STUDIA INFORMATICA 2000
Volume 21 Number 1 (39)

Rafał KNOPA, Henryk KRAWCZYK
Politechnika Gdańska, Katedra Architektury Systemów Komputerowych

INTEGRACJA I KOORDYNACJA WSPÓŁPRACY KOMPONENTÓW APLIKACJI INTERNETOWYCH[1]

Streszczenie. Przedstawiono szeroko akceptowane modele integracji i współpracy komponentów aplikacji internetowych. Zaprezentowano koncepcję wykorzystania mobilnych agentów w zarządzaniu złożonych aplikacji. W celu ilustracji tej koncepcji przedyskutowano projekt aplikacji dotyczącej wspomagania diagnostyki medycznej.

INTEGRATION AND COORDINATION OF COMPONENT COOPERATIONS IN INTERNET – BASED APPLICATIONS

Summary. Widely accepted integration and co-operation models for components of Internet applications are presented. The idea of management of complex applications by mobile agents is given. The project of a medical diagnostic recommender system which illustrates this idea is discussed.

1. Wstęp

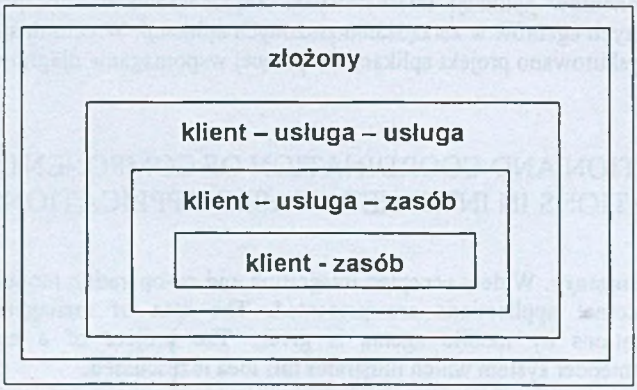
Ostatnie lata dynamicznego rozwoju Internetu wykształciły pojęcie aplikacji internetowej. W sposób najbardziej ogólny można ją określić jako zbiór komponentów realizujących określone zadanie, gdzie komponenty mogą być ulokowane na różnych serwerach, zaś komunikacja między nimi odbywa się poprzez Internet. Patrząc na aplikację internetową pod kątem jej architektury, wyróżnić możemy pewne elementy istotne dla sposobu konstrukcji i działania aplikacji. Są to następujące klasy komponentów:

- Zasoby – reprezentują konkretne dane lub wynik działania uruchomionych procesów. Udostępniane są za pomocą serwerów zasobów. Zasób jest to element bierny.

[1] Opracowano w ramach grantu KBN 8T11C 001 17

- Usługi – elementy wykonujące określone zadania. Usługi mogą korzystać z zasobów w celu realizacji zadań. Inne komponenty mogą wywołać usługę poprzez kontakt z serwerem usług. Usługę można określić jako element czynny.
- Interfejsy komunikacyjne – protokoły internetowe i interfejsy, za pomocą których zachodzi interakcja pomiędzy różnymi komponentami lub komponentami i ich użytkownikami. Użycie konkretnego interfejsu zależy od kontekstu i wymagań komponentów, które interfejs ma połączyć.
- Klient – element reprezentujący zewnętrznego użytkownika wraz z jego interfejsem korzystania z aplikacji.

Wykorzystanie sieci Internet głównie jako sieci usługowej powoduje, że większość połączeń w niej funkcjonujących opartych jest na architekturze klient-serwer. Architektura ta w większości zastosowań przenosi ciężar obowiązków związanych z zarządzaniem na serwer.



Rys. 1. Poziomy integracji aplikacji internetowych
Fig. 1. Internet-based applications levels of integration

