MODIMOS – SYSTEM MONITOROWANIA OBIEKTOWYCH APLIKACJI ROZPROSZONYCH

A. Laurentowski, J. Szymaszek, K. Zieliński

Katedra Informatyki AGH, Al. Mickiewicza 30, 30-059 Kraków

1. Wprowadzenie

Wśród środowisk programowania rozproszonego w sieciach komputerowych powstało w ostatnich latach wiele systemów reprezentujących różne modele przetwarzania. Można tu wymienić systemy STRAND, PVM, P4, ANSA, SR, Emerald, PCN, czy środowiska zgodne ze standardem CORBA. Nierealne jest zatem dążenie do uniformizacji środowisk programowych różnych organizacji czy pojedynczych użytkowników, których systemy informatyczne prędzej, czy później zaczną współpracować. Stąd wniosek, że istniejące oraz przyszłe systemy i aplikacje użytkowników są niejako „skazane" na współpracę. Uważamy, że przyszłe zaawansowane systemy informatyczne będą się składać ze współpracujących ze sobą modułów napisanych i uruchomionych w różnych językach i środowiskach. Prototypy takich systemów już powstały w ramach projektów badawczych [6, 7] i będziemy je tu dalej nazywać *systemami heterogenicznymi programowo*. Ich nieuniknioną właściwością jest duża złożoność, zwłaszcza na wstępnym etapie badań nad ich konstrukcją i zastosowaniem. Stąd wynika konieczność budowy odpowiednich systemów śledzenia i wizualizacji takich aplikacji. Niniejszy artykuł omawia prototypowy system tego typu, służący do monitorowania i wizualizacji heterogenicznych programowo, obiektowych aplikacji i środowisk rozproszonych. System ten, zwany MODIMOS (Managed Object-based Distributed Monitoring System), został zrealizowany w Katedrze Informatyki AGH.

Większość istniejących narzędzi monitorowania i wizualizacji nie nadaje się do środowisk obiektowych [14, 11]. Inne są najczęściej ograniczone do pojedynczego środowiska programowania [12] lub w ogóle nie działają w systemach równoległych i rozproszonych [13, 10]. Poza tym nie przewidują one współpracy komponentów wewnątrz systemu heterogenicznego programowo, a ich architektura ma często charakter zamknięty, tzn. nie są przystosowane do łatwej rozbudowy, poszerzania o nowe środowiska monitorowane. MODIMOS spełnia te wymagania. Jego użytkownik może obserwować strukturę obiektowej aplikacji rozproszonej, jej fizyczne odwzorowanie na komputery w sieci oraz zachowanie i dynamikę jej komponentów. Dzięki temu system może służyć do wykrywania nierównego rozdziału lub przeciążenia zasobów i ułatwiać ich wyważanie i dystrybucję. MODIMOS jest systemem otwartym, łatwym do rozbudowy, tzn. środowiska spełniające określone wymagania mogą być stosunkowo łatwo zaadaptowane do współpracy z systemem. W ten sposób MODIMOS może poszerzać zbiór różnych środowisk, których komponenty jest w stanie monitorować, a trzeba pamiętać, że środowisk tych jest już sporo i wciąż powstają nowe. Otwarta koncepcja systemu jest możliwa dzięki jego ustrukturalizowanej, wielowarstwowej architekturze. Definiuje ona podstawowe interfejsy oraz protokoły komunikacji, sprzęgające poszczególne warstwy ze sobą. Dotyczy to zarówno instrumentacji monitorowanego kodu, jak i mechanizmu przechwytywania i interpretacji zdarzeń, mechanizmów i technik współdziałania oraz stosowanych baz danych. Otwartość systemu jest możliwa również dzięki zastosowaniu najnowszych technik inżynierii oprogramowania, takich jak obiektowo zorientowane ramy [4] (ang. *frameworks*) i wzorce projektowe [5] (ang. *design patterns*).

