INFORMATYKA EKONOMICZNA BUSINESS INFORMATICS 22 • 2011

ISSN 1507-3858

Radosław Klimek, Paweł Skrzyński
AGH Akademia Górniczo-Hutnicza

ZASTOSOWANIE WERYFIKACJI DEDUKCYJNEJ W PROJEKTOWANIU OPROGRAMOWANIA KORPORACYJNEGO BUDOWANEGO W OPARCIU O PARADYGMAT SOA

Streszczenie: Praca dotyczy zagadnień modelowania i weryfikowania oprogramowania korporacyjnego, budowanego zgodnie z architekturą SOA. Zostały pokazane możliwości budowy takiego oprogramowania w oparciu o środowisko zorientowane na usługi. Przedstawiono zarys odpowiedniej metodyki modelowania. Język modelowania procesów biznesowych BPMN może być uznany za język wizualizacji środowiska usług sieciowych. Procesy takie mogą i powinny być zweryfikowane pod względem formalnej poprawności. Dobre możliwości weryfikacji daje tutaj podejście dedukcyjne z wykorzystaniem logiki modalnej i metody tablic semantycznych jako metody wnioskowania. Przedstawiono możliwości automatycznej budowy specyfikacji logicznej z wzorców projektowych BPMN, co ma kluczowe znaczenie w przypadku budowy modelu logicznego rzeczywistego systemu.

Słowa kluczowe: oprogramowanie korporacyjne, SOA, modelowanie biznesowe, BPMN, weryfikacja formalna, logika temporalna, logika epistemiczna, metoda tablic semantycznych.

1. Wstęp

Coraz większe znaczenie zyskuje *oprogramowanie korporacyjne*. Cechą je wyróżniającą jest to, że każdy jego komponent jest osadzony w ramach infrastruktury technologicznej (*technology infrastructure*), na którą składa się oprogramowanie (systemy operacyjne, API systemowe, bazy danych i katalogi, np. LDAP, programy zarządzające transakcjami, kolejki komunikatów, oprogramowanie warstwy pośredniej – middleware i adaptery, programy zarządzające użytkownikami, technologie zapewniające bezpieczeństwo etc.) oraz sprzęt (serwery, routery i inne narzędzia sprzętowe, zapory ogniowe – firewall, systemy storage, okablowanie etc). Oprogramowanie takie ma na celu, na różnych polach, wspomagać działanie przedsiębiorstwa.

Cechą odróżniającą oprogramowanie korporacyjne od oprogramowania „standalone" (np. edytora tekstu MS Word) jest to, że każdy komponent w ramach tej

infrastruktury jest dostępny dla różnych innych aplikacji/komponentów oraz systemów, przez co staje się dla korporacji zasobem (*resource*). Paradygmat SOA (*Service Oriented Architecture*) reprezentuje model architektury, której celem jest zwiększenie możliwości reakcji na zmiany infrastruktury technologicznej przedsiębiorstwa (dodawanie/zmiana komponentów programowych) przy jednoczesnej redukcji kosztów wprowadzenia tych zmian [Ertl 2008]. Architektura zorientowana na usługi ma umożliwić łatwą integrację różnego rodzaju działalności biznesowej oraz zapewnić interoperabilność heterogenicznych systemów. Cele te osiągnięte są poprzez wprowadzenie abstrakcyjnego pojęcia „usługi" (*service*), poprzez które opisywana jest logika rozwiązania problemu – pojedyncza usługa enkapsuluje w sobie logikę biznesową, która stanowi część rozwiązania problemu.

Oprogramowanie korporacyjne może być budowane z wykorzystaniem paradygmatu SOA. Architektura ta zyskuje coraz większe znaczenie i stanowi istotny kierunek rozwojowy współczesnej informatyki. Celem pracy jest przedstawienie możliwości formalnej weryfikacji architektury dedukcyjnej z wykorzystaniem logiki temporalnej i metody tablic semantycznych jako strategii wnioskowania. Proponuje się wykorzystanie notacji do modelowania oprogramowania korporacyjnego – notacja ta jest powszechnie przyjęta jako narzędzie wizualizacji architektur zorientowanych na usługi. Wnioskowanie dedukcyjne stanowi ważny, alternatywny wobec weryfikacji modelowej (*model checking*) kierunek rozwojowy dla formalnych metod wnioskowania odnośnie do poprawności systemów informatycznych. Logika temporalna jest uznanym narzędziem specyfikacji systemów. Pewne znaczenie może mieć także logika epistemiczna.

