Mgr inż. Jacek Pękala, email: pekala@m6.mech.pk.edu.pl
Politechnika Krakowska, Wydział Mechaniczny
Damian Dziedzic, email: damiandziedzic@gmail.com
Politechnika Krakowska, Wydział Mechaniczny
Mateusz Kołacz, email: mat.kolacz@gmail.com
Politechnika Krakowska, Wydział Mechaniczny

WALIDACJA DANYCH W PROCESIE ICH WYMIANY POMIĘDZY SYSTEMAMI INFORMATYCZNYMI W PRZEDSIĘBIORSTWIE PRODUKCYJNYM

Streszczenie: W niniejszym artykule przedstawiono koncepcję walidacji 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 kontroli poprawności danych po ich przekształcaniu. Zaprezentowano opracowane rozwiązanie, a także wyniki działania programu go implementującego.

Słowa kluczowe: walidacja danych, poprawność danych, XML, B2MML, XSD, ERP, MES

DATA VALIDATION IN THE PROCESS OF ITS EXCHANGE BETWEEN PRODUCTION ENTERPRISE SYSTEMS

Abstract: The paper presents the concept of data validation process after their transformation developed for the exchange of information between different systems in a manufacturing company. A brief overview of the mechanism of conversion of information used during their exchange between systems company is depicted. The issues of validation of data after conversion has been described. Developed solution, and the results of its software implementation is presented.

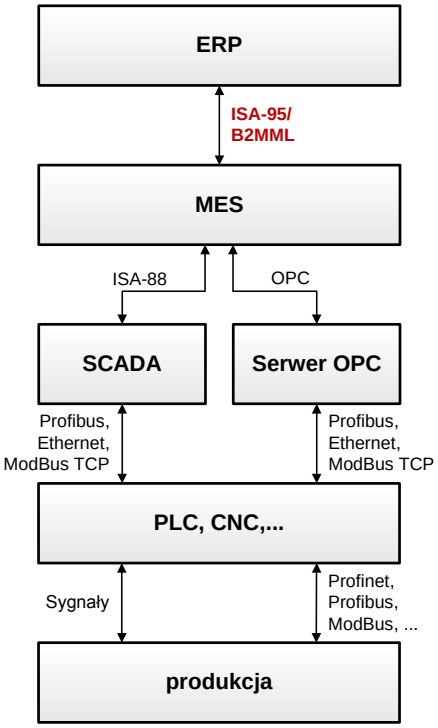
Keywords: data validation, data correctness, XML, B2MML, XSD, ERP, MES

1. WPROWADZENIE

We wszystkich firmach produkcyjnych pozyskiwanie danych z warstwy produkcyjnej i zarządzanie nimi służyć ma podnoszeniu wydajności i niezawodności produkcji. Akwizycja jest kluczowym elementem w procesie podejmowania decyzji i to na każdym szczeblu zarządzania w przedsiębiorstwie – od służb operacyjnych i utrzymania ruchu, poprzez wydziały inżynieryjne, aż po jednostki administracyjne [7]. 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 (Manufacturing Execution Systems) i ERP (Enterprise Resource Planning) [10]. Systemy MES, odpowiedzialne są za skuteczne prowadzenie procesu produkcyjnego na podstawie dokładnych i aktualnych danych produkcyjnych pochodzących z systemów niższego poziomu, takich jak SCADA (Supervisory Control and Data Aquisition) realizujących kontrolę, sterowanie oraz zajmujące się gromadzeniem danych z produkcji. Domeną

systemu klasy ERP jest zarządzanie zasobami przedsiębiorstwa w tym: łańcuchami dostaw materiałów, zasobów ludzkich, finansów, itp. Wymienione wyżej systemy są rozwiązaniami wzajemnie komplementarnymi. Ich zdolność do wzajemnego zrozumienia stanowi wartość dodaną dla przedsiębiorstwa. 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.