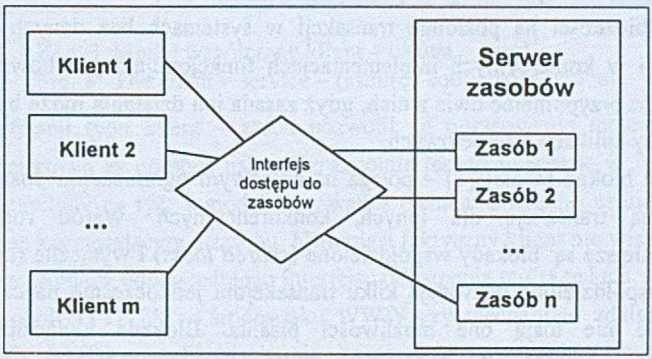
Pełny opis aplikacji wymaga jednak przedstawienia, w jaki sposób współpracują ze sobą wyżej wymienione klasy komponentów. Jest to też istotne przy konstrukcji złożonych rozwiązań – na przykład systemu handlu elektronicznego, czy systemów przepływu dokumentów. Jednak architektura większości aplikacji w najbardziej ogólnym pojęciu sprowadza się do wykorzystania wyżej wymienionych elementów w różnych konfiguracjach. Badając najczęściej używane konfiguracje, określić można pewną ich hierarchię, przedstawioną na rys. 1, którą nazywamy tutaj poziomami integracji. Nazwa ta wywodzi się stąd, że element na wyższym poziomie składa się ze zintegrowanych elementów poziomów niższych w nim zawartych. W skład poziomu najwyższego (złożonego lub hybrydowego) wchodzą dowolne kombinacje konfiguracji z poziomów niższych.

W pracy przedstawiono sposób funkcjonowania aplikacji dla wszystkich wymienionych poziomów integracji. Szczególną uwagę zwrócono na różne aspekty zarządzania wynikające ze specyfiki aplikacji internetowej. Zasygnalizowano istotną rolę agentów, oraz sformułowano sposób integracji i koordynacji pracy przedstawionych komponentów opracowywanego w ramach grantu KBN systemu ERS [9] (Endoscopy Recommender System).

2. Poziomy integracji

Zgodnie z rys. 1 należy przeanalizować cztery podstawowe poziomy integracji. Dla każdego z nich możliwych jest kilka konkretnych rozwiązań, przedstawionych poniżej:

- **Współpraca klient – zasób**



Rys. 2. Model współpracy klient – zasób
Fig. 2. The client – resource cooperation model

Struktura tego poziomu, przedstawiona na rys. 2, pozwala na realizację zdalnego korzystania z zasobu za pośrednictwem Internetu. Interfejs jest tutaj protokołem pozwalającym na dostęp typu zapis/odczyt do zasobu za pośrednictwem serwera. Typowe rozwiązania są następujące:

– **Klient – serwer plików:** jest to jedna z najczęściej stosowanych implementacji. Polega na udostępnieniu plików za pomocą standardowego protokołu internetowego. Protokołem tym najczęściej jest *File Transfer Protocol* (FTP) [10]. Synchronizacja dostępu do zasobów odbywa się tutaj w oparciu o wewnętrzne mechanizmy serwera. Cechą charakterystyczną dla protokołu jest to, że transmisja danych i wydawanie komend

inicjujących tę transmisję odbywa się na różnych poziomach, pomimo że sam protokół nie pozwala jednemu klientowi pracować wielowątkowo.

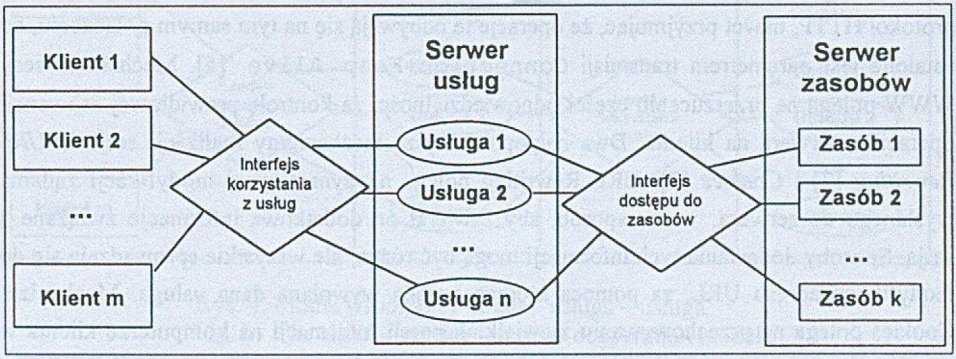
– **Klient – baza danych:** Kolejna bardzo często spotykana implementacja. Korzystanie z serwerów baz danych poprzez Internet wywodzi się z analogicznego zastosowania w sieciach LAN, inny jest tylko interfejs dostępu. Współczesne interfejsy dostępu do baz danych niekoniecznie są związane z jednym tylko typem baz, przeważnie umożliwiają korzystanie z różnych typów za pomocą tych samych mechanizmów. Sztandarowym przykładem jest tutaj interfejs *Open DataBase Connectivity* (ODBC) [8], który umożliwia korzystanie z większości liczących się systemów zarządzania bazami danych. Dużą rolę odgrywa tutaj stosowanie standardowego języka definicji danych i operacji na nich. Dla relacyjnych baz danych językiem takim jest SQL [6].

