ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: INFORMATYKA z. 34 Nr kol. 1381

Andrzej USZOK, Andrzej MIAZGA, Tomasz KAŹMIERCZUK
Akademia Górniczo-Hutnicza, Katedra Informatyki

UNIWERSALNE NARZĘDZIE DO TESTOWANIA SERWERÓW CORBA[1]

Streszczenie. Niniejszy artykuł prezentuje program umożliwiający testowanie funkcjonalności i konfiguracji dowolnych serwerów napisanych zgodnie ze standardem CORBA. W programie intensywnie wykorzystano rozbudowaną zdolność samoopisywania się programów działających w tej architekturze. Standard CORBA staje się obecnie podstawą dla budowy trzypiętrowych systemów WWW, i nie tylko, a prezentowane narzędzie znacząco ułatwia proces testowania takich systemów.

UNIVERSAL TOOL FOR TESTING CORBA SERVERS

Summary. The paper presents the tool enabling testing of functionality and configuration of any servers written in the CORBA system. The extended CORBA feature of self-description of CORBA servers is extensively used in the programme. The CORBA standard is now emerging as a fundation of the three-tired WWW and other applications and presented tool significantly facilitates the testing process of such systems.

1. Architektury trzypiętrowe współczesnych programów

Standard CORBA [1] [6], organizacji OMG, znacząco ułatwia tworzenie obiektowo-zorientowanych programów rozproszonych. Nowe programy WWW, ale nie tylko, są obecnie coraz częściej konstruowane z komponentów opisanych za pomocą języka CORBA IDL [2]. Umożliwia to opakowanie istniejącego już kodu programów napisanych w prawie dowolnym języku w powłokę obiektową. Na przykład progam bazy danych składający się z wielu milionów linii kodu napisanego w języku COBOLA może zostać łatwo zamieniony w komponent CORBA i dostarczać dane dla appletu, napisanego w Java, udostępniającego je poprzez stronę

[1] Artykuł opracowano w ramach grantu KBN nr 8T11C01210

WWW. Taka właśnie architektura systemu składająca się z trzech warstw nazwana została architekturą trzypiętrową [3]. Warstwy te to:

- **Klient** - warstwa prezentacji danych zrealizowana najczęściej jako applet Java potrafiący jednocześnie komunikować się z innymi komponentami przez protokół IIOP [1].
- **Warstwa Środkowa** (*Middleware*) - warstwa logiki programu zrealizowana jako zespół kooperujących ze sobą komponentów CORBA, koordynowanych przez obiektowy monitor przetwarzania transakcji (*TP monitor*).
- **Zaplecze** (*Back End*) - warstwa dostarczająca dane, czyli relacyjne i obiektowe bazy danych, e-mail, Lotus Notes i inne.

Skomplikowanie komponentów takich systemów jest najczęściej bardzo duże, jak na przykład w systemie *Active Badge next generation* (*ABng*) [7] budowanym zgodnie z zaprezentowaną architekturą w Katedrze Informatyki AGH. Z drugiej strony komponenty powinny być niezawodne,tak aby możliwe było ich użycie nie tylko w danym systemie, ale i w systemach budowanych w przyszłości, co jest jedną z ich podstawowych zalet. Dlatego niezbędne jest opracowanie metodologii i narzędzia testowania komponentów CORBA. W artykule tym zaprezentowano propozycję częściowego rozwiązania tego problemu.

2. Koncepcja narzędzia testującego

Prezentowane narzędzie testujące umożliwia wywoływanie dowolnej operacji, z dowolnymi argumentami na wybranym serwerze. Narzędzie na podstawie metawiedzy zawartej w bazach danych systemu CORBA prowadzi użytkownika przez kolejne etapy testowania funkcjonalności serwera. Metawiedza systemu CORBA zawiera bowiem informacje o poszczególnych aspektach konfiguracji i funkcjonalności programów. Jest ona dynamicznie aktualizowana w trakcie rozwijania i działania programu, co pozwala uzyskiwać bieżące informacje. Dwie podstawowe bazy metawiedzy o stanie systemu intensywnie wykorzystywane przez narzędzie to: *Interface Repository* (*IR*) [1] i *Trader* [4]. Należy podkreślić, że w trakcie budowy narzędzia starano się wykorzystać maksimum wiedzy zawartej w tych bazach danych, tak aby przedstawiać użytkownikowi możliwe warianty parametrów, a nie zakładać, że wprowadzi on je z pamięci lub innego źródła.

2.1. Interface Repository

Dostęp do zbioru obiektów rozproszonych w systemie możliwy jest poprzez ich publiczne interfejsy zdefiniowane w języku IDL. Korzystanie z usług dostarczanych przez obiekt wyma-

ga od klienta znajomości ich definicji. Definicje operacji obiektu mogą być udostępnione na dwa sposoby:

- Statycznie - przy mapowaniu specyfikacji interfejsu obiektu w IDL na konkretny język programowania generowane są wzorcowe metody, które można następnie wykorzystać podczas pisania aplikacji klienta.
- Dynamicznie - aplikacja klienta uzyskuje definicję operacji dostarczanych przez obiekt podczas wykonywania, pobierając je z *IR*.