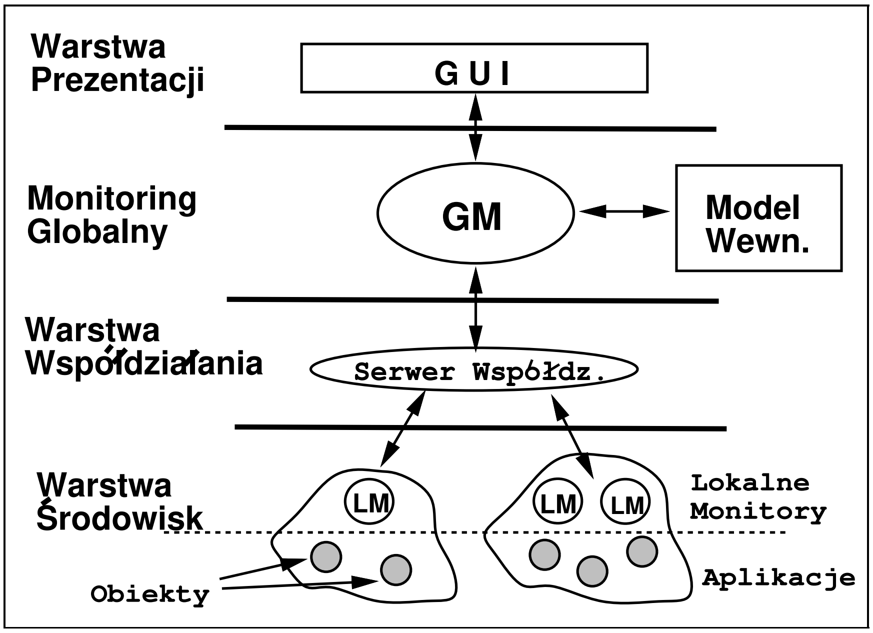
Kluczem do efektywnego śledzenia i wizualizacji dużych i złożonych systemów, jakimi niewątpliwie są rozproszone środowiska programowo heterogeniczne, jest nawigacja zorientowana na problem, umożliwiająca użytkownikowi takie zarządzanie strumieniem informacji, by mógł ograniczyć ją do minimum niezbędnego dla rozwiązania aktualnego problemu. Stąd nacisk położony w systemie MODIMOS na mechanizmy selekcji istotnej i filtrowania nieistotnej informacji. Mechanizmy te są (lub będą) zaimplementowane na kilku poziomach systemu. MODIMOS pracuje w dwu trybach: *off-line*, umożliwiającym analizę pliku ze śladem działania aplikacji, oraz *on-line*, kiedy system nasłuchuje zdarzeń z sieci i dzięki temu praca monitorowanej aplikacji analizowana jest na bieżąco.

Niniejszy artykuł jest podzielony następująco: w sekcji 2 opisana jest architektura systemu. Sekcja 3 omawia Uniwersalny Model Obliczeniowy (UMO), przyjęty dla potrzeb projektu. W sekcji 4 przybliżone zostały wybrane aspekty zastosowania systemu MODIMOS do monitorowania aplikacji zbudowanych zgodnie z modelem CORBA. W sekcji 5 omówiono wybrane aspekty implementacyjne wraz z przykładem zastosowania systemu. Całość artykułu kończą wnioski.

2. Architektura systemu

Jak już wspominano, MODIMOS umożliwia jednoczesną i jednolitą pod względem logicznym obserwację kilku środowisk programistycznych (lub aplikacji zbudowanej z komponentów działających w tych środowiskach). By to osiągnąć, zaprojektowano wielowarstwową architekturę pokazaną na rysunku 1.

Warstwę Środowisk tworzą odpowiednio zaadaptowane obiektowe środowiska programowania rozproszonego. Praktycznie każde środowisko tego typu może być zaadaptowane na potrzeby systemu MODIMOS pod warunkiem, że dostarcza podstawowe mechanizmy komunikacji ze światem zewnętrznym (np. gniazdka



Rysunek 1: Wielowarstwowa architektura systemu MODIMOS

– ang. *sockets*) oraz jego model obliczeniowy zawiera się w Uniwersalnym Modelu Obliczeniowym MODIMOS (patrz sekcja 3). Warstwa Środowisk składa się z dwóch poziomów: Poziomu Monitorowanych Aplikacji i Poziomu Lokalnych Monitorów. Poziom pierwszy tworzą obiekty aplikacji użytkownika, których kod źródłowy został zinstrumentowany przez odpowiednie preprocesory. Instrumentacja polega głównie na dodaniu tzw. *funkcji zawiadamiających*. Funkcje te zbierają informacje o zdarzeniach, które miały miejsce w śledzonej aplikacji i zawiadamiają o ich zajściu lokalne monitory. Monitory te to specjalne obiekty środowiska, w którym działają, mające dwa podstawowe zadania: filtrację strumienia zdarzeń wg przyjętych reguł oraz przesłanie wybranych zdarzeń przez Warstwę Współdziałania do Monitora Globalnego.

Środowiska mogą posiadać jeden lub więcej lokalnych monitorów, w zależności od swych właściwości i rozmiaru aplikacji. Ograniczanie strumienia danych wejściowych jest możliwe dzięki zastosowaniu odpowiednich reguł filtracji na obu poziomach Warstwy Środowisk. Zespół reguł tworzy *strategię* filtracji. Na Poziomie Monitorowanych Aplikacji użytkownik definiuje reguły za pomocą odpowiednich przełączników preprocesorów, pozwalających np. pominąć generację zdarzeń określonego typu lub zdarzeń dotyczących określonych obiektów. Z kolei na Poziomie Lokalnych Monitorów możliwe jest ustawianie podobnych reguł przy uruchamianiu danego monitora lub w trakcie pracy za pomocą specjalnego interfejsu zarządzania. Lokalne monitory mogą oczywiście spełniać wiele dodatkowych funkcji pomocniczych, jak kompresja danych czy translacja formatów kodowania, w zależności od potrzeb użytkownika i właściwości danego środowiska. W środowiskach usług zamkniętych (jak np. SR [2]) lokalne monitory są skonsolidowane z zasadniczą aplikacją, powstają i są niszczone wraz z nią. Z kolei w środowiskach otwartych, jak np. ANSA [1] czy CORBA [3], lokalne monitory są publicznie dostępnymi serwerami, uruchamianymi niezależnie na żądanie użytkownika.

