Algorytm sprawdzania kompletności danych w procesie ich wymiany między systemami informatycznymi przedsiębiorstw produkcyjnych z uwzględnieniem analizy składniowej

Jacek Pękala

Instytut Technologii Maszyn i Automatyzacji Produkcji, Politechnika Krakowska

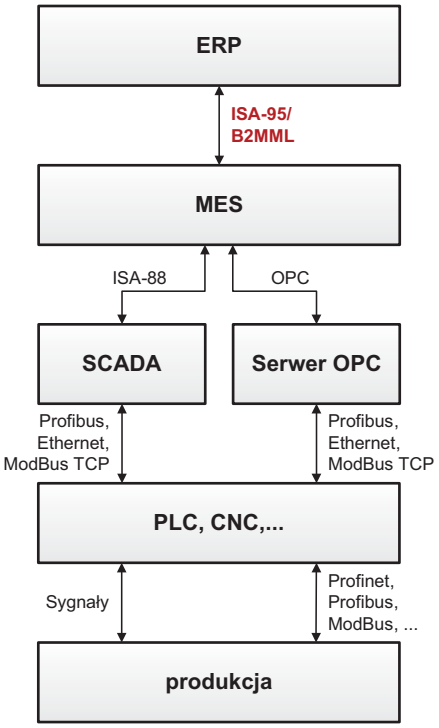
Streszczenie: W artykule przedstawiono koncepcję schematu sprawdzania kompletności danych po procesie ich transformacji opracowaną na potrzeby wymiany informacji pomiędzy różnymi systemami informatycznymi przetwarzającymi informacje w przedsiębiorstwie produkcyjnym. Scharakteryzowano pokrótce mechanizm konwersji informacji wykorzystywany podczas ich przepływu pomiędzy systemami informatycznymi przedsiębiorstwa. Opisano problematykę utraty części danych w trakcie ich przekształcania. Zaprezentowano opracowany algorytm a także wyniki działania programu go implementującego.

Słowa kluczowe: algorytm sprawdzania kompletności danych, transformacja danych, wymiana danych, analiza składniowa

W współczesnych strukturach informatycznych, których architektura jest często tworzona w oparciu o rozwiązania rozproszone, zastosowanie odpowiednich narzędzi komunikacyjnych jest niezbędne [1]. W mechanizmach wymiany informacji, obok samego przesyłania informacji, równie istotną rolę pełni ich przekształcanie, które konieczne jest z racji różnic chociażby w sposobie ich przechowywania. Stanowi ono szczególnie ważny element sprawnego obiegu informacji pomiędzy dwoma klasami systemów występujących w strukturze informatycznej przedsiębiorstwa produkcyjnego: MES i ERP. Ze względu na specyfikę pracy systemów informatycznych przedsiębiorstwa, wymiana danych pomiędzy nimi musi być pozbawiona przypadkowości, a metody komunikacji muszą być odgórnie narzucone. W obiegu danych pomiędzy systemami wykorzystuje się język B2MML. Specyfikacja języka została zbudowana na bazie standardu ISA-95 przy zachowaniu zgodności ze specyfikacją języka XML [2, 4]. Za przebieg transformacji odpowiedzialny jest blisko związany z językiem XML ogólnodostępny standard XSL. Pomimo posiadania właściwych narzędzi transformacji, przekształcanie danych nie pozbawione jest wad. Z reguły proces ten powoduje utratę części informacji w wyniku błędnie zdefiniowanych reguł transformacji, bądź ich braku. Celem niniejszej pracy jest przedstawienie najnowszej wersji rozwiązania pozwalającego na sprawdzanie kompletności danych przechodzących proces konwersji. Wykorzystano do tego został ogólnodostępny język XML przeznaczony do zapisu różnych danych w sposób zhierarchizowany.