IR jest bazą umożliwiającą przechowywanie i zarządzanie definicjami interfejsów obiektów. Definicja interfejsu zawiera szczegółowy opis operacji dostarczanych przez obiekt, czyli typy parametrów, wyjątki, które mogą wystąpić podczas ich wykonania i informacje kontekstowe, których mogą one używać. *IR* zawiera także definicje stałych, które tworzone są zwykle dla wygody programisty. Przeglądanie tej bazy ułatwia jej strukturalna budowa, zwykle tworzona na podstawie definicji modułów w IDL. Moduły używane są do grupowania logicznie powiązanych interfejsów, modułów, stałych, wyjątków i definicji typów. Specyfikacja CORBA określa zbiór operacji umożliwiających dostęp do definicji przechowywanych w *IR*. Operacje te zdefiniowane są w dwóch abstrakcyjnych interfejsach: *Container* i *Contained*. *Container* reprezentuje obiekt zawierający inne obiekty, które są typu *Contained*. Te z kolei posiadają operacje służące do odczytywanie swojej zawartości, co pozwala na nawigację poprzez strukturę *IR*, natomiast *Contained* dostarcza metod do opisu obiektu, który reprezentuje. Dzięki tym interfejsom, rozpoczynając od największego kontenera i kolejno zagłębiając się w strukturę, można uzyskać szukaną informację.

2.2. Trader

Jest to baza danych, która umożliwia klientom uzyskanie dodatkowych informacji o usługach dostarczanych przez obiekty określonego typu. Na podstawie uzyskanych wiadomości klient może wybrać konkretnego dostawcę usługi. Obiekt może zareklamować swoją usługę, poprzez podanie jej właściwości i lokalizacji interfejsu, który ją udostępnia. Dzięki temu, kiedy klient pyta *Tradera* o usługę posiadającą pewne określone własności, ten sprawdza opisy, które posiada i zwraca informację o lokalizacji interfejsu spełniającego podane wymagania. W rezultacie klient może uzyskać usługę bezpośrednio od otrzymanego obiektu. *Trader* może być podzielony na współpracujące obiekty rozproszone po sieci, co umożliwia szerokie wykorzystanie zawartych w nim informacji.

Z każdą usługą przechowywaną w *Traderze* związany jest jej typ, który zawiera informacje potrzebne do jej opisania. Obejmuje on typ interfejsu dostarczającego usługę (dokładny jego opis można uzyskać z *IR*) i typy własności. Własności mogą być trzech rodzajów:

- Zwykłe - wartość własności ustalana jest przy rejestracji usługi w *Traderze* i od tej pory jest niezmienna,
- Modyfikowalne - wartość własności może być zmieniana po rejestracji usługi,
- Dynamiczne - wartość własności nie jest podawana przy rejestracji; w przypadku gdy jest potrzebna, jest na bieżąco uzyskiwana od dostawcy usługi.

Jedną z podstawowych operacji *Tradera* jest *query*, która pozwala klientowi dokładnie określić parametry usług, którymi jest on zainteresowany. Do parametrów tych należą:

- Typ usługi.
- Ograniczenia - wyrażenie podlegające składni języka OMG *Constraint Language* [4] i składające się z nazw właściwości wybranego typu serwisu, stałych i operatorów. Służy on do określania dodatkowych wymagań odnośnie do funkcjonalności i własności usługi.
- Preferencje - wyrażenie określające porządek, w jakim zwracane będą usługi spełniające wymagania odnośnie do typu i pasujące do ograniczeń. W ten sposób na początku zwrócone będą oferty najlepiej pasujące.
- Zasady - specyfikują sposób, w jaki będzie wykonywana operacja poszukiwania usług. Są to w pewnym sensie parametry wewnętrznych algorytmów *Tradera*.
- Pożądane własności - określa te własności zwracanych usług, na podstawie których klient chce ostatecznie wybrać konkretną usługę.

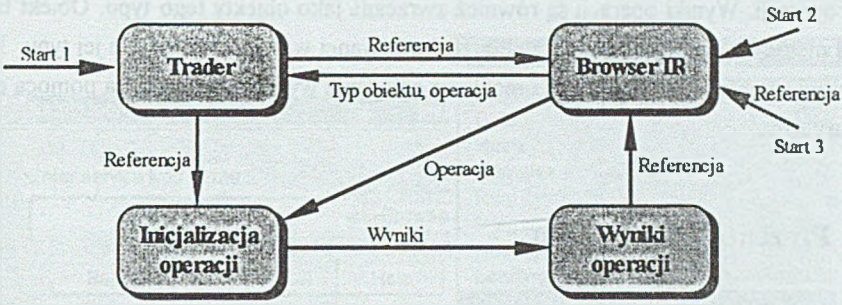
Trader może przechowywać także usługi zastępcze. Usługa taka jest tylko opisem zwykłej usługi dostępnej na innym obiekcie *Tradera*. Jeśli operacja wyszukiwania wybierze taką usługę, wtedy szczegółowe informacje zwracane do klienta pobierane są z odległego obiektu *Tradera*, która zawiera właściwy opis usługi.

2.3. Schemat funkcyjny narzędzia testującego

Kolejne fazy pracy z narzędziem przedstawione są na rys. 1. Jak widać, bazy danych przedstawione w poprzednich punktach są w trakcie testowania serwera intensywnie wykorzystywane. Program można rozpocząć na trzy sposoby:

- Start 1 - rozpoczęcie od *Tradera*, który pozwala na wybranie referencji do testowanego serwera. Następnie *Browser IR* umożliwia wybranie operacji z interfejsu tego serwera. Operacja ta jest inicjalizowana w module Inicjalizacja i następnie wywoływana. Wyniki operacji pokazywane są przez moduł Wyniki, który umożliwia przejście do modułu *Browser IR* w przypadku, gdy operacja zwróci referencję do obiektu. Wtedy dla tej referencji dokonywany jest wybór operacji, jej inicjalizacja, wywołanie i przegląd wyników. Tę pętlę można powtarzać dopóki kolejne operacje zwracają referencje.
- Start 2 - rozpoczęcie od *Browser IR*, który umożliwia wyszukanie interfejsu serwera i wybranie z niego operacji. Po tym następuje przejście do *Tradera* i wybranie konkretnego

serwera posiadającego wybrany wcześniej interfejs. Dalej następuje przejście do modułu Inicjalizacja, gdzie ustawiane są wartości parametrów operacji, po czym operacja jest wywoływana i pokazywane są jej wyniki. Analogicznie jak w pierwszym przypadku referencja zwrócona przez operację może być dalej wykorzystana do wywoływania kolejnych operacji.



