Prototype Display System ver. 2 (PDS-2)

Erwin Dzienis, Witold Karpeta *
Instytut Informatyki, Uniwersytet Warszawski

Streszczenie

Niniejszy dokument jest artykułem prezentującym prace prowadzone na *Wydziale Informatyki Uniwersytetu Warszawskiego* w zakresie narzędzi wspierających metodę kart. Szczegołowy opis metody kart będącej propozycją zorganizowania prac nad systematycznym dowodzeniem poprawności projektów informatycznych nie będzie tematem tego artykułu.

1. Wstęp

System PDS-2 powstaje w specyficznych warunkach środowiska akademickiego. Zakres planowanych prac sprawiał, że od początku stało się jasnym, że PDS-2 będzie musiał powstawać jako ciąg wielu samodzielnych prac magisterskich czy doktorskich. Dlatego wśród wymogów stawianych projektowi przyszłego systemu znalazło się dużo postulatów pozafunkcyjnych dotyczących autonomii modułów czy łatwości przyszłej rozbudowy.

Cele funkcyjne PDS-2 to:

- stworzenie jednego środowiska do pracy nad wieloma projektami;
- umożliwienie wielu użytkownikom jednoczesnej pracy zdalnej;
- zarządzanie treścią projektów czyli zbiorów kart i raportów o statusie kart i projektów;
- identyfikacja użytkowników i wykorzystanie tej informacji do wprowadzenia systemu uprawnień;

Wymogi pozafunkcyjne sprowadzają się do:

*Praca częściowo finansowana z grantu KBN nr 8 T11C 015 15

- łatwości korzystania z zewnętrznych narzędzi i dodatkowych modułów dla różnych jęz. programowania i specyfikacji.

- dużej autonomii modułów pozwalającej na prowadzenie nad nimi oddzielnych prac (różni studenci piszący swoje prace magisterskie);

- niezależności od platformy sprzętowej czy programowej (duża zmienność wyposażenia studenckiego laboratorium);

- korzystania z narzędzi i standardów rokujących nadzieje na obecność na rynku za kilka lat (kiedy do pracy nad PDS-2 włączą się nowi ludzie).

Potrzeba sprostania powyższym wymaganiom powoduje, że system PDS-2 jest sporym wyzwaniem, w którym należy skorzystać z wielu najnowocześniejszych koncepcji i technologii z dziedziny inżynierii oprogramowania. Rozwiązanie przez nas przyjęte przedstawiamy poniżej.

2. Ogólne spojrzenie na architekturę PDS-2.

Chcąc wywiązać się ze stawianych wobec PDS-2 postulatów silnej autonomii autorzy projektując system zainteresowali się rozwiązaniami wielowarstwowymi[1] wprowadzającymi podział zadań między warstwy o dobrze wyspecyfikowanych funkcjach. Na świecie obecnie najpopularniejszy stał się model trójwarstwowy rozróżniający:

- warstwę przechowywania danych (często realizowaną za pomocą gotowego rozwiązania RDBMS[2] firm trzecich);

- wartswę logiki aplikacyjnej (dbającą o zachowanie logicznej spójności przechowywanej informacji);

- warstwę prezentacji danych użytkownikowi (obecnie jest to najczęściej GUI[3]).

Ponieważ w PDS-2 chcieliśmy również zrealizować rozgłaszanie informacji o zmianach wartości przechowywanych obiektów, wyróżniliśmy jeszcze

