STUDIA INFORMATICA 2005
Volume 26 Number 3 (64)

Marcin GORAWSKI, Przemysław JABŁOŃSKI
Politechnika Śląska, Instytut Informatyki

UNIWERSALNE ŚRODOWISKO GRAFICZNE DO MODELOWANIA PROCESÓW EKSTRAKCJI I ODTWARZANIA

Streszczenie. Bardzo często przeprowadzenie procesu ekstrakcji i ładowania danych na potrzeby konkretnego systemu hurtowni danych wymaga stworzenia specjalizowanej aplikacji ETL [1]. W niniejszej pracy przedstawiono projekt i realizację uniwersalnego środowiska graficznego, pozwalającego na tworzenie procesów ETL, współpracującego ze środowiskiem bazowym udostępniającym zbiór specjalizowanych komponentów do budowy pojedynczego zadania ekstrakcji [2].

Słowa kluczowe: ETL, ETL-DR, modelowanie graficzne ETL

UNIVERSAL GRAPHICAL ENVIRONMENT FOR MODELLING EXTRACTION AND RESUMPTION PROCESSES

Summary. During data extraction optimization it is common to create a specialized ETL application adjusted to a particular data warehouse system [1]. In the following paper we present a project and realization of the universal graphical ETL development environment for modelling ETL processes. The presented environment interoperates with the base environment which provides a set of specialized components [2]. These components enable to define and execute any extraction process.

Keywords: ETL, ETL-DR, graphic modelling ETL

1. Wstęp

Formalnie proces przygotowania, przekształcenia i przenoszenia danych z zasobów pierwotnych do hurtowni danych nazywany procesem ETL (ang. *Extraction, Transformation and Load*) lub potocznie ekstrakcją danych. Ocenia się, że około jedną trzecią budżetu i nakładów pracy związanych z projektem hurtowni pochłania proces ETL [6].

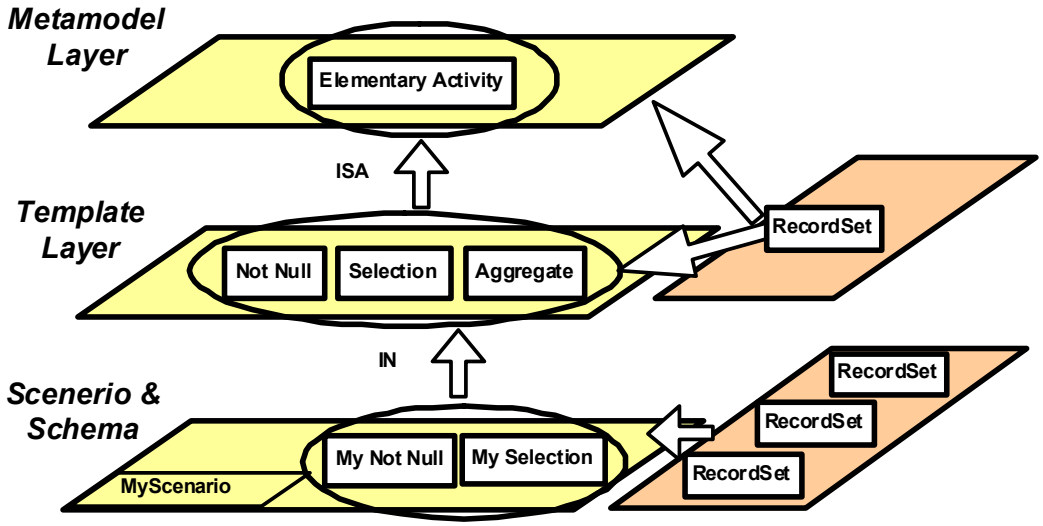
W pracy [2] opisano środowisko udostępniające zbiór specjalizowanych komponentów umożliwiających realizację pojedynczego zadania ekstrakcji. Konfiguracja środowiska odbywa się za pomocą zestawu poleceń zapisanych w tekstowym pliku konfiguracyjnym. W kolejnym etapie należy wygenerować odpowiedni plik *java*, a następnie skompilować go i uruchomić. W praktyce taki sposób pracy nastręcza wiele problemów i jest dość uciążliwy.

Zdecydowano się więc na opracowanie aplikacji udostępniającej graficzne środowisko projektowania procesu ekstrakcji i umożliwiającej współpracę z wymienionym wyżej zestawem komponentów. W trakcie pracy nad aplikacją wykorzystano doświadczenia przedstawione w pracy [3], opisującej środowisko umożliwiające graficzne projektowanie procesu ekstrakcji, którego koncepcja wykorzystuje połączenie modelu pojęciowego [4] procesu ETL z jego logicznym i fizycznym odpowiednikiem.

Całe opisane środowisko powstało w języku Java, dzięki czemu uzyskano możliwość pracy aplikacji na różnych platformach systemowych. Jako środowisko programowania wykorzystano pakiet JBuilder X firmy Borland, który umożliwia bardzo efektywne tworzenie oprogramowania w tym języku.

2. Model konceptualny ekstrakcji danych

Od początku rozwoju hurtowni danych proces ekstrakcji danych nie był traktowany z należytą uwagą. Spowodowało to, iż wiele firm produkujących oprogramowanie wspomagające proces ekstrakcji wypracowało własne rozwiązania, nie czekając na jednolite opisanie tego procesu przez środowisko naukowe. Próbę taką podjęto w pracy [4], gdzie przedstawiono definicję modelu konceptualnego opisującego mechanizmy odkrywania zależności pomiędzy atrybutami a odpowiednimi zadaniami ETL we wczesnych etapach projektowania hurtowni danych. Przedstawiony model konstruowany jest w konfigurowalny i rozszerzalny sposób tak, aby projektant mógł wzbogacić go o własne sekwencje zadań ETL. Rysunek 1 przedstawia schemat modelu konceptualnego, w skład którego wchodzi warstwa metamodelu, warstwa wzorca oraz warstwa użytkowa. Warstwa metamodelu składa się z ogólnych jednostek zdolnych do reprezentowania każdego scenariusza ETL. Warstwa wzorca zawiera zbiór specjalizowanych jednostek, będących uszczegółowieniem ogólnych jednostek metamodelu. Zbiór specjalizowanych jednostek został tak zdefiniowany, aby wspomagać realizację najczęstszych czynności scenariusza ETL. Warstwa użytkowa pozwala natomiast na tworzenie konkretnych scenariuszy składających się z instancji jednostek warstwy wzorca.



Rys. 1. Schemat modelu konceptualnego [5]
Fig. 1. Schema of the conceptual model [5]

Na rysunku 2 przedstawiono symbole graficzne reprezentujące podstawowe elementy modelu konceptualnego zdefiniowane dla warstwy metamodelu. Atrybuty i pojęcia są traktowane jako pierwszoplanowe elementy modelu [4].

Elementy modelu konceptualnego to [4, 3]:

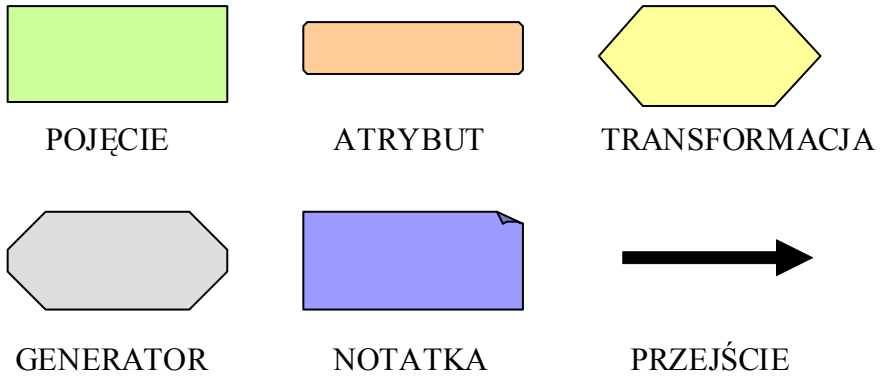
Atrybut – określa najmniejszy, pojedynczy obiekt danych (np. kolumnę danych) poprzez nazwę oraz typ.

