WebML, UML, MDE,
aplikacje internetowe

ŻYŁA Kamil[1]

WEBML I UML JAKO NARZĘDZIA PROJEKTOWANIA APLIKACJI INTERNETOWYCH

Niniejszy artykuł przedstawia najbardziej znaczące różnice pomiędzy notacją WebML oraz UML w dziedzinie projektowania aplikacji internetowych zarządzających dużą ilością danych. Poruszana problematyka dotyczy specyfikacji wymagań, a także warstwy danych oraz hipertekstu aplikacji.

WEBML AND UML AS THE TOOLS FOR DESIGNING DATA-INTENSIVE WEB APPLICATIONS

This paper presents the most significant differences between WebML and UML in the domain of designing data-intensive web applications. It also describes issues regarding application requirements, data layer and hypertext layer specification.

1. WSTĘP

Obecnie obserwuje się duży nacisk na szybkość i jakość wytwarzania oprogramowania oraz jego powszechną dostępność, z uwzględnieniem zagadnień bezpieczeństwa, skalowalności, użyteczności oraz wydajności. Dotyka to zarówno procesu implementacji oprogramowania, jak i poniekąd jego projektowania. Kolejnym trendem jest powszechne wykorzystanie aplikacji internetowych do świadczenia wszelkiego rodzaju usług, począwszy od aplikacji mapowych, przez współbieżną pracę z dokumentami, a skończywszy na e-bankingu. [2]

Na przeciw przedstawionym trendom wychodzi notacja WebML, będąca częścią konceptu inżynierii sterowanej modelami. Jest ona warstwą abstrakcji wywodzącą się z UML, lecz dedykowaną stricte aplikacjom internetowym. Umożliwia tworzenie łatwych do zrozumienia modeli przedstawiających: strukturę warstwy danych, rozmieszczenie treści i operacji pomiędzy stronami oraz metody nawigacji pomiędzy nimi. [1]

2. SPECYFIKACJA WYMAGAŃ

W ramach procesu specyfikacji wymagań, w przypadku WebML, wyróżnia się dwie fazy – zbieranie informacji oraz ich analizę. Produktem całości procesu, stanowiącym

[1] Politechnika Lubelska, Instytut Informatyki, ul. Nadbystrzycka 36 b, 20-618 Lublin, Polska, tel. +(48-81) 525 20 46, fax: +(48-81) 538 43 49, e-mail: kamilz@cs.pollub.pl

minimalny zestaw specyfikacji potrzebny do zaprojektowania i implementacji aplikacji internetowej, jest następujący zestaw dokumentów: specyfikacja grup użytkowników, specyfikacja przypadków użycia, specyfikacja słownika obiektów, specyfikacja widoków witryny, specyfikacja wyglądu aplikacji i specyfikacja wymagań niefunkcjonalnych. [1]

Specyfikacja wyglądu aplikacji jest zbiorem wytycznych dotyczących pozycjonowania oraz wyglądu elementów stron oraz szkicem ich rozmieszczenia na stronach aplikacji. Z kolei specyfikacja wymagań niefunkcjonalnych dotyczy przede wszystkim zagadnień związanych z wdrożeniem i architekturą aplikacji. W ich przypadku WebML nie wprowadza zmian – specyfikacja odbywa się zgodnie ze sprawdzonymi metodykami.

Specyfikacja grup użytkowników zawiera tekstową charakterystykę grup użytkowników korzystających z aplikacji, zakres ich uprawnień oraz hierarchię. Notacja UML może być wykorzystana, np. do zobrazowania hierarchii użytkowników.

Specyfikacja przypadków użycia ma za zadanie zobrazowanie interakcji pomiędzy aplikacją i jej użytkownikami. WebML w całości zaadaptował diagram przypadków użycia wraz ze scenariuszami. Dodatkowo zalecane jest tworzenie diagramów sekwencji, jako diagramów pomocniczych, w przypadku skomplikowanych przypadków użycia.

Specyfikacja słownika obiektów zawiera charakterystykę obiektów istotnych z punktu widzenia aplikacji, w tym ich komponentów i powiązań. W jej przypadku można użyć diagramu klas np. w celu uzyskania sformalizowanego obrazu powiązań pomiędzy obiektem a jego komponentami.

Specyfikacja widoków witryny opisuje obszary aplikacji umożliwiające realizację zidentyfikowanych przypadków użycia. Zastosowanie UML w ramach tego opisu jest raczej zbędne, aczkolwiek zależy od potrzeb projektanta. [3]

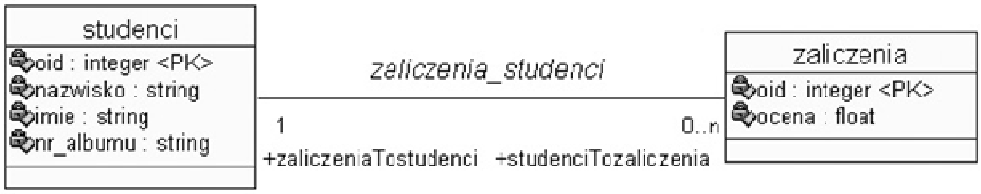
3. WARSTWA DANYCH APLIKACJI

Elementy modelu danych WebML pod względem reprezentacji są zbliżone do notacji UML (rys. 1, rys. 2), co nie powinno stanowić utrudnienia dla projektanta tworzącego diagramy klas lub diagramy związków encji. Siła UML tkwi w szczegółowości oraz możliwościach opisu, jednakże należy liczyć się ze znaczącym stopniem skomplikowania modelu.

