Rozdział

Metody projektowania aplikacji internetowych

Włodzimierz Dąbrowski
Politechnika Warszawska, Polsko-Japońska Wyższa Szkoła Technik Komputerowych, wlodek@pjwstk.edu.pl

Rafał Hryniów
Polsko-Japońska Wyższa Szkoła Technik Komputerowych; rhryniow@pjwstk.edu.pl

Tomasz Pieciukiewicz
Polsko-Japońska Wyższa Szkoła Technik Komputerowych; pietia@pjwstk.edu.pl

Streszczenie

W rozdziale przedstawiono specyfikę projektowania aplikacji internetowych oraz dokonano przeglądu najpopularniejszych metodyk zorientowanych na projektowanie aplikacji internetowych. Omówiono założenia metodyk OOHDM, WSDM oraz WebML i porównano ich podejścia do projektowania aplikacji internetowych. Autorzy przestawili również dalsze kierunki rozwoju metodyk zorientowanych na projektowanie tych systemów oraz zaproponowali pewne propozycje rozszerzenia metodyki WebML.

1. Wprowadzenie

Dynamiczny rozwój Internetu i technologii z nim związanych pociągnął za sobą również dynamiczny rozwój aplikacji dostępnych w sieci Internet udostępnianych za pomocą przeglądarki (aplikacje internetowe, e-aplikacje). Aplikacje tego typu były początkowo aplikacjami statycznymi udostępniającymi jedynie określone dane. Jednak bardzo szybko przekształciły się w duże, złożone i dynamiczne systemy udostępniające swoim użytkownikom dziesiątki a nieraz i setki funkcji. Ich rozwój często był rozwojem niekontrolowanym, dynamicznym wychodzącym naprzeciw zapotrzebowań rynku i użytkowników. Twórcy tego typu systemów szybko przekonali się, że dostępne metodyki (zarówno strukturalne jak i obiektowe) służące do projektowania aplikacji tradycyjnych nie spełniały ich oczekiwań i nie były w stanie sprostać specyfice e-aplikacji. W latach dziewięćdziesiątych zaczęły pojawiać się pierwsze próby wprowadzenia do metodyk projektowych odpowiednich prymitywów, notacji oraz abstraktów uwzględniających specyficzne aspekty projektowe (np. aspekty nawigacyjne).

Jedną z pierwszych takich metodyk była metodyka HDM (Hypermedia Design Method). Po niej pojawiły się następne takie jak OOHDM, WSDM, czy WebML scharakteryzowane w dalszej części pracy. Podejmują one próbę usystematyzowania podejścia do projektowania aplikacji internetowych abstrahując od szczegółów implementacyjnych. Większość z nich opiera się na notacji UML dodając odpowiednie dla niej elementy notacyjne i pojęciowe.

Bardzo dynamiczny rozwój aplikacji internetowych, potrzeb klientów i technologii powoduje, że metodyki te muszą podlegać ciągłej ewolucji i często nie nadążają za potrzebami projektantów pracujących w silnie zmieniającym się środowisku. W pracy przedstawiono krótką charakterystykę kierunków rozwoju metodyk projektowania aplikacji internetowych oraz przedstawiono propozycje autorów na ich rozszerzenie.

2. Specyfika projektowania aplikacji internetowych

Aplikacje internetowe znacząco różnią się od tradycyjnych aplikacji, zarówno z punktu widzenia przeznaczenia aplikacji jak również z punktu widzenia ich twórców oraz osób zajmujących się ich pielęgnacją. Różnice obejmują takie aspekty jak:

- Nawigację – aplikacje internetowe mają zazwyczaj znacznie bogatszy i bardziej elastyczny model nawigacji, co może zwiększa prawdopodobieństwo zagubienia się użytkownika w przypadku złego zaprojektowanie interakcji;
- Doświadczenie użytkowników - trudno jest robić jakiekolwiek założenia odnośnie umiejętności użytkowników aplikacji internetowych;
- Skalowalność - oszacowanie obciążenia systemów internetowych jest praktycznie niemożliwe, odnosząca sukces aplikacja internetowa może drastycznie zwiększyć liczbę użytkowników z niej korzystających w bardzo krótkim czasie;
- Zmienność – funkcjonalność realizowana przez aplikację internetowe jak również dane udostępniane przez te systemy podlegają zazwyczaj zdecydowanie częstszym zmianom niż ma to miejsce w przypadku tradycyjnych aplikacji;
- Interfejs użytkownika – stosowane obecnie technologie wytwarzanie aplikacji internetowych dają mniejsze możliwości budowania interfejs użytkownika, niż ma to miejsce w przypadku tradycyjnych aplikacji (choć różnica ta stopniowo zaciera się).

Aplikacje internetowe możemy podzielić na dwie główne kategorie:

- Aplikacje pełniące głównie funkcje informacyjne, udostępniające użytkownikom informacje na wybrany temat, np. portale i wortale internetowe, strony firmowe itp.;
- Aplikacje udostępniające użytkownikowi pewną funkcjonalność. Przykładami tej kategorii są sklepy internetowe, wyszukiwarki jak również intranetowe aplikacje stworzone w celu zaspokojenia konkretnych potrzeb firm.