Rys. 1. Schemat funkcyjny narzędzia testującego
Fig. 1. Functional scheme of the tool

- Start 3 - użytkownik bezpośrednio podaje referencję do testowanego serwera, po czym dalsze działania są analogiczne jak w pierwszym przypadku.

2.4. Główne problemy - budowa dynamicznych formatek

Jednym z głównych zadań programu jest dostarczenie użytkownikowi narzędzia umożliwiającego inicjalizację parametrów testowanej operacji. Typy parametrów mogą być dowolnie skomplikowane, co stwarza problemy z ich graficzną prezentacją. Przy dynamicznej inicjalizacji parametru wykorzystywane są dwie standardowe koncepcje specyfikacji CORBA: *TypeCode* i *Any*. *TypeCode* jest to interfejs używany do opisu dowolnie złożonego typu IDL, natomiast *Any* służy do przechowywania wartości dowolnego typu. Pobrane z *IR* opisy typów parametrów podane są właśnie za pomocą *TypeCode*. Każdy *TypeCode* zawiera pole *kind* określające rodzaj przechowywanego typu i sekwencję parametrów, dostarczających szczegółowych informacji o tym typie. Ilość parametrów i znaczenie każdego z nich zależy od rodzaju typu. W budowie tych parametrów w sposób rekurencyjny używany jest *TypeCode*. Na przykład dla typu *struct* parametry zawierają nazwę struktury i opis każdego z jej składników, tzn. jego typ i nazwę. Typ składnika jest to *TypeCode*, który może znów opisywać strukturę. *TypeCode* zawiera więc hierarchiczny opis typu - rozpoczynając od najwyższego poziomu można stopniowo poznawać kolejne jego elementy. Ten sposób konstrukcji *TypeCode* stanowi podstawę budowy interfejsu graficznego umożliwiającego dynamiczną inicjalizację para-

metru o dowolnym typie. Dokładniejszy opis tego interfejsu nazwanego edytorem struktury *Any* znajduje się w punkcie 3.4.

Na podstawie wartości wprowadzonych przez użytkownika tworzony jest obiekt typu *Any* przekazywany jako wartość parametru wywoływanej operacji do mechanizmu standardu CORBA *Dynamic Invocation Interface* (*DII*), który umożliwia dynamiczne wywołanie testowanej operacji. Wyniki operacji są również zwracane jako obiekty tego typu. Obiekt taki dostarcza metody umożliwiające pobranie przechowywanej wartości i poznanie jej typu. Typ ten jest opisywany przez *TypeCode*, co umożliwia oglądanie wyników operacji za pomocą edytora struktury *Any*.

3. Prezentacja programu

Program CORBA Browser umożliwia użytkownikowi wykonywanie operacji na obiektach dostępnych w systemie rozproszonym CORBA, a przez to testowanie ich poprawności. Aplikacja nie wymaga od użytkownika praktycznie żadnej znajomości systemu CORBA, a przez to pozwala mu korzystać z operacji dostarczanych przez obiekty rozproszone w sieci bez wykonywania jakichkolwiek operacji sieciowych. Zrealizowane jest to poprzez dostarczenie prostego i wygodnego w użyciu interfejsu graficznego, w którym większość operacji można wykonać poprzez wybieranie krok po kroku przedstawianych przez program propozycji.

3.1. Trzy sposoby użycia narzędzia

Pierwszym etapem programu jest wybór testowanego serwera. Narzędzie umożliwia zrealizowanie tego na trzy różne sposoby:

- Wyszukanie testowanego serwera w *Traderze*,
- Wyszukanie interfejsu serwera w *IR*, a następnie wybranie serwera tego typu za pomocą *Tradera*,
- Bezpośrednie podanie referencji do serwera.

Wybór jednej z tych możliwości udostępnia okno startowe programu. Przeglądanie zarówno *Tradera*, jak i *IR* wymaga uzupełnienia pól tej formatki z ich referencjami. Trzeci z wymienionych wariantów wymaga podania referencji do *IR* i serwera. Możliwy jest również wybór typu opisów pojawiających się w programie, które mogą być formalne lub bardziej przyjazne dla użytkownika.

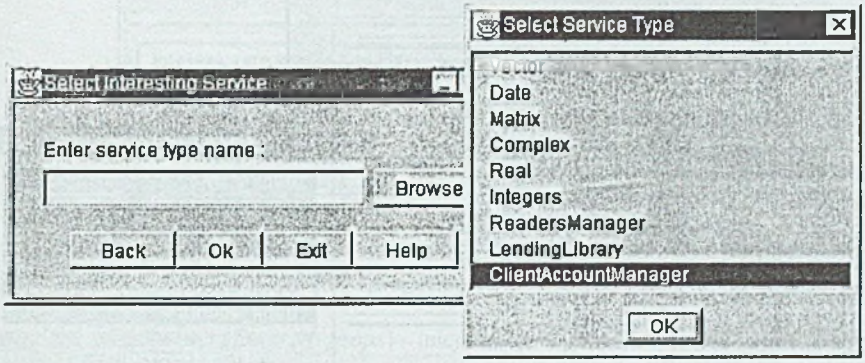
3.1.1. Określanie referencji serwera z użyciem Tradera

Ta opcja umożliwia przetestowanie nie tylko funkcjonalności serwera, ale również poprawności jego zarejestrowania w systemie w bazie danych *Tradera*. Wyszukiwanie referencji

serwera w *Traderze* wymaga wykonania kolejno czynności: wybór typu usługi testowanego serwera, inicjalizowanie parametrów wyszukiwania oraz wybór jednej z uzyskanych referencji, czyli wyselekcjonowanie testowanego serwera dostarczającego usługę.