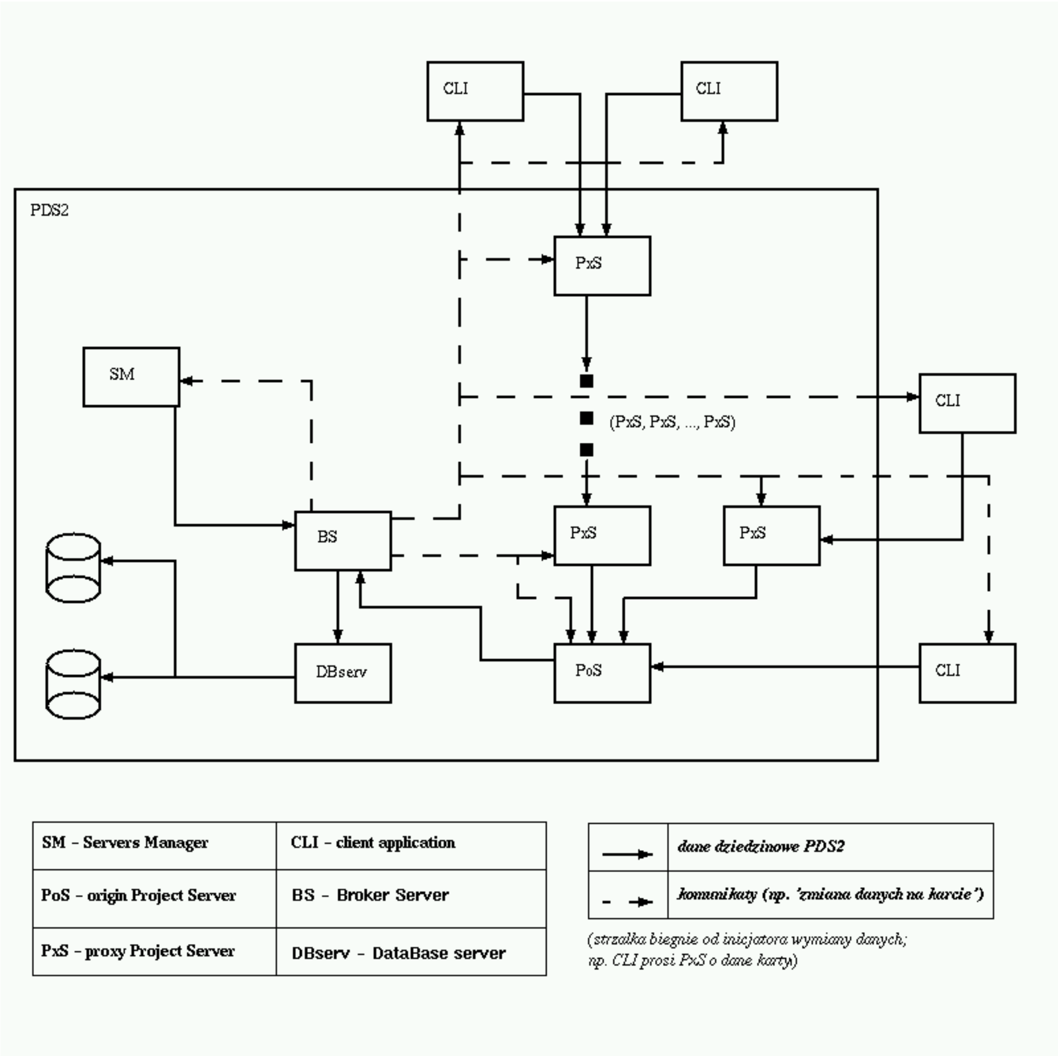
- warstwę rozgłaszania komunikatów (zrealizowaną za pomocą elastycznego i często obecnie stosowanego w systemach komercyjnych rozwiązania w postaci brokera komunikatów).

Zauważmy, że warstwa logiki aplikacji jest z reguły bardzo obszerna i w czasie używania systemu często istnieje potrzeba ciągłego powiększania jej o nowe funkcje. Stąd wynika wniosek, że realizacja wszystkich zadań tej warstwy w jednym programie o zamkniętej architekturze daleko wykracza poza zakres prac magisterskich. Z tego względu, a także dążąc do maksymalnie rozproszonego systemu spróbowaliśmy w naszej pracy wprowadzić jeszcze bardziej szczegółowy podział ról w ramach warstwy

[1] ang. multi tier
[2] ang. Relational Data Base Managment System
[3] ang. Graphical User Interface

logiki aplikacji wykorzystując rozpowszechnione przy obsłudze WWW pojęcie proxy serwera. Dlatego wyodrębniliśmy pewne minimum funkcjonalności tej warstwy i zawarliśmy je w specyfikacji podstawowego serwera (oznaczanego PoS[4] pozwalając, by wszystkie dodatkowe usługi były realizowane przez dodatkowe serwery połączone w strumień w sposób przypominający pracę programów w środowiskach typu Unix.