Aplikacje z pierwszej grupy powinny udostępniać użytkownikom łatwą i intuicyjną nawigację, dzięki której użytkownik nie poczuje się zagubiony w natłoku informacji.

Aplikacje te powinny również umożliwiać dostosowanie zawartości prezentowanych informacji do potrzeb konkretnego użytkownika. Osoby projektujące tego typu aplikacje powinny mieć jasny obraz struktury i rozmieszczenia danych w takiej aplikacji, aby nie dopuścić do dezaktualizacji danych przez nią udostępnianych. Główny nacisk podczas projektowanie tego typu aplikacji powinien zostać położony na łatwość nawigacji i utrzymania aktualności udostępnianych danych.

Podstawowym zadaniem aplikacji z drugiej grupy jest udostępnianie użytkownikowi konkretnych usług. Zazwyczaj ta grupa aplikacji korzysta ze znacznie większych zbiorów danych niż aplikacje „informacyjne". Dane te są jednak zwykle bardziej uporządkowane. W przypadku tego rodzaju systemów techniki projektowania typowych aplikacji mogą być znacznie bardziej użyteczne niż dla aplikacji „informacyjnych". Główny nacisk podczas projektowania tego typu systemów powinien być położony na udostępnioną funkcjonalność oraz łatwy do niej dostęp.

3. Przegląd metodyk

Pierwsze aplikacje internetowe były budowane bez uwzględnienia ich specyfiki projektowania i przeznaczenia, często bez tworzenia dokumentacji projektowej umożliwiającej ich efektywne utrzymywanie i rozbudowę. Nierzadko duże systemy internetowe powstawały przez rozbudowę początkowo niewielkiej aplikacji w miarę pojawiania się nowych potrzeb. Rosnące zapotrzebowanie na efektywne tworzenie takich aplikacji wymusiło powstanie metodyk zorientowanych na ich tworzenie. Metodyki te zazwyczaj wprowadzają własną notację lepiej przystosowaną do specyfiki aplikacji internetowych niż tradycyjne metodyki.

Poniżej znajduje się skrótowy przegląd trzech reprezentatywnych metodyk z tej grupy. Wszystkie dzielą tworzenie aplikacji internetowych na pięć etapów: zbieranie wymagań, budowę modelu koncepcyjnego, budowę modelu nawigacji, projektowanie interfejsu użytkownika i implementację. Jednak żadna z wymienionych metodyk nie obejmuje swoim zakresem całego cyklu życiowego aplikacji koncentrując się na wybranych dwu lub trzech etapach ich powstawania.

3.1 Obiektowy model projektowania Hipermediów (Object-Oriented Hypermedia Design Model, OOHDM)

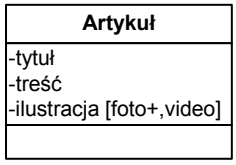
Metodyka OOHDM, opisana m.in. w [OOH2003], obejmuje cztery fazy tworzenia systemu internetowego. W niektórych z tych faz wykorzystuje znaną z UML notację – między innymi przypadki użycia i diagramy klas, dla pozostałych faz autorzy zaproponowali własną notację.

3.1.1. Zbieranie wymagań

Zebranie wymagań użytkownika jest pierwszym krokiem w tworzeniu systemu. Wymagania te powinny być przedstawione w postaci diagramów przypadków użycia oraz scenariuszy. Powinny być one zweryfikowane przez użytkownika. Podczas tego etapu wykorzystywana jest notacja zgodna z UML.

3.1.2. Budowa modelu koncepcyjnego

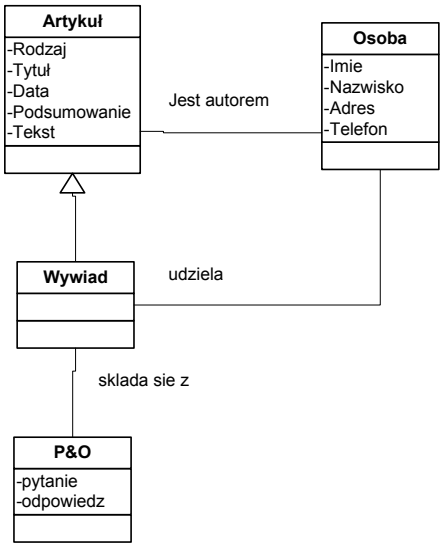
Model koncepcyjny przedstawia dane przechowywane w systemie. Wykorzystywane są w tym celu diagramy klas znane z UML, rozbudowane o pewne prymitywy – takie jak perspektywy atrybutów (używane w przypadku, gdy dany atrybut klasy może przyjąć wartości różnych typów).



Rysunek 1 Klasa z perspektywą atrybutu

Przykład takiego rozszerzenia przedstawiony został na rys. 1. Klasa Artykuł zawiera tu atrybut z perspektywą atrybutu – ilustracja może być albo fotografią (domyślnie, co oznaczone jest symbolem +), albo sekwencją video. W każdej instancji klasy zawierającej atrybut z domyślną perspektywą, atrybut musi mieć przypisaną wartość takiego typu, na jaki wskazuje domyślna perspektywa, a wartości innych typów są opcjonalne. To rozszerzenie notacji jest nadmiarowe, gdyż tą samą informację można łatwo przekazać bez wprowadzania dodatkowych prymitywów.