3.1.1.1. Wybór typu usługi testowanego serwera

Do wyboru typu usługi służy okienko programu „Select Interesting Service" (rys. 2).



Rys. 2. Okienko wyboru typu usługi testowanego serwera
Fig. 2. Selection of the tested server type window

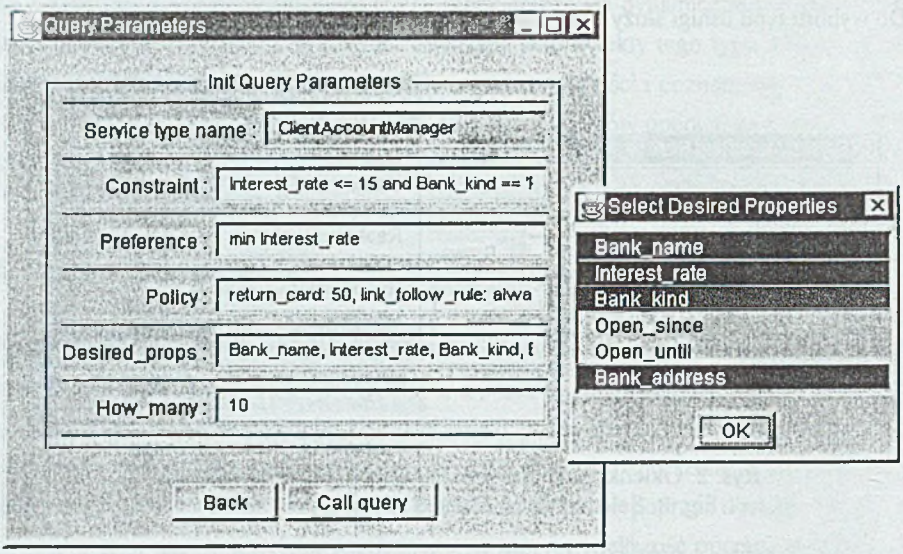
Nazwę typu usługi można wprowadzić bezpośrednio w polu tekstowym lub wybrać ją z listy. Lista ta jest tworzona na podstawie zawartości bazy danych typów *Tradera* i uzyskiwana jest za pomocą metody *list_types*. Przykładowa lista pokazana jest w okienku „Select Service Type". Po wybraniu typu usługi i naciśnięciu przycisku „OK" następuje sprawdzanie, czy opis interfejsu serwera dostarczającego tą usługę znajduje się w *IR*. Jeśli opis ten nie zostanie znaleziony, nastąpi wypisanie ostrzeżenia, po czym można wybrać inny typ usługi. Już w tym momencie można więc wykryć błędy w konfiguracji testowanego programu.

3.1.1.2. Inicjalizowanie parametrów wyszukiwania

Po wybraniu typu usługi testowanego serwera należy zainicjalizować parametry, które pozwalają na sprecyzowanie kryteriów wyszukiwania i własności pożądanej usługi. Do inicjalizacji tych parametrów służy okienko programu „Query Parameters" pokazane na rys. 3.
W okienku tym znajdują się następujące parametry operacji wyszukiwania:

- Typ usługi (*Service Type Name*) - pokazuje nazwę typu usługi już wybranego w poprzednio opisanym kroku.
- Ograniczenia (*Constraint*) - jest to wyrażenie nakładające ograniczenia na wartości właściwości wybranego typu usługi. Wyrażenie to jest budowane za pomocą specjalnego okienka (rys.4). Umożliwia ono budowanie krok po kroku wyrażenia zgodnego z regułami gramatyki języka Constraint, poprzez aktywowanie w danym momencie tylko tych przycisków, których wartości są prawidłowe według tej gramatyki. Wyrażenie to może się

składać z: operatorów porównania, logicznych i matematycznych oraz ze stałych i nazw właściwości danego typu usługi. Również w tym momencie wybór użytkownika jest wspomagany przez okienko z listą nazw właściwości uzyskanych z *Tradera*.

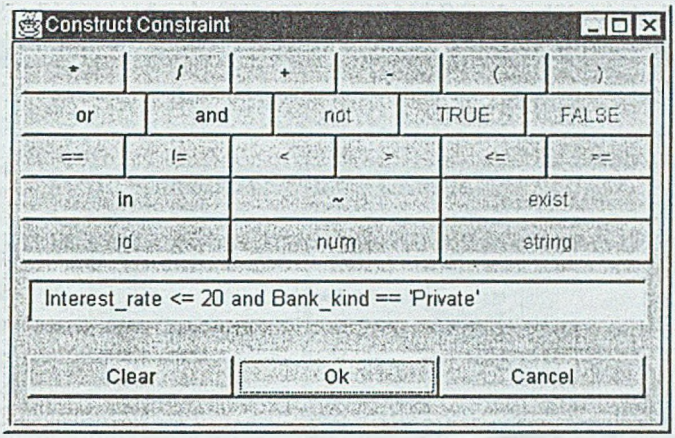


Rys. 3. Okienko programu umożliwiające zainicjalizowanie parametrów wyszukiwania referencji testowanego serwera

Fig. 3. Window enabling initialization of parameters for the query operation about the server

- Preferencje uporządkowania znalezionych ofert (*Preference*) - wyrażenie to może mieć jedną z następujących postaci:
  - max | min *wyrażenie* - oferty porządkowane są w kolejności odpowiednio od największej do najmniejszej lub od najmniejszej do największej według wartości zwracanej przez *wyrażenie*, gdzie *wyrażenie* jest tworzone według języka Constraint i musi zwracać liczbę,
  - with *ograniczenie* - oferty porządkowane są w kolejności: najpierw spełniające podane *ograniczenie*, później pozostałe,
  - random - kolejność ofert jest losowa,
  - first - oferty porządkowane są w kolejności, w jakiej były dopasowywane.

