Konferencja naukowa KNWS'04
„Informatyka - sztuka czy rzemiosło"
23-25 czerwca 2004, Czocha

KONCEPCJA WZORCÓW PROJEKTOWYCH W JĘZYKU JAVA

Michał Małecki

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

e-mail: m.malecki@iie.uz.zgora.pl

STRESZCZENIE

Na początku lat 90 w środowisku programistów zaczęto gorąco dyskutować na temat idei wzorców projektów. Pierwsze prace związane były z poszukiwaniem i opisywaniem wzorców bez odniesienia w konkretnym języku programowania. Dopiero praca The Design Patterns Smalltalk Companion opisywała wzorce, ale z punktu widzenia języka Smalltalk. Referat ma zaznajomić programistów w języku Java z wzorcami projektowymi. Zaprezentowane wzorce zostały podzielone na trzy kategorie: konstrukcyjne, strukturalne i czynnościowe. W drugiej części omówiono wzorce projektowe dla programistów Java 2 Enterprise Edition.

1. WPROWADZENIE

Każdy programista przed napisaniem fragmentu kodu tworzy go w swoim umyśle. Niestety nie zawsze wizja, która powstanie jest optymalnym rozwiązaniem problemu. Programista czerpie satysfakcję nie tylko z działającego programu, ale również z takiego kodu, który jest bardziej ogólnego zastosowania, łatwiejszy do ponownego użycia i wprowadzania zmian.

Jednym z głównych powodów, dla których informatyka szuka dobrych wzorców projektowych jest zapewnienie dobrych i prostych rozwiązań dla często występujących problemów. Termin wzorzec projektowy (ang. *design pattern*) określa wygodny sposób ponownego wykorzystania obiektowego kodu w innych projektach i przez innych programistów. Idea wzorców jest prosta: skatalogować te sposoby interakcji między obiektami, które będą najbardziej przydatne podczas pisania programów.

Wzorce projektowe określają sposoby komunikacji między obiektami, bez wdawania się w szczegóły implementacji innego obiektu. Zachowanie takiego podziału jest cechą dobrego programowania obiektowego.

W chwili obecnej nie ma jednej ustalonej definicji pojęcia wzorca projektowego. Poniżej przytaczamy kilka definicji wzorców projektowych pochodzących z literatury:

„Wzorce projektowe stanowią powtarzalne rozwiązanie zagadnień projektowych, z którymi się wciąż spotykamy." [1]

„Wzorce projektowe stanowią zbiór reguł określających jak osiągnąć pewne cele w dziedzinie programowania.” [2]

„Wzorce projektowe w największym stopniu dotyczą problematyki ponownego użycia powtarzających się motywów architektury programów, zaś szkielety aplikacji dotyczą szczegółów projektowych i implementacyjnych.” [3]

„Wzorzec adresowany jest do powtarzających się problemów, które pojawiają się w specyficznych momentach projektowania i stanowi dla nich rozwiązanie.” [4]

„Wzorzec identyfikuje i specyfikuje pewną abstrakcję, której poziom znajduje się powyżej poziomu abstrakcji pojedynczej klasy, instancji lub komponentu.” [5]

Jednak należy pamiętać, że wzorce projektowe nie dotyczą jedynie projektowania samych obiektów, lecz również interakcji między nimi. Można na nie patrzeć jak na wzorce w komunikowaniu się obiektów, ale nie jest to spojrzenie pełne, ponieważ niektóre wzorce, oprócz sposobów komunikacji, określają strategię dziedziczenia i kompozycji, czyli zawieranie się w sobie obiektów.

Wzorce nie są wymyślane lub opracowywane – są odkrywane. Proces poszukiwania wzorców projektowych jest nazywany eksploracją wzorców. Autorzy książki [5] skatalogowali 23 wzorce projektowe w trzech kategoriach:

wzorce konstrukcyjne – wykorzystuje się je do pozyskiwania obiektów zamiast bezpośredniego tworzenia instancji klasy,

wzorce strukturalne – pomagają łączyć obiekty w większe struktury, mają zastosowanie na przykład w implementacji złożonego interfejsu użytkownika,

wzorce czynnościowe – pomagają zdefiniować komunikację pomiędzy obiektami i kontrolować przepływ danych w złożonym programie.