Przykładowy diagram klas wygląda następująco:

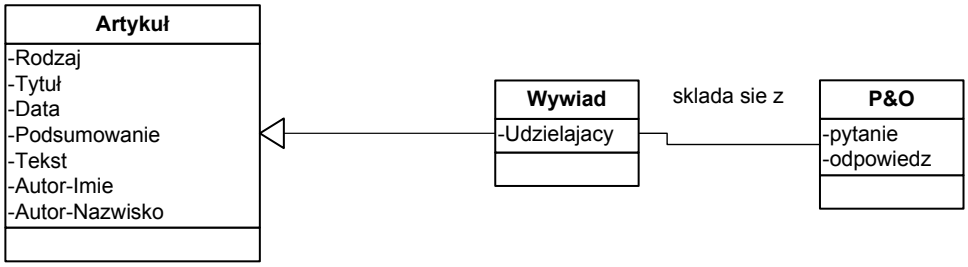


Rysunek 2 Przykładowy diagram klas należący do modelu koncepcyjnego

3.1.3. Model nawigacji

Model ten przedstawia strukturę nawigacji w aplikacji za pomocą tzw. *kontekstów nawigacyjnych*, zbudowanych w oparciu o *klasy nawigacyjne*.

Diagram klas nawigacyjnych to perspektywa zbudowana w oparciu o diagram klas z modelu koncepcyjnego, zapisana przy użyciu tej samej notacji. Przedstawia on jedno z możliwych spojrzeń na udostępniane przez system dane. Przykładowy diagram klas, będący perspektywą na diagram z Rys. 2 przedstawiony jest na Rys 3.:



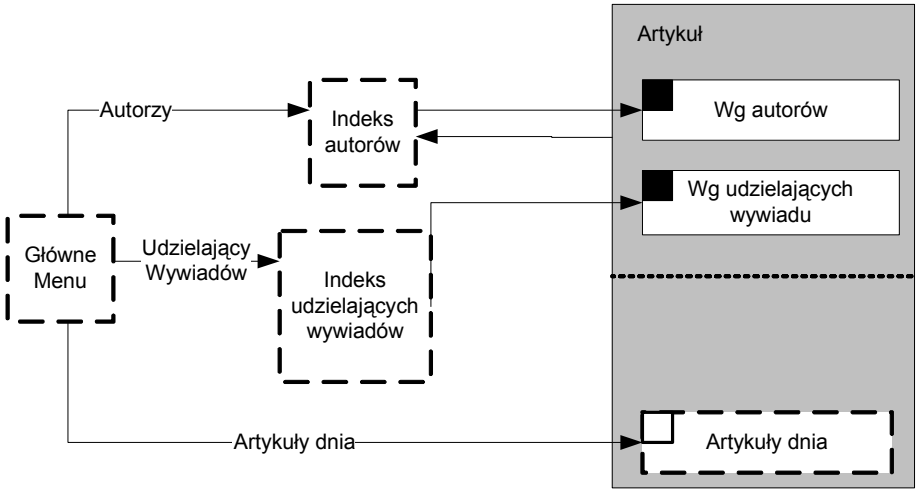
Rysunek 3 Diagram klas nawigacyjnych

W przypadku tej perspektywy, część danych z klasy Osoba zostało przeniesionych do klas Wywiad i Artykuł, zaś klasa Osoba nie występuje na diagramie.

Na podstawie diagramu klas nawigacyjnych budowany jest diagram kontekstów nawigacyjnych. W skład tego diagramu wchodzą następujące elementy:

- Węzły – reprezentują logiczne „okna” lub perspektywy na widok koncepcyjny;
- Odsyłacze – odzwierciedlają asocjacje z modelu koncepcyjnego, łączą obiekty innych klas nawigacyjnych;
- Indeksy – reprezentują indeksy elementów;
- Trasy (guided tours) – reprezentują sekwencje elementów;

Przykładowy diagram kontekstów nawigacyjnych wygląda następująco:



Rysunek 4 Przykładowy diagram kontekstów nawigacyjnych

W przypadku tego diagramu nawigację rozpoczynamy z Głównego Menu, będącego indeksem innych elementów. Możemy z niego przejść do jednego z dwóch indeksów – indeksu autorów lub indeksu udzielających wywiady, możemy też od razu przejść do artykułów dnia. Artykuły dnia są najprostszym rodzajem kontekstu – prostym zbiorem węzłów. Artykuły według autorów i według udzielających wywiadów są bardziej złożonymi kontekstami – stanowią grupy kontekstów (grupę kontekstów dla każdego autora i dla każdego udzielającego wywiadu). Czarne kwadraty w rogu każdego z tych kontekstów oznaczają, że istnieje dla nich indeks, jeśli kwadrat jest pusty, indeks taki nie istnieje. Jeśli między dwoma kontekstami dla danej klasy nawigacyjnej (oznaczonej szarym prostokątem) znajduje się przerywana linia, nie można bezpośrednio przejść z jednego kontekstu do drugiego. W przeciwnym wypadku jest to możliwe. Strzałki na diagramie oznaczają kierunek nawigacji pomiędzy elementami serwisu.