1. Wprowadzenie

We wszystkich firmach produkcyjnych pozyskiwanie danych z warstwy produkcyjnej i zarządzanie tymi danymi służyć ma podnoszeniu wydajności i niezawodności produkcji. Akwizycja jest kluczowym elementem w procesie podejmowania decyzji i to na każdym szczeblu zarządzania w firmie – od służb operacyjnych i utrzymania ruchu, poprzez wydziały inżynieryjne, aż po jednostki administracyjne [6]. Poziomy te mają również swoje odniesienie w zhierarchizowanej strukturze informatycznej przedsiębiorstwa. W obsłudze realizacji produkcji oraz obszarów zarządzania wysokiego szczebla w przedsiębiorstwie przemysłowym stosowane są dwie klasy systemów - odpowiednio MES i ERP. Systemy MES, odpowiedzialne są za skuteczne prowadzenie procesu produkcyjnego na podstawie dokładnych i aktualnych danych produkcyjnych pochodzących z systemów niższego poziomu. Domeną systemu klasy ERP jest zarządzanie zasobami przedsiębiorstwa w tym: łańcuchami dostaw materiałów, zasobów ludzkich, finansów itp. Wymienione wyżej systemy są rozwiązaniami wzajemnie komplementarnymi, a ich ewentualna interoperacyjność stanowi wartość dodaną dla przedsiębiorstwa. Ich współpraca i związana z nią wzajemna komunikacja jest równie ważna jak każda funkcjonalność, którą poszczególne systemy zapewniają niezależnie od obecności innych podmiotów w strukturze informatycznej. Istotnym elementem nowoczesnego systemu MES jest możliwość prostej integracji z systemami automatyki przemysłowej. Wykorzystują przy tym powszechnie stosowane, otwarte standardy komunikacyjne jak ISA-88 czy OPC. W przypadku wymiany informacji z nadrzędnym dla niego systemem ERP stosowany jest standard ISA-95 i powstała na bazie języka XML jego funkcjonalna implementacja – język B2MML. Wymiana informacji pomiędzy systemami klasy MES i ERP jest równie ważna dla przedsiębiorstwa jak przepływ danych między innymi poziomami. Stanowi ona przedmiot rozważania niniejszej pracy, a w szczególności analiza powstających podczas niej ubytków informacyjnych. Rys. 1 przedstawia model hierarchii systemów informatycznych w strukturze przemysłowej z uwzględnieniem standardów komunikacyjnych.

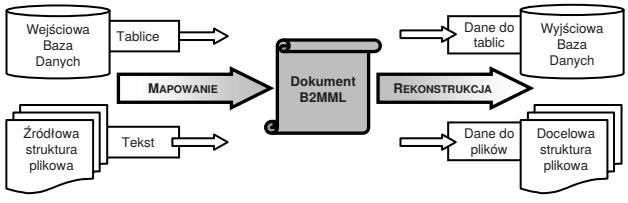


Rys. 1. Model struktury informatycznej w przedsiębiorstwie przemysłowym

Fig. 1. IT infrastructure model in an industrial enterprise

1.1. Transformacja danych za pomocą arkuszy stylów

XSLT to oparty na XML-u język przekształceń dokumentów XML. Pozwala na przetłumaczenie dokumentów z jednego formatu XML m.in. na dowolny inny format zgodny ze składnią XML-a, w tym także na wspomniany wcześniej B2MML. Dzięki dużej prostocie, łatwości implementacji i powszechnemu stosowaniu XML-a jako standardu dla zapisu informacji, XSLT jest uniwersalnym narzędziem znajdującym zastosowanie w wielu rodzajach oprogramowania [7].



Rys. 2. Schemat przepływu informacji pomiędzy systemami z uwzględnieniem procesu transformacji

Fig. 2. Schema of the information flow between IT systems including the transformation process

Danymi wejściowymi w procesie transformacji jest źródłowy dokument XML oraz arkusz stylów XSL, określający sposób transformacji dokumentu XML. Arkusz stylów składa się z szablonów. Każdy szablon opisuje jak zamieniać pewien fragment dokumentu wejściowego na fragment dokumentu wyjściowego. Dane te przetwarzane są przez procesor XSLT - aplikację, która potrafi interpretować arkusz XSLT i na podstawie danych wejściowych wygenerować dokument wyjściowy. Wykonanie transformacji polega na wywołaniu szablonu pasującego do konkretnego elementu. Rys. 2 przedstawia uproszczony schemat przepływu informacji z uwzględnieniem transformacji wiadomości do międzyoperacyjnego formatu B2MML. W uzupełnieniu do powyższego opisu należy dodać, iż dokument B2MML (jak każdy dokument typu XML) nie jest plikiem „płaskim". Ma strukturę drzewa, a dane w niej przechowywane są zhierarchizowane. XSLT stosuje szablony do elementów drzewa pasujących do zadanych wzorców, a zatem XSLT zawiera zbiór reguł opisujących przekształcenie jednego drzewa XML w nowe. Procesor w procesie transformacji tworzy nowe drzewo.