Pojęcie – pojęcie reprezentuje składnicę, z której dane są pobierane lub do której dane są zapisywane. Przykładem pojęcia może być plik lub tablica faktów czy też tablice wymiarów w docelowej bazie hurtowni danych. Pojęcie jest formalnie definiowane przez nazwę i określony zbór atrybutów, który reprezentuje.

Transformacja – w modelu transformacje są pewnymi abstrakcjami reprezentującymi część lub całość kodu wykonującego określoną, pojedynczą operację na danych. Wyróżniamy dwie kategorie transformacji. Do pierwszej możemy zaliczyć operacje filtrowania i czyszczenia danych, jak np. kontrola klucza głównego czy kontrola naruszeń klucza obcego. Natomiast druga obejmuje te operacje transformacji, podczas których przekształcany jest schemat danych wejściowych. Formalnie transformacja definiowana jest przez określony zbiór atrybutów wejściowych, określony zbiór atrybutów wyjściowych oraz symbol graficzny charakteryzujący istotę transformacji.

Generator – w definicji modelu komponent ten występuje pod nazwą „ograniczenie ETL”. Generator obejmuje tylko część zadań stawianych przed tym elementem, dlatego też został on wprowadzony jako osobny składnik. Obiekt ten umożliwia projektantowi narzucenie struktury lub wartości danej kolumnie. Może on być używany zarówno do wypełniania

pojedynczą wartością podanego atrybutu, jak i do przydzielania temu atrybutowi unikalnych wartości według określonej zasady generacji.



Rys. 2. Symbole graficzne elementów stosowanych w modelu konceptualnym
Fig. 2. Notation for the conceptual modeling of ETL activities

Notatka – notatki mogą zostać wykorzystane przez projektanta do opisu nieformalnych cech komponentów lub przedstawienia specyficznych wymagań odnośnie do elementu lub grupy elementów w czasie fazy projektowania.

Przejście – w definicji modelu komponent ten występuje pod nazwą „zależność udostępniająca" (ang. *Provider relationships*). Element ten odzwierciedla kolejne etapy transformacji wybranego atrybutu. Przejście zaczyna się zawsze w atrybucie, a kończy albo na wejściu transformacji albo na atrybucie w pojęciu docelowym.

Przedstawiony zbiór ogólnych jednostek metamodelu pozwala na opisanie dowolnego scenariusza ekstrakcji danych. Każdy z zaprezentowanych elementów może zostać zaimplementowany w dowolny sposób, specyficzny dla wymagań konkretnego procesu ETL.

3. Projekt systemu

W pracy [3] przedstawiane środowisko powstało z graficznego projektowania procesu ekstrakcji danych z wykorzystaniem gotowego pakietu komponentów napisanych w języku Java. Wspomniany pakiet udostępnia zestaw komponentów umożliwiających realizację pewnych podzadań składających się na proces ekstrakcji danych. Najogólniej komponenty te możemy podzielić na trzy grupy:

- komponenty ekstrakcji danych – służą do pozyskania danych ze źródła, jakim może być plik tekstowy lub baza danych;
- komponenty transformacji danych – służą do wykonywania operacji na danych;

- komponenty ładowania danych – służą do zapisu wynikowych danych do miejsca docelowego, jakim może być plik tekstowy lub baza danych.

Aby przeprowadzić proces ekstrakcji danych z wykorzystaniem tych komponentów należy przygotować właściwy plik konfiguracyjny, a następnie z wykorzystaniem jednego z dostępnych kompilatorów języka Java uruchomić proces ekstrakcji. Nie jest to najwygodniejszy sposób prowadzenia tego typu działań. Przedstawiane środowisko graficzne ma za zadanie w dużej mierze wyeliminować omówione niedogodności oraz zapewnić:

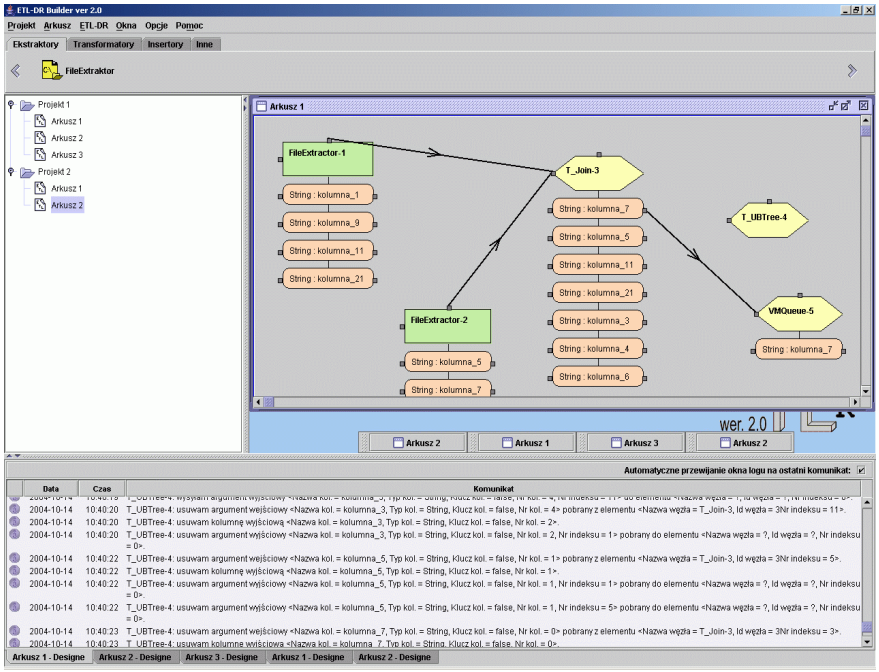
- możliwość graficznego projektowania procesów ekstrakcji z wykorzystaniem gotowych komponentów;
- wsparcie dla graficznego zarządzania właściwościami i parametrami tych komponentów;
- łatwe rozszerzanie o kolejne komponenty;
- testowanie poprawności zdefiniowanych projektów ekstrakcji i automatyczną generację poprawnych plików konfiguracyjnych;
- uruchomienie zdefiniowanych projektów ekstrakcji na podstawie wygenerowanych plików konfiguracyjnych.

4. Program ETL – DR Builder

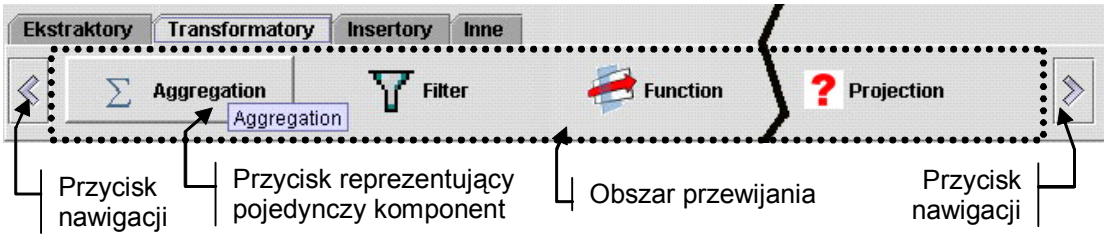
Aplikacja ETL-DR Builder umożliwia graficzne projektowanie procesów ekstrakcji danych z wykorzystaniem gotowych komponentów. Po uruchomieniu programu na ekranie pojawia się formatka, którą przedstawia rysunek 3. Tworzy ją kilka elementów, których szczegółowy opis znajduje się w następnych rozdziałach.