2. Architektura zorientowana na usługi

System tworzony w oparciu o *paradygmat SOA* składa się z luźno powiązanych ze sobą usług, które stanowią komponenty wielokrotnego wykorzystania realizujące logikę biznesową, mogące występować w różnych kompozycjach. Cechami charakterystycznymi oprogramowania rozwijanego w oparciu o paradygmat SOA są [Ertl 2008]:

- możliwość wywoływania usług niezależnie od technologii i lokalizacji sieciowej,
- komunikacja jeden do jeden: w danej chwili czasowej konsument usługi (*service consumer*) może wywołać tylko jedną usługę, przy czym komunikacja jest dwukierunkowa,
- przepływ inicjowany jest przez konsumenta usługi – wywołanie usługi,
- komunikacja synchroniczna.

Z punktu widzenia implementacyjnego SOA stanowi kombinację technologii, produktów, specyficznych API etc. Praktyczne wdrożenie takiej architektury jest specyficzne dla danej korporacji, przy czym częścią wspólną może być wykorzystanie platform (np. Java EE) ułatwiających tworzenie oprogramowania w oparciu

o taką architekturę poprzez zapewnienie możliwości tworzenia, wykonania i ewolucji rozwiązań zorientowanych na usługi (*service-oriented solutions*).

Na usługę można patrzyć jako na zbiór funkcjonalności w ramach określonego kontekstu służący rozwiązaniu określonego problemu. Na poziomie koncepcji nie ma żadnych założeń co do sposobu implementacji usługi (nie należy mylić pojęcia usługi z pojęciem usługi sieciowej (*web service*), która może być jednym, aczkolwiek nie jedynym, sposobem implementacji usługi).

Termin kompozycja usług (*service composition*) opisuje zbiór usług, które pozwalają na automatyzację jakiegoś zadania lub procesu biznesowego. Należy zaznaczyć, że termin ten odnosi się do agregatów, które zawierają przynajmniej dwie usługi oraz coś, co nazywane jest inicjatorem kompozycji (*composition initiator*) – w innym przypadku mówimy o komunikacji typu punkt-punkt (*point-to-point exchange*).

Jednym z głównych założeń paradygmatu projektowania oprogramowania zorientowanego na usługi jest takie przygotowanie usług, by umożliwić im efektywne działanie w ramach różnych, złożonych kompozycji.

Efektywna implementacja architektury korporacyjnej powinna brać pod uwagę paradygmat SOA. Zaletą takiego podejścia jest odejście od praktyk „izolacjonistycznych", w których każdy departament w korporacji buduje swój system, często duplikując funkcjonalność, co podnosi koszty, zwiększa czas rozwoju i utrudnia utrzymanie. W paradygmacie SOA system składa się z luźno powiązanych ze sobą usług, które stanowią komponenty wielokrotnego wykorzystania i mogą występować w różnych kompozycjach.

3. Notacja BPMN a paradygmat SOA

BPMN [*Business Process*… 2011] jest standardową notacją graficzną przeznaczoną do opisu procesów biznesowych dostarczoną przez Business Process Management Initiative (BPMI), która została zaakceptowana przez OMG. BPMI współpracuje z organizacją OASIS w zakresie opracowania standardów dla e-biznesu. Organizacja ta jest ogólnoświatowym konsorcjum non-profit zajmującym się opracowywaniem, łączeniem i przyjmowaniem standardów dla e-biznesu. Jej działania obejmują tworzenie standardów bezpieczeństwa, usług WWW, zgodności z XML, transakcji biznesowych, publikacji elektronicznych, map tematycznych oraz zasad bezproblemowej współpracy wewnątrz, a także pomiędzy rynkami zbytu.

Zasadniczym celem powstania BPMN było dostarczenie notacji łatwej i zrozumiałej dla analityków biznesowych, którzy znają i tworzą procesy biznesowe, oraz specjalistów technicznych, którzy odpowiadają za ich implementację i automatyzację, a także kierowników i menedżerów, którzy tymi procesami zarządzają i je monitorują.