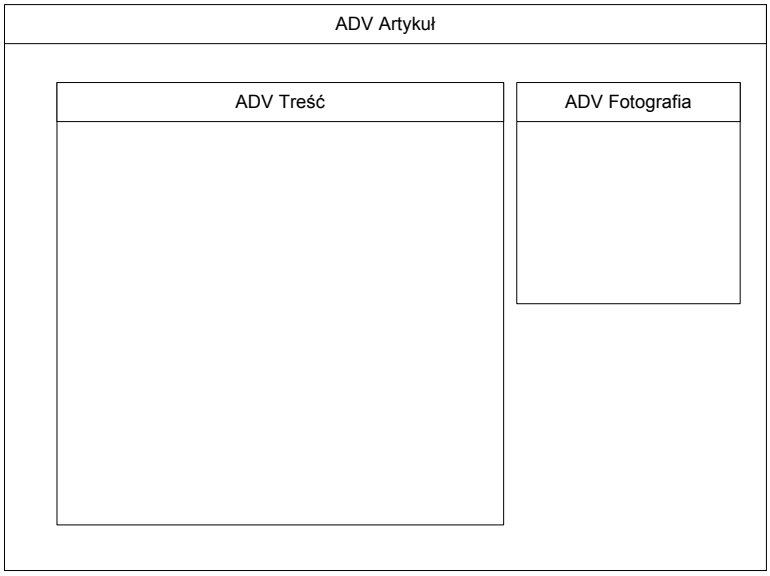
Model nawigacji może zawierać więcej niż jeden diagram klas nawigacyjnych i kontekstów nawigacyjnych – każdy z nich będzie reprezentować spojrzenie na system z punktu widzenia innego aktora– np. autora tekstów, czytelnika itp. Może też opisywać różne części systemu.

3.1.4. Model abstrakcyjnego interfejsu użytkownika

Interfejs użytkownika w OOHDM opisywany jest za pomocą Abstrakcyjnych Widoków Danych – ADV (Abstract Data Views). Reprezentują one zawartość interfejsu, nie odwołując się do konkretnej implementacji. Pozwalają one określić:

- Strukturę interfejsu użytkownika, przy użyciu agregacji i dziedziczenia jako mechanizmów abstrakcji;
- Układ elementów interfejsu;
- Powiązanie poszczególnych elementów interfejsu z obiektami nawigacyjnymi;
- Reakcje elementów interfejsu na zdarzenia zewnętrzne, w szczególności związek między zdarzeniem zewnętrznym a nawigacją – w tym celu używane są tzw. ADV-

Charts (będące wersją diagramów stanów), opisane w [CAR1994], oraz sieci Petriego;



Rysunek 5 Przykładowy prosty Abstract Data View

Rys. 5 przedstawia prosty interfejs służący do przeglądania artykułów prasowych w notacji używanej w OOHDM.

3.2 Metoda projektowania stron sieciowych (Web Site Design Metod, WSDM)

Metoda ta opisana w [DTL1998] obejmuje trzy pierwsze etapy wytwarzania systemów internetowych, nie zajmuje się projektem implementacyjnym, interfejsu użytkownika ani też samą implementacją. Jest to metoda zorientowana na użytkownika a nie na dane. Przeznaczona jest głównie do tworzenia aplikacji o charakterze informacyjnym. Metoda ta jest niezależna od użytej notacji służącej do modelowania przechowywanych danych.

3.2.1 Klasyfikacja użytkowników

Pierwszym etapem w projektowaniu systemów przy użyciu metodyki WSDM jest przeprowadzenie klasyfikacji użytkowników korzystających z przyszłego systemu. Klasyfikacja ta odbywa się w trzech krokach:

- Identyfikacja aktywności – aktywności są funkcjami udostępnianymi przez system;
- Identyfikacja klas użytkowników – dla każdej aktywności identyfikowane są grupy użytkowników, które będą z niej korzystać;
- Opis klas użytkowników – stworzenie tekstowego opisu każdej z klas użytkowników zawierającego informację na temat ich oczekiwań dotyczących

przekazywanych przez system informacji oraz charakterystyki ich umiejętności istotnych z punktu widzenia systemu.

W przypadku, gdy w opisie klasy użytkowników występują znaczne różnice w ich charakterystyce, klasa ta powinna zostać podzielona na *perspektywy*. Przykładowy opis klasy użytkowników znajduje się poniżej:

- **Czytelnik** – czytelnik poszukuje artykułów na interesujące go tematy. Poziom doświadczenia użytkowników w korzystaniu z Internetu może być bardzo zróżnicowany od „brak doświadczenia" do „zawodowy informatyk".
- **Autor artykułu** – użytkownik wprowadzający artykuły. Interesują go głównie informacje o własnych publikacjach. Posiada doświadczenie w korzystaniu z Internetu.

3.2.2 Model koncepcyjny

Model koncepcyjny w WSDM składa się z dwóch części: modelu danych oraz modelu nawigacyjnego. Model danych opisuje wymagania informacyjne różnych klasach użytkowników. Dla każdej klasy użytkowników tworzony jest oddzielny model danych obejmujący interesujący ich fragment danych przechowywanych w systemie. Metodyka ta nie określa, w jaki sposób dane te mają być opisywane, można zastosować zarówno notacje relacyjną jak i obiektową. W przypadku, gdy oczekiwania użytkowników różnią się w obrębie jeden klasy użytkowników możliwe jest użycie wariantów perspektyw w celu ich opisania. Ponadto tworzony jest (o ile nie istnieje) model biznesowy obiektów przedstawiający wszystkie dane dostępne w systemie.