Mechanizm działania procesora XSLT podczas procesu transformacji można podzielić na dwie zasadnicze części. W pierwszej dokument XML jest przygotowywany do przekształcenia, w drugiej wykonywana jest procedura transformacji. W kroku przygotowawczym dokonywane jest przede wszystkim parsowanie arkusza XSLT oraz źródłowego dokumentu XML. W wyniku parsowania utworzone zostają ich drzewa. Następnie, z dokumentów usuwane są nadmiarowe białe znaki, a w dalszej kolejności do drzewa XSLT dołączane są standardowe reguły [7]. Po spreparowaniu dokumentów procesor przechodzi do zasadniczej części transformacji. Najpierw tworzony jest główny element drzewa wyjściowego, a następnie przetwarzane są elementy drzewa wejściowego, począwszy od elementu głównego, w efekcie czego zwracane jest drzewo wyjściowe. Szerszy opis procesu transformacji znaleźć można w [5].

1.2. Wady przekształceń danych

Największym mankamentem przekształceń dokonywanych za pomocą arkuszy stylów XSL jest powstawanie niekompletnych plików wynikowych, tzn. pozbawionych części danych znajdujących się w dokumencie wejściowym. Część danych z pliku źródłowego jest podczas konwersji tracona. Jest to przede wszystkim wynik źle zdefiniowanych plików przekształceniowych XSL, które pozbawione są definicji szablonów odpowiedzialnych za konwersję konkretnych informacji. W przypadku braku takiego szablonu informacje takie są całkowicie ignorowane przez procesor XSLT i pomijane przy tworzeniu nowej struktury danych w pliku wyjściowym.

Osobną kwestią pozostaje częsty problem braku pewnych informacji w pliku źródłowym, a których to system docelowy oczekuje. Wynika to bezpośrednio ze specyfiki pracy danego systemu, który informacji potrzebnych współpracującym z nim systemom nie obsługuje ponieważ sam ich nie potrzebuje. Arkusze stylów samodzielnie nie utworzą nie istniejących danych nawet jeśli posiadają przygotowane dla ich przekształcenia szablony. Nie istnieje zatem możliwość uzupełnienia dokumentu wynikowego w procesie transformacji. Pojawia się zatem pytanie na ile przekształcone pliki są kompletne i jakie różnice w struk-

turze danych występują między plikami źródłowym i wynikowym. W tym celu należy przyjrzeć się strukturze dokumentów a także poszczególnym operacjom na niej dokonywanym w procesie konwersji. Podczas wysyłania przez system wejściowy wiadomości dedykowanej innemu systemowi przechodzi ona proces podwójnej transformacji; najpierw bowiem trafia ona do warstwy oprogramowania pośredniczącego, która tłumaczy metadane i strukturę wiadomości na język B2MML (*schema conversion*), a następnie ponownie tłumaczy dane z formatu B2MML na postać docelową. Transformacja metadanych i struktury informacji na język właściwy dla systemu wyjściowego wymaga nie tylko przekształcenia metadanych ale także jej części semantycznej, jeśli zachodzi taka konieczność (*data conversion*). Sytuacja taka może mieć miejsce np. przy zmianie formatu zapisu daty, kiedy jeden z systemów określa ją w porządku dzień-miesiąc-rok, a drugi miesiąc-dzień-rok. Dopiero po tym zabiegu i uzyskaniu gwarancji, że właściwy system go otrzyma dokument może zostać przekazany dalej [5].