Kolejnym celem było zapewnienie możliwości wizualizacji językom opartym na XML, które umożliwiają wykonanie procesów – takich jak BPEL4WS (*Business Process Execution Language for Web Services*). Zanim podjęto próbę standa-

ryzacji notacji dla takich potrzeb, zauważono, że analitycy biznesowi mają tendencję do posługiwania się różnego rodzajami grafami przepływu (*flowcharts*). Z drugiej strony notacja musiała brać pod uwagę rosnącą popularność języków opartych na usługach sieciowych i XML, które umożliwiały automatyczne wykonanie procesów. Języki te były zgodne z paradygmatem strukturalnym, natomiast BPMN musiał mieć jednocześnie korzenie w diagramach grafowych, co było trudne do pogodzenia. Języki takie jak BPEL są zoptymalizowane pod kątem interoperacyjności systemów zarządzania procesami (BPM, *Business Process Management Systems*), co czyni je trudnymi do zrozumienia dla człowieka niebędącego specjalistą technicznym. W związku z tym pojawił się rozdźwięk pomiędzy tym, co było używane przez analityków biznesowych do opisu procesów (nieformalne notacje grafowe), oraz językami używanymi przez techników do implementacji systemów. Notacja BPMN stanowi nie tylko próbę zasypania tej luki, ale uwzględnia też ludzki poziom interoperabilności, który nie był uwzględniany przez języki typu BPEL, nastawione na interoperabilność systemów informatycznych.

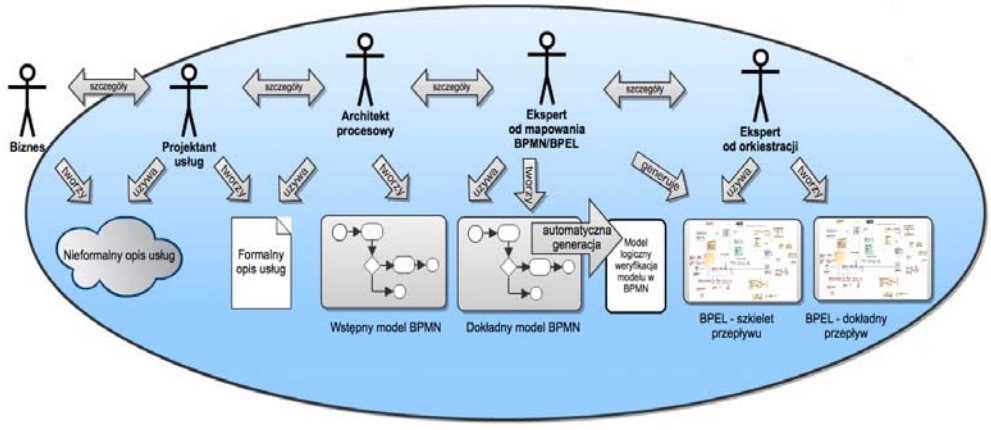
4. Metodyka modelowania

Wymienione cechy sprawiają, że BPMN został zaadaptowany do proponowanej metodyki rozwijania oprogramowania korporacyjnego. Dodatkowo, w celu ułatwienia tworzenia modelu BPM, postuluje się przed przystąpieniem do opisu procesów biznesowych w notacji BPMN dokonanie próby stworzenia specyfikacji tekstowej tych procesów. Opis taki dostarcza informacji poglądowej o procesie i stanowi również cenną dokumentację. Wadą opisów tekstowych jest brak formalizmu, który uniemożliwia ich późniejszą automatyzację. Opis tekstowy powinien zawierać następujące informacje:

- cel i krótka charakterystyka procesu;
- wymagane zasoby;
- wytwarzane dokumenty;
- warunki początkowe niezbędne do uruchomienia procesu;
- warunki końcowe;
- główny ciąg zdarzeń;
- systemy, osoby niezbędne do prawidłowego przebiegu procesu (w tym osoby oraz systemy, z którymi proces się komunikuje).

Zebranie tych informacji od osób posiadających wiedzę biznesową nie jest trudne, a wydatnie przyczynia się do ułatwienia stworzeniu formalnego opisu procesu w notacji BPMN – jest to kolejny krok w łańcuchu wytwarzania oprogramowania wprowadzonym przez metodykę. Proces modelowania BPMN otrzymuje na wejściu specyfikację tekstową oraz inne specyfikacje, np. w postaci WSDL opisujących usługi sieciowe wystawiane przez systemy, z którymi proces się komunikuje. Proces tworzenia modelu w notacji BPMN proponuje się przeprowadzić w dwóch fazach:

- W pierwszej fazie tworzony jest wstępny diagram/diagramy będące odzwierciedleniem opisu tekstowego. Diagram na wyjściu tej fazy powinien zawierać tylko podstawowe symbole BPMN (tzw. Core BPMN Elements), dzięki czemu jest on łatwy w zrozumieniu przez analityków biznesowych, którzy powinni dokonać weryfikacji stworzonego modelu,
- W drugiej fazie diagramy BPMN są uszczegóławiane poprzez zastosowanie rozszerzonych elementów notacji BPMN (np. zdarzenia startowe pośrednie, końcowe, typy aktywności etc.) oraz uwzględnienie dodatkowych informacji (np. WSDL). Diagramy powstałe w tej fazie mają w pełni wykorzystywać możliwości ekspresji, jakie daje BPMN, w związku z tym są one w pełni zrozumiałe już tylko dla ekspertów biznesowych. W tej fazie dokonywana jest formalna analiza i weryfikacja stworzonego modelu poprzez zastosowanie logik modalnych i wnioskowania dedukcyjnego, która jest także przedmiotem niniejszego artykułu.



Rys. 1. Proces wytwarzania oprogramowania korporacyjnego

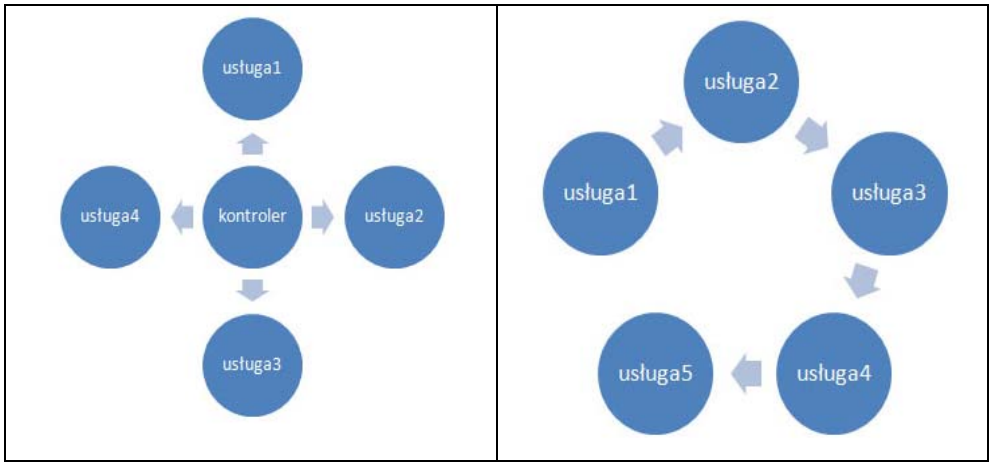
Źródło: opracowanie własne.

Stworzone w drugiej fazie modelowania diagramy BPMN umożliwiają automatyczne wygenerowanie opisu procesów w języku wykonywalnym, jakim jest BPEL – język ten jest dla SOA tym, czym dla relacyjnych baz danych język SQL [Juric, Krizevnic 2010]. Proces ten może, lecz nie musi, przebiegać w sposób automatyczny ze względu na fakt, że model BPMN zawsze może zostać skonwertowany do odpowiadającego modelu w BPEL oraz ze względu na to, że nie jest to model w pełni formalny (w odróżnieniu od BPEL). Kolejnym krokiem omawianego procesu wytwarzania oprogramowania jest orkiestracja BPEL. Proces ten na wejściu otrzymuje szablon BPEL wygenerowany ze stworzonego modelu zapisanego w notacji BPMN i jego celem jest uzupełnienie kodu, które umożliwi jego automatyczne wykonanie na silniku procesowym. W trakcie generowania szablonu

bardzo często zdarza się, że stworzone zostały puste zadania (*tasks*), których reprezentacja w BPMN nie była możliwa. Ponadto konieczna może być implementacja przepływów komunikatów (*message flows*) oraz stworzenie (lub transformacja) niezbędnych struktur danych. Ze względu na specyfikę BPEL zalecanym sposobem realizacji komunikacji jest technologia usług sieciowych (*web services*). Proces opisany jest na rys. 1.

5. Orkiestracja i aranżacja modeli

Usługi sieciowe w środowisku SOA odgrywać będą w działalności firm coraz większą rolę, można spodziewać się w tym zakresie współpracy wielu odbiorców i partnerów biznesowych, a rzeczywiste przyspieszenie nastąpi wówczas, gdy firmy zaczną integrować swoje usługi. Generalnie, istnieją dwa zasadnicze sposoby realizacji usług sieciowych, a są nimi *orkiestracja* (lub aranżacja) i *choreografia* usług, por. rys. 2.



Rys. 2. Orkiestracja a choreografia w usługach sieciowych

Źródło: opracowanie własne.