W zarządzaniu aplikacji korzystającej z bazy danych jednym z najważniejszych elementów jest takie wykonywanie operacji, aby nie została naruszona spójność danych. Ma to szczególnie istotne znaczenie w wywołaniach współbieżnych. We współczesnych bazach danych w zasadzie wszystkie operacje realizowane są w oparciu o transakcje. Problem kontroli współbieżności na poziomie transakcji w systemach baz danych jest znany od kilkunastu lat i w komercyjnych implementacjach funkcjonują wypróbowane metody jej realizacji. Warto przypomnieć dwie z nich, gdyż zasada ich działania może być przeniesiona na inne poziomy aplikacji internetowych:

- Mechanizm blokad (*locks*) [1] – polega na czasowym ograniczeniu dostępu do danych przez jedną transakcję dla innych, konkurencyjnych. Wśród rodzajów blokad najpopularniejsze są: blokady współdzielone (*shared locks*) i wyłączne (*exclusive locks*). Blokady współdzielone pozwalają kilku transakcjom jednocześnie na czytanie jednego zasobu, ale nie mają one możliwości pisania. Blokada wyłączna pozwala na wykonywanie operacji tylko jednej transakcji naraz, za to dozwolona jest realizacja odczytu i zapisu. Wśród algorytmów kontroli współbieżności opartych na blokadach najbardziej popularny jest algorytm dwufazowy, polegający na tym, że transakcja może zakładać blokady na kolejne zasoby do momentu zwolnienia pierwszej z nich – potem transakcja może już tylko zwalniać wcześniej założone blokady. W ten sposób możemy wyszczególnić dwie fazy – fazę zakładania blokad (*growing phase*) i ich zwalniania (*shrinking phase*).
- Mechanizm stempli (*timestamps*) [1] – polega na tym, że każdej transakcji w momencie jej rozpoczęcia przydzielany jest unikalny identyfikator, wynikający z czasu systemowego, a dla zasobów, na których działają transakcje, przydzielana jest wartość identyfikatora transakcji, która ostatnia dokonała operacji odczytu lub zapisu (dwie osobne wartości). Ten mechanizm rozpowszechniony jest szczególnie w rozproszonych bazach danych. Jego wykorzystanie opiera się głównie na tzw. algorytmie

optymistycznym, który pozwala transakcji wykonywać działania do momentu, gdy trzeba je zaakceptować – wtedy są one przez bazę akceptowane lub odrzucane, zgodnie z wynikiem porównań stempli transakcji związanych z zasobem.

- **Współpraca klient – usługa – zasób**



Rys. 3. Model współpracy klient – usługa – zasób
Fig. 3. The client – service – resource cooperation model

Rozwój aplikacji typu klient – zasób pozwolił na opracowanie metod korzystania z zasobów zewnętrznych za pomocą usług. Szczególnie jest to widoczne w implementacjach WWW. Przedstawiony na rys. 3 model wprowadza element pośredni, usługę, gdzie usługa staje się klientem korzystającym z zasobu. Natomiast faktyczny klient nie widzi zasobu, tylko funkcje, które za pomocą odpowiedniego interfejsu udostępnia mu ta usługa. Interfejsem tym może być protokół HTTP [5] w środowisku WWW czy mechanizm zdalnego wywołania procedur (RPC, RMI) w aplikacjach nie bazujących na standardowych protokołach internetowych.

Najpopularniejsza implementacja takiego rozwiązania polega na wykorzystaniu serwera WWW jako serwera usług, implementacji *Common Gateway Interface* (CGI) jako usług, protokołu HTTP + CGI jako interfejsu korzystania z usług oraz bazy danych jako zasobu. Problemy zarządzania zawęzić można tutaj do poziomu samych usług. Efektywność działania wymaga od serwera, by dla każdego wywołania usługi (o ile nie zostało to wyraźnie zabronione) tworzyć zawsze jej osobną instancję. Prawdopodobna jest taka sytuacja, że do synchronizacji operacji na zasobach zewnętrznych nie wystarczą mechanizmy serwera zasobów (stosowane w modelu klient – zasób). Ma to miejsce, zwłaszcza, gdy usługa korzysta z zasobów na różnych serwerach. Pojawia się problem, jak synchronizować dwie instancje, żeby wykonywane przez nie operacje nie kolidowały ze sobą? Rozwiązaniem może być implementacja samego serwera, który nadzoruje operacje na zasobach zewnętrznych bądź zapewnienie synchronizacji operacji przez zewnętrzny proces. Istnieją również rozwiązania

programistyczne dające łatwe możliwości tego typu synchronizacji. Przykładem mogą być serwlety, w których można wykorzystać mechanizm wbudowanego monitora. Rozwiązanie oparte na zewnętrznym procesie pozwala również kontrolować współbieżność dostępu do zasobu różnych usług.