2. Sprawdzanie kompletności danych

2.1. Klasy różnic w strukturze danych

Na podstawie analizy przebiegu procesu przekształcania danych, treści arkuszy stylów oraz porównania zawartości plików przed i po przekształceniu określone zostały przez autora cztery podstawowe klasy zmian dokonywanych w strukturze dokumentów i wynikających z nich różnic:

- zmiana nazwy węzłów (metadanych), w tym także przestrzeni nazw,
- zmiana formatu zapisu danych (wartości węzła lub atrybutu),
- przeniesienie danych z bądź na listę atrybutów poszczególnych węzłów (związane z brakiem atrybutu węzła bądź istnieniem nadmiarowego),
- zmiana struktury (schematu) dokumentu poprzez przeniesienie podległości poszczególnych węzłów na inne (związane z brakiem węzła w odpowiedniej lokalizacji bądź istnieniem nadmiarowego).

Typy zmian przedstawione na przykładzie z pracy [4] zostały przedstawione w tab. 1.

2.2. Analiza składniowa

W ramach potencjalnych różnic występujących w strukturze danych szczególną uwagę skupia problem stanowiący złożenie dwóch a nawet spośród podstawowych typów wymienionych w pkt. 2.1. Całkiem bowiem prawdopodobna wydaje się być taka sytuacja, w której pliki porównywane posiadać będą różnice w strukturze danych związane z przeniesieniem węzła w inne miejsce struktury przy jednoczesnej zmianie jego nazwy oraz modyfikacji formatu zapisu informacji. Przykładem takiego węzła jest element *<StartTime>* (powstały z elementu *<BeginTime>*) znajdujący się z wiadomości B2MML przedstawionej w tab. 1, którego wartość podawana jako czas występuje w dwóch różnych postaciach.

Tab. 1. Porównanie zawartości dokumentu przed i po jego transformacji

Tab. 1. Comparison of document content before and after its transformation

Wiadomość natywna	Wiadomość wynikowa (B2MML)
<MasterProductionPlan> <Number> 0105200501095646 </Number> <ReleaseDate> 01-05-2012T09.56.46.048 </ReleaseDate> <Task> <TaskNumber> 00010000431 </TaskNumber> <Recipe id=0010+00430006> <BeginTime> 01-05-2012T00.00.00 </BeginTime> </Recipe> ...etc. </Task> </MasterProductionPlan>	<ProductionSchedule> <ID> 0105200501095646 </ID> <PublishedDate> 2012-01-05T09.56.46.048 </PublishedDate> <ProductionRequest> <ID ref=00010000431/> <ProductProductionRuleID 0010+00430006 </ProductProductionRuleID> <StartTime> 2012-01-05T00.00.00 </StartTime> ...etc. </ProductionRequest> </ProductionSchedule>
- zmiana nazwy węzła (metadanych) - przeniesienie informacji z/do listy atrybutów - zmiana formatu zapisu danych - zmiana struktury (schematu) dokumentu	

Pierwotna postać algorytmu sprawdzania kompletności danych rozpatrywała każdą z różnic z osobna. Jeśli zatem jakiś węzeł nie występował w strukturze, bądź było ich więcej niż w wersji pierwotnej algorytm wykrywał tę różnicę, zgłaszał ją i przechodził do dalszych elementów drzewa. Zatem w hipotetycznej sytuacji, w której jeden węzeł został przesunięty na niższy poziom i zaczął podlegać innemu elementowi, algorytm zgłaszał niekompletność danych. W rzeczywistości jednak konkretna informacja nadal zawarta była w wiadomości, tyle, że w innym miejscu. Problem ten nie byłby zbyt trudny do wykrycia gdyby element po przeniesieniu posiadał taką samą nazwę, albowiem każdy węzeł pliku źródłowego porównywany jest podług tego kryterium z węzłami w pliku, który przeszedł proces transformacji. Pojawia się zatem pytanie jak skojarzyć dwa elementy ze sobą, które nie dość, że znajdują się na różnych poziomach hierarchii dokumentu, to jeszcze posiadają inne nazwy, mimo, iż przechowują tę samą informację. Jednym z rozwiązań może być dołączenie słowników leksykalnych kojarzących poszczególne słowa i wyrażenia poprzez asocjacje z ich synonimami i odpowiednikami. Innym rozwiązaniem, proponowanym przez autora jest przeprowadzenie analizy składniowej na meta danych oraz danych. W razie wykrycia podobieństw w składni tagów (nazw węzłów) można przyjąć, że konkretna wartość porównywanego elementu pochodzi od elementu, do którego jest porównywana.