Aranżacja jest stosowana najczęściej i opisuje sekwencję kroków, jakie muszą być wykonane, aby zrealizować cały proces. Sekwencja tych kroków, składających się zarówno z wewnętrznych, jak i zewnętrznych usług, jest kierowana i zarządzana przez punkt centralny, kontroler, który odpowiada za realizację całości przedsięwzięcia. Choreografia zapewnia odmienny sposób realizacji usług sieciowych, który nie wymaga już obecności kontrolera sterującego realizacją całości przedsięwzięcia. Wynika jednak z tego, że poszczególne procesy posiadają wiedzę w zakresie realizacji poszczególnych usług, usług składających się na całość, przy czym po zakończeniu danej usługi realizowana jest usługa kolejna, wynikająca m.in. z wiedzy na temat całego przedsięwzięcia.

Orkiestracja jest w praktyce stosowana częściej, ale wymaga kontroli nad wszystkimi aspektami procesów biznesowych, z uwzględnieniem podejścia zstępującego. Inną zaletą jest tutaj względnie łatwe odwzorowania procesów w architekturę SOA. Z drugiej jednak strony trudności mogą się pojawić w trakcie skalowania w przypadku złożonych i skomplikowanych procesów, aczkolwiek taki model przetwarzania jest wspierany przez wiele istniejących na rynku narzędzi. Choć orkiestracja jest zrozumiała i popularna, to jednak nie odwzorowuje wszystkich możliwych sytuacji przetwarzania. Choreografia poprzez odmienny sposób organizacji może z kolei być przydatna w realizacji systemów opartych na zdarzeniach i obsługiwanych przez agentów. Zachowanie wynikowe jest rezultatem działania wielu indywidualnych fragmentów systemu, z których każdy samodzielnie decyduje o przebiegu obliczeń i samodzielnie planuje obliczenia. Ważnym elementem tego jest też możliwość indywidualnego i samodzielnego doboru parametrów konfiguracyjnych poszczególnych procesów. Podejście choreograficzne jest przydatne także wtedy gdy procesy zmieniają się dynamicznie i trudne jest określenie z góry celu konkretnego działania, a więc odmiennie niż w orkiestracji, gdzie procesy są względnie statyczne i możliwe do zdefiniowania *a priori*. Dokładniejsze informacje na ten temat można znaleźć np. w pracy [Maréchaux].

6. Aparat logiczny – logiki modalne

Sformalizowanie procesu myślenia i wnioskowania jest niezbędne w wielu obszarach, także w projektowaniu i badaniu systemów informatycznych. Doświadczenie i intuicja, chociaż bardzo przydatne w tworzeniu systemów, to jednak nie mogą być tutaj uznane za wystarczające. Pewną analogią do tej sytuacji może być intuicyjna znajomość języka w porównaniu ze znajomością języka wraz z jego strukturami gramatycznymi – ta druga sytuacja jest cenniejsza i zwiększa wiarygodność posługiwania się nim oraz weryfikacji poprawności odpowiednich struktur i wypowiedzi. W sformalizowaniu procesu projektowania systemów informatycznych ważną rolę odgrywa logika formalna, przy czym szczególnie dogodna wydaje się tutaj logika modalna.

Logika modalna pozwala na badanie pojęcia konieczności i możliwości. Istotny i charakterystyczny w logice jest operator modalny, który powoduje, że sama znajomość prawdziwości zdania poprzedzonego takim operatorem nie jest wystarczająca do stwierdzenia prawdziwości logicznej całego zdania z operatorem (funktorem). Takie podejście do modalności nosi miano modalności logicznej jako tej, która odnosi się do prawdziwości zdań. Z kolei sposób interpretacji operatorów modalnych ma wpływ na dziedzinę rozważań, którymi mogą być np. zagadnienia związane z upływającym czasem i następstwami czasowymi, co prowadzi do logiki temporalnej, jak i zagadnienia dotyczące wiedzy wraz z możliwością i pewnością informacji dotyczącej tej wiedzy i prowadzące do logiki epistemicznej. Oba te ro-

dzaje interpretacji, w ogólności nie jedyne zresztą, mogą być przedmiotem zainteresowania na gruncie analizy i weryfikacji systemów i architektur SOA, a także modeli biznesowych.

