumentów XMLowych. Kod JDOMowy jest dużo krótszy i bardziej czytelny niż kod napisany przy użyciu niskopoziomowych parserów w stylu SAX.

- stworzenie interfejsu do komunikacji z bazą danych

  Moduł komunikacji z bazą danych został napisany w ten sposób, by w przyszłości możliwa była łatwa zmiana bazy eXist na inną, np. szybszą bazę danych. Moduł oprócz realizowania połączenia z bazą, utrzymywania stanu, wysyłania zapytań i zwracania wyników zapytań opcjonalnie wyświetla informacje diagnostyczne: czas wykonania zapytania, liczba modyfikacji w bazie spowodowanych przez zapytanie oraz nieprzetworzone zapytania i ich wyniki. Informacje te są bardzo pomocne w

strojeniu bazy danych oraz diagnostyce i szukaniu błędów w programie i w bazie danych.

- udział w implementacji i optymalizacji prototypowych zapytań XQuery

  XQuery jest nowym, funkcyjnym językiem zapytań stosowanym w XMLowych bazach danych. Pozwala na proste i eleganckie zapisywanie niektórych typów zapytań, jednak jego implementacja w bazie danych eXist pod względem pełności i szybkości nie zawszy była zadowalająca. Podczas prac implementacyjnych okazało się, że czas wykonania niektórych, teoretycznie bardzo prostych zapytań przez bazę eXist był niedopuszczalnie długi. Znalezienie zapytań, które znajdowały potrzebne Enlilowi informacje, a jednocześnie wykonywały się odpowiednio szybko było zadaniem trudnym, wymagającym wielu eksperymentów, znajomości zasad działania bazy eXist, a w pewnym sensie szczęścia.

- implementacja obsługi zapytań XQuery składowanych w plikach zewnętrznych

  Przechowywanie zapytań w zewnętrznych plikach, a nie bezpośrednio w kodzie Enlila było pomysłem Jana Posiadały. Składowanie zapytań poza kodem pozwala na ich aktualizacje bez potrzeby rekompilacji programu. Umożliwia to znaczące skrócenie czasu procesu tworzenia i dostrajania zapytań. Enlil w czasie działania wczytuje zapytania z plików, uzupełnia odpowiednio oznaczone miejsca w zapytaniach danymi które przekazuje użytkownik w czasie pracy z programem. Przetworzone w ten sposób zapytania przekazywane są bazie danych.

- tworzenie pomocniczych programów do konwersji danych wewnętrznych programu między różnymi formatami

  W trakcie prac implementacyjnych klient kilkukrotnie częściowo zmieniał wymagania, specyfikacje programu oraz format danych (schemę, sposób kodowania tabliczek itp.) wejściowych programu. Często potrzebne były programy, które zamienią pliki w jednym formacie na inny. Do przygotowania tych programów użyłem popularnych języków przystosowanych do pracy na tekstach: perl, sed itp. Języków tych użyłem też do napisania programów generujących duże zbiory tabliczek, które używane były do testowania wydajności bazy danych, kiedy nie były jeszcze dostępne prawdziwe tabliczki.

- instalacja i zarządzanie zespołową listą mailingową oraz repozytorium kodu

Lista mailingowa była głównym sposobem (poza spotkaniami) komunikacji członków zespołu. Własna lista mailingowa pozwoliła na wolną od reklam wymianę dokumentów i myśli. Niestety parokrotne problemy z serwerem na którym działała lista powodowały czasowe nieprzesyłanie listów. Do przechowywania i dystrybucji kodu wykorzystywany był system kontroli wersji Subversion. Subversion określany jest jako CVS zrobiony dobrze. Zespół odkrył, że wtyczka SVN dla Eclipse jest zrobiona nieco gorzej, zdarzały się problemy z synchronizacją kodu z repozytorium.