Model nawigacyjny składa się z *tras nawigacyjnych*, po jednej dla każdej perspektywy użytkownika. Trasa nawigacyjna określa, w jaki sposób użytkownik porusza się po dostępnych mu danych. Trasa taka jest opisana za pomocą komponentów (dzielących się na informacyjne, nawigacyjne i zewnętrzne) oraz odsyłaczy.

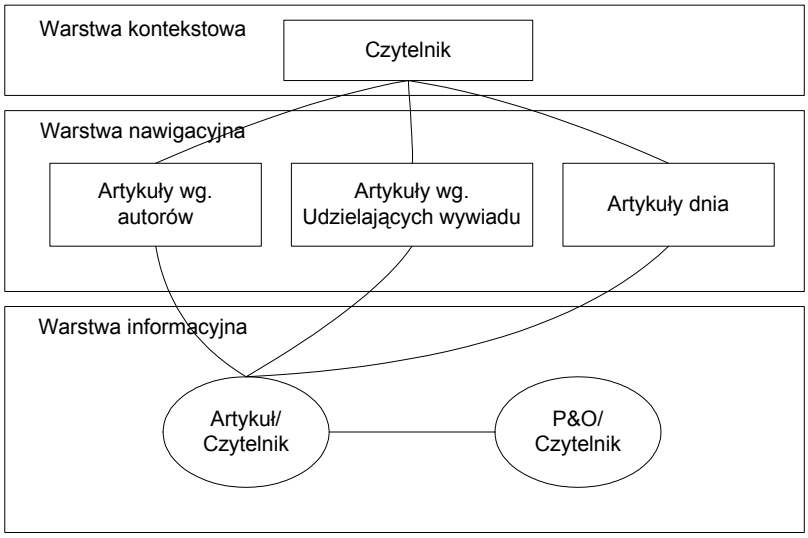
Komponenty informacyjne związane są z typami obiektów (które w modelu danych mogą być reprezentowane przez klasy lub encje). Jako że każdy typ obiektu może być związany z innymi (przez relacje/asocjacje – zależnie od modelu), komponenty informacyjne mogą być powiązane ze sobą za pomocą odsyłaczy. Komponent zewnętrzny to po prostu odnośnik do komponentu w innym systemie. Komponent nawigacyjny może być traktowany jako grupa odsyłaczy.

Metodyka WSDM opisuje algorytm budowy tras nawigacyjnych. Każda trasa nawigacyjna składa się z trzech warstw:

- Warstwy kontekstowej – składającej się z elementu nawigacyjnego o takiej samej nazwie, co perspektywa;
- Warstwy informacyjnej – w której każdy z typów obiektów wchodzących w skład perspektywy staje się obiektem informacyjnym lub zewnętrznym – zależnie od tego, czy dane są dostępne w systemie, czy też poza nim; Między obiektami warstwy informacyjnej tworzone są odsyłacze odpowiadające powiązaniom między typami obiektów;

- Warstwy nawigacyjnej – łączącej warstwy kontekstową i informacyjną, uwzględniając potrzeby użytkownika. Jeśli do tej samej informacji należy zapewnić dostęp na kilka różnych sposobów, tworzy się pośrednie odsyłacze i komponenty.

Przykładowy model nawigacyjny dla czytelnika serwisu prasowego wygląda następująco:



Rysunek 6 Model nawigacyjny w WSDM

Czytelnik na Rys. 6 ma możliwość przeglądania artykułów według autorów, osób udzielających wywiadu lub też obejrzenia artykułów dnia. Każda z tych możliwości prowadzi do oglądania perspektywy czytelnika na obiekt typu artykuł. Od artykułu można nawigować w kierunku pytań i odpowiedzi.

3.3 Web Modeling Language (WebML)

WebML [BCF2000] jest najbardziej rozbudowaną z opisywanych metodyk. WebML służy do wysoko poziomowego, niezależnego od platformy specyfikowania zorientowanych na dane aplikacji internetowych. Obejmuje on nie tylko budowę modelu danych i modelu nawigacyjnego, lecz również projektowanie interfejsu użytkownika oraz obejmuje proces personalizacji tworzonej aplikacji. Specyfikacja aplikacji internetowej w WebML składa się z czterech ortogonalnych modeli:

- Model strukturalny – opisujący dane przechowywane w systemie przy użyciu dowolnej notacji obiektowej lub relacyjnej;
- Model hipertekstu – opisujący jeden lub więcej hipertekstów mogących być umieszczonych na stronie. Każdy z hipertekstów opisuje tzw. widok, składający się z dwóch modeli:
  - modelu kompozycji – określającego, z czego zbudowany będzie hipertekst oraz jakie treści wchodzą w skład stron

    - model nawigacji – określającego sposób połączenia stron oraz treści w celu stworzenie hipertekstu.
- Model prezentacji – opisujący wygląd i układ stron w sposób niezależny od docelowego urządzenia i użytych środków implementacji.
- Model personalizacji – opisujący użytkowników i grupy użytkowników korzystających z systemu oraz dane i reguły specyficzne dla poszczególnych grup i użytkowników.