Powyższe idee znalazły swoje odzwierciedlenie w planowanej architekturze systemu PDS-2 widocznej na diagramie 1. Ponieważ podstawowym elementem PDS-2 jest projekt informatyczny realizowany przy użyciu metody kart, na rysunku przedstawiona jest przykładowa sytuacja pracy nad jednym projektem. Rysunek dla wielu projektów nie zawierałby żadnych nowych elementów i kopiowałby jedynie symbole przedstawione w przykładzie.



Rysunek 1: Architektura PDS-2

[4]z ang. origin Project Server

W kolejnych rozdziałach omówimy pokrótce warstwy PDS-2. W tym miejscu warto zwrócić uwagę czytelnika na rolę centralnego zarządcy serwerów (SM).
W systemie tak mocno rozproszonym jak PDS-2 pojawia się problem wyszukiwania informacji o tym, gdzie który serwer projektu można znaleźć. Temu zapotrzebowaniu na przeciw wychodzi idea stworzenia jednego miejsca, w którym każdy serwer oferujący swe usługi mogłby rejestrować się podając informację niezbędną do odnalezienia go przez klienta. Z drugiej strony klient musi znać sposób znalezienia serwera o interesującej go funkcjonalności.
W PDS-2 postanowiliśmy rozszerzyć znaczenie SM i oprócz funkcji informacyjnych obarczyć go jeszcze zadaniami administrowania serwerami. Projekt przewiduje, że będzie to miejsce, gdzie będzie można definiować projekty; określać prawa użytkowników w tychże; uruchamiać i zatrzymywać serwery skonfigurowane w sposób to umożliwiający (czyli albo będące na tej samej maszynie, albo osiągalne za pomocą komend zdalnych).

Wszystko to jednak wymaga czasu i na dziś w PDS-2 zrezygnowaliśmy z rozbudowanego systemu zarządzania użytkownikami i uprawnieniami. Zamiast tego przyjęliśmy rozproszony model w myśl, którego każdy serwer (PoS czy PxS) może mieć własne metody uwierzytelniania i prowadzić własną politykę bezpieczeństwa. Rozwiązanie takie jest o tyle uzasadnione, że w modelu scentralizowanym SM musiałby posiadać pełnię wiedzy o możliwościach zarejestrowanych serwerów i ich wymaganiach co do ogrniczania dostępu użytkowników do funkcji przez nie oferowanych. W wybranym rozwiązaniu każdy serwer ma prawo poprosić użytkownika o dodatkowe informacje uwierzytelniające (dla wygody człowieka część z tych informacji może być automatycznie wysyłana przez odpowiednio skonfigurowane oprogramowanie klienckie).

3. Warstwa prezentacji wyników.

Idea skorzystania w obsłudze PDS-2 z całej „fermy serwerów" (poprzez wprowadzenie zasady „robię, co umiem, a resztę życzeń użytkownika przekazuję kolegom przede mną") zaciera różnice między serwerem a klientem, dlatego w następnym rozdziale poświęconym warstwie logiki aplikacji znajdują się informacje istotne także dla programów współpracujących z użytkownikiem.

W tym miejscu warto wspomnieć o wymaganiach dotyczących tylko i wyłącznie prezentacji danych. W skrócie można ująć je w ten sposób:

- klient ma zapewnić okienkowe środowisko pracy nad wieloma dokumentami jednocześnie;
- program klienta także podlega nakazowi łatwej przenośności między systemami;

Z powyższych powodów prace nad dedykowanym klientem PDS-2 skupiają się wokół napisania apletu korzystającego z Javy jako języka (środowiska) oferującego przenośność i popularność porównywalną z tą osiągniętą przez przeglądarki WWW.