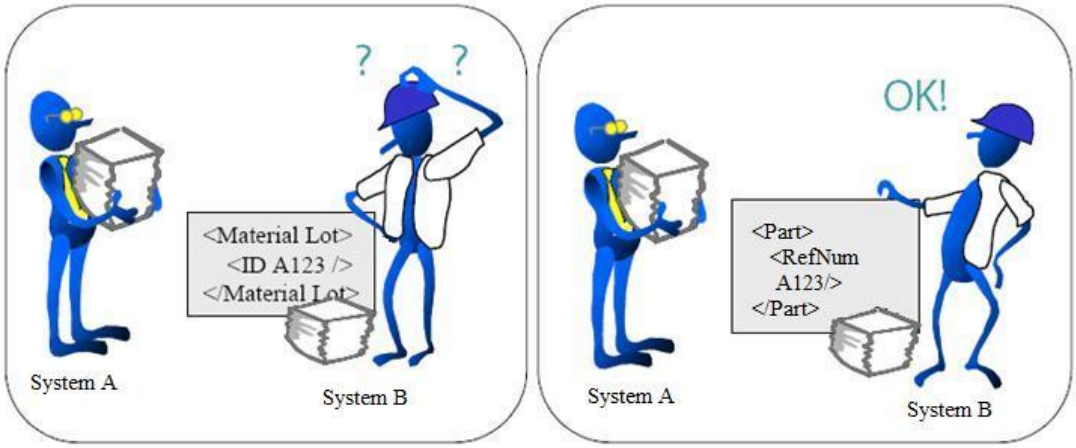
Istotnym elementem nowoczesnego systemu MES jest możliwość prostej integracji z systemami automatyki przemysłowej. Wykorzystują przy tym powszechnie stosowane, otwarte standardy komunikacyjne jak ISA-88 czy OPC (OLE for Process Control) [1]. W przypadku wymiany informacji z nadrzędnym dla niego systemem ERP stosowany jest standard ISA-95 i powstała na bazie języka XML (eXtensible Markup Language) jego funkcjonalna implementacja – język B2MML (Business To Manufacturing Marku Language). Rys. 1 przedstawia model hierarchii systemów informatycznych w strukturze przemysłowej z uwzględnieniem standardów komunikacyjnych.



Rys. 1. Model struktury informatycznej w przedsiębiorstwie przemysłowym z uwzględnieniem standardów komunikacyjnych

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 poprawności przekształconych danych. Jest ona konieczna z powodu braku spójności w strukturze przechowywania danych w różnych systemach. Informacje są gromadzone w systemach w sposób odmienny. Wynika to z różnic koncepcji modeli obiektowych opracowanych przez różnych dostawców oprogramowania. Informacje przepływające między podmiotami muszą być zatem każdorazowo sprawdzone pod kątem zgodności z modelami obiektowymi zdefiniowanymi u odbiorcy zanim do niego zostaną przekazane. W przeciwnym wypadku

nie będą one zrozumiałe dla systemu je otrzymującego. Rys. 2 ilustruje podjętą w artykule problematykę.



Rys. 2. Problematyka poprawności danych

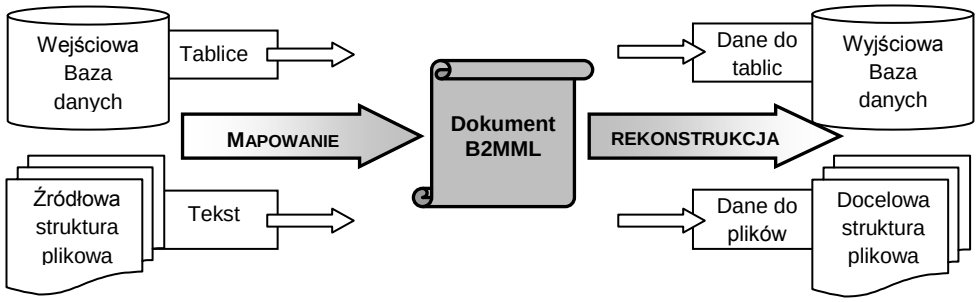
2. STANDARD ISA-95 I JĘZYK B2MML

ANSI/ISA-95 Enterprise-Control System Integration to międzynarodowy standard zatwierdzony przez grupę producentów, dostawców systemów informatycznych i ich opiniodawców. To opisana w kilku dokumentach, składająca się z pięciu części metodyka szeroko pojętej integracji systemów [6]. Standard nie przedstawia technicznego rozwiązania problemu, za to określa fundamenty pod jego realizację. Jeśli przyjąć, że standard ISA-95 prezentuje teorię dotyczącą integracji systemów zarządzania (ERP) i systemów odpowiedzialnych za realizację produkcji (MES), to za jego ramię wykonawcze uznać należy język B2MML (Business to Manufacturing Markup Language). W pracy [3] autor podaje prostą definicję języka B2MML jako opartą na języku XML implementację standardu ANSI/ISA-95. Język B2MML zawiera zbiór schematów XML zapisanych w języku XSD, w których zawarte są zaczerpnięte z treści standardu ISA-95 definicje modeli obiektowych. Celem nadrzędnym stawianym językowi jest pośredniczenie w procesie integracji systemów poprzez konwersję danych i struktury wiadomości przesyłanych pomiędzy tymi systemami. Połączenie XML i ISA-95 przynosi wiele wymiernych korzyści w procesie transferu informacji. Poza otwartością, prostotą i niezależnością, schematy XML są łatwo adaptowalne do potrzeb wymiany danych, która wymaga zachowania jednolitości i spójności struktury danych. Znaczącą zaletą języków XML i B2MML jest czytelność informacji wynikająca z przejrzystej struktury. Należy jednak pamiętać, że język B2MML nie jest standardem, a pewną interpretacją standardu, która w drobnych szczegółach może być inaczej rozumiana przez różnych dostawców systemów i użytkowników.