Proces projektowania systemów internetowych przy użyciu WebML składa się z opisanych poniżej kroków.

3.3.1 Zbieranie wymagań

WebML nie opisuje szczegółowo procesu zbierania wymagań. Informuje jedynie, że czynność taka powinna być wykonana. Określa też jakie informacje powinny być zebrane. Należą do nich:

- Główny cel strony;
- Docelowa grupa odbiorców;
- Przykłady zawartości;
- Wskazówki dotyczące stylu;
- Wymagania dotyczące personalizacji;
- Ograniczenia związane z systemami spadkowymi.

3.3.2 Projekt struktury danych

Podczas tego etapu projektowany jest model danych. Projekt ten może być wykonany przy użyciu dowolnej metodyki i notacji. Efektem tego etapu powinien być plik XML, zgodny ze specyfikacją zawartą w WebML, zawierający definicję danych i zależności między nimi.

3.3.3 Projekt hipertekstu

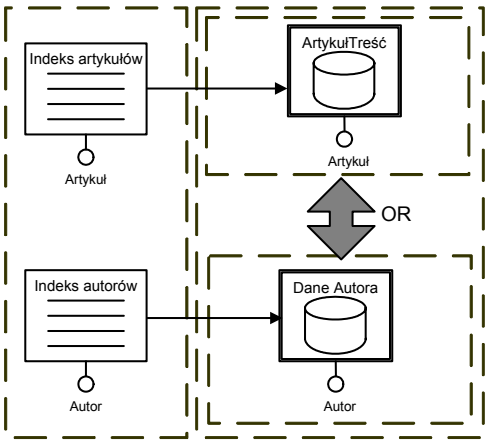
Tworzenie projektu hipertekstu składa się z dwóch etapów. Pierwszym etapem jest zaprojektowanie ogólnej struktury hipertekstu poprzez identyfikację wszystkich stron i jednostek treści, powiązań między nimi oraz mapowanie pomiędzy jednostkami treści a elementami zdefiniowanymi podczas projektowania modelu danych. W ten sposób tworzony jest szkielet systemu, który jest następnie iteracyjnie udoskonalany.

Drugim etapem jest projekt każdej ze stron i jednostek treści niezależnie od innych. W tym etapie możliwe jest stworzenie dodatkowych powiązań między stronami, konsolidacja atrybutów w obrębie jednostek treści jak również tworzenie nowych stron i jednostek treści określonych przez specjalne wymagania (np. filtry, alternatywne indeksy itp.).

WebML wyróżnia sześć rodzajów jednostek treści:

- Jednostki danych – zawierają podzbiór informacji z modelu danych. Dla danej encji lub klasy może być zdefiniowana więcej niż jedna jednostka danych;
- Jednostki złożone - zawierają wiele instancji określonego rodzaju;
- Jednostki indeksów – zawierają listę instancji danego kontenera (encji, relacji lub komponentu) jako listy. Każdy obiekt występujący w kontenerze jest jednym z elementów listy;
- Jednostki filtrowania – zawierają pola umożliwiające przeszukanie obiektów z danego kontenera (encji, relacji lub komponentu);
- Jednostki przeglądania – pozwalają na przeglądanie pojedynczej instancji klasy i sekwencyjnego przechodzenia między kolejnymi instancjami;
- Jednostki bezpośrednie – używane są jako specjalny rodzaj indeksów. Pozwalają na modelowanie relacji 1:1.

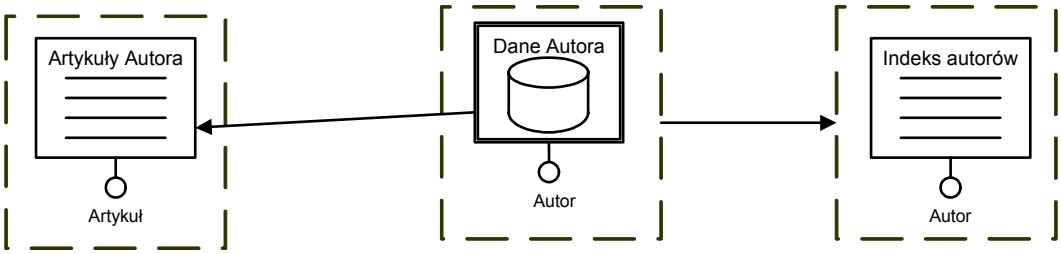
Jednostki treści są grupowane na stronach. Pojedyncza strona składa się z innych stron (podstron) lub jednostek treści.



Rysunek 7 Projekt strony w WebML

Na Rys. 7 znajduje się przykład strony zaprojektowanej za pomocą notacji graficznej WebML. Strona główna składa się z dwóch podstron zaznaczonych za pomocą prostokątów obramowanych przerywanymi liniami.. Lewa podstrona zawiera indeksy (listy) artykułów oraz autorów. Zależnie od dokonanego wyboru prawa podstrona zawierać będzie treść artykułu bądź też dane autora.

Model nawigacyjny określa połączenia pomiędzy stronami i jednostkami treści oraz sposób nawigacji pomiędzy nimi. WebML wyróżnia dwa rodzaje połączeń: *kontekstowe* i *bezkontekstowe*. Połączenia bezkontekstowe są niezależne od aktualnego stanu strony (np. powrót do strony głównej). Połączenia kontekstowe zależą od stanu np. pobranie artykułów danego autora. Połączenia bezkontekstowe w notacji graficznej WebML modelowane są jako powiązanie między stronami, połączenia kontekstowe jako powiązania między jednostkami treści.