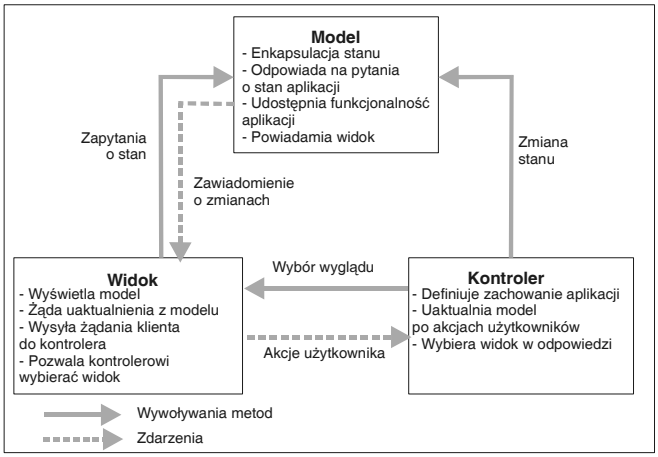
Proces zdobywania wiedzy na temat wzorców składa się z następujących faz: akceptacja, rozpoznanie i przyswojenie. Najpierw akceptujemy przesłankę o tym, że wzorzec jest dla nas ważny. Następnie rozpoznajemy, jakie ma dla nas znaczenie i gdzie możemy go wykorzystać. W końcu przyswajamy sobie wszystkie szczegóły dotyczące go, abyśmy potrafili za jego pomocą rozwiązać jakiś problem.

2. WZORCE PROJEKTOWE W JAVA 2 ENTERPRISE EDITION

Popularnym wzorcem, który bardzo wcześnie pojawił się w literaturze dotyczącej programowania jest model-widok-kontroler, (ang. *Model-View-Controler – MVC*):

- Data Model (model danych) – część programu odpowiedzialna za przechowywanie i przetwarzanie danych,
- View (widok) – prezentuje interfejs użytkownika,

- Controller (kontroler) – pośredniczy między użytkownikiem a widokiem.



Rys. 1. Architektura wzorca MVC

Katalog wzorców projektowych Java 2 Enterprise Edition

Wszystkie prezentowane w tym punkcie wzorce odnoszą się do prezentowanej wcześniej ogólnej koncepcji wzorca MVC. Istnieje również możliwość korzystania z tych wzorców poza rozwiązaniami bazującymi na wzorcu MVC.

Ze względu na stałą ewolucję prezentowanych wzorców, w nawiasach kwadratowych podano nazwy, pod którymi dany wzorzec jest również znany w innych publikacjach.

Data Access Object (DAO)
[Data Access Component]

Wzorzec DAO jest stosowany do rozdzielenia logiki biznesowej od logiki dostępu do danych. Umożliwia łatwa wymianę zasobów takich jak bazy danych bez konieczności modyfikacji logiki biznesowej. Wzorzec warto stosować w przypadku kiedy może dochodzić do częstych zmian typów zasobów, a zasoby te mają specyficzne funkcję dostępu. Problem ten rozwiązywany jest w J2EE poprzez usunięcie kodu programu zapewniającego dostęp do zasobów z Enterprise Java Beans (zwane ziarnami) i utworzenie abstrakcyjnego interfejsu dostępu do zasobów.

Fast-Lane Reader
[Bimodal Data Access]

Efektem zastosowania wzorca jest przyspieszanie dostępu do danych w trybie do odczytu poprzez nie używanie Enterprise Java Beans. Dzięki użyciu tego wzorca można osiągnąć wyraźny wzrost wydajności przy dostępie do dużych struktur danych. Można go stosować

tylko do danych, które nie będą modyfikowane i równocześnie dostęp do najświeższych danych nie jest krytyczną cechą systemu. Mechanizm nie zastępuje istniejącego sposobu dostępu do danych poprzez komponenty EJB – jedynie go uzupełnia.