Kolejna warstwa architektury systemu MODIMOS to Warstwa Współdziałania. Ma ona na celu zapewnienie uniwersalnej i ogólnej platformy współdziałania komponentów systemu heterogenicznego programowo, jakim jest sam MODIMOS. Warstwa ta, dzięki rozwiązaniom strukturalnym (patrz sekcja 4) i odpowiedniemu protokołowi komunikacyjnemu [8] umożliwia współpracę monitorów lokalnych różnych środowisk z Monitorem Globalnym (GM). GM przechwytuje zdarzenia nadchodzące z całego systemu monitorowanego, zbiera informację o śledzonych jednostkach i ich stanie w specjalnej bazie danych, zw. Modelem Wewnętrznym. Model ten odzwierciedla chwilową strukturę i stan tej części jednostek (obiektów) aplikacji, które wg działającej aktualnie strategii nie są pominięte (odfiltrowane). Monitor Globalny współpracuje ze znajdującym się w Warstwie Prezentacji Graficznym Interfejsem Użytkownika (GUI). Interfejs ten służy do graficznej reprezentacji struktury i stanu monitorowanej aplikacji oraz dostarcza użytkownikowi dodatkowych metod selekcji napływającej informacji.

3. Uniwersalny Model Obliczeniowy

System monitorowania danego środowiska programowego powinien być oparty na jego modelu obliczeniowym, tzn. śledzenie i wizualizacja powinny dotyczyć jednostek tego modelu. Tymczasem MODIMOS jest z założenia przeznaczony do zastosowania dla potrzeb wielu, często znacznie różniących się, obiektowych środowisk programowania rozproszonego. Różnice między nimi dotyczą również modeli obliczeniowych. Stąd wynikła potrzeba opracowania uniwersalnego modelu, na którym system MODIMOS mógłby być oparty. W tym celu przeanalizowano modele obliczeniowe najważniejszych dostępnych środowisk obiektowych i wyprowadzono wniosek, że ów uniwersalny, abstrakcyjny model powinien być nadzbiorem (unią) istniejących modeli. Takie podejście gwarantuje, że wszystkie możliwe w praktyce poziomy abstrakcji będą obecne w modelu uniwersalnym. Model ten nazwano Uniwersalnym Modelem Obliczeniowym (UMO), choć należy pamiętać, iż dotyczy on jedynie obiektowych środowisk rozproszonych.

Aplikacja w obiektowym środowisku programowym składa się najczęściej, na poziomie logicznym, z pewnej grupy niezależnych, komunikujących się ze sobą obiektów. Uniwersalny Model Obliczeniowy definiuje jednak więcej poziomów abstrakcji, a mianowicie: środowisko, aplikację, kontener, obiekt, interfejs, metodę i wywołanie. By dane środowisko mogło być monitorowane w systemie MODIMOS, jego model obliczeniowy powinien zawierać się w UMO. Poniżej przytoczono krótki opis poszczególnych jednostek abstrakcyjnych.

Środowisko. Przez środowisko jest rozumiany działający system wykonawczy (ang. *run-time system*), umożliwiający uruchomienie obiektów (zasobów) definiowanych przez użytkownika.

Aplikacja. Pojęcie aplikacji nie jest trudne do zdefiniowania w środowiskach zamkniętych, jak np. SR. Tworzą ją tam wszystkie obiekty i zasoby zdefiniowane przez użytkownika w danym programie, najczęściej wraz z dedykowaną instancją środowiska dla ich wykonania. Inaczej ma się sprawa ze środowiskami otwartymi, jak np. ANSA czy CORBA, gdzie działają ogólnie dostępne serwery usług, uruchamiane i wykorzystywane przez różnych użytkowników. Nie da się tam zakreślić granic między aplikacjami w sposób arbitralny czy automatyczny. Jedynie dany użytkownik jest w stanie zdefiniować zasoby, wchodzące w skład jego aplikacji.

Kontener. Kontenery to obszary pamięci dzielone przez obiekty w nich rezydujące. Zazwyczaj kontenery są implementowane przez procesy systemu operacyjnego.

Obiekt. Definicja obiektu jest tradycyjna i powszechnie znana. Przypomnimy tylko, że w niektórych środowiskach (np. ANSA) obiekt może mieć więcej niż jeden interfejs.

Interfejs. Interfejs definiuje metody i atrybuty, których obiekt dostarcza dla użytku innych obiektów.

Metoda. Metoda to operacja wchodząca w skład interfejsu obiektu. MODIMOS rozróżnia i pozwala śledzić osobno instancje metod każdego obiektu, czyli w istocie wątki wykonania związane z daną operacją.

Wywołanie. Wywołanie jest abstrakcyjnym pojęciem reprezentującym stan, w którym obiekt–klient wywołuje usługę (metodę) z interfejsu obiektu–serwera. Sumaryczna liczba wywołań powinna być równa liczbie instancji metod w aplikacji. Takie podejście umożliwia spójne i symetryczne śledzenie i wizualizację komunikacji w obiektowym systemie rozproszonym.