  Tworzenie tego wyrażenia jest wspierane również przez okienko przedstawione na rys. 4. oraz pomocnicze okno wyboru typu wyrażenia preferencji.
- Zasady wyszukiwania wewnątrz serwera (*Policies*) - parametr ten jest sekwencją par (nazwa, wartość). Program dostarcza nie zaprezentowane tutaj okienka pozwalające określić wartość dla każdego dozwolonego elementu sekwencji.



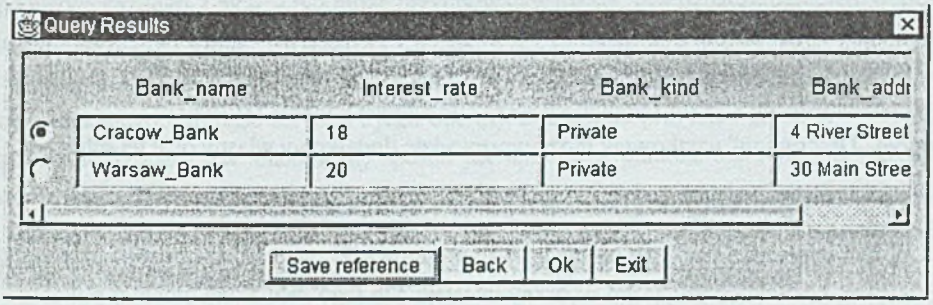
Rys. 4. Okienko programu do tworzenia ograniczeń na wyszukiwanie
Fig. 4. Query constraint construction window

- Pożądane właściwości (*Desired_props*) - inicjalizacja tego pola jest wspierana przez okienko „Select Desired Properties" pokazane na rys. 3., z którego można wybrać te nazwy właściwości, których wartości powinny być zwracane wraz z wybraną referencją serwera.
- Ilość (*How_many*) - określa ilość ofert, które zostaną zwrócone bezpośrednio do klienta.

Po zainicjalizowaniu parametrów wyszukiwania należy nacisnąć przycisk „Call query", co spowoduje wywołanie operacji wyszukiwania w *Traderze*.

3.1.1.3. Wybór referencji testowanego serwera

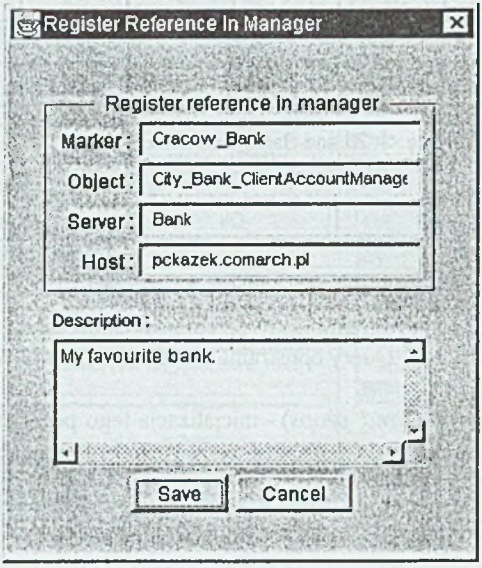
Referencje serwerów pasujące do zadanych kryteriów zwrócone przez operację wyszukiwania wyświetlone są w specjalnym okienku (rys. 5).



Rys. 5. Okienko z referencjami zwróconymi przez operację wyszukiwania
Fig. 5. Window with offers returned by the query operation

W okienku tym przedstawione są kolejno zwrócone referencje. Nazwy kolumn są nazwami wybranych własności usług. W poszczególnych polach tabeli pokazane są wartości tych

właściwości dla kolejnych usług. Znajdujący się w tym okienku przycisk „Save reference" służy do zapamiętania referencji do zaznaczonego obiektu dostarczającego usługę. Po naciśnięciu tego przycisku pokazuje się okienko widoczne na rys.6.



Rys. 6. Zapisywanie referencji w podręcznym notatniku referencji
Fig. 6. Saving object reference in the reference bookmark

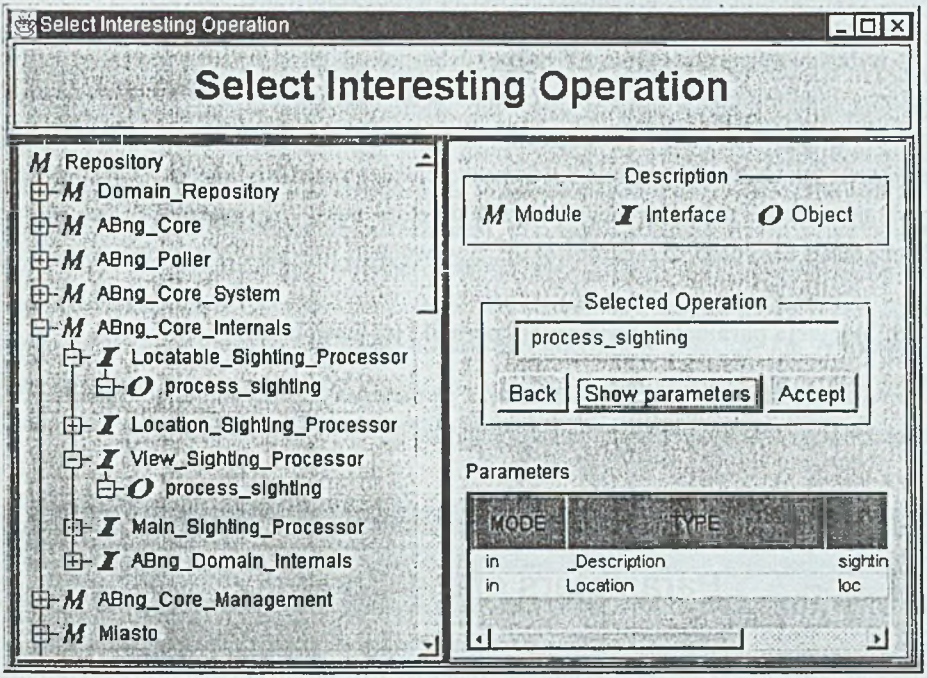
Okienko to przedstawia dokładny opis referencji do obiektu uzyskany na podstawie analizy informacji zawartych w referencji, co obejmuje:

- Nazwę własną obiektu (*Marker*),
- Nazwę typu obiektu (*Object*),
- Nazwę własną serwera, w którym znajduje się obiekt (*Server*),
- Nazwę hosta, na którym znajduje się serwer zawierający obiekt (*Host*).

W polu „Description" użytkownik może wprowadzić dodatkowy własny opis tej referencji.

3.1.2. Przegladanie funkcjonalności serwerów z użyciem IR

W tym przypadku najpierw następuje wybór testowanej operacji, zaś później etap wyboru testowanego serwera oferującego wybrany interfejs. Do tego celu służy okienko pokazane na rys. 7. Okienko to umożliwia znalezienie interfejsu dostarczającego pożądaną testowaną usługę poprzez przeszukiwanie struktury *IR*. Przeglądanie to rozpoczyna się od najwyższego poziomu. Wybierając nazwę modułu można zobaczyć moduły i interfejsy w nim zawarte. W ten sposób można zagłębiać się w strukturze drzewa i oglądać jego wybrane elementy.



Rys. 7. Okienko umożliwiające przeglądanie zawartości *IR*
Fig. 7. Interface Repository browsing window

Po wybraniu nazwy konkretnego interfejsu pokażą się jego operacje. Po zaznaczeniu operacji jej nazwa pojawi się w polu tekstowym „Selected Operation", a po naciśnięciu przycisku „Show parameters" w polu „Parameters" wyświetlony zostanie opis jej parametrów, czyli tryb, typ i nazwę każdego z nich. Tryb może przyjmować wartości: *in* - oznacza parametr wejściowy, *out* - wyjściowy i *inout* - wejściowy i wyjściowy. Typ parametru może prosty: *short*, *long*, *float*, *double*, *char*, *boolean*, *octet* lub złożony: *struct*, *union*, *string*, *array*, *sequence*. Po wybraniu szukanego interfejsu serwera i jego operacji naciśnięcie przycisku „Accept" powoduje przejście do *Tradera* i w analogiczny sposób jak opisane w punkcie 3.2 wybranie referencji testowanego serwera. Jednak w tym przypadku w okienku „Select Service Type" pokazanym na rysunku 2 widoczne są tylko te typy, które dostarczają interfejs wybrany przez użytkownika z *IR*. Po wyborze obiektu następuje przejście do etapu inicjalizacji operacji.

3.1.3. Bezpośrednie podanie referencji testowanego serwera

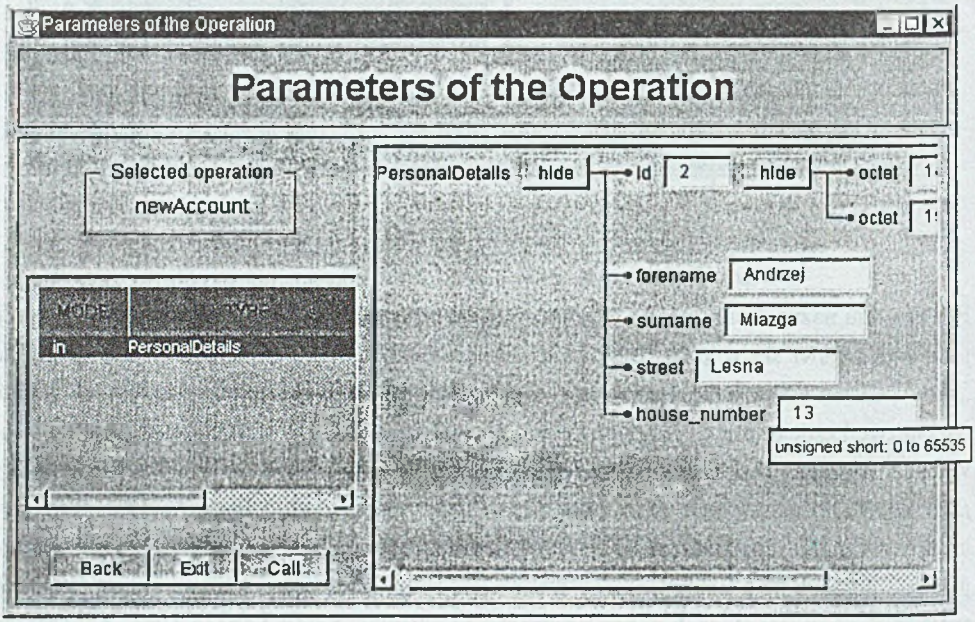
Trzecią możliwością jest bezpośrednie wprowadzenie referencji serwera na początku działania narzędzia. W tym przypadku następuje bezpośrednie przejście do przeglądania interfejsu tego serwera w *IR*. Informacja o tym, jaki jest to interfejs, uzyskiwana jest z podanej referencji serwera.

3.2. Przeglądanie funkcjonalności serwera i wybór testowanej operacji

Po wybraniu referencji serwera z *Trader*a lub bezpośrednim jej podaniu następuje wyświetlenie jego interfejsu pobranego z *IR* w okienku podobnym jak na rys. 7. W tym przypadku jednak możliwe jest wyłącznie przeglądanie interfejsu wybranego serwera, nie zaś całego *IR*. Analogicznie jak opisano w punkcie 3.1.2 można oglądać parametry każdej operacji obiektu. Po wybraniu jednej z metod następuje przejście do etapu inicjalizacji jej parametrów, jeśli posiada ona jakieś argumenty, lub bezpośrednie jej wywołanie, jeśli jest bezargumentowa.