Logika temporalna [Klimek 1999] umożliwia opisywanie zależności i następstw czasowych zdarzeń świata dyskretnego. Procesy biznesowe modelowane np. w języku BPMN, a następnie odwzorowane do języka wykonawczego BPEL dla środowiska SOA mogą wyrażać zarówno aspekty żywotnościowe, jak i bezpieczeństwa modelowanej architektur systemu. Przykładowo, takimi formułami dla usług sieciowych mogą być:

$$\Box \neg (\text{web-service1} \wedge \text{web-service2})$$
$$\Diamond \text{ web-service}$$
$$\Box (\text{web-service1} \Rightarrow \Diamond \text{web-service2})$$
$$\Diamond \text{ web-service1} \Rightarrow (\neg \text{ action } U \text{ trigger})$$

We wszystkich przypadkach formuł mamy operatory jednoargumentowe, z wyjątkiem drugiego operatora ostatniej formuły. Formuły opisują własności żywotnościowe, a więc sytuację zajścia pewnego dobrego zdarzenia. Pierwsza formuła natomiast wyraża bezpieczeństwo systemu poprzez żądanie niewystąpienia pewnego złego zdarzenia.

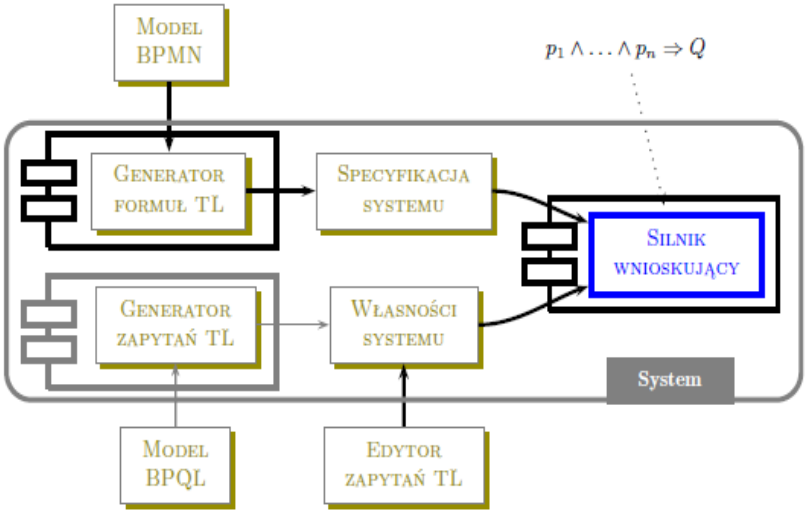
Logika epistemiczna odnosi się do wiedzy i przekonania. Dziedziną może być tu oczywiście wiedza o realizacji usług sieciowych, w szczególności usług sieciowych realizowanych z paradygmatem choreografii. Podstawowym funktorem modalnym jest tu dwuargumentowy $K(a,\alpha)$, co interpretujemy, że agent *a* wie (ma wiedzę), że α. Alternatywnie możemy także powiedzieć, że we wszystkich możliwych światach (stanach obliczeń) agent *a* wie, że α. Funktor możemy też traktować jako jednoargumentowy, oczywiście o ile nie prowadzi to do wątpliwości, o jakiego agenta chodzi, i wówczas zapisujemy $K\alpha$. Należy zwrócić uwagę, że w logice epistemicznej zawsze prawdziwe jest zdanie $Kp \rightarrow p$. Podstawowym założeniem logiki jest podział możliwych światów na dwa: te, które są zgodne z postawionym pytaniem, i te, które są z nim niezgodne. Zdanie $\sim K \sim p$ oznacza, że nie jest tak że wiadomo, że $\sim p$, w konsekwencji możemy zdefiniować, że $Mp = \sim K \sim p$. Operator *K* jest dualny do operatora *M*.

W dalszych rozważaniach, aby pokazać możliwości formalnej analizy i weryfikacji systemów, ograniczymy się do logiki temporalnej.

7. System dedukcyjny

Dla wspomnianych w poprzednim punkcie obu logik modalnych można zbudować skuteczny system wnioskowania. Wnioskowanie bazuje na metodzie tablic seman-