Z każdym elementem Uniwersalnego Modelu Obliczeniowego wiązna jest para komplementarnych zdarzeń: utworzenie i zniszczenie danego elementu. Zdarzenia te są raportowane przez funkcje zawiadamiające do lokalnych monitorów i stamtąd dalej do Monitora Globalnego. Stąd z każdą jednostką UMO

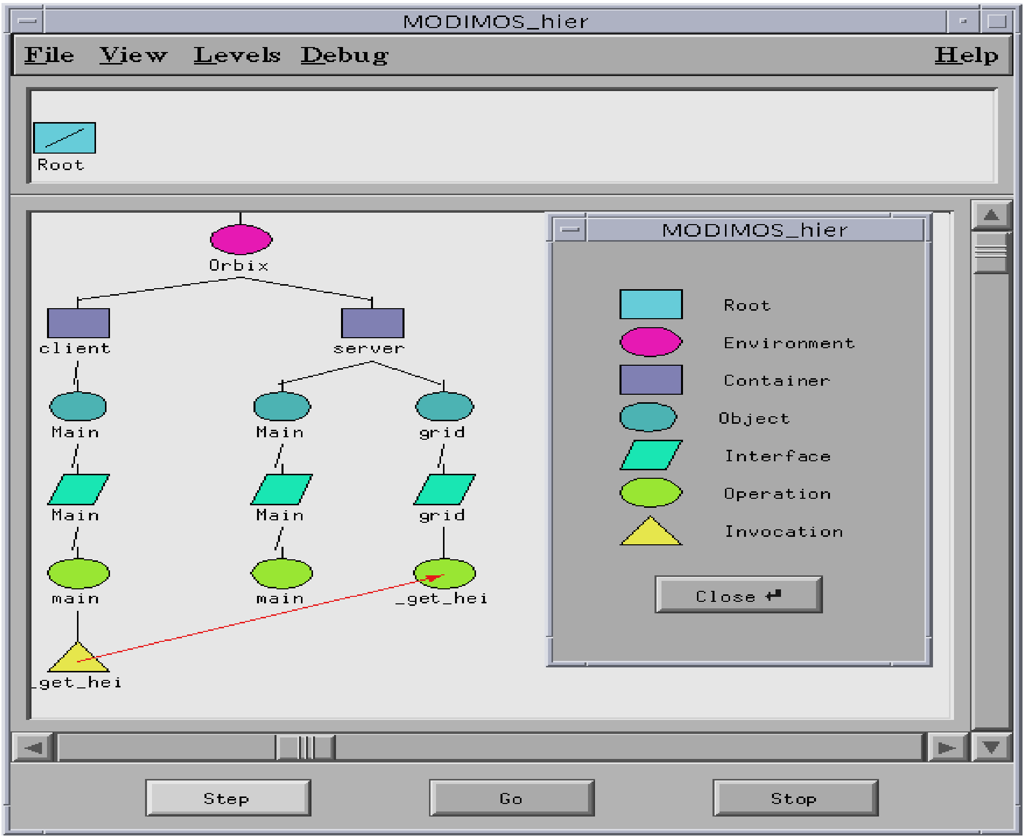
związane są dwie funkcje notyfikujące. Zbiór funkcji zawiadamiających jest w sposób uniwersalny opisany interfejsem monitorowania w języku IDL (*Interface Definition Language*) [3], co umożliwia łatwe przenoszenie do różnych języków i środowisk. Interfejs ten definiuje zarówno interfejsy monitorujące wszystkich lokalnych monitorów, jak i Monitora Globalnego.
Aby rozszerzyć system MODIMOS o nowe środowisko należy:

- przeanalizować jego model obliczeniowy i znaleźć jego odwzorowanie do UMO,
- dokonać translacji interfejsu monitorowania (funkcji zawiadamiających) na język programowania danego środowiska,
- napisać odpowiedni preprocesor instrumentujący kod źródłowy aplikacji w danym środowisku
- zaimplementować monitor(y) lokalny(e) na podstawie interfejsu monitorowania
- zaimplementować fragment podsystemu komunikacji w Warstwie Współdziałania

Osobny problem w systemach heterogenicznych programowo stanowi unikalna identyfikacja obiektów. Każde środowisko posiada własny, niekompatybilny z innymi, system identyfikacji. Tymczasem w aplikacji heterogenicznej programowo, a zwłaszcza w systemie monitorowania, istnieje potrzeba istnienia globalnie unikalnych identyfikatorów. Można rozwiązać ten problem, wykorzystując oryginalne identyfikatory z poszczególnych środowisk i trakując je jako ciągi bajtów, ew. uzupełnione krótkimi przedrostkami, nadającymi im globalną unikalność. Takie identyfikatory „naturalne" są jednak długie i stosowanie ich jako kluczy przeszukiwań w wielu typach baz danych powodowałoby nieefektywność operacji wyszukania i zapisu. Stąd w systemie MODIMOS zastosowano „sztuczne" identyfikatory o strukturze hierarchicznej, tworzone przez kod instrumentujący specjalnie na potrzeby systemu. Identyfikator hierarchiczny składa się z komponentów, z których każdy odpowiada pewnemu poziomowi logicznemu UMO i zawiera liczbę instancji elementów danego poziomu w elemencie nadrzędnym. Taki typ identyfikatora, kodowany binarnie lub znakowo, znacznie usprawnia operacje na niektórych typach baz danych, stosowanych w projekcie.