- testowanie i dbanie o przenośność kodu

  Jednym z wymagań dla Enlila było działanie na wielu systemach operacyjnych. Mimo teoretycznej niewrażliwości Javy na system operacyjny występowały drobne różnice w sposobie wyświetlania niektórych fragmentów interfejsu użytkownika. Częste testy na systemie Linux i Windows pozwoliły znaleźć i usunąć większość usterek.

- współtworzenie dokumentacji projektu

Rozdział 5
Podsumowanie

W pracy przedstawiono program Enlil, przeznaczony dla użytkowników zajmujących się gromadzeniem i analizą danych pochodzących ze starosumeryjskich tabliczek. Archiwizacja danych w wygodnym formacie XML zapewnia zachowanie oryginalnej struktury tekstów oraz efektywną pracę z zbiorem (wyszukiwanie, aktualizacja, itp.). Format ten może stać się podwaliną pod ogólnoświatowy standard zapisu danych będących wynikiem transkrypcji tekstów starożytnych Sumerów. Zastosowane technologie i rozwiązania implementacyjne odpowiadają potrzebom potencjalnego użytkownika aplikacji i są łatwe do dalszej rozbudowy, dzięki czemu stanowią atrakcyjną alternatywę dla dotychczasowych metod pracy historyka – sumerologa.

Załącznik A
Podział na pakiety

enlil – zawiera klasę *Enlil* z metodą *main* oraz klasy odpowiedzialne za wyskopoziomowe zarządzanie oknami i ich współpracę.

attributes – klasy odpowiadające różnym typom atrybutów (*Integer, Boolean, String*), dzięki czemu zmiana *Schemy* na poziomie atrybutów sprowadza się do drobnych korekt kodu (jedna linijka w odpowiednim miejscu).

components – zawiera zdefiniowane przez programistów komponenty graficzne („tabelka" do wyświetlania wyników wyszukiwania).

database – klasy odpowiedzialne za komunikację z bazą danych i dwukierunkową konwersję danych (z XML na format wewnętrzny systemu Enlil i odwrotnie).

disposer – usuwa z pamięci najdłużej nieużywane tabliczki; zaimplementowany jest jako demon uaktywniający się w adaptacyjnie ustalanych odstępach czasu.

enlilExceptions – zawiera definicje wyjątków używanych w programie.

groups – implementacja hierarchii grup.

interfaces – interfejsy użytkownika takich obiektów, jak tabliczka, tekst, kolumna, itp.

lineItems – klasy odpowiadające elementom linii (znak, znak złożony, słowo, itp.).

lineItemsInterfaces – interfejsy graficzne elementów linii.

parsers – zestaw klas służących do parsowania elementów JDom'owych do klas javowych.

searchResults – definicja interfejsów dla wyników wyszukiwania. Celem wprowadzenia tego pakietu jest stworzenie możliwości zdefiniowania sposobu wyświetlania wyników dowolnych zapytań. Dzięki temu znacznie ułatwione jest dodawanie nowych zapytań nawet przez programistę nie zaznajomionego ze strukturą całego programu.

syllabarius – narzędzie ułatwiające użytkownikowi kontrolę poprawności i spójności wprowadzanych danych.

sumerianTextStructure – zawiera definicje obiektów odpowiadających strukturze sumeryjskiego tekstu (tekst, tabliczka, kolumna, linia).

undo – pakiet odpowiedzialny za mechanizm *undo* w systemie, czyli wycofanie zmian wprowadzonych do tabliczki.

utils – zbiór pomocniczych klas wykorzystywanych przez pozostałą część programu (np. klasa *Cast*).

Załącznik B
Przykładowy przepływ (otwarcie tabliczki za pośrednictwem drzewa grup)

Dwukrotne kliknięcie na nazwę tabliczki widniejącą w drzewie grup powoduje wywołanie *listenera*. Pobierany jest deskryptor tekstu przypisanego do tego węzła w drzewie. Za pomocą metody *getSumerianText()* z klasy *SumerianTextDesc* wydobywany jest żądany tekst.