3. TRANSFORMACJA DANYCH OPARTA O ARKUSZE 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 [9].

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. 3 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 trakcie transformacji tworzy nowe drzewo.



Rys. 3. Schemat przepływu informacji pomiędzy systemami z uwzględnieniem procesu transformacji

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 [9]. 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 stanowiący obiekt wcześniejszych badań autora znaleźć można w [4].

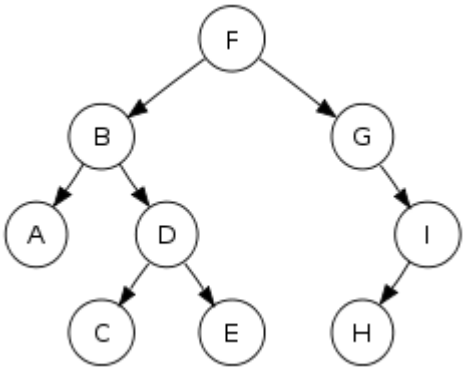
4. POPRAWNOŚĆ DANYCH

Poza wspomnianą w rozdziale 3 transformacją danych, równie istotną rolę w procesie przepływu informacji pełni następujące po nim badanie kompletności danych. Proces ten jest niezbędny do przeprowadzenia albowiem wynika z błędów bądź braku reguł przekształceniowych w trakcie konwersji. Weryfikacja kompletności danych stanowiła przedmiot wcześniejszych badań autora i została szerzej omówiona w pracy [5]. W efekcie przeprowadzenia operacji transformacji oraz kompletyzacji danych, powstały pakiet informacji powinien być gotowy do przekazania do systemu docelowego. Wymaga on jednak uprzedniego sprawdzenia czy struktura danych jest poprawna. W tym celu wykorzystywany jest język definicji schematów rozszerzalnego języka znaczników - XSD (XML Schema Definition) [8]. Pochodzi on z tej samej rodziny języków co XML czy XSL, a zatem idealnie nadaje się do kontroli pakietów danych zapisanych w formacie B2MML. Komitet SP-95 działający przy World Batch Forum (Światowe Forum Przetwarzania Wsadowego) i rozwijający zarówno standard ISA-95, jak i oparty na nim język B2MML, dostarcza definicje schematów danych poszczególnych modeli obiektowych zawartych w normie w postaci plików o rozszerzeniu .xsd. A zatem, posiadając wiadomość przekształconą przy użyciu języka XSL na format B2MML, mamy możliwość sprawdzenia czy utworzony plik posiada prawidłową definicję.

W części badawczej Autorzy skupił się na problemie sprawdzenia poprawności danych. Na podstawie analizy przebiegu procesów przekształcania i uzupełniania danych, treści plików z danymi po konwersji, jak i plików z definicjami poszczególnych schematów określone zostały przez Autorów trzy podstawowe problemy związane z walidacją danych:

- wielopoziomowa, zagnieżdżona struktura,
- częste odwołania do zewnętrznych definicji schematów,
- budowa rekurencyjna.

Praktycznie każdy obiekt opisany szeregiem informacji składa się z mniejszych obiektów również posiadających swoje atrybuty. Podległość poszczególnych elementów jest często prezentowana w postaci tzw. drzewa [2]. Drzewa to określone struktury danych, gdzie jednemu obiektowi (węzłowi) podlegać mogą inne, a im kolejne (rys. 4).