Na dalszych miejscach listy znajdują się takie postulaty:

- klient powinien nasłuchiwać komunikatów od BS i informować użytkownika o wszystkich zmianach danych obiektów PDS-2 otwartych w programie klienckim;
- klient musi ułatwiać kontakt z SM i wybranie serwera odpowiedniego do potrzeb klienta;
- klient powinien umożliwiać edycję znaczników PDS-2 wykraczających poza HTML[5];
- w wypadku powstania serwerów wymagających dodatkowego bezpieczeństwa klient mógłby zostać wzbogacony o odpowiednie funkcje pozwalające mu stać się generatorem haseł jednorazowych czy zapewnić komunikację za pomocą łącz szyfrowanych.

4. Obsługa żądań użytkownika - „ferma serwerów".

Kluczem do osiągnięcia w PDS-2 maksimum autonomii jest pojęcie „fermy serwerów", czyli idea łączenia wielu programów, z których każdy przetwarza jedynie część żądania użytkownika resztę przekazując innym. Ale po kolei.

Podstawową jednostką organizującą pracę w PDS-2 jest projekt. Projekt ma swoich uczestników pracujących nad nim jednocześnie. Projekt składa się z wielu kart zawierających kod źródłowy i teksty specyfikacji bądź dowodów. W ramach projektu mamy dobrze określony język programowania i specyfikacji i/lub formalizm przyjęty przy przeprowadzaniu dowodów.
Dlatego każdy projekt ma swój oddzielny PoS zapewniający minimum funkcjonalne jakim jest przechowywanie kart w strukturze wyznaczonej przez budowę kodu źródłowego. Każdy projekt może mieć tylko jeden PoS, który jest niezbędny do pracy nad projektem.
Wszystkie dodatkowe funkcje (np. te związane ze wsparciem dla języków specyfikacji kart) będą realizowane poprzez niezależne dodatkowe serwery PxS. Taki dodatkowy serwer będzie umiał z komunikatu zawierającego żądanie klienta wybrać tę część, która jest realizowana przez niego a resztę komunikatu przekazać kolejnym serwerom połączonym w łańcuch, na którego końcu znajdzie się PoS.

W środowisku, gdzie ma wspólnie pracować wiele różnych programów bardzo ważną kwestią staje się wybór protokołów komunikacyjnych między programami i formatu wymienianych danych. W momencie rozpoczęcia prac projektowych nad PDS-2 na świecie coraz większą popularność zdobywało WWW[6] i związane z nim standardy. Były to metody wymiany danych, które potwierdziły już swoją przydatność na milionach komputerów i przyzwyczaiły użytkowników do pewnych funkcjonalności, z których trudno byłoby już im zrezygnować.
Dlatego projektując PDS-2 postanowiliśmy zastosować w naszej pracy HTTP[7] i HTML jako podstawowego sposobu wymiany informacji.

[5] z ang. HyperText Markup Language
[6] z ang. World Wide Web
[7] z ang. HyperText Transfer Protocol

W trakcie prac doszliśmy do takich wniosków:

- para HTTP i HTML doskonale nadaje się do wymiany danych, ale do wymiany komunikatów powinniśmy skorzystać z protokołu umożliwiającego ustawiczne nasłuchiwanie informacji od brokera (BS);
- HTTP nie będzie wymagał żadnych zmian do pracy jako protokół wymiany danych dziedzinowych (informacji zawartych na kartach);
- organizacja obiektów PDS-2 jest skomplikowana i będziemy musieli rozszerzyć HTML o nowe znaczniki, by osiągnąć planowaną funkcjonalność.