Niezależnie od sprawdzenia nazw węzłów, podobny mechanizm analizy tekstowej zastosować można w przypadku wartości danych, których format również może ulegać zmianie. Dopiero łącząc obydwa mechanizmy weryfikacji kompletności danych można określić czy porównywane węzły przekroczyły arbitralnie wyznaczony próg podobieństwa i tym samym skojarzone będą jako tożsame bądź nie. By cel ten zrealizować analiza składniowa została oparta o odległość Levenshteina, której algorytm obliczeniowy należy do grupy algorytmów edycyjnych (tekstowych).

2.2.1. Odległość Levenshteina

Odległość Levenshteina to zaproponowana przez Vladimira Levenshteina w 1965 roku miara odmienności napisów (skończonych łańcuchów znakowych) [3]. Nieformalnie przyjmuje się, że odległość Leveshteina pomiędzy dwoma słowami jest równa liczbie jednoznakowych modyfikacji niezbędnych do zmiany jednego słowa w drugie. Odległość definiuje się jako minimalną liczbę zmian potrzebnych do przekształcenia jednego łańcucha w drugi, przy czym modyfikacje odbywać się mogą tylko poprzez operacje wstawiania, usuwania lub zamiany znaku. Wzór (1) przedstawia matematyczną definicję odległości Levenshteina, gdzie:
lev – oznacza odległość Levenshteina,
a, b – dwa łańcuchy znakowe,
i, j – długość łańcuchów znakowych odpowiednio a i b [3].

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} 0 & , i = j = 0 \\ i & , j = 0 \text{ i } i > 0 \\ j & , i = 0 \text{ i } j > 0 \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+[a_i \neq b_j] \end{cases} & \text{, lub} \end{cases}$$

(1)

Wartość odległości Levenshteina dla przykładowych słów: *marka* i *ariada* wynosi 4, ponieważ potrzeba co najmniej czterech działań do przekształcenia pierwszego słowa w drugie: usunięcia litery m, zamiany *k* na *i*, oraz dodania *d* oraz *a*.

2.3. Algorytm sprawdzania kompletności danych

W oparciu o przedstawioną w pkt. 2.1 klasyfikację różnic pomiędzy plikami oraz analizę składniową wykorzystującą obliczenia odległości Levenshteina omówioną pkt. 2.2 opracowany został algorytm sprawdzania kompletności danych (rys. 4). Ma on za zadanie porównywać zawartość plików przed i po transformacji pod kątem omówionych typów różnic z uwzględnieniem analizy składniowej.

Mechanizm działa według następującej procedury:

1. Uruchomienie algorytmu porównywania dokumentów XML.
2. Odczyt z interfejsu minimalnego stopnia podobieństwa wymaganego do określenia dwóch różnych elementów jako podobne.
3. Odczyt porównywanych plików XML.
4. Dekompozycja wczytanych plików XML do listy węzłów (dwie listy – elementy źródłowe i docelowe).
5. Oznaczenie (oflagowanie) wszystkich węzłów znajdujących się na liście jako 'nieodnalezione'.
6. Przydzielenie każdemu węzłowi na liście jego ścieżki w drzewie (XPath).
7. Wykonanie zestawu instrukcji na każdym nieodnalezionym węźle z listy źródłowej, w tym:
   a. Sprawdzenie czy dany węzeł źródłowy został znaleziony w liście docelowej (sprawdzenie czy w liście docelowej istnieje element o takiej samej ścieżce w drzewie).
   b. Selekcja pośród nieodnalezionych węzłów docelowych wybieramy wszystkich 'podobnych' do porównywanego węzła. Domyślnie jako 'podobne' traktujemy wszystkie nieodnalezione węzły, które posiadają taką samą nazwę jak węzeł poszukiwany. Jeżeli przed porównywaniem zaznaczona została opcja ignorowania nazw to jako 'podobne' traktowe są wszystkie nieodnalezione węzły.
   c. Selekcja węzła o największym podobieństwie wśród węzłów podobnych. Podobieństwo określamy jako sumę punktów przyznawanych za podobieństwo:
      - wartości tekstowej węzłów (z wykorzystaniem algorytmu Levenshteina),
      - atrybutów,
      - nazwy elementu (z wykorzystaniem algorytmu Levenshteina),
      - przestrzeni nazw,
      - elementów potomnych węzłów.
   d. Sprawdzenie czy węzeł wybrany jako najbardziej podobny (taki, który uzyskał najwięcej punktów podczas porównywania) jest wystarczająco podobny (przekracza minimalny stopnień podobieństwa określony w punkcie 2).
   e. Zamiana ścieżki w drzewie elementu porównywanego na ścieżkę elementu źródłowego (wiąże się to także ze zmianą ścieżki XPath dla wszystkich węzłów potomnych) w przypadku spełnienia warunku określonego w pkt. 7d.
   f. Zgłoszenie informacji o odnalezieniu przeniesionego węzła. Na liście węzłów docelowych został odnaleziony węzeł o takiej samej ścieżce, co węzeł źródłowy. Obydwa węzły zostają oznaczone jako znalezione.
8. Po oznaczeniu węzłów jako 'znalezione' algorytm przechodzi do wykonywania szeregu operacji na znalezionych elementach, w tym:
   a. Porównanie wartości tekstowej węzłów (bez wartości tekstowych węzłów potomnych (dzieci) jeśli są one zagnieżdżone).
   b. Weryfikacja czy wartości tekstowe węzłów są równe.
   c. W razie wystąpienia różnicy następuje zgłoszenie komunikatu zawierającego informacje o odległości

Levenshteina pomiędzy porównywanymi wartościami.

d. Porównanie wszystkich atrybutów węzłów. Porównanie obejmuje zarówno odnalezienie atrybutów źródłowych w węźle docelowym, jak również porównanie ich wartości.
e. Sprawdzenie czy węzły posiadają identyczny zestaw atrybutów. Jeżeli atrybuty są różne (brak atrybutu w węźle docelowym, zbędny atrybut w węźle docelowym, różne wartości atrybutu) to zgłaszany jest komunikat.
f. Porównanie przestrzeni nazw do których należą porównywane węzły.
g. Weryfikacja czy przestrzenie węzłów są identyczne. Jeśli węzły należą do różnych przestrzeni to zgłaszany jest odpowiedni komunikat.

9. Sprawdzenie czy porównano wszystkie węzły (sprawdzenie pozwalające zakończyć pętlę rozpoczynającą się w punkcie 7).
10. Sprawdzenie czy w zakończonej pętli zostały zgłoszone nowe problemy.
11. Weryfikacja czy na listach węzłów (źródłowych/docelowych) znajdują się jeszcze elementy nieodnalezione.
12. Zgłoszenie komunikatu dla każdego nieodnalezionego elementu z listy węzłów.
13. Koniec.

Przez zgłoszenie komunikatu należy rozumieć ekspozycję informacji o występującym problemie, zawierającej dane na temat: klasy różnicy, szczegółowej informacji dotyczącej różnicy, linii dokumentu, w której wystąpił problem.

3. Implementacja algorytmu

Na potrzeby realizacji celu niniejszej pracy opracowana została aplikacja, której głównym zadaniem jest porównywanie plików w formacie XML, a w dalszej kolejności określenie stopnia kompletności danych dokumentu porównywanego do wzorcowego. Aplikacja ma możliwość sprawdzenia zgodności dokumentu ze specyfikacją XML, a także zawartości jego syntaktycznej i semantycznej. W programie znajduje się prosty edytor kodu pozwalający na edycję dokumentów. Aplikacja została napisana w języku C#, a do jego uruchomienia konieczne jest posiadanie zainstalowanej platformy .NET Framework 4.0.

3.1. Opis programu

3.1.1. Okno główne

Główne okno programu (rys. 3) podzielone jest na trzy sekcje:
- pasek narzędzi,
- pola edycyjne porównywanych plików.
- pasek wyświetlający analizowane różnice.

Pasek narzędzi zawiera rozwijalne menu zawierające przyciski pozwalające na tworzenie, otwieranie, zapisywanie plików oraz zamykanie aplikacji, a także wstążkę z zestawem narzędzi pozwalającą na ustawienie parametrów działania aplikacji, uruchomienie procedury porównania, łatwą nawigację po wykrytych różnicach, określenie progu podobieństwa oraz obserwację prostych statystyk dotyczących wykrytych błędów.