Rys. 1. Model danych WebLM

Semantyka na poziomie wystarczającym do modelowania warstwy danych aplikacji internetowych jest bardzo podobna. Relacje WebML odpowiadają asocjacjom UML i są charakteryzowane przez role, nazwy ról i liczność. Koncept generalizacji i specjalizacji nie

różni się pomiędzy notacjami. Większe różnice pojawiają się w przypadku klas i encji. Koncept klasy generalizuje koncept encji, co pozwala na specyfikowanie nie tylko atrybutów, ale również metod operujących na instancjach klasy. [1]

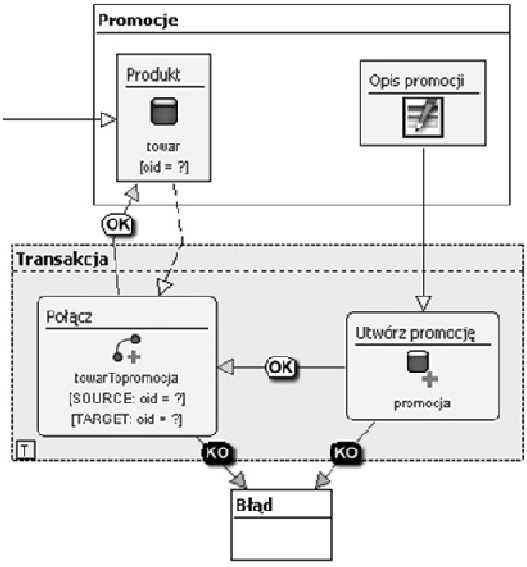


Rys. 2. Model danych UML

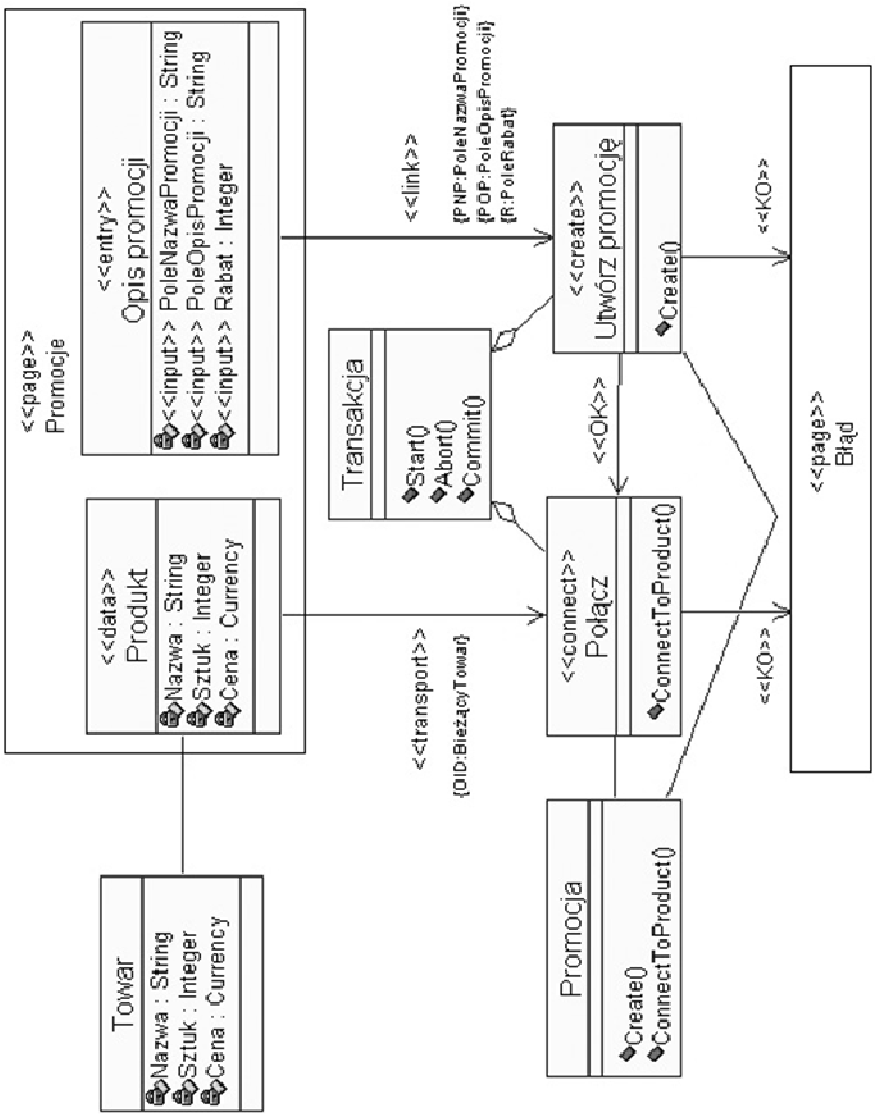
4. WARSTWA HIPERTEKSTU APLIKACJI

UML pozwala na budowanie modeli o wysokiej szczegółowości przedstawiających działanie i strukturę aplikacji. Jednakże w przypadku inżynierii sterowanej modelami, gdzie dąży się do ukrywania nieistotnych szczegółów implementacyjnych, może okazać się to zbędne. W przeciwieństwie do MDE, w ramach klasycznego procesu wytwarzania oprogramowania szczegółowy opis jest niemalże wymagany. W związku z tym wybór pomiędzy UML a WebML, to wybór pomiędzy precyzją opisu a prezentacją ogólnej koncepcji działania aplikacji. [1]

W celu zilustrowania omawianej problematyki na rysunku 3 i 4 przedstawiono stronę aplikacji internetowej służącą do zapisywania w bazie danych (w ramach transakcji) nowych promocji na wybrane produkty.



Rys. 3. Projekt strony aplikacji w notacji WebML



Rys. 4. Projekt strony aplikacji w notacji UML

Komponenty notacji WebML prezentujące treść mogą być reprezentowane w notacji UML w następujący sposób [1]:

1. Widoki witryny i obszary są reprezentowane jako zagnieżdżone pakiety.
2. Strony są reprezentowane jako klasyfikatory o stereotypie <<page>>.
3. Komponenty prezentacji danych są reprezentowane jako klasy umieszczane wewnątrz konstruktu strony i oznaczane stereotypem będącym nazwą typu komponentu WebML.

4. Encja źródłowa i selektor komponentu są reprezentowane jako asocjacja, pomiędzy klasą reprezentującą komponent WebML a klasą reprezentującą encję źródłową, opisana przez wyrażenie OCL będące warunkiem selektora.
5. Linki pomiędzy stronami lub komponentami aplikacji są reprezentowane jako skierowane asocjacje opatrzone opisem.

Komponenty notacji WebML wykonujące operacje mogą być reprezentowane w notacji UML w następujący sposób [1]:

1. Komponenty operacji są reprezentowane jako klasy, stanowiące interfejs pomiędzy komponentem inicjującym operację a encjami będącymi jej przedmiotem i oznaczane stereotypem będącym nazwą typu komponentu WebML. Klasa reprezentująca operację udostępnia na zewnątrz tylko jedną metodę.
2. Komponenty logowania, wylogowania i poczty są reprezentowane w postaci klas posiadających jedną metodę zgodną z nazwą komponentu.
3. Linki wychodzące z komponentów operacji są reprezentowane jako skierowane asocjacje opatrzone opisem.
4. Transakcje są reprezentowane jako klasy posiadające metody *start()*, *commit()* i *abort()* oraz połączone asocjacjami (w sensie kompozycji) z klasami operacji.

5. WNIOSKI

Z punktu widzenia klasycznego podejścia do tworzenia oprogramowania, użyteczność notacji WebML jest dyskusyjna. Może ona służyć jako: uzupełnienie tworzonej dokumentacji, metoda uproszczonej prezentacji koncepcji aplikacji skierowana do uczestników procesu wytwarzania oprogramowania lub narzędzie tworzenia dokumentacji zorientowanej na użytkownika. Ostatnie zastosowanie jest szczególnie istotne, ponieważ umożliwia porozumienie na płaszczyźnie deweloper-klient, który nie zawsze jest zaznajomiony z wykorzystywanymi przez dewelopera technologiami [3]. UML z kolei jest używany przede wszystkim w fazie projektowania aplikacji.

Z punktu widzenia wytwarzania oprogramowania sterowanego modelami, użyteczność UML jest dyskusyjna. Największy zysk z jego wykorzystania można uzyskać na etapie specyfikacji wymagań, gdzie uzupełnia on niedobory notacji WebML, służąc jako dodatkowe narzędzie opisu istotnych aspektów aplikacji. Przewagą WebML, na dalszych etapach tworzenia oprogramowania, jest możliwość generowania w pełni funkcjonalnych rozwiązań na podstawie zbudowanych modeli. [4]

6. BIBLIOGRAFIA

[1] Ceri S., Fraternali P., Bongio A., Brambilla M., Comai S., Matera M.: *Designing Data-Intensive Web Applications*, Morgan Kaufmann, 2002

[2] Żyła K., Kęsik J.: *Usability comparison of WebRatio and symfony for educational purposes*, Prace Instytutu Elektrotechniki, zeszyt 247, IEL, Warszawa 2010

[3] Żyła K., Kęsik J.: *Creation of the user-oriented requirements specification of the data intensive web application basing on WebML*, Informatyk Zakładowy, Lublin 2010

[4] Żyła K., Kęsik J.: *Zintegrowane środowisko programistyczne IDE WebRatio 5.1 wspomagające nauczanie MDE*, Logistyka 6/2010, Instytut Logistyki i Magazynowania, Poznań 2010