Ostatni punkt wymaga w tym miejscu dłuższego komentarza. Podczas projektowania PDS-2 XML[8] nie był jeszcze rozwiązaniem popularnym i -na dobrą sprawętrwały dopiero nad nim prace w komitetach redakcyjnych, a potrzeby PDS-2 były skromniejsze i mogły zostać zaspokojone w prostszy sposób. Dlatego rozszerzyliśmy znaczniki HTMLa o informacje potrzebne nam do przekazywania informacji o obiektach PDS-2 w sposób zgodny z duchem powstającego właśnie standardu XML. W przyszłości pozwoli to stworzyć opis w XMLu wszystkich obiektów PDS-2 i używać tego opisu we wszystkich warstwach systemu (zaczynając od serwerów przechowujących dane w plikach, a kończąc na klientach wspomagających prace użytkownika). Poza tym popularne przeglądarki WWW mają tę przyjemną cechę, że ignorują znaczniki, które wykraczają poza HTML. Ta własność pozwalała nam żywić nadzieję, że do oglądania dokumentów przygotowanych w PDS-2 wystarczy ogromnie popularne narzędzie, jakim stały się te programy na przestrzeni ostatnich lat. W ten sposób osiągnęliśmy postulowaną możliwość korzystania z istniejących narzędzi do pracy z PDS-2. Wymaga to, co prawda od serwerów projektu (PoS i PxS) przygotowywania wersji z natywnymi znacznikami PDS-2 zdublowanymi znacznikami HTMLa (np. w ten sposób można uzyskać powiązania między użyciem zmiennej a jej deklaracją umieszczając w dokumencie obok znacznika PDS-2 jak najbardziej HTMLowy znacznik href), ale na pewno warto podjąć się tej dodatkowej pracy w zamian za możliwość korzystania z narzędzi ocenianych bardzo wysoko przez miliony użtkowników WWW na całym świecie.

5. Sposób rozgłaszania informacji o zmianach obiektów.

Istotnym elementem systemu PDS-2 jest rozgłaszanie informacji o zmianach zachodzących w poszczególnych obiektach systemu, takich jak np. karty. Ma to na celu informowanie klientów systemu o tym, że obiekt który jest lokalnie otwarty do czytania został przez kogoś innego zmieniony i że należy uaktualnić przechowywany obraz tego obiektu na podstawie informacji zawartych w repozytorium danych.

Architektura części rozgłaszającej informacje o zmianach obiektów opiera się na idei brokera wiadomości, będącego warstwą pośrednią pomiędzy częściami aplikacji, ukrywającą przed nimi wszystkie szczegóły dotyczące komunikacji, takie jak np. protokół sieciowy. Jednocześnie zapewnione jest bezpieczeństwo przesyłanych

[8] z ang. Extended Markup Language

wiadomości do odbiorcy poprzez kolejkowanie ich w wypadku czasowej przerwy w łączności, czy to spowodowanej chwilową niedostępnością aplikacji-adresata czy to brakiem dostępu do sieci.

Idea działania brokera jest następująca - każda wiadomość przesyłana przy użyciu brokera ma określony typ. Obiekt pragnący otrzymywać wiadomości danego typu musi się uprzednio w tym celu zarejestrować na serwerze brokera. Wiadomości wysyłane poprzez brokera docierają do wszystkich obiektów zarejestrowanych na otrzymywanie danego typu wiadomości. W ogólności można to porównać do metody działania list dyskusyjnych, gdzie każdy może wysłać wiadomość a otrzymają ją tylko osoby zapisane uprzednio do danej listy.

Dodatkowym elementem w mechanizmie brokera wiadomości jest zapewnianie, że każda wiadomość dotrze do wszystkich odbiorców. Opiera się ono na kolejkowaniu wiadomości, które nie mogły zostać przesłane dalej czy to z powodu niedostępności sieci czy to z powodu wyłączenia aplikacji, która miała odebrać wiadomość. Zgromadzone wiadomości są potem przesyłane do odbiorcy po ponownym nawiązaniu połączenia.