4.1. Obsługa przybornika i pliku konfiguracyjnego

Zadaniem przybornika jest umożliwienie użytkownikowi skorzystanie z gotowych komponentów podczas projektowania procesu ekstrakcji. Budowany jest on w czasie startu aplikacji na podstawie pliku konfiguracyjnego. Użytkownik może wybierać komponenty spośród czterech zakładek, na których umieszczane są kolejno: komponenty pobierające dane, komponenty transformujące dane, komponenty zapisujące dane. Ostatnia zakładka przewidziana jest dla wszystkich pozostałych typów komponentów. W ramach danej zakładki z wykorzystaniem przycisków nawigacyjnych możliwe jest przewijanie panelu z przyciskami w prawo lub w lewo. Jeżeli ruch panelu w daną stronę jest niemożliwy, przycisk nawigacji jest nieaktywny.



Rys. 3. Główna formatka programu ETL-DR Builder
Fig. 3. Main form of the application ETL-DR Builder



Rys. 4. Wygląd i organizacja przybornika
Fig. 4. Appearance and the organization of the toolbox

Przybornik ze względu na swą funkcję powinien umożliwiać z jednej strony łatwe i uniwersalne dodawanie komponentów, a z drugiej strony zapewnić ich czytelną graficzną reprezentację. Aby komponent mógł zostać wykorzystany przez środowisko graficzne, musi składać się z pliku klasy podstawowej, zawierającego podstawowe informacje o rodzaju i funkcjach realizowanych przez ten komponent, a także poprawnie obsługiwać komunikację z przybornikiem i panelem projektowym. Wymagana jest także druga klasa, zapewniająca właściwy graficzny edytor właściwości tego komponentu.

Przybornik budowany jest dynamicznie w oparciu o informacje zawarte w pliku konfiguracyjnym, który zawiera niezbędne dane na temat dodawanych komponentów. Gramatykę pliku konfiguracyjnego przedstawia tabela 1.

Tabela 1

Gramatyka pliku konfiguracyjnego

Nazwa parametru	Opis parametru
ComponentClass[1)]	Wskazuje ścieżkę dostępu oraz nazwę pliku głównej klasy komponentu.
ComponentCustomizerClass[1)]	Wskazuje ścieżkę dostępu oraz nazwę pliku graficznego edytora właściwości komponentu.
ComponentWrapperClass[2)]	Wskazuje ścieżkę dostępu oraz nazwę pliku graficznej reprezentacji komponentu wyświetlanej na panelu edycyjnym, o ile nie zapewnia tego główna klasa komponentu.
ComponentIcon[3)]	Wskazuje ścieżkę dostępu oraz nazwę pliku tzw. ikony, która będzie wyświetlana w przyborniku dla komponentu.
ComponentDisplayName[3)]	Definiuje nazwę, jaka będzie wyświetlana w przyborniku dla komponentu.
ComponentTipText[3)]	Definiuje treść podpowiedzi wyświetlaną w przyborniku dla komponentu.
[1)] parametr wymagany; [2)] parametr wymagany, jeżeli klasa główna komponentu nie jest klasą graficzną (tzn. nie jest rozszerzeniem klasy JComponent); [3)] parametr opcjonalny	

Kolejne dwie tabele 2 i 3 przedstawiają dwa pełne zestawy wpisów, jakie mogą wystąpić przy opisie komponentu w pliku konfiguracyjnym:

- Zestaw pierwszy – klasa główna komponentu jest klasą graficzną i sama dostarcza graficzną reprezentację komponentu.

Tabela 2

Przykład opisu komponentu, który nie korzysta z klasy reprezentacji graficznej

Nazwa parametru		Przykładowa wartość parametru
ComponentClass	=	FileExtractorComponent.class
ComponentCustomizerClass	=	FileExtractorCustomizer.class
ComponentIcon	=	ExSourceFile32.gif
ComponentDisplayName	=	FileExtraktor
ComponentTipText	=	Ekstraktor plikowy - FileExtraktor
Parametr wymagany ***Parametr opcjonalny***		

- Zestaw drugi – klasa główna komponentu nie jest klasą graficzną, dodatkowy parametr wskazuje klasę dostarczającą graficzną reprezentację komponentu.

Tabela 3

Przykład opisu komponentu, który wykorzystuje klasę reprezentacji graficznej

Nazwa parametru		Przykładowa wartość parametru
ComponentClass	=	FileExtractorComponent.class
ComponentCustomizerClass	=	FileExtractorCustomizer.class
ComponentWrapperClass	=	FileExtractorComponentWrapper.class
ComponentIcon	=	ExSourceFile32.gif
ComponentDisplayName	=	FileExtraktor
ComponentTipText	=	Ekstraktor plikowy - FileExtraktor
Parametr wymagany ***Parametr opcjonalny***		

4.2. Obsługa panelu zarządzania projektami i arkuszami

Panel zarządzania projektami i arkuszami służy do zarządzania zestawem projektów i arkuszy. Arkusz to pojedynczy proces ekstrakcji zdefiniowany w ramach danego projektu. Projekt służy natomiast do grupowania arkuszy. Projekty i arkusze wizualizowane są w formie drzewa, w którym węzły nadrzędne odpowiadają projektom, a węzły podrzędne odpowiadają arkuszom. Rysunek 5 pokazuje nam przykładową strukturę projektów i arkuszy oraz menu kontekstowe wyświetlane dla węzłów reprezentujących projekt i arkusz. Menu kontekstowe dla projektu definiuje następujące opcje:

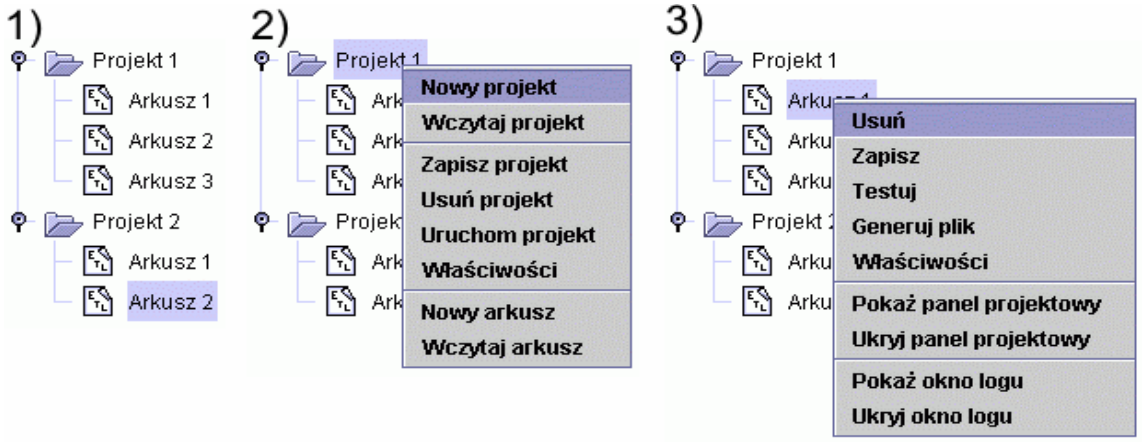
Nowy projekt – polecenie umożliwia stworzenie nowego projektu.

Wczytaj projekt – polecenie umożliwia wskazanie pliku, z którego ma być odczytana struktura projektu i dodana do panelu zarządzania. Wskazany plik musi posiadać rozszerzenie *project*.

Zapisz projekt – polecenie umożliwia wskazanie miejsca i nazwy pliku, gdzie ma zostać zapisana struktura wskazanego projektu. Zapisany plik będzie miał rozszerzenie *project*.

Usuń projekt – polecenie umożliwia usunięcie wskazanego projektu. Wraz z projektem usuwane są podległe arkusze, okna projektowe, okna informacyjne.

Uruchom projekt – polecenie aktywuje dialog służący do uruchomienia wszystkich arkuszy wchodzących w skład projektu. Dialog opisany jest w punkcie 4.5.2.