Rysunek 8 Model nawigacyjny

Pokazany na Rys. 8 model nawigacyjny pokazuje zarówno połączenie kontekstowe (umożliwiające przejście od danych autora do listy jego artykułów), jak i połączenie bezkontekstowe (umożliwiające przejście od danych autora do indeksu autorów).

3.3.4 Projekt prezentacji

W WebML strony są generowane w oparciu o arkusze stylów określające wygląd strony oraz zawartość jednostek treści na stronie. WebML określa dwa rodzaje arkuszy stylów: *generyczne* (untyped) - arkusz stylów określający wygląd strony niezależnie od zawartości oraz *specjalizowane* (typed) używane do określenia wyglądu stron zawierających określone treści.

3.3.5 Projekt użytkowników i grup

WebML pozwala na modelowanie grup i użytkowników. Grupy określają zbiór użytkowników o podobnych charakterystykach. Użytkownik zawsze należy do co najmniej jednej grupy. Każdy użytkownik lub grupa jest modelowana za pomocą specjalnych encji zawierających specyficzne właściwości danej grupy lub użytkownika.

3.3.6 Projekt personalizacji

W WebML zdefiniowano dwa rodzaje personalizacji:

- Personalizację deklaratywną – projektant określa elementy, które mogą być personalizowane. W trakcie generowania strony system wstawia niezbędne informacje wynikające z preferencji użytkownika;
- Personalizację proceduralną – wykonywanie zdefiniowanych w XML reguł biznesowych w odpowiedzi na określone zdarzenia systemowe.

3.4 Porównanie prezentowanych metodyk

Zarówno OOHDM jak i WSDL koncentrują się podczas projektowania systemu internetowego na potrzebach użytkownika (widać to szczególnie w WSDL). WebML znacznie większy nacisk kładzie na obsługę danych zawartych w systemie oraz szczegółowy projekt interfejsu użytkownika. WebML stanowi zdecydowanie najbardziej

rozbudowaną metodykę projektowania aplikacji internetowych. Jedynie WebML zapewnia wsparcie ze strony istniejących narzędzi wspierających projektowanie.

Żadna z opisanych metodyk nie zajmuję się całością procesu budowy aplikacji internetowej, w związku z pominięciem fazy implementacji oraz związku poszczególnych elementów aplikacji z fragmentami kodu, co może utrudnić późniejsze utrzymanie i pielęgnację takiej aplikacji.

Wszystkie zaprezentowane metodyki stanowią znaczący krok w przód w stosunku do powszechnie wykorzystywanych rozwiązań (a raczej ich braku).

4. Perspektywy rozwoju

Zdaniem autorów kolejnym krokiem w rozwoju metodyk projektowania systemów internetowych powinno być rozwiniecie istniejących metodyk na całość procesu wytwórczego, w szczególności wyraźne powiązanie projektu z ostateczną postacią zaimplementowanego systemu tzn. wskazanie modułów realizujących poszczególne funkcjonalności, miejsca przechowywanie poszczególnych elementów danych itp. Razi też niezdecydowanie twórców metodyk, co do użytej notacji stosowanej przy tworzeniu modelu danych.

Wybranie modelu obiektowego pozwoliłoby na uściślenie zarówno stosowanej terminologii, która czasem może wydawać się niejasna, jak również skorzystać z pewnych właściwości modelu obiektowego nie występujących w modelu relacyjnym i niewykorzystywanym w tych metodykach. Decyzja taka pozwoliłaby też na bardziej precyzyjne określenie zależności pomiędzy poszczególnymi modelami.

Należy jeszcze dodać, iż praktyczny brak narzędzi wspierających te metodyki (z wyjątkiem WebML) zdecydowanie ogranicza ich zastosowanie w rzeczywistych projektach.

5. Propozycja rozszerzenia WebML

Wskazane w rozdziale 4 braki istniejących metodyk można zniwelować przy użyciu stosunkowo prostych środków. W związku z tym, że metodyką obejmującą najszerszy zakres projektowania aplikacji internetowych, a co za tym idzie wymagającą najmniej zmian, jest WebML, poniżej przedstawione są modyfikacje tej metodyki.

5.1. Wybór notacji dla fazy zbierania wymagań oraz projektu struktury danych.

Jako sposób zbierania wymagań proponuje się przypadki użycia, pozwalające na dokładne określenie wymagań klienta co do aplikacji internetowej. Zaletą przypadków użycia jest szerokie rozpowszechnienie tej metody oraz wsparcie ze strony wielu narzędzi.

Jako notację do projektu struktury danych proponuje się diagram klas w notacji UML, będący obecnie de facto standardem w zakresie modelowania danych.