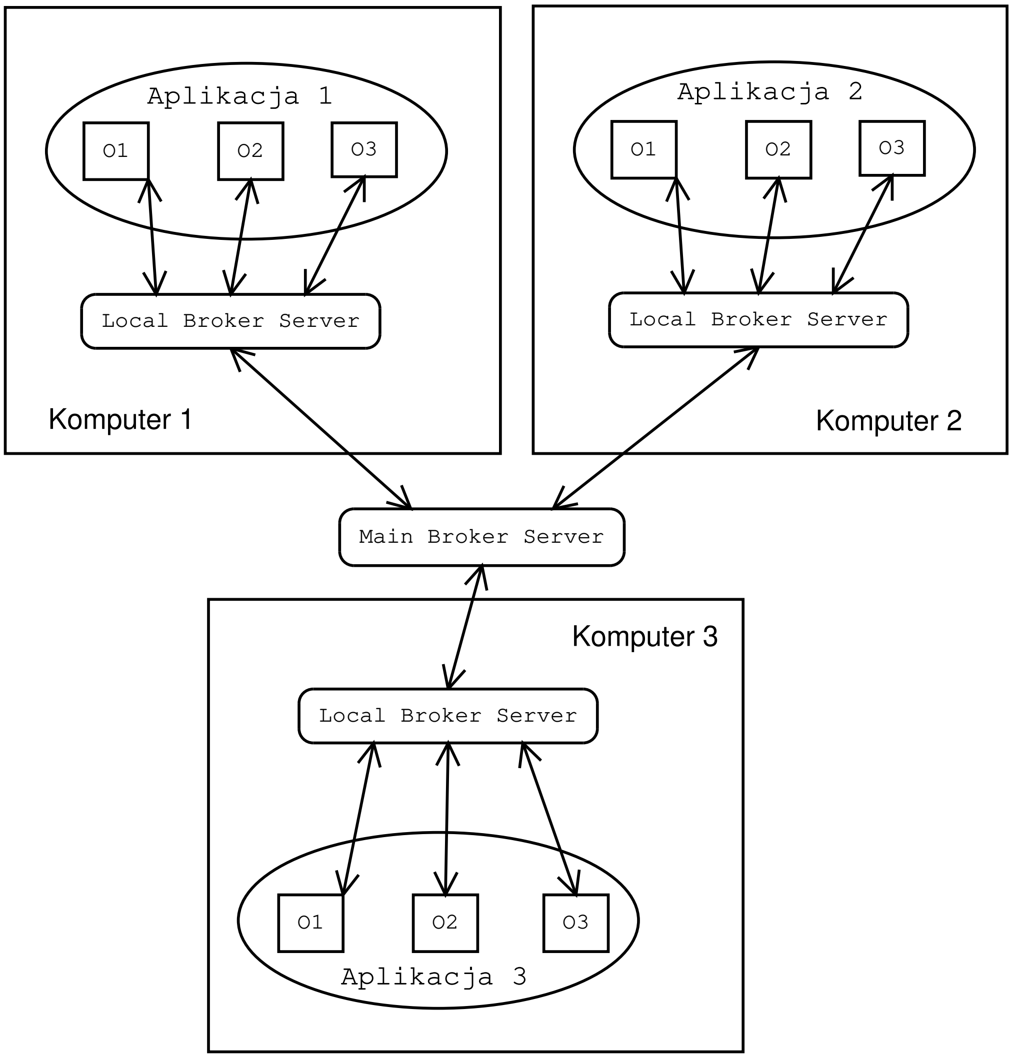
Od strony technicznej wygląda to tak, że na każdym komputerze gdzie pracują aplikacje komunikujące się za pomocą brokera wiadomości znajduje się lokalny serwer brokera (rys.2), który obsługuje rozsyłanie wiadomości do i od lokalnych obiektów i aplikacji korzystających z brokera. Trzyma on informacje o wszystkich typach wiadomości na które zarejestrowały się poszczególne obiekty oraz przechowuje kolejki wiadomości jeszcze nie dostarczonych do danego obiektu oraz kolejkę z wiadomościami jeszcze nie wysłanymi do głównego serwera.

Na jednym z komputerów umieszczony jest główny serwer brokera wiadomości odpowiedzialny za rozsyłanie wiadomości pomiędzy komputerami. Do niego trafiają wszystkie wiadomości przekazane serwerom lokalnym. Tutaj również trzymane są kolejki wiadomości nie wysłanych jeszcze do danego lokalnego serwera.