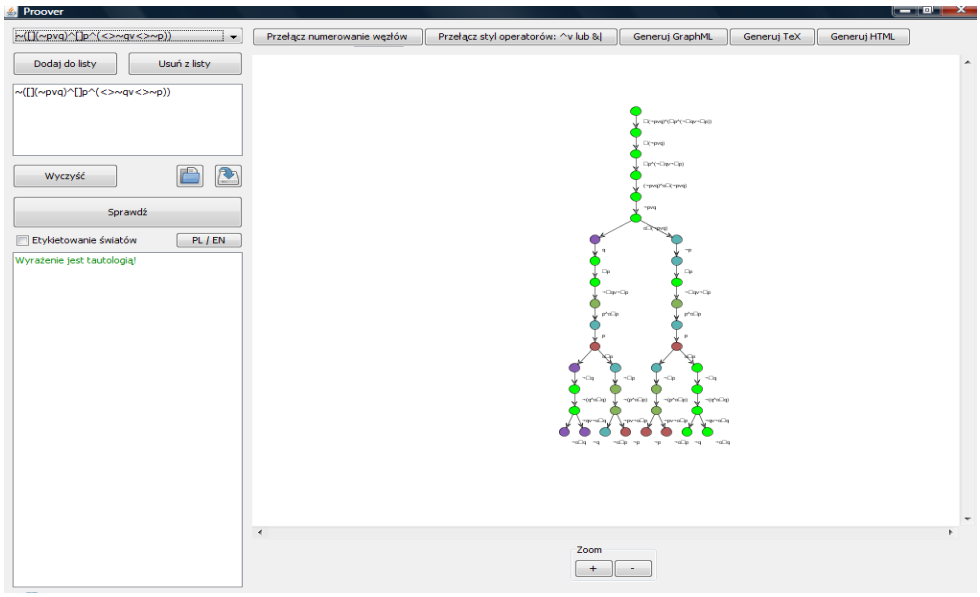
tycznych, która jest możliwa do zastosowania w przypadku obu logik [D'Agostino i in. (eds) 1999]. *Metoda tablic semantycznych* polega, w pewnym uproszczeniu, na umieszczeniu w korzeniu drzewa wnioskowania badanej formuły, a następnie na podstawie dobrze zdefiniowanych reguł postępowania dekompozycji formuły na elementy mniejsze, aż do osiągnięcia zdań elementarnych. Kolejnym krokiem jest badanie wszystkich gałęzi drzewa. Jeżeli w gałęzi są umieszczone zdania elementarne komplementarne, to oznacza to, że gałąź taka zawiera sprzeczności i traktujemy ją jako gałąź zamkniętą. Jeżeli wszystkie gałęzie są zamknięte, to oznacza to, że formuła umieszczona w korzeniu nie zawiera wartościowania spełniającego formułę. Ponieważ metoda tablic semantycznych jest apagogiczna, a więc w korzeniu umieszczono zaprzeczenie formuły badanej, oznacza to, że formuła początkowa jest prawdziwa. Dokładniejsze omówienie metody wnioskowania metodą tablic semantycznych, w kontekście modeli biznesowych, wraz z odpowiednim przykładem można znaleźć w pracy [Klimek 2011].



Rys. 3. System wnioskujący

Źródło: opracowanie własne.

Proponowany system wnioskowania metodą tablic semantycznych został przedstawiony na rys. 3, wcześniej także w pracy [Klimek i in. 2010]. System składa się z kilku komponentów. Pierwszy zapewnia funkcjonalność generującą specyfikację logiczną, rozumianą jako zbiór formuł logiki temporalnej. Formuły są generowane bezpośrednio ze wzorców projektowych BPMN. Powiązanie ich później symbolem koniunkcji, wraz z żądaniem spełnienia pewnej własności Q, daje ogólną specyfikację systemu: p1 ∧ p2 ∧ … ∧ pn => Q. Zarówno specyfikacja systemu, jak i badane własności stanowią wejście dla silnika wnioskującego, który



Rys. 4. Prototypowy silnik wnioskujący

Źródło: opracowanie własne.

realizuje proces dedukcyjny metodą tablic semantycznych. Na rysunku 4 pokazano jeden z wykonanych[1], prototypowych silników wnioskujących.

8. Automatyczne generowanie specyfikacji logicznej

W odniesieniu do przedstawionej koncepcji weryfikacji metodą dedukcyjną kluczowe znaczenie ma zbudowanie specyfikacji logicznej badanego systemu. Specyfikację stanowi zbiór formuł logiki temporalnej opisujących własności systemu. Automatyzacja procesu pozyskiwania specyfikacji ma w praktyce podstawowe znaczenie. Bazuje ona *wzorcach projektowych* BPMN (por. np. [van der Aalst i in. 2003]) w ten sposób, że każdy z 23 znanych wzorców ma przypisany zbiór predefiniowanych formuł logiki temporalnej dobrze opisujący własności wzorca. Proces budowy specyfikacji obejmuje następujące kroki postępowania (por. [Klimek 2011]):

1) analiza całego modelu biznesowego w celu wydzielenia wszystkich występujących wzorców projektowych,

2) translacja modelu biznesowego do tzw. *wyrażenia logicznego* W, które zawiera wszystkie wykorzystane w modelu wzorce, wraz z ich zagnieżdżeniami,

[1] Projekt wykonany przez studenta Przemysława Eliasa i innych (AGH Kraków, kierunek informatyka stosowana), pod kierunkiem autorów pracy.