Warto uwzględnić również tzw. mechanizmy sesji WWW. Logiczny ciąg operacji wykonywany w środowisku WWW trudno kontrolować jako całość w oparciu tylko o protokół HTTP, nawet przyjmując, że operacje te odbywają się na tym samym połączeniu, co ustalone jest parametrem transmisji `Connection=Keep Alive` [5]. Mechanizm sesji WWW polega na przerzuceniu części odpowiedzialności za kontrolę prawidłowej sekwencji operacji z serwera na klienta. Dwa najpopularniejsze mechanizmy realizacji sesji to *URL Rewriting* [7] i *Cookies* [7]. URL Rewriting polega na dynamicznej modyfikacji żądania wysłanego do serwera w ten sposób, aby zawierał on dodatkowe informacje związane z sesją. Sposoby dodawania tych informacji mogą być różne, ale wszystkie sprowadzają się do modyfikacji adresu URL, za pomocą którego została wywołana dana usługa. Mechanizm Cookies polega na przechowywaniu niewielkich porcji informacji na komputerze klienta w ten sposób, aby usługa mogła dynamicznie je odczytać. Operacje oparte na sesji WWW są powszechnie realizowane na przykład w sklepach internetowych, gdzie klient może najpierw pozbierać „do koszyka" towary poprzez wędrówkę po różnych stronach, a na końcu sfinalizować zakup. Informacje o towarach przechowywane są wtedy za pomocą mechanizmów sesji bez angażowania w to serwera. Wadą zastosowań mechanizmu sesji jest to, że tylko naprawdę niewielkie porcje danych mogą być przechowywane za jego pomocą.

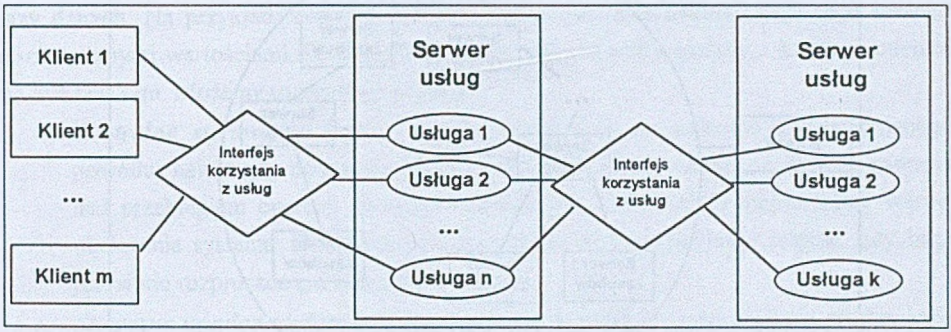
- **Współpraca klient – usługa – usługa**

Korzystanie z zasobów za pośrednictwem usług w pewnych zastosowaniach okazało się niewystarczające i stąd zrodziła się koncepcja korzystania z usług za pośrednictwem innych usług. Na rys. 4 przedstawiona jest struktura, w której klient wywołując usługę, uruchamia za jej pomocą proces innej usługi, niedostępnej dla niego bezpośrednio.

Przykładowa implementacja tego modelu polega na korzystaniu z zadań uruchamianych w sieci LAN czy na komputerach dużej mocy za pośrednictwem usług serwera WWW. Innym zastosowaniem, ostatnio bardzo popularnym, jest również wykorzystanie środowiska WWW do operacji związanych z zarządzaniem interfejsem dostępu do baz danych lub serwerem bazy danych, np. monitorowanie operacji czy ustawianie parametrów konfiguracyjnych.