Rys. 3. Okno główne programu
Fig. 3. Main window

W części edycyjnej wyświetlana jest zawartość wczytanych plików XML z rozróżnieniem na plik źródłowy (czyli ten, do którego dokonywane będzie porównanie) oraz docelowy (czyli taki, który będzie porównywany).

Ostatnia, najniżej położona sekcja aplikacji prezentuje listę wykrytych różnic pomiędzy plikami.

3.1.2. Opis działania

Po otworzeniu aplikacji istnieje możliwość stworzenia nowych plików lub wczytania ich celem późniejszej analizy kompletności danych. Po wczytaniu bądź stworzeniu dokumentów przechodzimy dokonujemy prostych ustawień związanych z przeprowadzaną analizą porównawczej. Dostępne są następujące opcje:
- **Przestrzenie** – jeśli użytkownik dokonał wyboru tej opcji program będzie sprawdzał przestrzenie nazw elementów,
- **Główne problemy** – jeśli opcja jest zaznaczona, na liście problemów znajdować się będą tylko istotne różnice. W przeciwnym razie wyświetlone zostaną wszystkie występujące problemy,
- **Ignoruj nazwy** – po zaznaczeniu tej opcji każdy znaleziony element będzie porównywany z innym nieznalezionymi bez względu na jego nazwę i położenie.

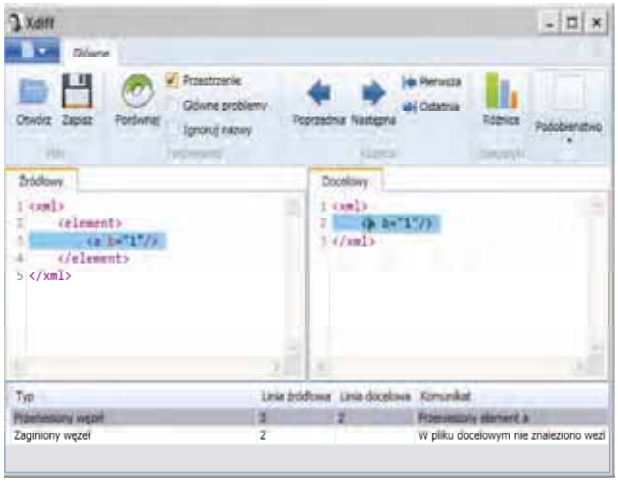
W następnej kolejności należy określić stopień podobieństwa dla nieznalezionych elementów. Próg ten definiuje się za pomocą suwaka. Po ustaleniu wszystkich ustawień przejść można do procesu porównywania dokumentów XML poprzez naciśnięcie przycisku ‘Porównaj’. Po zakończeniu analizy treści porównywanych dokumentów w dolnej części okna aplikacji pojawia się lista różnic między nimi.

By sprawnie poruszać się podczas przeglądania znalezionych różnic istnieje możliwość skorzystania z przycisków nawigacyjnych ‘Poprzednia’, ‘Następna’, ‘Pierwsza’, ‘Ostatnia’, które pozwalają na swobodne przemieszczanie się po tekście dokumentu do miejsc, w których wystąpiły różnice. Prostą statystykę podsumowujące przeprowadzoną analizę, dotyczącą liczby wystąpień poszczególnych różnic można znaleźć pod przyciskiem różnice.

Typy zgłaszanych przez program problemów w strukturze dokumentów i wynikających z nich różnic są następujące:

- **Różne wartości tekstowe węzłów** – problem zgłaszany w przypadku gdy dwa węzły o taj samej nazwie i położeniu w strukturze drzewa xml posiadają różne wartości tekstowe,
- **Zbędny atrybut** – problem występujący gdy w docelowym węźle występuje atrybut nieistniejący w węźle źródłowym,
- **Zaginiony atrybut** – informacja wyświetlana, gdy w źródłowym węźle występuje atrybut nieistniejący w węźle docelowym,
- **Różne wartości tekstowe atrybutów** – komunikat zgłaszany kiedy węzły posiadają ten sam atrybut, lecz z różnymi wartościami,
- **Zbędny węzeł** – wiadomość wyświetlana w sytuacji, w której w pliku docelowym występuje węzeł, którego nie ma w pliku źródłowym,
- **Zaginiony węzeł** – informacja zgłaszana w sytuacji, gdy w pliku docelowym nie można odnaleźć węzła występującego w pliku źródłowym,
- **Przeniesiony węzeł** – problem zgłaszany w momencie gdy w pliku źródłowym znajduje się węzeł, którego nie odnaleziono bezpośrednio w pliku docelowym, lecz odnaleziono do niego „podobny" węzeł,
- **Różne przestrzenie nazw węzłów** – wiadomość wyświetlana w sytuacji, gdy dwa węzły należą do różnych przestrzeni nazw,
- **Błędy w strukturze dokumentu** – komunikat zgłaszany kiedy wczytywany dokument nie jest poprawnym plikiem zapisanym zgodnie ze specyfikacją XML.

Rys. 4. przedstawia przykładowy efekt działania programu.



Rys. 4. Przykładowe wyniki działania programu
Fig. 4. Sample results from the program

4. Podsumowanie

Prezentowane rozwiązanie zostało przetestowane na przykładowych dokumentach XML wygenerowanych przez system MES firmy Wonderware oraz plik powstałych poprzez ich konwersję do formatu zgodnego ze specyfikacją B2MML. Wyniki działania programu są na obecnym etapie jego rozwoju zgodne z oczekiwaniami. Rozwiązanie to wpisuje się w teorię związaną z wymianą danych pomiędzy systemami klasy ERP i MES jako element warstwy pośredniczącej (middleware) pomiędzy systemami. Pozwala ona porównywać dane pochodzące z różnych środowisk ze sobą i na podstawie różnic między nimi ułatwić uzupełnianie pomijanych w procesie wymiany informacji. Stanowi ona wartość dodaną do rozwiązania, którego nadrzędnym celem jest interoperacyjność systemowa na szczeblu zarządczym w przedsiębiorstwie produkcyjnym. Obecna postać algorytmu może być rozwijana o kolejne metody analizy składniowej.

Bibliografia

1. Chwajoł G., *The Evolution of Middleware used in distributed manufacturing control systems*, "Mechanika i informatyka" 2005, 18-20.
2. Gould L., *B2MML Explained*, "Automotive Design & Production" 2007, [www.autofieldguide.com/articles/b2mml-explained].
3. Levenshtein V.I., *Binary codes capable of correcting deletions, insertions, and reversals.* "Soviet Physics Doklady" 10/1966, 707–710.
4. Pękala J., *Wykorzystanie języka B2MML jako narzędzia integracji informacji w przedsiębiorstwie produkcyjnym w ramach standardu ISA-95*, „Research Reports Project CII-SK-0030-03-0708" 2008, 304-309.
5. Pękala J., Gadzina K., *Transformacja danych z wykorzystaniem formatu B2MML jako element integracji systemów informatycznych przedsiębiorstwa*, „Pomiary, Automatyka, Robotyka" 2/2012, 151-156.
6. Skura K., Smalec Z., *Integracja systemów informatycznych w automatyzacji procesów produkcyjnych.* „Pomiary, Automatyka, Robotyka" 7-8/2005, 6-11.
7. [pl.wikipedia.org/wiki/XSL_Transformations] *XSL Transformations* (23 czerwca 2012).

Data completeness verification algorithm in the process of its exchange between production enterprise systems with syntax analysis included

Abstract: The paper presents newest version of data completeness verification algorithm used in the process of its transformation elaborated for the information exchange between different systems in manufacturing company. Data conversion mechanism used in information flow between enterprise systems was characterized. Problems of data loss during the conversion process were described. Results of working of the program which implements the algorithm were presented.

Keywords: data completeness verification algorithm, data transformation, data exchange, syntax analysis

mgr inż. Jacek Pękala
Asystent naukowo-dydaktyczny, Instytut Technologii Maszyn i Automatyzacji Produkcji, Politechnika Krakowska.
e-mail: pekala@m6.mech.pk.edu.pl