4. Monitorowanie aplikacji w środowisku typu CORBA

W niniejszym punkcie zostaną przedstawione rozważania dotyczące zastosowania systemu MODIMOS do monitorowania aplikacji pracujących w środowiskach przetwarzania zgodnych z modelem CORBA. Dla przykładu, na rysunku 2 przedstawiono główne okno systemu MODIMOS z otwartą „legendą”, stanowiącą opis ikon wizualizujących poszczególne poziomy UMO. Przedmiotem monitorowania jest bardzo prosta aplikacja, zw. `grid`, stanowiąca przykład załączany do dystrybucyjnej wersji środowiska Orbix. Prostokąty reprezentują kontenery (serwery), elipsy - obiekty, romby - interfejsy, kolejne elipsy - metody oraz trójkąty - wywołania. Na rysunku przedstawiony jest moment wywołania operacji `_get_height`. Sciezka w górnym okienku, zawierająca jeden element ("Root"), informuje, że okno przedstawia początek hierarchii monitorowanych jednostek.



Rysunek 2: Okno systemu MODIMOS wizualizujące prostą aplikację w środowisku Orbix.

Wymagania postawione systemowi MODIMOS w zakresie wizualizacji aplikacji wykonującej się w środowisku typu CORBA można w odniesieniu do tej grupy systemów przedstawić następująco:

Wizualizacja struktury. MODIMOS umożliwia wizualizacje modelu obliczeniowego wspólnego dla wszystkich systemów rozproszonych bazujących na podejściu obiektowym. W przypadku systemów typu CORBA obiekty posiadają, jak dotychczas, tylko jeden interfejs. Celowa wydaje się zatem eliminacja wizualizacji poziomu interfejsów tak aby uniknąć dodatkowych narzutów. W systemie MODIMOS jest to bardzo proste dzięki mechanizmowi horyzontalnej selekcji, który umożliwia usunięcie dowolnego z rozważanych w sekcji 3 poziomów abstrakcji.

Wizualizacja atrybutów. W systemie MODIMOS nazwa obiektu jest wyświetlana poniżej ikony natomiast pozostałe atrybuty mogą być wyświetlane na żądanie w osobnym okienku. Zawiera ono większość atrybutów, które mogą interesować użytkownika, takich jak: nazwa i definicja interfejsu, fizyczne umiejscowienie w sieci, aktualne parametry oraz rodzaj synchronizacji zdalnego wywołania operacji.

Reprezentacja dynamicznego zachowania. Przepływ sterowania pomiędzy różnymi przestrzeniami adresowymi, następujący w wyniku wywołania operacji z interfejsu zdalnego obiektu jest reprezentowany przez pojawienie się ikony wywołania operacji powiąznej z klientem, ikony aktywizacji kodu operacji w serwerze oraz połączenie obydwu elementów graficznych strzałką.

Konfigurowalność. MODIMOS posiada wiele parametrów umożliwiających dostosowanie jego działania do wymagań użytkownika. Dotyczy to w szczególności kolorów ikon, domen, grup, typów oraz stanów komponentów modelu UMO. Wszystkie te dane są mogą być ustawione za pomocą specjalnego narzędzia służącego do administrowania systemu zwanego **modadmin**. Program ten pozwala na generację odpowiednich plików konfiguracyjnych, zawierających specyfikacje wymagań użytkownika w postaci wyrażeń zapisanych w języku ESL (Entity Specification Language). Przykładowo, wyrażenie takie może specyfikować, że obiekt `"IIOP_Halfbridge"` powinien być zielony i pulsujący.

Zarządzanie danymi. MODIMOS posiada zaawansowane mechanizmy zarządzania danymi, które można podzielić na trzy podstawowe grupy:

- **Filtracja** na poziomie instrumentacji kodu, lokalnych monitorów, Globalnego Monitora (GM) oraz interfejsu graficznego (GUI). Przykładowo mechanizmy filtracji dostępne na poziomie GUI są następujące:
  - *Ignoruj Nowe Zdarzenia*: mechanizm ten pozwala na „zamrożenie” stanu okna. Aktualny stan jednak zostać odtworzony automatycznie na podstawie stanu Modelu Wewnętrznego.
  - *Ignoruj Nowe Obiekty*: mechanizm ten umożliwia ograniczenie obserwacji wyłącznie do śledzenia aktualnie wizualizowanych jednostek.