Na szczególną uwagę zasługuje nowe podejście do komunikacji pomiędzy usługami, jakim jest system mobilnych agentów. Metodologia korzystania z systemu, opisana przez OMG w dokumentacji standardu [4], określa mobilnego agenta jako proces, który potrafi

przenosić się w sieci wraz z całym swoim kodem lub aktualnym zbiorem stanów pomiędzy systemami agentów (*Agent Systems*), gdzie jest uruchamiany. Wśród cech charakteryzujących system mobilnych agentów wyróżnić można:

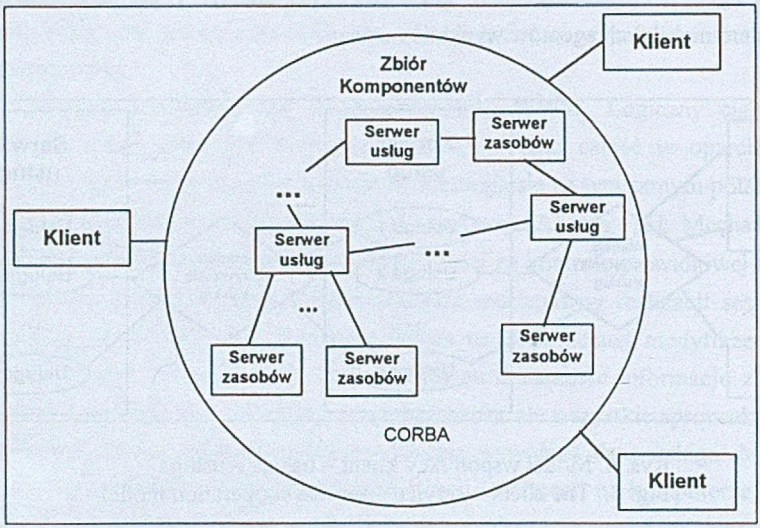


Rys. 4. Model współpracy klient – usługa – usługa
Fig. 4. The client – service – service cooperation model

- *Agent Authority* – identyfikator danego agenta, zawierający informację o jego właścicielu (osobie lub organizacji).
- *Agent System Type* – typ systemu agentów opisujący jego parametry pod kątem możliwości uruchomieniowych, czyli np. języka programowania, który akceptuje.
- *Place* – wyróżnione miejsce w systemie agentów zawierające środowisko wykonania dla danego agenta. Za pomocą takiego miejsca można kontrolować dostęp agenta do systemu agentów.
- *Region* – zbiór systemów agentów mających tego samego właściciela, ale niekoniecznie tego samego typu. Idea regionów pozwala na rozszerzenie skalowalności systemu, gdyż może on wtedy dystrybuować obciążenie pomiędzy systemy agentów tego samego typu.

System mobilnych agentów, w porównaniu do innych rodzajów usług, posiada wiele zalet. Przykładowo: aplikacja udostępnia bibliotekę funkcji umożliwiających wykonywanie zaawansowanych obliczeń. Klient potrzebuje wykorzystać sekwencję tych funkcji i po ich wykonaniu otrzymać określony wynik. W modelu statycznym usługa uruchomiona przez klienta musi każdą z tych funkcji wywoływać kolejno przez interfejs komunikacyjny i czekać na odpowiedź. Natomiast mobilny agent realizujący żądaną sekwencję jest w stanie przenieść się w całości do systemu agentów aplikacji, tam lokalnie wykonać wszystkie operacje i wrócić do klienta z wynikiem.

- **Współpraca złożona**



Rys. 5. Hybrydowy model współpracy komponentów
Fig. 5. Hybrid model component cooperation

Modele współpracy określone dla poprzednich poziomów pozwalają na tworzenie współpracy hybrydowej (złożonej) w sposób przedstawiony na rys. 5. Zakłada się tutaj, że usługa może korzystać z zasobu lub innej usługi, zaś zasób jest elementem biernym, gdyż on sam nie może już skorzystać z innych komponentów. Co więcej, w taką współpracę mogą być włączeni różni użytkownicy w dowolnych momentach (tzw. przetwarzanie złożone - ang. *collaborative computing*). W ten sposób można stworzyć taki wariant współpracy, w którym dostęp do rozproszonych zasobów kontrolowany jest przez system rozproszonych usług.