3.3. Inicjacja parametrów wybranej operacji i jej wywołanie

Po wybraniu operacji należy zainicjalizować jej parametry, z którymi operacja będzie wywołana. Do tego celu służy okienko pokazane na rys. 8.



Rys. . Okienko programu inicjalizacji parametrów testowanej operacji
Fig. 8. Window enabling initialization of operation's parameters

W okienku tym po lewej stronie wypisywane są parametry wybranej operacji, z których należy zainicjalizować te, które mają tryb „in" lub „inout" (w pokazanym na rysunku przypadku operacja posiada tylko jeden parametr). W celu zainicjalizowania parametru należy go wybrać, wtedy po prawej stronie okna w edytorze struktury *Any* wyświetlona zostanie formatka zbu-

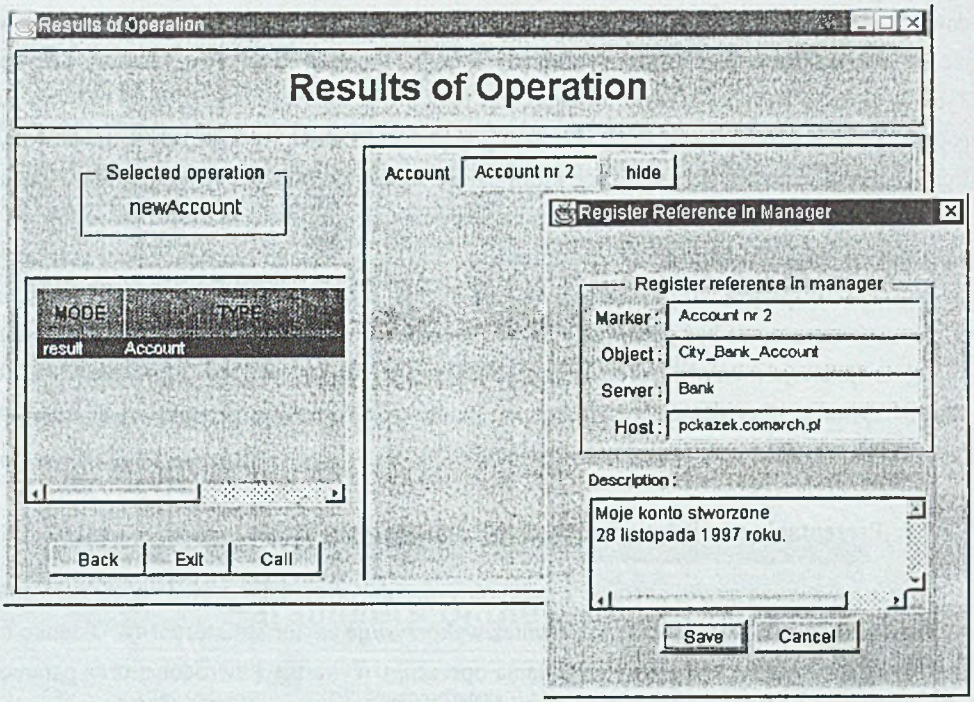
dowana na podstawie *TypeCode* tego parametru. Formatka ta pozwala na łatwe inicjalizowanie wszystkich, nawet bardzo skomplikowanych typów poprzez stworzenie struktury drzewa. Typy złożone są węzłami drzewa formatki, zaś typy proste są liśćmi. Inicjalizacja odbywa się poprzez kolejne rozwijanie węzłów, do czasu gdy widoczne staną się liście, które umożliwiają wprowadzenie wartości. Typ wartości, którą należy wpisać do pola tekstowego, można zobaczyć ustawiając kursor myszki na nazwie pola. Wpisanie niepoprawnej wartości powoduje wyświetlenie okienka z odpowiednim ostrzeżeniem. Przykład formatki dla typu *struct* pokazany jest na rys. 8. W przypadku gdy wygenerowane drzewo jest zbyt duże i nie jest widoczne na ekranie, wówczas może być przewijane. Jeżeli parametrem operacji jest referencja do obiektu, to jego inicjalizacja możliwa jest dzięki notatnikowi interfejsów obiektów lub *Traderowi*. Po zainicjalizowaniu wszystkich parametrów „in" i „inout" i wybraniu przycisku „Call" nastąpi dynamiczne wywołanie operacji na testowanym serwerze.

3.4. Prezentacja wyników operacji i ewentualnych wyjątków

Wyniki zwrócone przez operację i przygotowane do prezentacji graficznej są następnie wyświetlane w okienku (rys. 9), które również wykorzystuje edytor struktury *Any*. Okienko to pozwala na przeglądanie wyników wywołania operacji, tzn. wartości zwróconej oraz parametrów „inout" i „out", w sposób podobny jak przy inicjalizowaniu parametrów. Wartości zwróconych parametrów nie mogą być modyfikowane. W przypadku kiedy wykonanie operacji zakończy się w sposób niepoprawny zwróceniem wyjątku, pojawia się nie zaprezentowane tutaj okno, które wyświetla typ wyjątku oraz wykorzystując edytor struktury *Any* umożliwia dostęp do zawartości tego wyjątku.

3.5. Obsluga systuacji, gdy zwracany rezultat jest referencją obiektu

Jeżeli operacja zwróciła referencję, to użytkownik ma możliwość zapisania jej w notatniku referencji obiektów, a następnie może przeglądać interfejs tego obiektu. Następuje wtedy przejście do etapu wyboru operacji opisanego w punkcie 3.2. Użytkownik może więc wybrać dowolną z operacji dostępnych na danym serwerze, zainicjalizować jej parametry, wywołać ją i obejrzeć jej wyniki. Przykład takiej sytuacji pokazany jest na rys. 9.