Rys. 5. Wygląd drzewa projektów i arkuszy (1) oraz menu kontekstowego dla węzła projektu (2) i węzła arkusza (3)
Fig. 5. Appearance of trees of projects and sheets (1) and of context menu for the project node (2) and the sheet node(3)

Właściwości – polecenie aktywuje dialog opisu projektu. Składa się on z dwóch zakładek. Pierwsza z nich o nazwie Właściwości pozwala na ustawienie nazwy i opisu projektu. Zawiera także informacje o dacie utworzenia projektu, liczbie arkuszy zdefiniowanych w projekcie oraz o liczbie uruchomień projektu. Kolejna zakładka o nazwie Historia służy do wizualizacji informacji o historii uruchomień projektu. Gromadzone informacje to data rozpoczęcia uruchomienia, data zakończenia uruchomienia, czas wykonania, status wykonania (OK, ERROR) oraz liczba wykonywanych arkuszy.

Nowy arkusz – polecenie umożliwia dodanie nowego arkusza. Wiąże się to ze stworzeniem nowego okna projektowego oraz nowego okna informacyjnego.

Wczytaj arkusz – polecenie umożliwia wskazanie pliku, z którego zostaną odczytane informacje na temat arkusza. Po poprawnym odczycie arkusz zostanie dodany do aktualnego projektu. Wskazany plik musi posiadać rozszerzenie *sheet*.

Menu kontekstowe dla arkusza definiuje następujące opcje:

Usuń – polecenie umożliwia usunięcie wskazanego arkusza wraz z powiązanym oknem projektowym oraz powiązanym oknem informacyjnym.

Zapisz – polecenie umożliwia wskazanie miejsca i nazwy pliku, gdzie mają zostać zapisane informacje o arkuszu, powiązanym oknie projektowym, powiązanym oknie informacyjnym. Zapisany plik będzie miał rozszerzenie *sheet*.

Testuj – polecenie umożliwia przeprowadzenie testów dla wskazanego arkusza. Od tego momentu wszystkie komponenty zapisujące dane zdefiniowane w arkuszu mogą zostać wykorzystane jako źródło danych przez inne arkusze zdefiniowane w ramach tego samego projektu.

Generuj plik – polecenie aktywuje dialog umożliwiający generację pliku konfiguracyjnego i jego uruchomienie. Dialog opisany jest w punkcie 4.5.1.

Właściwości – polecenie aktywuje dialog opisu arkusza. Składa się on z trzech zakładek. Pierwsza z nich o nazwie Informacje pozwala na ustawienie nazwy i opisu arkusza. Zawiera także informacje o dacie utworzenia arkusza i liczbie jego uruchomień. Kolejna zakładka o nazwie Właściwości pozwala na ustawienie globalnych parametrów pliku konfiguracyjnego. Należą do nich: nazwa pliku, w którym zapisywany jest stan wykonania procesu ekstrakcji (parametr obowiązkowy), domyślny katalog roboczy, domyślny rozmiar pakietu, domyślny rozmiar kolejki oraz tryb pracy procesu ekstrakcji. Poza wskazaniem nazwy pliku pozostałe parametry są nieobowiązkowe. Jeżeli użytkownik chce, aby dany parametr znalazł się w pliku konfiguracyjnym, musi to dodatkowo wskazać przez zaznaczenie odpowiedniej opcji (tzw. 'check box-a'). Ostatnia zakładka o nazwie Historia służy do wizualizacji informacji o historii uruchomień arkusza. Gromadzone informacje to: data rozpoczęcia uruchomienia, data zakończenia uruchomienia, czas wykonania, status wykonania (OK, ERROR) oraz informacje o komponentach, które brały udział w uruchomieniu: liczbie komponentów zapisujących dane (insertorów), liczbie komponentów transformujących dane (transformatorach), liczbie komponentów pobierających dane (ekstraktorach), liczbie dodatkowych komponentów projekcji związanych z wykorzystaniem niejawnej selekcji.

Pokaż panel projektowy – polecenie umożliwia pokazanie okna projektowego powiązanego ze wskazanym arkuszem. Jeżeli okno jest niewidoczne, to zostaje ono pokazane na pulpicie ponad wszystkimi innymi oknami projektowymi. Jeżeli natomiast znajduje się ono już na pulpicie, ale jest przysłonięte innymi oknami to zostaje pokazane jako pierwsze.

Ukryj panel projektowy – polecenie umożliwia ukrycie okna projektowego powiązanego ze wskazanym arkuszem.

Pokaż okno logu – polecenie umożliwia przywrócenie widoczności ukrytego okna informacyjnego powiązanego ze wskazanym arkuszem.

Ukryj okno logu – polecenie umożliwia ukrycie okna informacyjnego powiązanego ze wskazanym arkuszem.

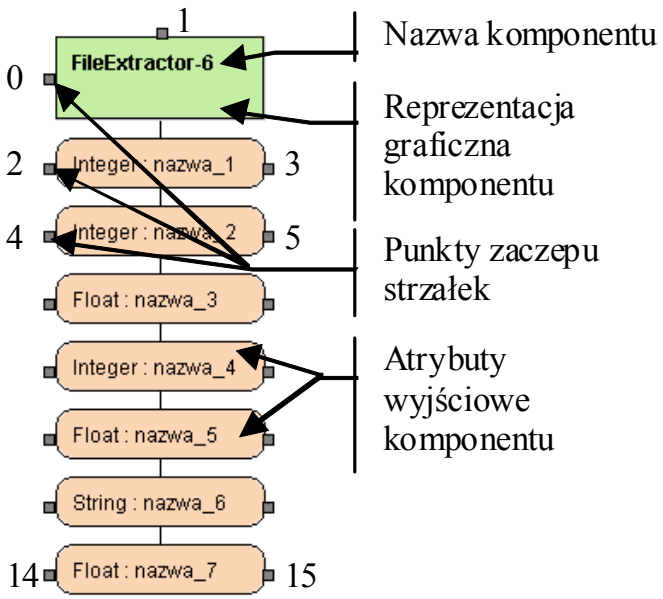
4.3. Obsługa obszaru edycyjnego

Obszar edycyjny pozwala na graficzne konstruowanie procesów ekstrakcji z wykorzystaniem zdefiniowanych komponentów. Umiejscowienie komponentu w panelu projektowym sprowadza się do kliknięcia odpowiedniego przycisku w przyborniku, a następnie kliknięcia w obszarze projektowym w miejscu, gdzie komponent ma zostać umieszczony. Informacją dla użytkownika o poprawnym wskazaniu komponentu w przyborniku jest zmiana kursora

myszki na krzyżyk w obrębie panelu projektowego. Każdy komponent wizualizowany w panelu projektowym składa się z następujących elementów:

- reprezentacji graficznej;
- nazwy komponentu;
- zestawu atrybutów wyjściowych;
- zestawu punktów zaczepu strzałek umożliwiających wprowadzenie i/lub wyprowadzanie atrybutów.

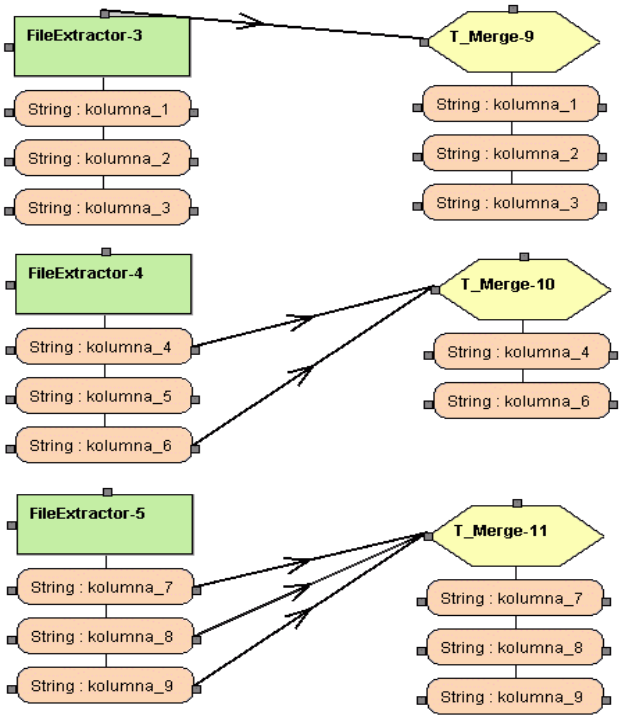
Rysunek 6 obrazuje elementy składające się na graficzną reprezentację komponentu.