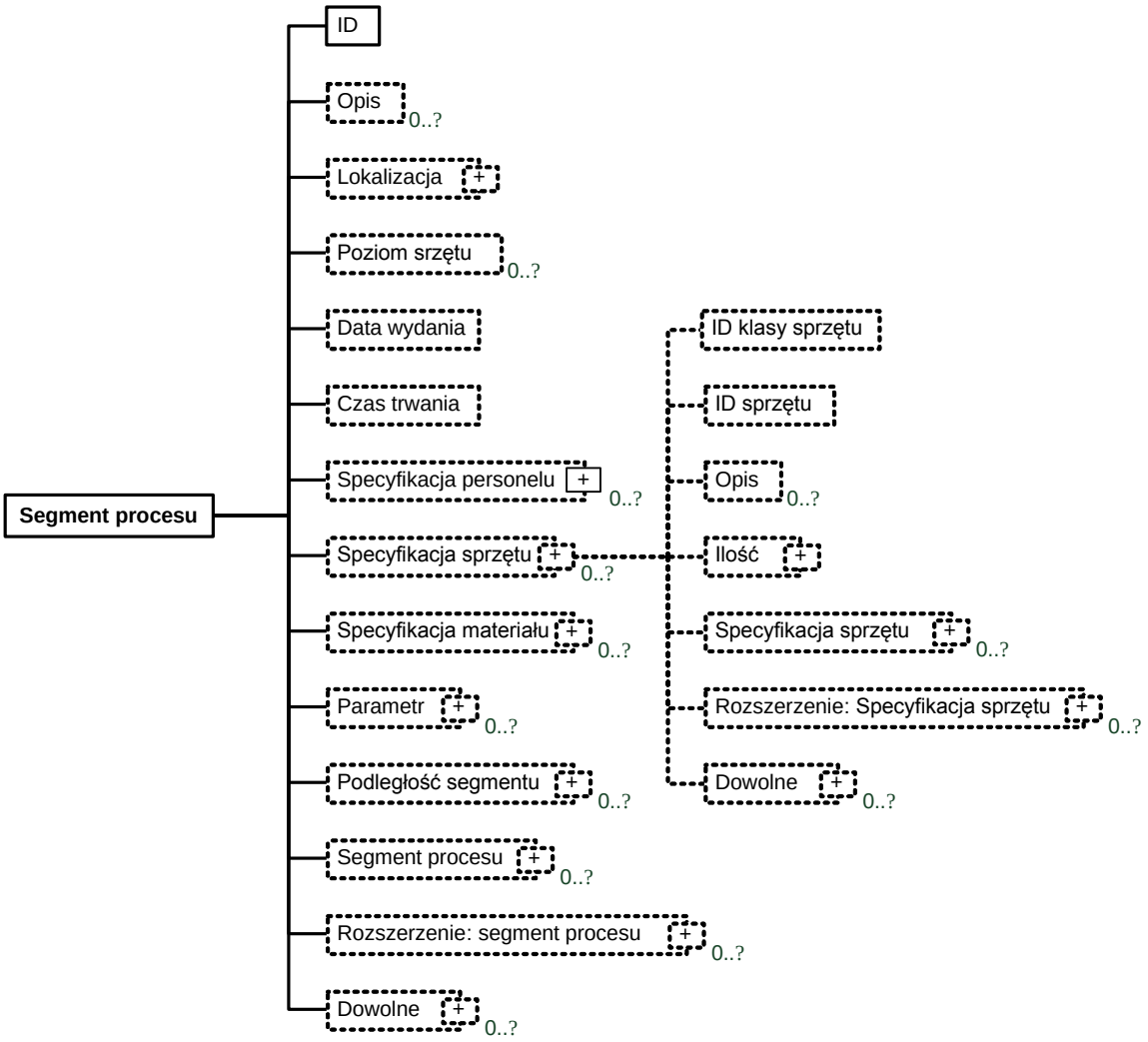
Rys. 4. Przykładowe drzewo danych [2]

Drzewa te w sposób naturalny definiują hierarchię. W przypadku obiektów opracowanych w ramach języka B2MML są one bardzo rozbudowane. Na każdym poziomie informacji w takim schemacie znajdują się elementy, które posiadają własną

strukturę danych. Im więcej elementów posiada dane drzewo, tym trudniej zapewnić jego poprawność jako całości, albowiem nawet jeden najdrobniejszy błąd może nieść niechciane konsekwencje. Na rys. 5 zademonstrowana została przykładowa definicja schematu zgodnego z modelami obiektowymi zawartymi w standardzie ISA-95. W strukturze tej widać rozwinięcie drzewa do trzeciego poziomu, jednak kolejne elementy znajdują się także na niższych poziomach. Takie wielopoziomowe, zagnieżdżone struktury danych pomimo przejrzystości zapisu wymagają bardzo rozbudowanych definicji schematów. Kaskadowość struktury zwiększa ryzyko popełnienia w niej błędu. Tak rozbudowane struktury wymagają bardzo pieczołowitego przygotowania definicji schematów.