Front Controller
[Front Component]

Wzorzec służy do zcentralizowania zarządzania widokiem – nawigacją, szablonami, bezpieczeństwem itp. – w aplikacjach Web-owych. Wprowadza on jeden obiekt obsługujący wszystkie żądania przychodzące od klienta. W praktyce wiele aplikacji Web-owych składa się z dużej ilości powiązanych ze sobą stron html. Obsługa takich serwisów jest utrudniona między innymi ze względu na konieczność dokonywania zmian w wielu plikach po przeniesieniu serwisu w inne miejsce. Dodatkowo zmiana wyglądu stron wymaga ich ponownej aranżacji. Problemy te można zmniejszyć korzystając z jednego, wspólnego dla wszystkich wywołań klientów kontrolera.

Page-by-Page Iterator
[Paged List, Value List Handler]

Usprawnienie dostępu do dużych list danych poprzez pobranie za jednym razem całej listy. W systemach gdzie następuje konieczność pobrania dużej listy informacji w celu ich zaprezentowania pomocne jest korzystanie z wzorca Page-byPage Iterator, który umożliwia pobranie całej zawartości listy za jednym razem. W wzorcu udostępnione są metody umożliwiające nawigację po pobranej liście danych.

Session Facade
[Session Entity Facade, Distributed Facade]

Wzorzec Session Facade dostarcza interfejs przypominający mechanizm *workflow* do ustawiania stanu ziaren *enterprise*. Jeden obiekt sesyjny pośredniczy w komunikacji z wieloma bean-ami sesyjnymi i encjowymi. Zastosowanie wzorca Session Facade skutecznie obniża ruch w sieci oraz uelastycznia system w przypadku zmian w docelowo wywoływanych bean-ach – ich zmiana nie powoduje zmian w obiektach klienta.

Value Object
[Data Transfer Object, Replicate Object]

Wzorzec ten stosowany jest w celu zmniejszenia ruchu w sieci, kiedy w trakcie działania aplikacji następuje równoczesny odczyt wielu atrybutów bean-a. Informacja zawarta w atrybutach jest przenoszona na stronę klienta, co wymiernie podnosi wydajność aplikacji.

3. PODSUMOWANIE

Korzyści wynikające z wykorzystywania wzorców projektowych są niepodważalne. Niestety nie wszyscy programiści korzystają z nich w trakcie projektowania i implementacji swoich rozwiązań. Podstawowymi przyczynami takiego stanu wydają się być brak wiedzy na temat wzorców, długotrwały okres pełnego przyswojenia, tak aby móc za jego pomocą rozwiązać jakiś problem oraz niekiedy niechęć do „odgórnego narzucenia" sposobu rozwiązania problemu.

W Javie wzorce projektowe można spotkać na każdym kroku. Dla przykładu, biblioteka Java Fundation Classes (JFC), która od pewnego czasu jest zawarta w podstawowym API, wykorzystuje kilka popularnych wzorców projektowych. Realizacja graficznego interfejsu użytkownika z wykorzystaniem biblioteki JFC w pewien sposób pomaga przyswoić wzorce projektowe.

W rozwiązaniach J2EE używanie wzorców projektowych jest wręcz koniecznością ze względu na ich złożoność i liczbę osób uczestniczących w projektach. Z pomocą przychodzi nam w tym przypadku firma Sun, która stara się opracować i upowszechnić jak największą ilość wzorców projektowych J2EE.

LITERATURA

[1] Alpert, Sherman, K. Brown, B. Woolf, *The Design Patterns Smalltalk Companion,* Addison-Wesley Pub Co, 1998

[2] Pree, Wolfgang, *Design Patterns for Object-Oriented Software Development,* Addison-Wesley Pub Co, 1994

[3] Coplien, Schmidt, *Patterns Languages of Program Design,* Addison-Wesley Pub Co, 1995

[4] Buschman, Meunier, *Pattern Oriented Software Architecture,* New York, John Wiley and Sons, 1996

[4] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison-Wesley Pub Co, 1995