Jeśli wartość pola *sText* w deskryptorze jest równa *null*, to tekst jest sprowadzany z bazy danych (metoda *importSumerianText()* z klasy *Enlil*); wywoływana jest metoda *importSumerianText()* z klasy *DBComm,* a następnie:

- wywołana zostaje metoda *query()*, która wywołuje *doQuery()*
- *doQuery()* wywołuje *queryDB()*, która kieruje zapytanie do bazy danych eXist i oczekuje na wynik
- wynikiem zapytania jest obiekt klasy ResourceSet, który *queryDB()* przekazuje wyżej do *doQuery()*
- na podstawie otrzymanego *ResourceSet doQuery()* konstruuje dokument JDom'owy i przekazuje go do metody *query()*, a ta zwraca go do *importSumerianText()*.

Metoda *importSumerianText()* przekazuje dokument do metody *getTablets()* z klasy *JDomConverter*. Ta ostatnia metoda ma na celu konwersję dokumentu JDom do obiektów z pakietu *SumerianTextStructure* (kolumny, powierzchnie, itp.):

- *getTablets()* generuje na podstawie dokumentu JDom listę tabliczek wchodzących w skład sprowadzonego tekstu
- dla każdej tabliczki wywołuje metodę *getSurfaces()*, która zwraca listę powierzchni danej tabliczki
- *getSurfaces()* wywołuje dla każdej powierzchni *getColumns()*, zwracającą listę kolumn
- *getColumns()* wywołuje dla każdej kolumny metodę *getLines()*, która zwraca listę linii
- *getLines()* wywołuje *getLineItems()*, która dla każdego znaku wywołuje *getLineItem()*

- *getLineItem()* wywołuje parser odpowiedni dla danego obiektu (znaku, słowa, itp.).

Sterowanie wraca następnie po kolei „w górę", aż otrzymamy listę wypełnionych tabliczek, która jest zwracana przez *getTablets()* do *importSumerianText()*. Na podstawie tej listy tworzony jest nowy obiekt klasy *SumerianText* i uzupełniane jest pole *sText* w deskryptorze. Sterowanie powraca do *listenera*.

Dalsza część przepływu dotyczy wizualizacji sprowadzonego tekstu. Wywoływana jest metoda *displayText()* z klasy *Enlil*, która wyświetla tabliczkę w trybie READ_ONLY. *DisplayText()* z klasy *Enlil* wywołuje metodę *displayText()* z klasy *EnlilMainWindow*, a ta - najpierw metodę *startMonitoring()* (z klasy *Enlil*), która inicjalizuje nową pozycję w tablicy haszującej wykorzystywanej przez *undo* (uruchomienie monitorowania tabliczki dla *undo*). Następnie w *displayText()*, jeśli dany tekst jest już otwarty, to jest przesuwany na górę (staje się aktywny); w przeciwnym przypadku tworzony jest nowy interfejs tekstu (*SumerianTextInterface*), w którym wywoływana jest metoda *reader()*, tworząca okno i w przeglądarce wyświetlająca zawartość tabliczki.

Załącznik C
Zawartość płyty

Na załączonej płycie znajdują się:

- aplikacja Enlil wraz ze skryptem instalacyjnym *build.bat*
- plik README.txt z opisem procedury instalacji
- Java Runtime Environment w wersji 5.0 Update 4
- baza danych eXist
- katalog ze źródłami projektu
- katalog z dokumentacją javadoc
- dokumentacja techniczna
- zestaw przykładowych tabliczek
- niniejsza praca

Bibliografia

Technologie

http://java.sun.com

http://xml.org

http://subversion.tigris.org

http://www.exist-db.org

http://www.w3.org/TR/xquery

http://www.eclipse.org/swt

http://www.eclipse.org

http://www.jdom.org

Inne

http://www.urnammu.republika.pl/gilgamesz.htm - epos o Gilgameszu

http://etcsl.orinst.ox.ac.uk/ - zbiór tekstów literatury sumeryjskiej

http://psd.museum.upenn.edu/epsd/index.html - strona projektu słownika sumeryjskiego Uniwersytetu z Pensylwanii