Kolejnym, częstym problemem występującym w przypadku rozbudowanych struktur danych, są odwołania w definicji schematu do innych definicji schematu. Przykładowo, zilustrowany na rys. 5, obiekt o nazwie *Segment procesu* może być zdefiniowany w schemacie o tej samej nazwie ale zawiera on także element podległy *Specyfikacja personelu*, który posiada własny plik z definicją schematu. A zatem w jednej definicji schematu może znajdować się wiele elementów zdefiniowanych w osobnych, oddzielnych schematach. Fakt ten znacznie utrudnia proces walidacji danych, zwłaszcza przy kaskadowej strukturze, albowiem węzły podległe posiadające własne definicje schematu, też posiadają własne elementy podległe posiadające własne definicje schematu itd.

Ostatnim problemem zidentyfikowanym przez Autorów są rekurencyjne odwołania zawarte w definicjach schematu. Na rys. 5 widać, iż obiekt Segment procesu może się składać z elementu Segment procesu, a zatem w definicji jego schematu muszą być zawarte zapisy odwołujące się do niego samego i pozwalające na takie sformułowanie danych. Rekurencyjność schematów może w skrajnym przypadku powodować nieskończone zagnieżdżenia struktur, co wydaje się być niewskazane w kontekście przeprowadzania walidacji danych.



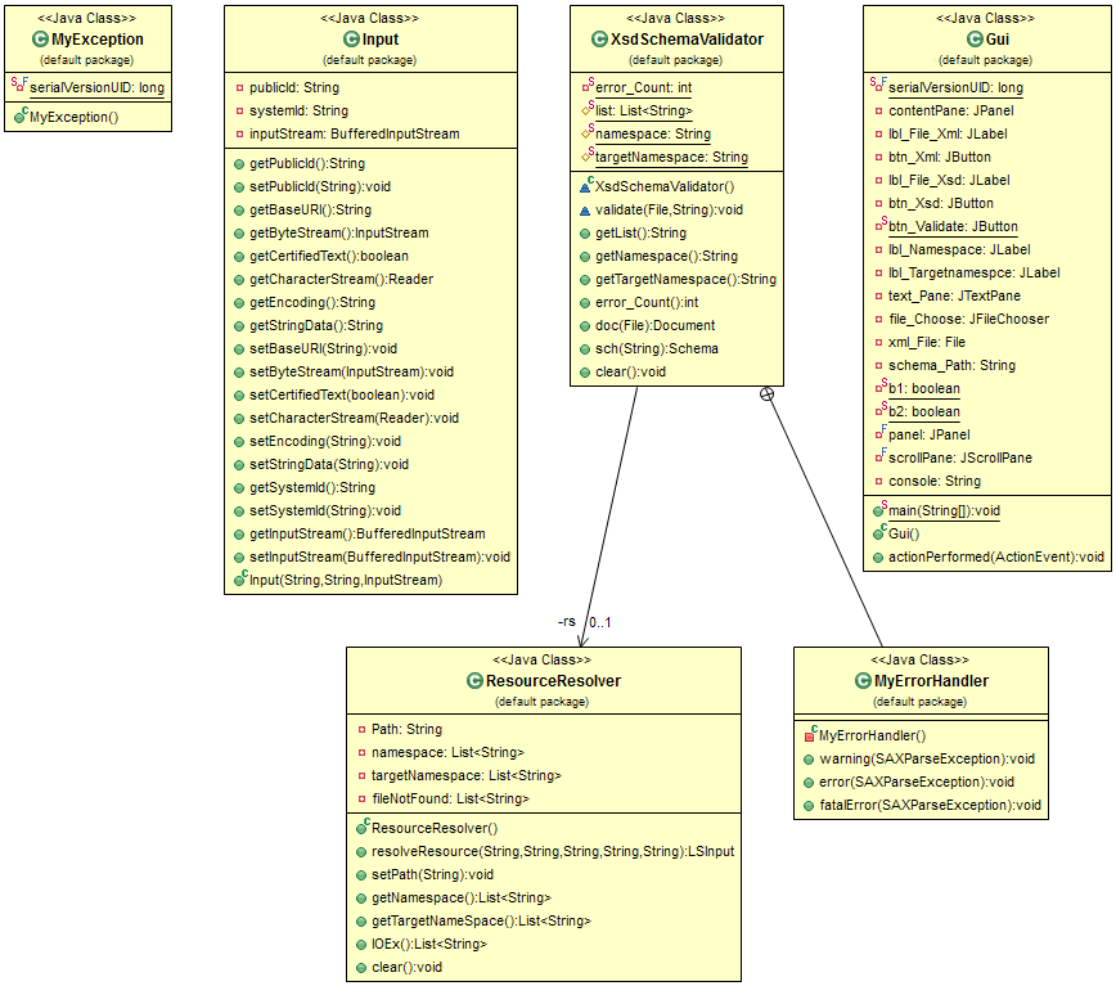
Rys. 5. Przykładowy schemat danych zgodny ze specyfikacją B2MML