5.2 Rozszerzenie metodyki o elementy związane z implementacją.

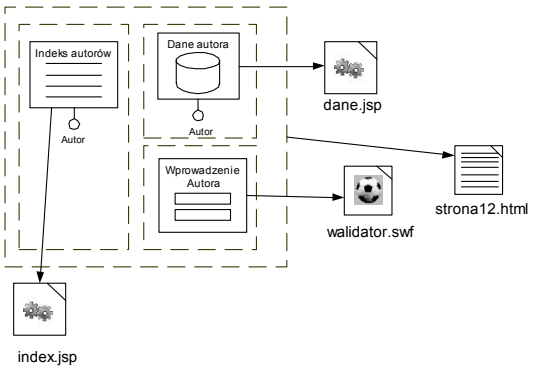
W związku z brakiem jakiegokolwiek wsparcie ze strony WebML dla przedstawienia zależności między projektem aplikacji a jej implementacją, co jest bardzo istotnym elementem z punktu widzenia utrzymania i pielęgnacji aplikacji, proponuje się rozszerzenie notacji graficznej WebML o nowy diagram pozwalający ukazać te zależności.

Diagram ten będzie przedstawiał miejsce implementacji poszczególnych elementów strony, jak również sposób w jaki będą zaimplementowane. Każdy element strony może być zaimplementowany na jeden z 3 sposobów:

- Statycznie – jako statyczna strona HTML, XML z XSLT;
- Dynamicznie po stronie serwera – za pomocą np. PHP, JSP, Servlet, ASP.NET itp.;
- Dynamicznie po stronie klienta – jako Applet, ActiveX, Macromedia Flash lub JavaScript itp.

Diagram ten powinien być budowany w następujący sposób: diagram przedstawiający projekt strony w WebML rozbudowujemy o dodatkowe elementy. Elementy te to ikony reprezentujące statyczne i dynamiczne komponenty systemu wraz z nazwami plików gdzie znajduje się ich implementacja oraz powiązania między elementami projektu strony (strona i jednostki treści) a komponentami systemu.

Poniższy diagram implementacyjny przedstawia fragment systemu umożliwiający przeglądanie indeksu autorów, wybranych z tego indeksu autorów oraz wprowadzanie nowych autorów. Zgodnie z informacjami przekazywanymi przez diagram, szkielet strony jest zrealizowany za pomocą statycznej strony www o nazwie strona12.html, indeks autorów jest przygotowywany przez dynamiczny kod wykonywany po stronie serwera zapisany w pliku o nazwie index.jsp, dane autora wyświetlane są przez dynamiczny kod wykonywany po stronie serwera zapisany w pliku o nazwie dane.jsp, zaś wprowadzanie danych autora realizowane jest przez dynamiczny komponent wykonywany po stronie klienta walidator.swf.



Rysunek 9 Przykładowy diagram implementacyjny

6. Podsumowanie

Jedną z charakterystycznych cech aplikacji internetowych jest to, że często dokonywane są zmiany w tych aplikacjach. Może to być zmiana wyglądu, przesunięcie elementów czy dodanie nowej funkcjonalności. Dzięki zastosowaniu metodyk oraz modeli w procesie projektowym, utrzymywanie tego typu aplikacji staje się łatwiejsze, szybsze i w efekcie tańsze. Dlatego też stosowanie przy projektowaniu aplikacji internetowych metodyk uwzględniających specyfikę tego typu projektu w szybkim czasie może przynieść ich twórcom oczywiste korzyści. Metodyk lub metod takich powstaje coraz więcej. Trudno jest jednak wskazać metodykę w tym zakresie dojrzałą i w pełni spełniającą oczekiwania projektantów. W prezentowanym rozdziale omówiono, z konieczności bardzo skrótowo, najbardziej znane i najistotniejsze metodyki ukierunkowane na projektowanie aplikacji internetowych. Wszystkie z nich są w ciągłej fazie rozwoju i wymagają dalszych prac i uzupełnień, choć najbardziej zaawansowaną wydaje się być metodyka WebML. Dzięki zastosowaniu wybranej metodyki i ewentualnemu dostosowaniu jej do własnych potrzeb i możliwości możliwe jest uzyskanie narzędzia przydatnego nie tylko w procesie projektowania systemu hipermedialnego, lecz również w jakże istotnym procesie pielęgnacji i utrzymania takiego systemu. Łatwa do znalezienia i zrozumienia informacja, mówiąca o miejscu implementacji poszczególnych elementów systemu może zdecydowanie zmniejszyć czasochłonność modyfikacji i utrzymania systemu. Wybór konkretnych notacji wykorzystywanych podczas projektowania systemu umożliwia stworzenie oprogramowania wspierającego taki proces od jego rozpoczęcia do zakończenia.

7. Bibliografia

[OOH2003] http://www.telemidia.puc-rio.br/oohdm/oohdm.html

[CAR1994] Carneiro, L.M.F.; Coffin, M.H.; Coewan, D.D.; Lucena, C.J.P.L; “ADVCharts: a Visual Formalism for Highly Interactive Systems”, in M.D. Harrison, C. Johnson, eds, Software Engineering in Human-Computer Interaction, Cambridge University Press, 1994.

[DTL1998] http://www7.scu.edu.au/programme/fullpapers/1853/com1853.htm

[BCF2000] Web Modeling Language (WebML): a modeling language for designing Web sites , S. Ceri, P. Fraternali, and A. Bongio, Proc. of the 9th World Wide Web Conference (WWW9), Amsterdam, May 2000

[BCF2002] Designing Data-Intensive Web Applications, Stefano Ceri et al., Morgan Kaufmann 2002