Mechanizmy zarządzania dla współpracy złożonej mają wiele wspólnych cech z rozproszoną bazą danych. Dla klienta baza widoczna jest jako całość, nie interesuje go, w którym miejscu znajdują się zasoby lub usługi. Chce on uzyskać do nich dostęp w możliwie najbardziej efektywny sposób. Z powodu wymagań na szybkość lub efektywność pracy stosuje się metody równoważenia obciążenia [11] (*load balancing*), jak mirroring lub lokalne odwzorowanie (*mapping*) komponentów. Mirroring polega na tym, że komponenty są w całości duplikowane w kilku miejscach i użytkownik sam wybiera sobie miejsce, które najbardziej odpowiada jego potrzebom. Lokalne odwzorowanie zakłada, że komponenty są równomiernie rozłożone w całym zbiorze i klient widzi je jako całość. O rzeczywistym komponencie decyduje system, wybierając spośród kilku możliwości w zależności od parametrów, takich jak: aktualne obciążenie, odległość od klienta. Jest to rozwiązanie o tyle lepsze od mirroringu, że uwzględnia aktualny stan wykonywania usług. Oczywiście przy

wyborze dokonywanym przez klienta może on podać wartości powyższych parametrów dla poszczególnych komponentów, ale to nie gwarantuje, że klient skorzysta z optymalnej możliwości wykonywania usług.

Z przedstawionymi metodami wiążą się problemy zarządzania typowe dla rozproszonej bazy danych. Na przykład: z dwóch miejsc wychodzi żądanie uaktualnienia tego samego zasobu różnymi wartościami. Po zakończeniu aktualizacji zasób widziany z każdego miejsca ma być taki sam. Możemy to osiągnąć poprzez:

- *Centralne sterowanie*: jest to rozwiązanie raczej nie zalecane z następującego powodu: nawet gdy dostęp do komponentów jest równoważony, centralna kontrola nad przebiegiem operacji wymaga monitorowania i powoduje niepotrzebny wzrost obciążenia systemu. Dochodzą do tego nakłady komunikacyjne – istotne, gdy baza jest silnie rozproszona, o zasięgu światowym.
- *Statyczna wymiana informacji*: lokalne centra kontrolne komunikują się między sobą według ustalonego algorytmu synchronizacji. Rozwiązanie to jest dobre dla prostych, ustalonych z góry operacji. Zaawansowane operacje wyszukiwania czy aktualizacji trudno jest zrealizować w takim układzie, poza tym rekonfiguracja systemu może też nastręczać tu pewne kłopoty.
- *Wykorzystanie mobilnych agentów*: jest to rozwiązanie, które pomimo dużej złożoności implementacyjnej daje największe możliwości. Dobrze przygotowany agent potrafi dynamicznie dostosować się do warunków panujących w danym systemie, co pozwala zwiększyć efektywność przetwarzania. Poza tym wędrującego agenta można modyfikować w różny sposób, co ułatwia proces rekonfiguracji systemu.

Istnieją obecnie dobrze udokumentowane standardy koordynacji usług dla architektur rozproszonych typu CORBA (*Common Object Request Broker Architecture*) [3].

3. Przykład aplikacji

Na podstawie przedstawionych modeli i opisów ich funkcjonowania można spróbować stworzyć projekt aplikacji internetowej (lub raczej systemu), który, realizując odpowiednie funkcje, zachowywać będzie własności zintegrowanych poziomów. Przykładem takiego projektu może być System ERS (Endoscopy Recommender System) [9]. W zamierzeniu jest on systemem mającym umożliwić wykonywanie zaawansowanych operacji na danych uzyskiwanych z badań endoskopowych, lub ściślej, z dziedziny gastroenterologii. Wśród jego funkcji wyróżnić można m. in.:

- Obróbkę obrazów w celu poprawienia jakości i umożliwienia diagnozowania.
- Archiwizowanie danych w globalnie dostępnej bazie, w formie rekordów pacjenta.
- Wspomaganie diagnozowania poprzez funkcje porównywania obrazów i szukania podobieństw wśród danych zawartych w bazie.
- Ustalanie wyników diagnozy przez grupę ekspertów za pośrednictwem Internetu.

Standard składowania i przesyłania danych w systemie zgodny będzie z normą DICOM, określającą aspekty transmisji i przechowywania obrazów medycznych.

Model architektury systemu ERS opisujący współpracę jego komponentów, przedstawiony na rys. 6, bazuje na usystematyzowanej koncepcji hybrydowej (porównaj rys. 5 i 6). Podstawowe komponenty wykorzystywane w systemie to: zasoby, usługi, system sterowania, interfejsy, przeglądarka/wyszukiwarka i system endoskopowy. Jako zasoby wykorzystuje się bazy danych zawierające informacje o pacjentach (rekordy pacjentów), takie jak: dane personalne, wywiady, badania, rozpoznanie, leczenie. Poza tym analiza poszczególnych obrazów medycznych wymaga dużej mocy obliczeniowej. Dlatego też jako zasoby wykorzystuje się superkomputery (komputery dużej mocy obliczeniowej), zainstalowane w dostępnych centrach komputerowych (np. CI TASK).