5. WALIDATOR

Na potrzeby rozwiązania zdiagnozowanych i opisanych powyżej problemów opracowany został przez autorów algorytm umożliwiający walidację dowolnego pliku XML na podstawie dowolnego pliku XSD zawierającego schemat. W efekcie prac związanych z realizacją ww. celu, utworzony został przez autorów Walidator - oprogramowanie implementujące opracowany algorytm. Aplikacja została napisana w języku Java, a do jej uruchomienia potrzebne są biblioteki wbudowane w JavaSE7. Biblioteki te w pełni wystarczyły zrealizowania celu, nie było potrzeby importowania zewnętrznych bibliotek.

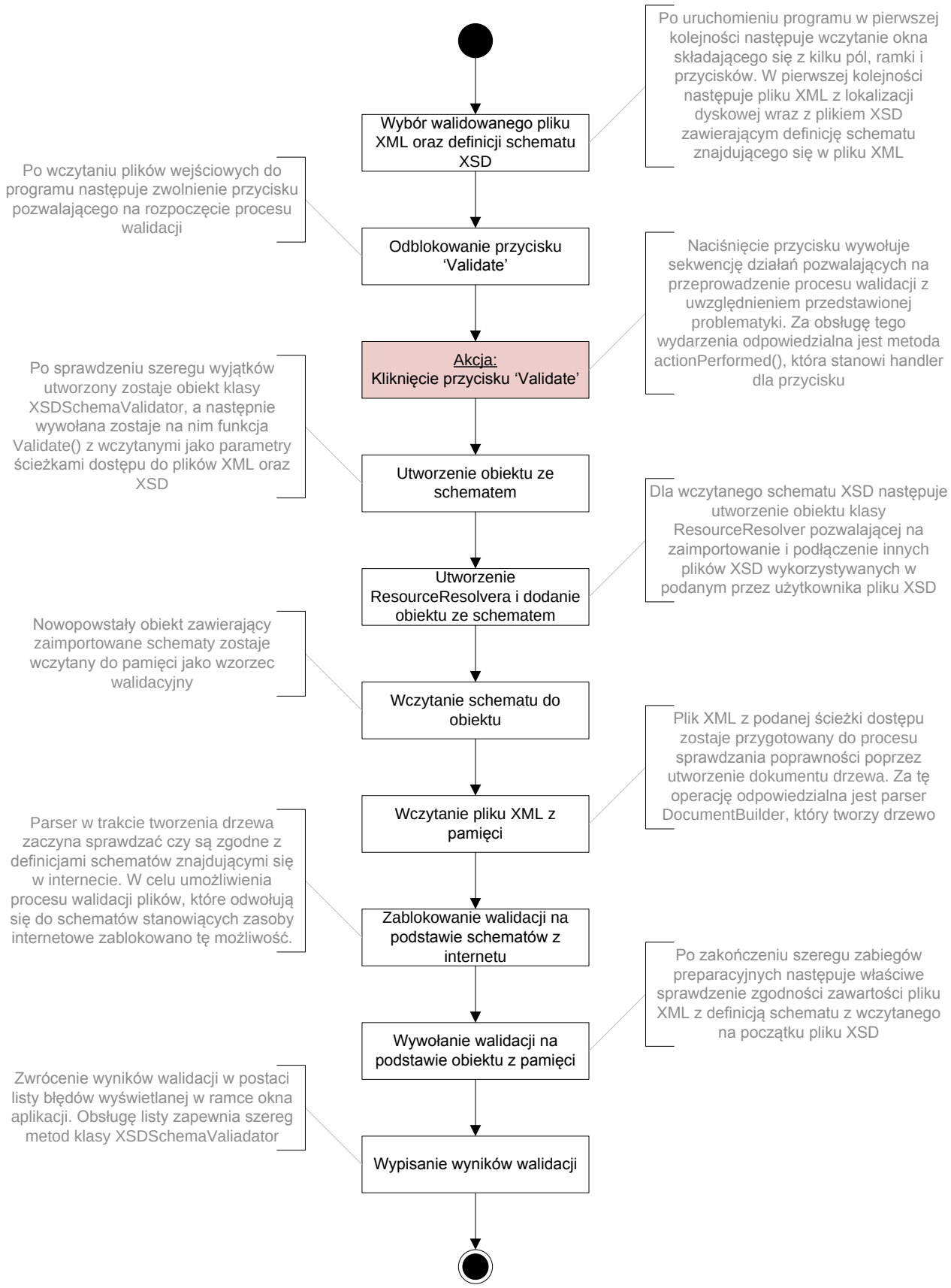
Podczas realizacji oprogramowania przyjęto proste, liniowe działanie programu, co podparte zostało prostym interfejsem. Ilość kodu jest niewielka, większość jest przystępna i dodatkowo opatrzona komentarzem. W celu zapewnienia niezawodności oraz minimalizacji ilości kodu wykonane zostały rzutowania pewnych obiektów. Mogą one budzić wątpliwości, jednak stanowią one irrelewantny element kodu w kontekście celu jego działania i głównego zadania mu powierzonego. Na rys. 6. przedstawiony został diagram klas użytych w utworzonym oprogramowaniu. W programie znaleźć można predefiniowane w bibliotekach Javy klasy pozwalające na wprowadzanie plików, obsługę wyjątków czy