Rys. 6. Opis reprezentacji graficznej komponentu
Fig. 6. Description of the graphical representation of the component

Ponieważ projektowanie procesu ekstrakcji wiąże się z definiowaniem przepływu danych pomiędzy komponentami, każdy z nich udostępnia zaczep umożliwiający wprowadzanie atrybutów. Zawsze jest to zaczep o numerze 0. W przypadku ekstraktorów jest on nieaktywny, gdyż element ten ze swojej definicji sam jest źródłem atrybutów. Pozostałe zaczepy wykorzystywane są do pobierania atrybutów i przekazywania ich do innych komponentów. Pobieranie może być zrealizowane na dwa sposoby. Pierwszy to pobieranie grupowe atrybutów przeprowadzane z wykorzystaniem zaczepu o numerze 1. Z komponentu pobierane są wtedy wszystkie atrybuty wyjściowe. Drugi natomiast to pobieranie pojedynczych atrybutów z wykorzystaniem zaczepów związanych z danym atrybutem. Taki sposób pobierania atrybutów powoduje jednak wystąpienie zjawiska niejawnej projekcji. Ma to ścisły związek ze sposobem zdefiniowania komponentów w środowisku bazowym [2] i w

efekcie powoduje konieczność definicji dodatkowego komponentu dokonującego projekcji wskazanych atrybutów na rzecz komponentu docelowego. Dodatkowy komponent nie jest bezpośrednio wstawiany do panelu projektowego, ale jego definicja pojawia się w tekstowym pliku konfiguracyjnym. Jest to proces całkowicie automatyczny i nie wymaga dodatkowej interwencji użytkownika. Należy jednak mieć to zjawisko na uwadze przy optymalizacji projektowanego procesu.



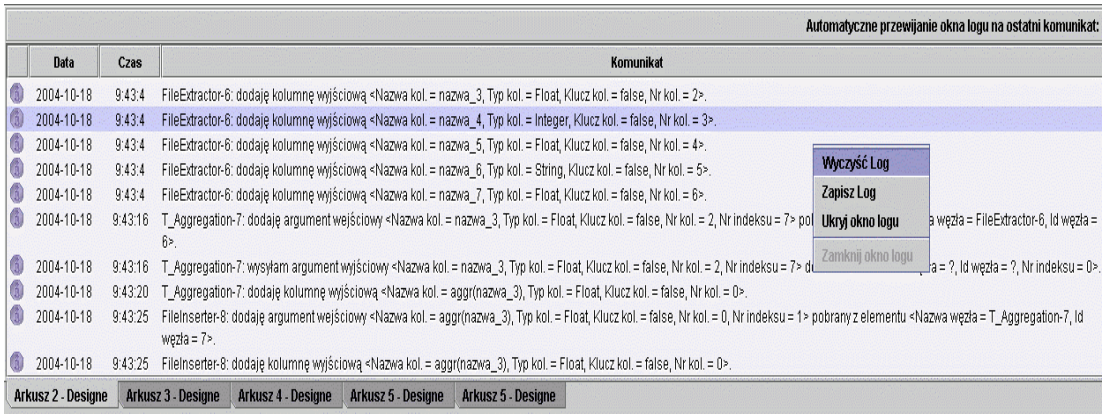
Rys. 7. Przykład łączenia komponentów, dla którego zachodzi (b, c) i nie zachodzi (a) zjawisko niejawnej projekcji

Fig. 7. Example of the combinations of components for which the effect of the implicit projection occurs (b, c) and for which the implicit projection does not take place (a)

Rysunek 7 pokazuje nam sytuacje, kiedy przy łączeniu komponentów mamy do czynienia ze zjawiskiem niejawnej projekcji (rysunek b i c) oraz kiedy takie zjawisko nie występuje (rysunek a). Przekazanie wszystkich atrybutów jak na rysunku c) nie jest jednoznaczne z przekazaniem atrybutów w sposób grupowy. Strzałka łączącą komponenty musi być wyprowadzana z zaczepu wyjściowego komponentu źródłowego. Ma ona wtedy kolor czerwony i jest rysowana linią przerywaną. W chwili, kiedy kursor myszki znajdzie się w obszarze innego zaczepu, linia reprezentująca strzałkę rysowana jest jako ciągła. Jeżeli w zaczepie tym można skończyć prowadzoną strzałkę, linia dodatkowo zmieni kolor na zielony, jeżeli jest inaczej, linia pozostanie czerwona.

4.4. Obsługa panelu informacyjnego

Panel informacyjny służy do drukowania komunikatów przekazywanych przez komponenty i inne elementy aplikacji. W obecnej wersji okna informacyjne tworzone są dla każdego arkusza i służą do drukowania komunikatów przekazywanych przez komponenty umieszczone w powiązanym panelu projektowym. Dodatkowo okna informacyjne tworzone są także w celu drukowania komunikatów związanych z testowaniem arkuszy i projektów. Rysunek 8 pokazuje przykłady okien informacyjnych. Każdy komunikat oprócz właściwej treści zawiera także datę, godzinę oraz typ komunikatu reprezentowany odpowiednim symbolem graficznym.



Rys. 8. Przykłady okien informacyjnych.
Fig. 8. Examples of information windows

W każdym oknie informacyjnym dostępne jest menu kontekstowe składające się z następujących poleceń:

Wyczyść Log – umożliwia wyczyszczenie zawartości wskazanego okna informacyjnego.

Zapisz Log – umożliwia zapis do pliku wszystkich komunikatów znajdujących się w ok.nie informacyjnym.

Ukryj okno logu – umożliwia ukrycie okna informacyjnego. Przywrócenie widoczności okna można zrealizować za pomocą polecenia menu kontekstowego Pokaż okno logu dla powiązanego arkusza lub też korzystając z opcji menu głównego Okna Pokaż okno logu arkusza, wcześniej zaznaczając powiązany arkusz.

Zamknij okno logu – umożliwia całkowite usunięcie okna informacyjnego. Opcja ta jest niedostępna dla okien informacyjnych powiązanych z arkuszem, które drukują informacje przekazywane przez komponenty znajdujące się w powiązanym panelu projektowym. Usunięcie takiego okna informacyjnego następuje w momencie usunięcia arkusza lub projektu, do którego arkusz należy.

Komunikat może należeć do jednego z trzech predefiniowanych typów (tabela 4).