- **Selekcja** już wizualizowanej informacji wg preferencji użytkownika. Mechanizm ten pozwala na usunięcie z okna wszystkich obiektów, które nie są interesujące dla użytkownika i zawężenie w ten sposób liczby obserwowanych zdarzeń. Szczególnym przypadkiem selekcji jest selekcja horyzontalna, polegająca na wyborze do monitorowania całych poziomów UMO.
- **Wyszukiwanie** specyficznych danych wśród zebranej informacji jest realizowane przy użyciu specjalnego modułu wyszukującego, którego działanie jest sterowane przy użyciu wyrażeń zapisanych w języku ESL. Umożliwia to precyzyjne określenie atrybutów poszukiwanych zdarzeń i obiektów. Przykładowo, wyrażenie `"name = "*bank*" & level = container"` opisuje wszystkie serwery, których nazwa zawiera łańcuch „bank".

5. Wybrane aspekty implementacji systemu

Implementacja systemu MODIMOS składa się ogólnie z wielu obiektów CORBA. Centralny element architektury stanowi obiekt Globalnego Monitora zintegrowany z GUI, stanowiący osobny serwer. Lokalne Monitory, pracujące w monitorowanych środowiskach typu CORBA, są klientami Globalnego Monitora i raportują mu zaistniałe zdarzenia. Lokalne monitory są serwerami dla zinstrumentowanej aplikacji użytkownika, która raportuje zdarzenia do nich. Poszczególne składowe systemu MODIMOS mogą być zaimplementowane w dowolnym środowisku zgodnym z CORBA.

5.1 Instrumentacja i Lokalne Monitory

Przyjęto, że technika instrumentacji kodu aplikacji powinna być prosta i nadawać się zarówno do instrumentacji ręcznej jak i automatycznej. Założono także, że proces instrumentacji nie powinien zmieniać już istniejącego kodu. Uwzględnienie tych założeń doprowadziło do powstania koncepcji obiektowej instrumentacji opartej na pojęciu sensora jednostki [15].

Każda monitorowana jednostka (środowisko, proces, obiekt itd.) jest obserwowana przez specjalny obiekt zwany sensorem jednostki. Obiekt ten jest odpowiedzialny za raportowanie do LM zdarzeń dotyczących danej jednostki, powstaje i jest niszczony wraz z nią. W środowiskach, w których język C++ jest językiem implementacyjnym również sensory są reprezentowane przez obiekty C++. Przestawiona koncepcja instrumentacji została użyta w trzech środowiskach: Orbix 1.3, Orbix 2.0 oraz Orbeline. Instrumentacja aplikacji w środowisku Orbix, wykorzystująca mechanizm filtrów, nie wymagała żadnych zmian w kodzie użytkownika. W środowisku Orbeline zmiany te były potrzebne jedynie w niewielkim stopniu. Instrumentacja dotyczyła plików C++, generowanych przez kompilator IDL, i to jedynie w pewnych określonych miejscach. Zasadniczy kod sensorów zawarto w postaci skompilowanej w odpowiedniej bibliotece dla każdego ze środowisk. Dodatkowo, dla Orbixa zbudowano specjalny preprocesor pozwalający na automatyczną instrumentację.

Lokalne Monitory w obydwu rozważanych środowiskach są obiektami uaktywnionymi jako osobne serwery. Ich liczba jest zależna od wielkości aplikacji i może być dowolnie zmieniana.

5.2 Warstwa współdziałania

Warstwa współdziałania systemu MODIMOS została skonstruowana zgodnie modelem ORBa rdzeniowego, przy czym GM oraz GUI są obiektami tego ORBa. Zapewnia to skalowalność systemu oraz otwartość na dalsze rozszerzenia.

Budowa warstwy współdziałania została przedstawiona na rys. 3. Jako ORB rdzeniowy wybrano dowolny ORB zgodny ze specyfikacją CORBA 2.0, tj. pracujący w protokole IIOP. Lokalne monitory zaimplementowane w tym środowisku mogą zatem komunikować się z GM bez żadnych trudności. W przypadku, gdy monitorowane środowisko nie jest zgodne z CORBA 2.0 konieczne jest użycie półmostów. Konstrukcja półmostów systemu MODIMOS została opisana szczegółowo w [16]. Ich zadanie polega na translacji protokołu danego ORBa oraz reprezentacji referencji obiektów danego ORBa do postaci obowiązującej w domenie IIOP.