błędów. Najistotniejsze w kontekście realizacji głównego celu działania aplikacji wydają się klasy XsdSchemaValidator oraz ResourceResolver. Pierwsza z nich bezproblemowo radzi sobie ze sprawdzaniem schematów plików XML ale bez zagnieżdżonych struktur danych i odwołań do innych definicji. Klasa ResourceResolver została dodana by obsłużyć liczne zagnieżdżenia i kaskadowość struktur danych.



Rys. 6. Diagram klas walidatora

W trakcie prac nad aplikacją, wszelkie funkcje programu nie związane z jego przeznaczeniem nie stanowiły problemu, w przeciwieństwie do tych funkcjonalności, które odpowiedzialne były za uznanie plików XML za poprawne. W celu ich wyeliminowania, ustalenia przyczyn lub ominięcia problemów dogłębnie został przeanalizowany sposób działania różnych parserów jak i walidatora, skutkiem czego konieczne było dołączenie ResourceResolvera.



Rys. 7. Przebieg procesu walidacji

Już na tym etapie wysnuć można było niniejsze wnioski:

- Narzędzie SAX jako proste API do obsługi xml'owych plików w Javie, przystosowany jest do działania na jednym pliku XML. Jest bardzo szybki, nie zużywa wielu zasobów, ale za to nie ma rozbudowanej obsługi zdarzeń, w związku z czym przy jego użyciu trudno nimi manipulować.

- Biblioteka DOM jako narzędzie potrafiące reprezentować złożone pliki XML w modelu obiektowym lepiej się nadaje do realizacji założonego celu. Zalecane jest przy operacjach na wielu plikach XML. Jest bardziej złożony od SAX i w przypadku chęci dodania możliwości walidacji plików XML, które zawierają część informacji w innych plikach XML jest to lepsze rozwiązanie, ponieważ wystarczy dopisać dla niego ResourceResolvera. Jego znaczącą wadą jest fakt, że zużywa dużo zasobów, w zamian oferując lepszą obsługę zdarzeń występujących w parserze.

- Narzędzie StAX stworzone jako rozwiązanie pomiędzy DOM a SAX, przystosowane głównie do strumieniowego parsingu nie nadawał się do realizacji zadania.

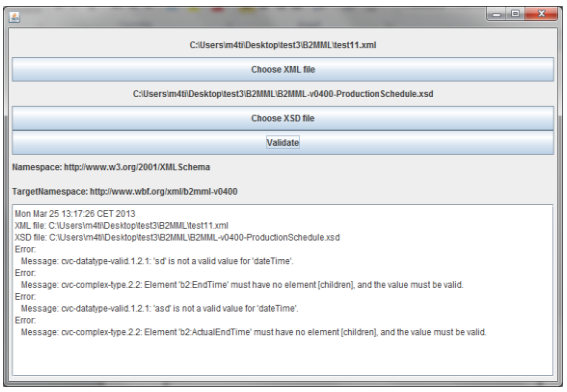
- Dokumentacja udostępniona przez Oracle dotycząca działania jak i używania parserów, wraz z opisem używania udostępnionym przez Google oraz IBM jest ogromna i można w niej znaleźć wiele ciekawych niuansów różniących działanie obydwu parserów.