Tabela 4
Symbole graficzne określające rodzaj komunikatu okna informacyjnego

Rodzaj komunikatu	Symbol graficzny
Komunikat informacyjny	
Komunikat ostrzegawczy	
Komunikat błędu	

4.5. Obsługa i uruchamianie procesu ekstrakcji danych

Środowisko graficzne umożliwia uruchomienie pojedynczego procesu ekstrakcji danych zdefiniowanego w arkuszu lub też zestawu procesów ekstrakcji danych zdefiniowanych w ramach projektu. Poprawny proces musi składać się z jednego lub więcej węzłów pobierających dane (ekstraktorów), z jednego lub więcej węzłów transformujących dane (transformatorów) oraz z jednego lub więcej węzłów zapisujących dane (insertorów). Dodatkowo połączenia pomiędzy komponentami muszą być zdefiniowane poprawnie, a same komponenty muszą być poprawnie skonfigurowane. Sposób uruchomienia zaprojektowanego procesu ekstrakcji jest ściśle związany ze środowiskiem bazowym komponentów [2]. Wymagane jest w pierwszej fazie przygotowanie tekstowego pliku konfiguracyjnego, a następnie w drugiej fazie uruchomienie klasy menadżera, dostarczanego przez środowisko bazowe, dla wskazanego pliku. Generacja pliku konfiguracyjnego odbywa się automatycznie na podstawie graficznego projektu ekstrakcji. Realizowane jest to poprzez odpytanie każdego komponentu o tekstowy opis jego obecnego stanu. Opis ten jest zgodny z wymogami środowiska bazowego [2]. Przed przystąpieniem do generacji pliku odbywa się seria testów, mająca sprawdzić poprawność zaprojektowanego procesu ekstrakcji. Ma to na celu zapewnienie maksymalnej poprawności tworzonych plików konfiguracyjnych. Testy przeprowadzane są zarówno dla pojedynczego arkusza, jak i dla projektu.

Dla pojedynczego arkusza dostępne są następujące testy:

- Test 1 – sprawdza poprawność połączeń pomiędzy komponentami. Stawiane wymagania zależą od typu badanego komponentu (dokładny opis w tabeli 5).
- Komponenty, które nie spełniają przedstawionych wymagań, nie są uwzględniane w kolejnych testach.
- Test 2 – sprawdza poprawność zaprojektowanego grafu ekstrakcji. Wymagane jest, aby w grafie zdefiniowano przynajmniej jeden komponent pobierający dane, przynajmniej jeden komponent transformujący dane oraz przynajmniej jeden komponent zapisujący dane. Niepoprawne zakończenie tego testu przerywa procedurę testowania.

Tabela 5

Wymagania testowe nr 1

Rodzaj komponentu	Wymagania
Komponenty pobierające dane – ekstraktory	Sprawdzanie czy komponenty te przekazują atrybuty innym komponentom.
Komponenty transformujące dane – transformatory	Sprawdzanie czy komponenty te pobierają atrybuty i czy przekazują je innym komponentom.
Komponenty zapisujące dane – insertory	Sprawdzanie czy komponenty pobierają dane.

- Test 3 – sprawdza poprawność konfiguracji wewnętrznej komponentów. Odbywa się to poprzez odpytanie komponentów z wykorzystaniem funkcji isReady. Jako wynik wywołania tej funkcji zwracana jest wartość logiczna true – konfiguracja poprawna, false – konfiguracja niepoprawna. Jeżeli którykolwiek z komponentów zgłosi błąd swojej konfiguracji, procedura testowania jest przerywana.
- Test 4 – sprawdza poprawność połączeń pomiędzy komponentami pod kątem zapętleń. Wykrywanie zapętleń odbywa się, co prawda, już w czasie tworzenia połączeń, jednak dla pewności wprowadzono dodatkowe sprawdzenie w postaci tego testu. Realizowany on jest z wykorzystaniem prostego i wydajnego algorytm Roy-Warshala. Jeżeli wykryta zostanie pętla, procedura testowania jest przerywana.
- Test 5 – sprawdza aktualność definicji atrybutów wyjściowych tych ekstraktorów, które pobierają dane z insertorów zdefiniowanych w innych arkuszach. W ramach jednego projektu użytkownik ma możliwość wskazania jako źródło danych dla ekstraktora dowolnego insertora zdefiniowanego w innym arkuszu. Jeżeli w tym czasie definicja zbioru atrybutów insertora ulegnie zmianie, mamy do czynienia z niespójnością. Test nr 5 ma na celu wyeliminowanie takich sytuacji. Wykrycie niespójności powoduje przerwanie procedury testowej.

Po wykonaniu testów generowany jest plik konfiguracyjny, który może zostać uruchomiony w środowisku bazowym [2].

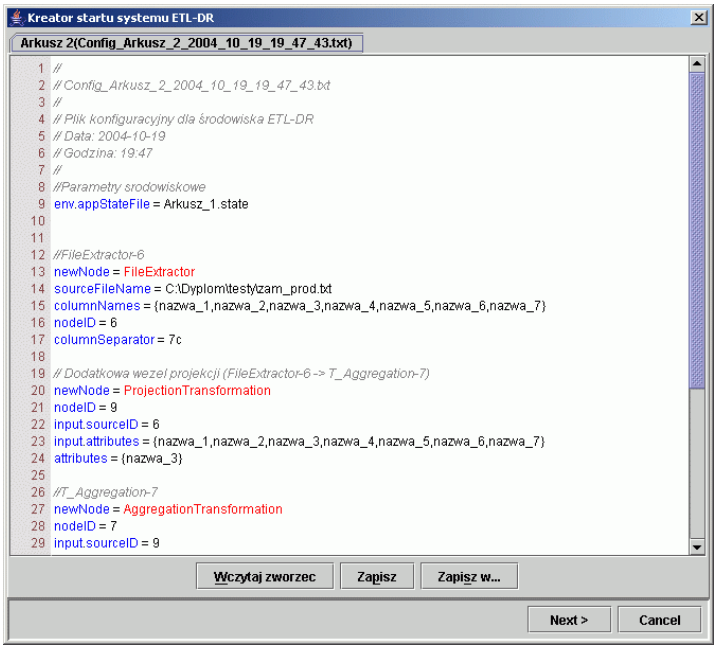
Procedura testowania projektu składa się z przetestowania wszystkich arkuszy w nim zdefiniowanych oraz przeprowadzenia dodatkowo testu nr 6.

- Test 6 – sprawdza poprawność wykorzystania insertorów zewnętrznych pod kątem zapętleń. W tym wypadku również wykorzystywany jest algorytm Roy-Warshala. Wykrycie pętli powoduje przerwanie procedury testowej.

Po poprawnym zakończeniu testów uruchamiany jest specjalny kreator określający niezbędne warunki uruchomienia projektu lub arkusza.

4.5.1. Uruchomienie pojedynczego arkusza

Uruchomienie arkusza odbywa się poprzez wybranie z menu kontekstowego polecenia Generuj plik dla tego arkusza. W pierwszej fazie generowany jest plik konfiguracyjny, którego zawartość może być dowolnie modyfikowana w oknie edytora (rysunek 9).

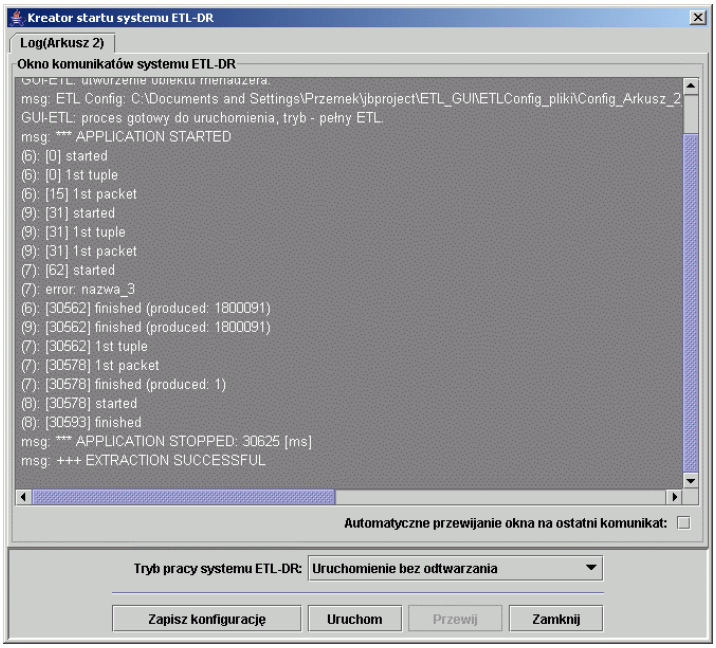


Rys. 9. Okno edytora pliku konfiguracyjnego
Fig. 9. Configuration file editor window

Edytor umożliwia ponowne wczytanie wygenerowanego wzorca oraz zapis pliku we wskazanym katalogu. Po zakończeniu edycji użytkownik może zrezygnować z uruchomienia naciskając przycisk Cancel lub też nacisnąć przycisk Next, który powoduje przejście do kolejnego okna dialogowego (rysunek 10), pozwalającego na wykonanie zdefiniowanego procesu. Użytkownik może wybrać jeden z następujących trybów wykonania:

- uruchomienie bez odtwarzania – wykonywany jest pełny proces ETL,
- uruchomienie z odtwarzaniem – podjęta zostaje próba odtworzenia,
- badanie stanu wykonania – zwracany jest status wykonania procesu ETL.