Rys. 9. Okienko prezentacji rezultatów testowanej operacji
Fig. 9. Window presenting results of the operation

4. Implementacja narzędzia

Program napisany został w środowisku OrbixWeb 2.01 [5], który jest implementacją standardu CORBA dla języka Java 1.1. Implementacja ta pozwala na wykorzystanie zalet języka Java, z których podstawową jest możliwość uruchamiania programu jako applet w przeglądarkach internetowych. Do niedawna popularne przeglądarki internetowe (Netscape, Internet Explorer) nie były zgodne z najnowszą wersją języka Java, co uniemożliwiło testowanie w nich programu. Jednak obecna wersja Netscape 4.04 może już być skonfigurowana do pracy z wersją 1.1 języka Java, co pozwala na uruchamianie programu poprzez sieć. Do pracy programu wymagane są trzy aplikacje: *IR*, *Trader* i dodatkowy serwer umożliwiający operacje na plikach (czytanie konfiguracji, zapis i odczyt referencji serwerów dla notatnika). Serwer ten jest wymagany ze względu na ograniczenia dotyczące appletu, które nie pozwalają mu wykonywać operacji na systemie plików.

Do wywoływania operacji testowanego serwera w sposób dynamiczny, to znaczy wywoływania operacji nieznanych w czasie kompilacji programu, wykorzystywany jest *Dynamic Invocation Interface* (*DII*) dostarczany przez standard CORBA i zrealizowany w OrbixWeb. *DII* zawiera szereg interfejsów (*Request*, *NVList*, *NamedValue*), które umożliwiają dynamiczne zainicjalizowanie parametrów dowolnej operacji, wywołanie jej na testowanym serwerze i otrzymanie jej wyników (włączając ewentualne wyjątki).

Podczas tworzenia narzędzia napotkano na wiele trudności spowodowanych szeregiem niedokładności i błędów w implementacji standardu CORBA dostarczanej przez OrbixWeb. Wymagało to dodatkowego nakładu pracy w celu ominięcia tych niedokładności, między innymi konieczność heurystycznego wyszukiwania interfejsu obiektu w *IR* i niskopoziomową analizę intensywnie wykorzystywanych klas *TypeCode* i *Any*.

5. Podsumowanie

W artykule przedstawiono narzędzie umożliwiające testowanie funkcjonalności komponentów programów napisanych w środowiskach zgodnych ze standartem CORBA. Uniwersalność tego narzędzia uzyskano dzięki intensywnemu wykorzystaniu metawiedzy opisu programów dostępnej w środowisku CORBA. Narzędzie pełni funkcję ogólnego klienta serwerów CORBA dostosowywanego do konkretnego serwera na podstawie jego opisu zawartego w *Traderze* i *Interface Repository*. Kluczowym elementem programu jest edytor struktury *Any*, którego konstrukcja jest bardzo złożona. Wysoką użyteczność tego narzędzia testującego potwierdziło jego intensywne użycie w trakcie realizacji projektu *Active Badge next generation*.

LITERATURA

[1] The Common Object Request Broker Architecture and Specification, Revision 2.1. OMG, 1997, http://www.omg.org/corba/corbiiop.htm.

[2] Rohit, G.: CORBA: The Future of Web Computing. Sun Developer NEWS, 1996, nr 3, http://www.sun.com/developers/devnews/summer97/corba.html.

[3] Orfali R., Herkey D., Edwards J.: CORBA, Java and the Object Web. Byte 1997, nr 10, s. 95 - 100.

[4] CORBAservices: Common Object Services Specification, *Trader* Object Service. OMG, 1997, ftp://www.omg.org/pub/docs/formal/97-07-26.ps.

[5] OrbixWeb for Java, Iona Technologies, Dublin, 1997, http://www-usa.iona.com/Products/Orbix/OrbixWeb/index.html.

[6] Steinder M., Uszok A., Zieliński K.: Architektura współpracy rozproszonych systemów obiektowych zgodnych ze specyfikacją CORBA. ZN Pol. Śl. S. Informatyka z. 30, Gliwice 1996.

[7] Szymaszek J., Uszok A., Zieliński K.: Active Badges using CORBA. Spring'97 Workshop Open Signalling for ATM, Internet and Mobile Networks (OPENSIG), Cambridge, Anglia, 1997.

Recenzent: Dr inż. Katarzyna Stąpor

Wpłynęło do Redakcji 11 grudnia 1997 r.

Abstract

The CORBA standard is getting a widespread adoption as a middleware for a three-tier architecture of emerging sophisticated WWW applications. This article presents a tool enabling extensive testing of CORBA servers (fig. 1). It serves as a generic client customised for the given server using meta-information describing the server stored in Interface Repository (IR). In the tool the *Trader* can be used to select a reference of the tested server. The user is led by the following steps of this process: selection of the service name (fig. 2), specification of query parameters (fig. 3), definition of constrains for offer properties (fig. 4) and selection of server references from the returned ones (fig. 5). However, server reference can be as well specified by hand. Knowing the server, tool presents the user with description of its functionality retrieved from the IR (fig. 7). User can choose any method of server's interface. Now, forms are created dynamically enabling initialisation of all parameters. This is the most complex part of the programme and was called *Any editor*. When a method is called all initialised parameters are sent to the DII mechanism and through it to the server. Results or exceptions returned by the server are available for a user's inspection. If some of the results are object references they can be saved in the object reference bookmark (fig. 6). Servers denoted by these references can be as well tested using the tool. The presented testing programme has already proved its usefulness in the CORBA oriented Active Badge next generation project [7].