Graficzny interfejs oprogramowania jest prosty. Składa się z trzech przycisków i pola wyświetlającego wyniki walidacji. Przebieg procesu walidacji i działania programu zarazem został przedstawiony na rys. 7.

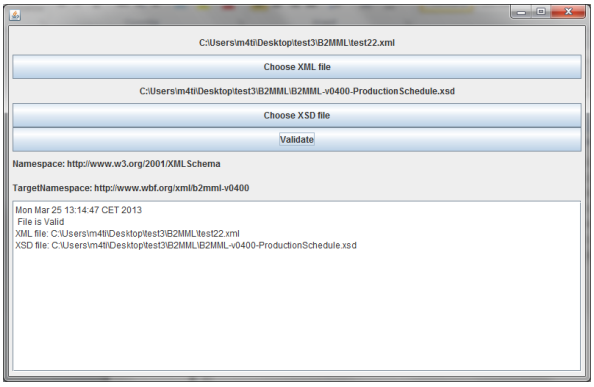
Po uruchomieniu programu pojawia się okno aplikacji. W pierwszej kolejności należy wybrać walidowany plik XML oraz definicję jego schematu. Po wyborze dwóch powyższych plików odblokowuje się przycisk 'Validate', który uruchamia proces sprawdzania poprawności danych.

W polu tekstowym znajdującym się w dolnej części okna pojawiają się wyniki przeprowadzonej walidacji. Na rys. 8 zaprezentowane zostały przypadki, w których pliki nie przeszły walidacji (a) i pomyślnie przeszły walidację (b).

a)



b)



Rys. 8. Okno walidatora po procesie walidacji demonstrujące:
(a) błędy schematów (b) poprawną strukturę

6. PODSUMOWANIE

Utworzone oprogramowanie pozwalające na przeprowadzenie procesu sprawdzania poprawności danych zawartych w plikach XML oraz przy użyciu schematów XSD działa zgodnie z oczekiwaniami. Prezentowane rozwiązanie zostało przetestowane na przykładowych dokumentach XML wygenerowanych przez system MES firmy Wonderware oraz plikach XSD zawierającymi definicje schematów zgodne specyfikacją B2MML. Wyniki działania programu są na obecnym etapie jego rozwoju poprawne. Aplikacja znajduje się w fazie rozwoju i może być optymalizowana pod kątem zużywanej pamięci. Konieczne wydaje się także poprawienie obsługi błędów (np. wyświetlanie numeru linii, w której występuje błąd) pozwalające na lepszą i szybszą diagnozę niezgodności informacji z ich definicjami.

Oprogramowanie uwzględnia obsługę problemów przedstawionych w pkt. 3. Rozwiązanie to wpisuje się w teorię związaną z wymianą danych pomiędzy systemami klasy ERP i MES jako element warstwy pośredniczącej (middleware) pomiędzy systemami. Warstwa takowego oprogramowania zapewniać ma interoperacyjność tychże systemów polegającą nie tylko na wymianie informacji ale wzajemnym ich rozumieniu.

LITERATURA

[1] Cahill, J.: *Integrating Manufacturing Operations with B2MML Standards*, http://www.emersonprocessxperts.com, 2007

[2] Drzewo (informatyka) - pl.wikipedia.org/wiki/Drzewo_(informatyka)

[3] Gould L., *B2MML Explained*. Automotive Design & Production, 2007, http://www.autofieldguide.com/articles/b2mml-explained

[4] Pękala J., Gadzina K.: *Transformacja danych z wykorzystaniem formatu B2MML jako element integracji systemów informatycznych przedsiębiorstwa*, Pomiary, Automatyka, Robotyka, 2/2012, pp. 151-156

[5] Pękala J.: *Data completeness verification in the process of providing the interoperability of production enterprise systems*, Innovations in management and production engineering, Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, Opole 2013, pp. 214-223

[6] Scholten B.: *The road to integration: A guide to applying the ISA-95 Standards in Manufacturing*, ISBN 978-0-9792343-8-5, 2007

[7] Skura K., Smalec Z.: *Integracja systemów informatycznych w automatyzacji procesów produkcyjnych*, Pomiary, Automatyka, Robotyka, 7-8/2005, pp. 6-11.

[8] XML Schema - pl.wikipedia.org/wiki/XML_Schema

[9] XSL Transformations - pl.wikipedia.org/wiki/XSL_Transformations.

[10] Zając J., Chwajoł G.: *Integracja informacji w systemach sterowania wytwarzaniem*. PIAP AUTOMATION'2005, 2005, pp. 96-105