Spośród różnych usług najistotniejsze są następujące kategorie:

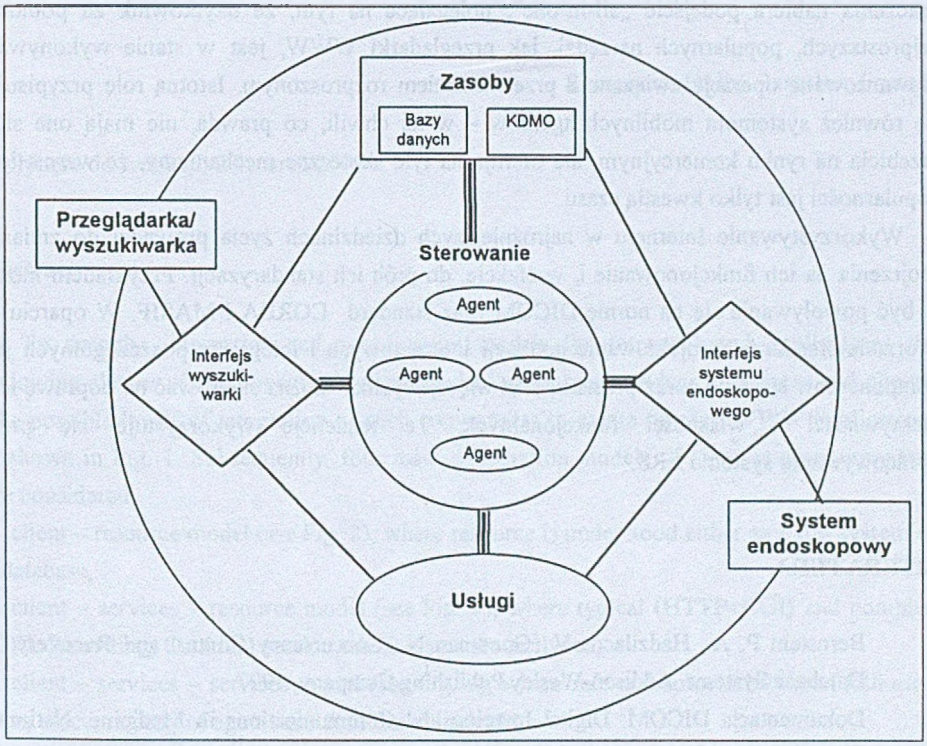
- Archiwizacja mająca na celu wstępną obróbkę i zapamiętanie rekordów pacjenta.
- Wyszukiwanie określonej informacji o pacjentach lub o rozwoju choroby, czy o postępach leczenia.
- Klasyfikacja, która bezpośrednio dotyczy analizy obrazów medycznych i rozpoznawania na nich charakterystycznych obiektów.
- Wspomaganie diagnozowania, które ma na celu udzielanie odpowiednich rekomendacji co do ujawnienia się danej jednostki chorobowej, bądź postępowania przy jej leczeniu.

Sterowanie lub zarządzanie polega na zapewnieniu koordynacji funkcjonowania poszczególnych komponentów systemu ERS i bazuje na koncepcji mobilnych agentów. W celu umożliwienia takiej współpracy definiowane są odpowiednie interfejsy pomiędzy komponentami. Poza tym do zarządzania całością systemu ERS wykorzystuje się odpowiednią metawyszukiwarkę współpracującą z dostępnymi wyszukiwarkami.

System endoskopowy pozwala na uzyskiwanie danych z badań w postaci obrazów medycznych oraz zapewnia wstępne ich przygotowanie do przeprowadzenia diagnozy i stworzenia rekordu pacjenta. Istnieje tutaj ścisła współpraca pomiędzy tym systemem i usługami archiwizacji.

Przedstawione rozwiązanie zakłada silne rozproszenie systemu. Założenie to zostało przyjęte z myślą o przyszłości, bowiem w planach jest stworzenie globalnej sieci systemów

ERS lub kompatybilnych, które, pracując w oparciu o normę DICOM, stanowić będą złożoną bazę konsultacyjno-diagnostyczną działającą w Internecie.