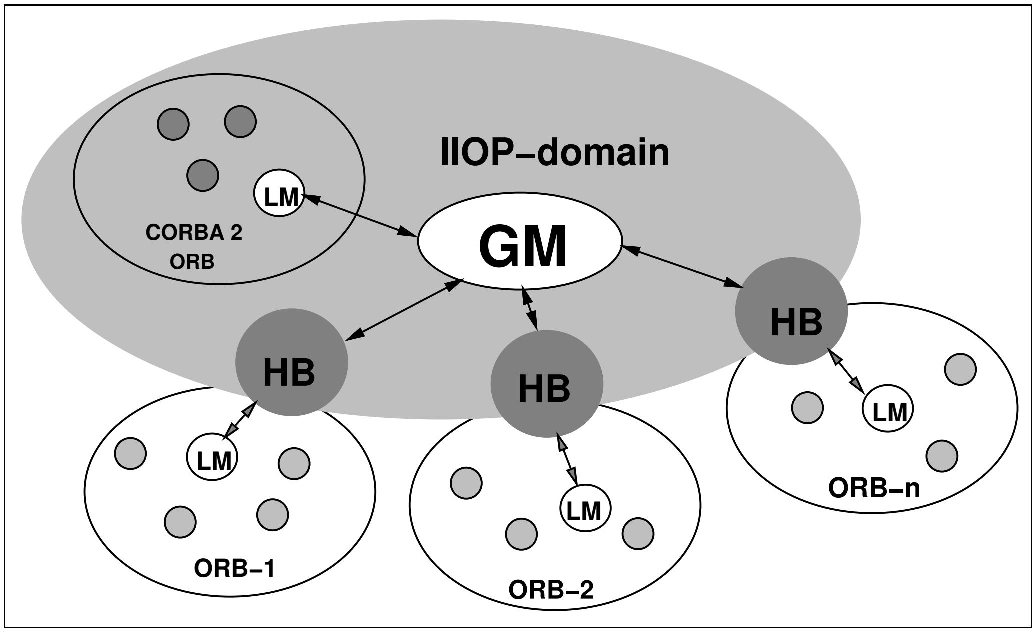
5.3 Globalny Monitor oraz GUI

Globalny Monitor został zbudowany w środowisku CORBA 2.0. Jego najważniejszą składową stanowi Model Wewnętrzny (MW) – baza danych zbudowana w oparciu o drzewo odwzorowujące strukturę monitorowanej aplikacji. Każde obserwowane zdarzenie dokonuje odpowiedniej zmiany MW, polegającej na utworzeniu lub usunęciu węzła tego drzewa.

Zarządzanie dużą ilością zdarzeń odbieranych przez GM jest możliwe dzięki filtrom, których działanie bazuje na koncepcji języka ESL. Każdy filtr jest obiektem, który sprawdza, czy obserwowane zdarzenie jest zgodne z posiadaną specyfikacją, reprezentowaną w postaci wyrażenia w języku ESL. Filtry można łączyć kaskadowo, tworząc bardzo silne mechanizmy filtracji. Opracowano także specjalne filtry umożliwiające zbieranie danych statystycznych.

5.4 Przykład wykorzystania systemu MODIMOS

Opisane własności systemu MODIMOS sprawiają, iż może on być efektywnie wykorzystany do monitorowania złożonych, heterogenicznych programowo ap-



Rysunek 3: Warstwa współdziałania systemu MODIMOS

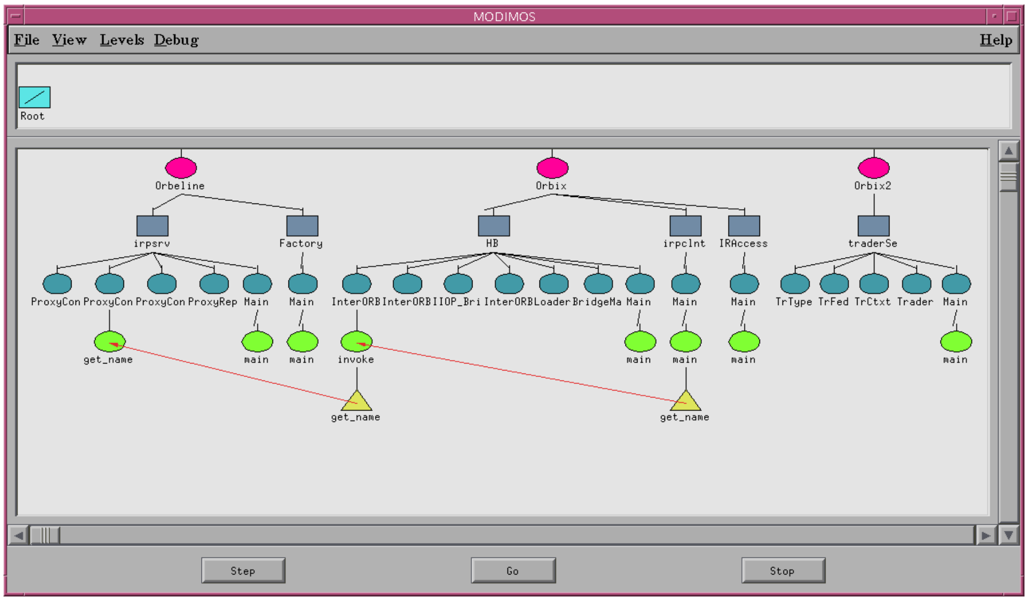
likacji działających w środowiskach zgodnych ze standardem CORBA. Jednym ze sposobów integracji heterogenicznych komponentów takich aplikacji jest użycie mostów i półmostów [16], co wprowadza dodatkowy stopień złożoności. W takich systemach użycie narzędzi do monitorowania i wizualizacji staje się wręcz niezbędne, zwłaszcza w fazie uruchamiania i testowania. Poniżej przedstawiono przykład zastosowania systemu MODIMOS do wizualizacji aplikacji wykorzystującej infrastrukturę półmostu, złożonej z klienta działającego w domenie Orbix 1.3 i serwera (reprezentującego Repozytorium Interfejsów) w środowisku Orbeline. MODIMOS umożliwia dokładne prześledzenie poszczególnych faz działania aplikacji, a w szczególności półmostu, jak m.in.:

- Dynamiczne wiązanie (ang. *binding*) pomiędzy klientem a serwerem w przypadku gdy te dwa procesy znajdują się w dwu różnych środowiskach. Operacja ta jest złożona i wymaga odwołania do usługi kojarzenia żądań klientów z ofertami serwerów (ang. *trading*).