Rejestracja obiektu na otrzymywanie danego typu wiadomości przebiega następująco: najpierw obiekt wysyła zgłoszenie rejestracji do swojego lokalnego serwera. Ten odnotowuje to dodając ten typ do listy typów na zarejestrowanych dla tego obiektu. Następnie sprawdza czy jakiś inny obiekt już rejestrował się na otrzymywanie wiadomości tego typu i jeżeli nie było jeszcze takiego obiektu, to wysyła do głównego serwera zlecenie na przesyłanie mu wiadomości tego typu. W ten sposób jeśli żaden obiekt na danej maszynie nie zarejestruje się na jakikolwiek typ, żadne wiadomości nie zostaną tam przesłane. Pozwala to ograniczyć przesyłanie wiadomości poprzez wysyłanie ich tylko tam, gdzie ktoś na nie czeka.

Samo rozsyłanie wiadomości działa w podobny sposób — najpierw wiadomość jest przekazywana lokalnemu serwerowi, następnie ten przekazuje ją serwerowi głównemu oraz wszystkim lokalnym obiektom zarejestrowanym na typ tej wiadomości. Główny serwer następnie rozsyła tę wiadomość do wszystkich pozostałych lokalnych serwerów obsługujących jakiekolwiek obiekty zarejestrowane na typ tej wiadomości. Ostatnim etapem jest przekazanie wiadomości przez lokalne serwery do obsługiwanych przez nie obiektów.

W przypadku wystąpienia trudności przy przesyłaniu wiadomości do lub z głównego serwera, wiadomości te są wstawiane do kolejek (odpowiednio w serwerze lokal-



Rysunek 2: Architektura brokera wiadomości

nym lub głównym) i przesyłane dalej w momencie ponownego uzyskania połączenia. Wiadomości mające być przesłane z lokalnych serwerów do poszczególnych obiektów są zawsze wstawiane do kolejek i przesyłane dalej na wyraźne żądanie ze strony konkretnego obiektu.

Połączenia nawiązywane są zawsze od strony klienta tzn. połączenia do lokalnych serwerów są zapoczątkowane przez obiekty a połączenia do serwera głównego przez serwery lokalne. Połączenia są utrzymywane przez cały czas aż do momentu zniszczenia obiektu korzystającego z brokera. W momencie gdy połączenie zostanie zerwane obie strony rozpoznają ten fakt i zaczynają kolejkować wiadomości do przesłania drugiej stronie, ale to klient próbuje nawiązać połączenie ponownie.

Przy pomocy opisanego powyżej brokera wiadomości został w systemie PDS-2 stworzony mechanizm rozgłaszania informacji o zmianach obiektów.

Obiekty chcące wiedzieć o zmianach jakiegoś innego obiektu muszą ten fakt odnotować rejestrując się na typ wiadomości reprezentujący to zdarzenie. W momencie gdy jakiś obiekt PDS-2 (np. karta) zostanie zmieniony wysyłana jest wiadomość odpowiedniego typu. Dzięki brokerowi zostaje ona dostarczona do wszystkich zainteresowanych tym faktem obiektów na potencjalnie wielu fizycznych komputerach. W ten sposób rozsyłanie informacji o zmianach obiektów w bardzo naturalny sposób przekłada się na rozsyłanie wiadomości przy pomocy brokera.

6. Warstwa przechowywania danych.

Repozytorium systemu jest przechowywane w bazie danych. Ponieważ sposoby dostępu do środowisk baz danych istotnie się różnią powstało wymaganie stworzenia serwera obsługującego zlecenia do bazy danych, który udostępniałby na zewnątrz jednolity interfejs dostępu i ukrywał w sobie wszelkie elementy różniące poszczególne bazy danych. W szczególności różnice dotyczą takich elementów jak korzystanie z sekwencji, obsługa transakcji do bazy danych czy choćby dostępne typy danych. W chwili obecnej serwer może obsługiwać darmową bazę Postgres-SQL lub jedno z dwóch komercyjnych środowisk — DB2 albo Oracle.