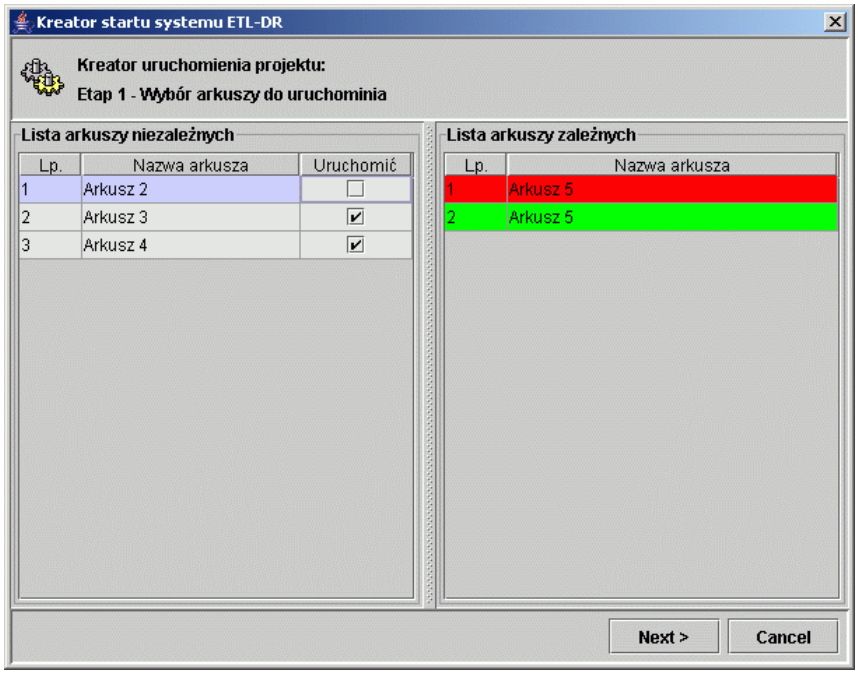
Dodatkowo istnieje możliwość zapisania stworzonej konfiguracji uruchomieniowej. Dla pojedynczego arkusza sprowadza się to do zapisu tekstu pliku konfiguracyjnego. Dzięki temu w dowolnym momencie można powtórnie wykonać zdefiniowany proces poprzez wczytanie tego pliku. Zapisany plik będzie miał rozszerzenie *srun*.



Rys. 10. Okno uruchomienia i wykonywania procesu
Fig. 10. Extraction process execution window

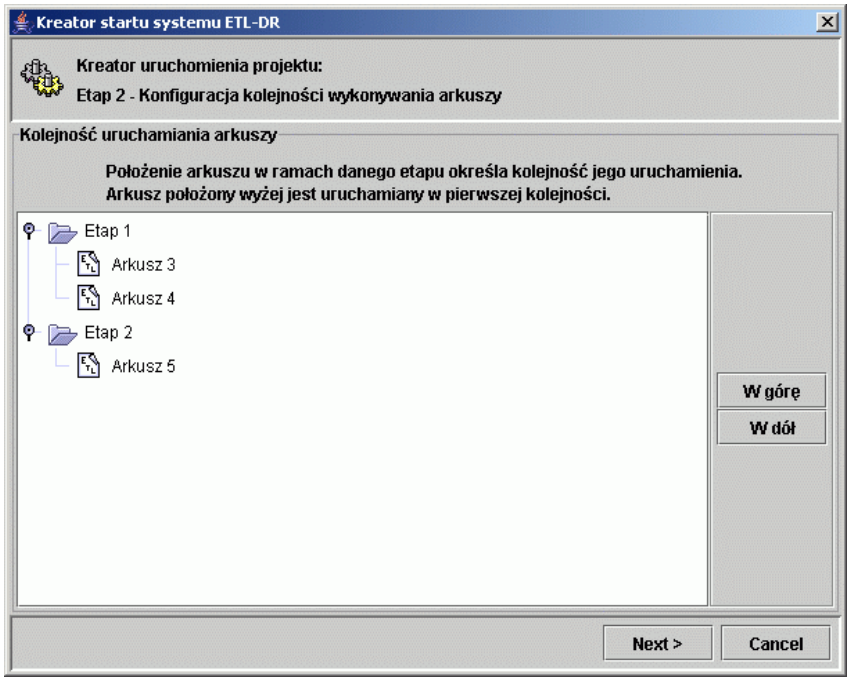
4.5.2. Uruchomienie projektu

Uruchomienie zestawu arkuszy należących do jednego projektu odbywa się poprzez wybranie z menu kontekstowego polecenia <Uruchom projekt> dla tego projektu. Zanim zostanie wygenerowany odpowiedni zestaw plików konfiguracyjnych, użytkownik ma możliwość dostosowania sposobu wykonania projektu do własnych potrzeb. Odbywa się to z wykorzystaniem zestawu okien dialogowych składających się na kreator startu projektu. Okno pierwsze obrazuje rysunek 11. Służy ono do wyboru arkuszy, które mają być uruchomione. Wyboru można dokonać tylko spośród arkuszy, których komponenty ekstrakcji nie korzystają z insertorów zewnętrznych jako źródła danych. Są to tzw. arkusze niezależne, prezentowane w tabeli po prawej stronie. W tabeli po lewej stronie prezentowane są natomiast arkusze zależne. Kolor podświetlenia zielony lub czerwony informuje o tym, czy dany arkusz zostanie uruchomiony czy nie. Zależy to tylko i wyłącznie od tego, czy będący źródłem danych arkusz niezależny także zostanie uruchomiony.

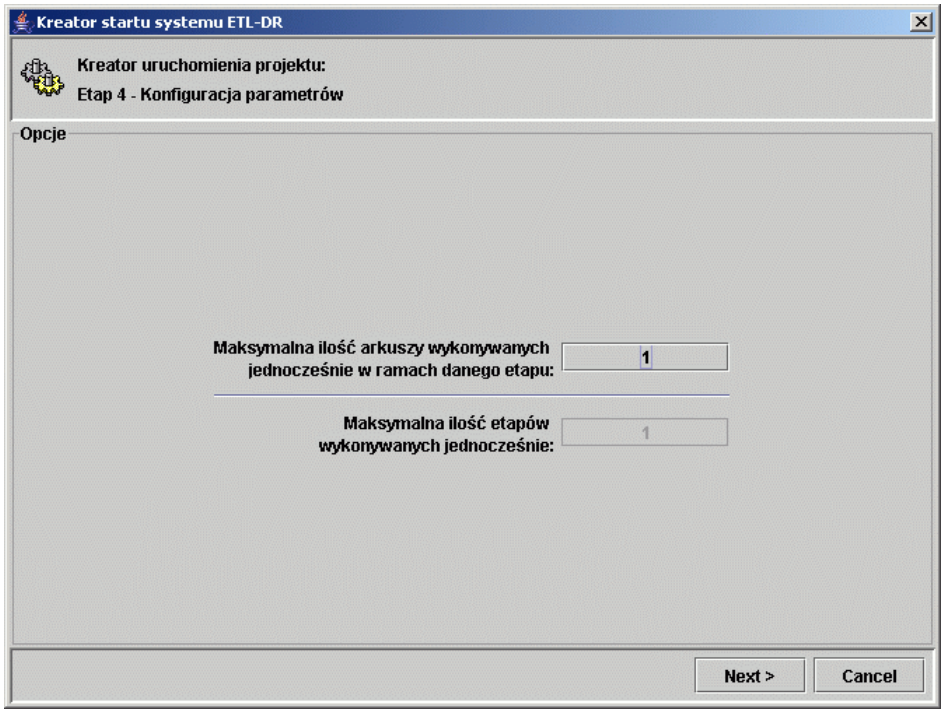


Rys. 11. Okno etapu pierwszego uruchamiania projektu
Fig. 11. Running of the project – step 1