Rys. 6. Architektura systemu ERS
Fig. 6. ERS System architecture

Należy zaznaczyć, że powyższy model architektury jest dopiero wstępnym projektem i nie wiadomo, czy wszystkie jego założenia odnośnie do elastycznego zarządzania uda się zrealizować w proponowanym kształcie (ograniczenia czasowe i finansowe). Jednak jest to podejście wzorujące się na opisanych wcześniej modelach, głównie na modelu hybrydowym, co prowadzi do zminimalizowania ryzyka nieudanego zakończenia projektu, gdyż unika się problemów, które mogą się pojawić w przypadku tworzenia całkiem nowej koncepcji architektonicznej.

4. Uwagi końcowe

Przedstawiono aktualne tendencje rozwoju aplikacji internetowych. Coraz większego znaczenia nabiera podejście „all-in-one", polegające na tym, że użytkownik za pomocą najprostszych, popularnych narzędzi, jak przeglądarki WWW, jest w stanie wykonywać zaawansowane operacje związane z przetwarzaniem rozproszonym. Istotną rolę przypisuje się również systemom mobilnych agentów – w tej chwili, co prawda, nie mają one siły przebicia na rynku komercyjnym, ale oferują na tyle skuteczne mechanizmy, że wzrost ich popularności jest tylko kwestią czasu.

Wykorzystywanie Internetu w najróżniejszych dziedzinach życia prowadzi do zmiany spojrzenia na ich funkcjonowanie i, w efekcie, do prób ich standaryzacji. Przykładem może tu być powoływanie się na normę DICOM oraz standard CORBA i MASIF. W oparciu o tworzone standardy, projektowanie aplikacji internetowych i integracja poszczególnych jej komponentów staje się coraz prostsza, stąd więcej wysiłku można skierować na poprawę ich efektywności i własności funkcjonalnych. Te tendencje wykorzystuje się przy opracowywaniu systemu ERS.

LITERATURA

1. Bernstein P. A., Hadzilacos V., Goodman N.: Concurrency Control and Recovery in Database Systems. Addison-Wesley Publishing Company, 1997.
2. Dokumentacja DICOM: Digital Imaging and Communications in Medicine. National Electrical manufactures Association, USA, 1999.
3. Dokumentacja OMG: CORBA Services: Common Object Services Specification styczeń 1997.
4. Dokumentacja OMG: Mobile Agent System Interoperability Facilities Submission OMG, 1997.
5. Fielding R., Gettys J., Mogul J., Frystyk H., Berners-Lee T.: Hypertext Transfer Protocol - HTTP/1.1. http://www.cis.ohio-state.edu/htbin/rfc/rfc2068.html, styczeń 1997.
6. Gruber M.: SQL. Helion, Gliwice 1996.
7. Hunter J., Crawford W.: Java Servlet Programming. O'REILLY, USA, październik 1998.
8. Knopa R., Krajewska K, Krawczyk H., Lipczyński M.: Architektura systemu ERS - opracowanie grantowe. KASK, ETI, Politechnika Gdańska, Gdańsk styczeń 2000
9. Microsoft Corporation: Microsoft ODBC. http://www.microsoft.com/data/odbc, marzec 1999

10. Postel J., Reynolds J.: File Transfer Protocol (FTP). http://www.cis.ohio-state.edu/htbin/rfc/rfc959.html, październik 1985.
11. Praca Zbiorowa.: KASKBOOK – Przetwarzanie sieciowe i rozproszone. Politechnika Gdańska, Gdańsk 1999.

Recenzent: Dr inż. Jarosław Francik

Wpłynęło do redakcji: 27 marca 2000 r.

Abstract

To describe integration and coordination models for Internet-based applications the following classes of components are distinguished: resources, services, interfaces and clients. The possible levels of integration of such components to create running WWW-applications is shown in Fig. 1. Subsequently, four main cooperation models of application components are considered:

- client – resource model (see Fig. 2), where resource is understood either as a file system or a database,
- client – services – resource model (see Fig. 3), where typical (HTTP+CGI) and complex (URL Rewriting, Cookies) session mechanisms are analyzed,
- client – services – services model (see Fig. 4), where role and some attributes (authority, type, place, region) of agents are considered in detail,
- hybrid model (see Fig. 5) where the above all models can be used in any application with various coordination strategies (centralized, static, dynamic), CORBA architectures are discussed.

Basing on these considerations and taking into account principal requirements for a recommender system, which can be used to support medical diagnosis, a general architecture of such a system is determined (see Fig. 6). The system is described by the hybrid application model and needs for implementation new standards such as DICOM, MASIF and CORBA. In that way we create a pattern Internet application for storing, retrieving and recognition of medical images taken during patient examination.