Serwer zleceń do bazy danych stanowi element mający całkowicie oddzielić główną część systemu PDS-2 od konkretnego środowiska bazy danych użytego jako repozytorium danych. Ma on zwiększyć przenośność całego systemu poprzez wydzielenie do osobnego modułu wszystkich operacji wykonywanych bezpośrednio w bazie danych.

Zasada działania serwera zleceń jest następująca — zlecenie operacji na bazie danych w postaci polecenia SQL jest przysyłane za pomocą brokera wiadomości wraz z identyfikatorem aplikacji wydającej to zlecenie. Następnie to polecenie SQL jest przekształcane do postaci składniowej odpowiadającej wykorzystywanej bazie danych. Krok ten jest potrzebny z tego powodu, że o ile praktycznie wszystkie bazy danych akceptują pewien standard języka SQL, to jednak występują pomiędzy nimi drobne różnice np. w nazwach dostępnych typów danych. Zastosowanie tej transformacji rozwiązuje ten problem i pozwala na stosowanie w systemie PDS-2 jednej spójnej składni SQL, która w razie potrzeby jest dopiero w końcowym etapie dostosowywana do konkretnej bazy danych.

Po dokonaniu transformacji polecenie SQL jest przekazywane do obiektu zapewniającego bezpośredni kontakt z bazą danych. Implementacja klasy tego obiektu różni się oczywiście dla każdej z baz danych. Jest to jednocześnie jedyny element obok opisu reguł przekształcania poleceń SQL, który należy specjalnie przygotować dla każdej bazy danych. Nie jest to jednak zadanie trudne, gdyż klasa ta implementuje jedynie najbardziej podstawowe elementy kontaktu z bazą danych takie jak wykonanie polecenia SQL i odebranie wyników tego polecenia. Wynik polecenia jest następnie odsyłany do aplikacji, która wydała to polecenie.

Serwer został zaprojektowany w ten sposób aby zminimalizować pracę przy zmianie środowiska bazy danych. Biblioteka z realizacją bezpośredniego dostępu do bazy jest kompilowana jako dynamicznie dołączana, dzięki czemu mając przygotowane takie moduły dla różnych baz danych wystarczy jedynie wymieniać je bez konieczności rekompilacji jakiejkolwiek innej części systemu.

Całość komunikacji z i do serwera zleceń jest również zaimplementowana z wykorzystaniem brokera wiadomości. Zlecenia do bazy danych stanowią jeden typ wiadomości, których treścią oprócz samego polecenia SQL jest także identyfikator (przydzielany na użytek brokera wiadomości) obiektu wydającego zlecenie. Komunikat zwrotny z wynikiem polecenia SQL jest typu określonego przez ten identyfikator. Oczywiście na wiadomości ze zleceniami do bazy danych jest zarejestrowany jedynie serwer zleceń a na wiadomości dla konkretnego obiektu tylko ten obiekt.

Dzięki takiemu rozwiązaniu możliwe jest jednoczesne wykorzystanie kilku instancji baz danych np. jednej służącej w normalny sposób jako repozytorium i drugiej służącej jako kopia bezpieczeństwa. Pozwala również na umieszczenie repozytorium na innym komputerze niż reszta systemu.

7. Podsumowanie

Niniejszy artykuł zaprezentował prace, jakie na wydziale informatyki Uniwersytetu Warszawskiego prowadzone są nad systemem PDS-2, mającym być narzędziem wspomagającym korzystanie z metody kart.

Ze względu na szczególne warunki powstawania projektu (zbiór niezależnie tworzonych prac magisterskich) istotnym pozafunkcyjnym wymogiem postawionym przed autorami była jak największa niezależność poszczególnych modułów systemu oraz możliwość łatwego rozszerzenia o nowe elementy. Wymóg ten został spełniony dzięki koncepcji „fermy serwerów”.

Powstająca obecnie wersja systemu będzie zapewniała podstawową funkcjonalność dla pracy wielu użytkowników w środowisku rozproszonym. Planowane w przyszłości rozszerzenia będą pozwalały na wykorzystanie większej ilości języków programowania i specyfikacji, jak też umożliwią zastosowanie systemów automatycznego dowodzenia twierdzeń.