3) wygenerowanie na podstawie wyrażenia logicznego, oraz predefiniowanego zbioru P formuł logiki temporalnej przypisanych do poszczególnych wzorców, *specyfikacji logicznej* L, a więc zbioru formuł logiki temporalnej opisujących cały model, przy czym generowanie takie odbywa się wg odpowiedniego algorytmu uwzględniającego zagnieżdżenia wzorców.

Generowanie specyfikacji logicznej może odbywać się wielokrotnie, por. rys. 3, a więc gdy tylko zajdzie taka potrzeba i zostanie zmodyfikowany analizowany model systemu.

9. Wnioski

Przedstawiono problem modelowania architektury korporacyjnej z wykorzystaniem paradygmatu SOA. Zaproponowano odpowiednią metodykę modelowania. Oprogramowanie korporacyjne odgrywa kluczową rolę w działalności firmy i dlatego istotne jest formalne zweryfikowanie poprawności modelu, co przekłada się bezpośrednio na jakość stworzonego oprogramowania, natomiast wczesne wykrycie błędów, które zapewnia przedstawiona metodyka, pozwala na redukcję kosztów jego wytworzenia. Weryfikacja odbywa się z wykorzystaniem podejścia dedukcyjnego, logiki temporalnej i logiki epistemicznej oraz wnioskowania metodą tablic semantycznych. Metodyka, zdaniem autorów, znakomicie więc wpisuje się w podstawowe założenia paradygmatu SOA.

Literatura

Business Process Modelling Notation, V 2.0, OMG Document Number: formal/2011-01-03, URL: http://www.omg.org/spec/BPMN/2.0.

Ertl T. [2008], *SOA Design Patterns*, Prentice Hall. New Jersey.

D'Agostino M., Gabbay D.M., Hahnle R., Posegga J. (eds) [1999], *Handbook of Tableau Methods*, Kluwer Academic Publishers, Dordrecht.

Juric M.B., Krizevnik M. [2010], *Overview of WS-BPEL 2.0 for SOA Composite Applications with Oracle SOA Suite 11g*, Packt Publishing. http://www.packtpub.com/sites/default/files/7948-chapter-9-bpel-with-oracle-service-bus-and-service%20.pdf.

Klimek R. [1999], *Wprowadzenie do logiki temporalnej*, Wydawnictwa Naukowo-Techniczne AGH, Kraków.

Klimek R. [2011], *Weryfikacja procesów biznesowych metodą dedukcyjną z wykorzystaniem logiki temporalnej*, „Pomiary Automatyka Robotyka", nr 12, s. 190–193.

Klimek R., Skrzyński P., Turek M. [2010], *Deduction based verification of business models*, [w:] *Advanced Information Technologies for Management*, eds. J. Korczak, H. Dudycz, M. Dyczkowski (Research Papers of Wroclaw University of Economics no. 147), Publishing House of Wroclaw University of Economics, Wrocław, s. 173–188.

Maréchaux J.L., *Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus*, IBM technical paper, http://www.ibm.com/developerworks/library/ws-soa-eda-esb/.

Rosen M. [2008], *Orchestration Or Choreography*?, www.bptrens.com, April.
van der Aalst W.M.P., ter Hofstede A.H.M., Kiepusewski B., Barros A.P. [2003], *Workflow Patterns*, "Distributed and Parallel Databases", 4(1), s. 5–51.

DEDUCTION BASED FORMAL VERIFICATION OF SOA COMPLIANT ENTERPRISE SOFTWARE

Summary: The work concerns the issues of modelling and verification of enterprise software which is built according to SOA principles. The possibility of building such software is shown. An appropriate modelling methodology is outlined. Business process modelling notation (BPMN) can be considered as a visualization language of web services. Such processes can and should be verified in a formal way. Deductive approach using modal logic and semantic tableaux method gives a good opportunity for such formal verification. The possibility of automatic construction of a logical specification is presented. This automatic construction is based on design patterns of BPMN.

Keywords: enterprise software, Service Oriented Architecture, business modeling, BPMN, formal verification, temporal logic, epistemic logic, semantic tableaux method.