Kolejne okno (rysunek 12) umożliwia wskazanie kolejności uruchamiania arkuszy w ramach etapów. Etapy określane są tak, aby dany arkusz miał dostępne wszystkie wymagane przez niego dane źródłowe. W etapie pierwszym wykonywane są wszystkie arkusze niezależne, a w kolejnych te arkusze zależne, dla których wszystkie wymagane informacje wejściowe są już dostępne. Kolejne okno umożliwia edycję wygenerowanych plików konfiguracyjnych. Edytor pod względem funkcjonalności jest zgodny z tym, co napisano wcześniej. Dodatkowym elementem jest tylko drzewo arkuszy umożliwiające szybkie dotarcie do powiązanego edytora z wykorzystaniem menu kontekstowego. Następne okno (rysunek 13) umożliwia określenie ilości arkuszy należących do jednego etapu, jaka może być wykonywana jednocześnie. Parametr ten należy dobierać rozważnie, aby nie doprowadzić do błędu braku pamięci. Domyślnie wartość ta jest ustawiona na jeden. Drugi parametr w obecnej wersji menadżera wykonywania projektu nie jest uwzględniany, dlatego też jego edycja jest zablokowana. Kolejne okno, już ostatnie, umożliwia uruchomienie projektu. Tu również użytkownik z wykorzystaniem drzewa wykonywanych arkuszy i menu kontekstowego może szybko dostać się do powiązanego okna komunikatów. Sam proces uruchamiania jest identyczny z opisanym wyżej dla pojedynczego arkusza. Jedyna różnica występuje dla opcji zapisu konfiguracji uruchomieniowej. W tym przypadku tworzony plik będzie miał rozszerzenie *prun* i będzie zawierał teksty wszystkich plików konfiguracyjnych projektu.



Rys. 12. Okno etapu drugiego uruchamiania projektu
Fig. 12. Running of the project – step 2



Rys. 13. Okno etapu czwartego uruchamiania projektu
Fig. 13. Running of the project – step 3

4.5.3. Uruchomienie procesu na podstawie pliku konfiguracji startowej

Wczytanie pliku konfiguracji startowej odbywa poprzez wybranie z menu głównego aplikacji polecenia <ETL-DR><Uruchom ETL z pliku kreatora konfiguracji startowej>. Powoduje to wyświetlenie okna dialogowego służącego do wskazania pliku z konfiguracją startową oraz katalogu, w którym zapamiętane pliki konfiguracyjne zostaną odtworzone. Wybrany plik musi posiadać rozszerzenie *srun* lub *prun*. Następnie odtworzone pliki wyświetlane są w oknie edytora, gdzie użytkownik ma możliwość ich edycji oraz uruchomienia tak jak opisano to powyżej.

4.5.4. Uruchomienie procesu na podstawie pliku konfiguracyjnego

Uruchomienie procesu na podstawie tekstowego pliku konfiguracyjnego odbywa się za pomocą polecenia menu głównego aplikacji <ETL-DR><Uruchom ETL z pliku konfiguracyjnego>. Powoduje to pojawienie się okna dialogowego umożliwiającego wskazanie konkretnego pliku. Następnie plik wyświetlany jest w oknie edytora, gdzie można go edytować oraz przejść do okna wykonania pliku.

5. Zakończenie

W pracy [4] przedstawiono definicję modelu konceptualnego opisującego mechanizmy odkrywania zależności pomiędzy atrybutami a odpowiednimi zadaniami ETL we wczesnych etapach projektowania hurtowni danych. Przedstawiony model daje projektantowi swobodę jego konfiguracji i umożliwia wzbogacenie go o własne sekwencje zadań ETL. W pracy [3] podjęto udaną próbę połączenia proponowanego modelu konceptualnego z jego logicznym i fizycznym modelem, ze szczególnym uwzględnieniem:

- zależności pomiędzy zadaniami ETL i rozważanymi źródłami danych,
- uchwycenia połączeń strumienia zadań/prac w scenariuszu ETL,
- optymalizacji jego wykonania w stworzonej aplikacji ETL.

W oparciu o wyniki tej pracy stworzona została aplikacja umożliwiająca wykorzystanie zbioru specjalizowanych komponentów [2], realizujących pojedyncze zadanie ekstrakcji. Aplikacja umożliwia swobodne rozszerzanie palety komponentów o nowe elementy. Wszystkie tworzone elementy środowiska testowane były na bieżąco w czasie procesu implementowania. Ostatecznym sprawdzianem ich poprawności stały się całościowe testy przeprowadzone po zakończeniu implementacji projektu.

Przeprowadzone testy wydajnościowe wykazały spory narzut czasowy związany z komunikacją pomiędzy zaimplementowanymi komponentami a środowiskiem edytora graficznego. Dogłębna analiza koncepcji wykorzystanej do budowy aplikacji [3], a rozwijana w prezen-

towanym systemie, wykazała, iż celowa jest zamiana dotychczasowego sposobu komunikacji. Obecnie wykorzystywana metoda, polegająca na odpytywaniu komponentów o ich stan, mogłaby zostać zmieniona na formę komunikatów wysyłanych przez komponenty tylko w momencie zmiany ich stanu.

LITERATURA

1. Gorawski M.: 3 perspektywy procesu ekstrakcji danych. Praca zbiorowa „Strategie informatyzacji i zarządzanie wiedzą". WNT, 2004, s. 295-295-341.
2. Gorawski M., Marks P. Ł Środowisko komponentów ETL-DR. Raport 2/PHDiSI/2004, Instytut Informatyki, Politechnika Śląska, Gliwice 2004.
3. Gorawski M., Siódemak P.: Graficzne projektowanie aplikacji ETL. Studia Informatica, Vol. 24, No 4(56), 2003, pp.345-365.
4. Vassiliadis P., Simitsis A.,Skiadopoulos S.:Conceptual Modeling for ETL Process. In Proc. 5th Inter. WorkShop on Data Warehousing and OLAP (DOLAP 2002), USA 2002.
5. Vassiliadis P., Simitsis A.,Skiadopoulos S.:On the Logical Modeling of ETL Processes". In Proc. 14th Conference on Advanced Information System Engineering (CAiSE '02), pp. 782-786, Canada 2002.
6. Demarest M.: The politics of data warehousing – http://www.hevanet.com/demarest /marc/dwpol.htmll

Recenzent: Dr hab. Zygmunt Mazur Prof. Pol. Wrocławskiej

Wpłynęło do Redakcji 25 kwietnia 2005 r.

Abstract

ETL process (data extraction) in practice can take even up to 70% of the overall time destined for data warehouse realization. This paper presents ETL universal graphic development environment that makes process of building new ETL application much easier by use of friendly user interface. ETL environment consists of components layer and „ETL Builder" graphic editor layer. Components layer contain the set of specialized components defined by base environment [3]. These components enable to model any ETL process. "ETL Builder" layer enables visualization of projects and project sheets management.

Adresy

Marcin GORAWSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, Marcin.Gorawski@polsl.pl.

Przemysław JABŁOŃSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, Hektor@zeus.polsl.gliwice.pl