- Tworzenie w moście reprezentantów (ang. *proxies*) dla obiektów serwera.

- Przeplatanie się operacji związane z wielowątkową implementacją półmostu.

Na rys. 4 przedstawiono okno systemu MODIMOS w momencie wizualizacji wywołania operacji `get_name` na obiekcie `ProxyContained` serwera `irpsrv`



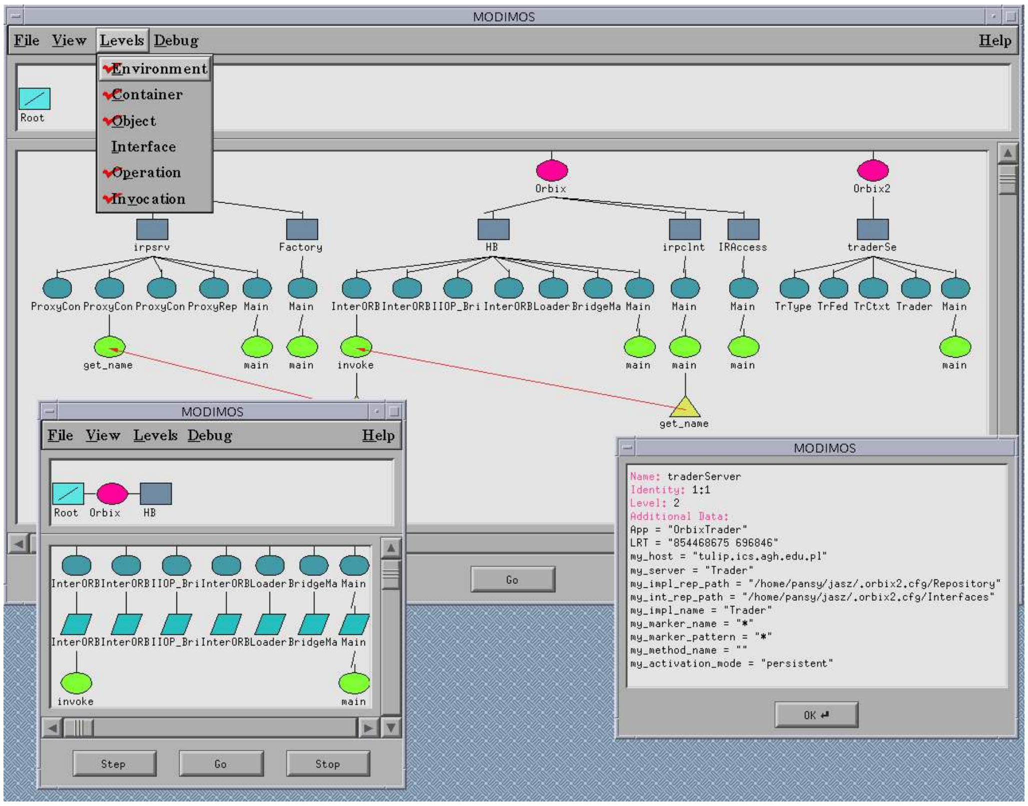
Rysunek 4: Wizualizacja aplikacji z widoczną infrastrukturą półmostu

inicjowanego przez funkcję `main` klienta. Obiekt `InterORBProxy` półmostu (serwera `HB`) pośredniczy w tym wywołaniu.

Rys. 5 przestawia z kolei uzycie dwóch mechanizmów dostarczanych przez system MODIMOS, przydatnych do wizualizacji złożonych aplikacji. Są to: nawigacja w strukturze monitorowanej aplikacji oraz selekcja horyzontalna. Nawigacja polega na możliwości zmiany widoku w oknie, tak że wybrany kursorem monitorowany element staje się korzeniem aktualnie wyświetlanego poddrzewa. Mechanizm ten może być zrealizowany w bieżącym lub nowo utworzonym oknie i właśnie ten drugi przypadek jest widoczny na rys. 5. Na rysunku tym aktywna jest również opcja selekcji horyzontalnej z usunięciem poziomu interfejsów. Widoczne jest również okno wizualizujące atrybuty serwera `traderServer`.

6. Wnioski

Potrzeba monitorowania aplikacji rozproszonych staje się coraz powszechniejsza wobec rosnącego stopnia ich złożoności. Jest to związane nie tylko z koniecznością zrozumienia zachodzących w tych systemach procesów, lub określenia przyczyn nieprawidłowego ich działania, lecz także zupełnie nowych zastosowań. Należą do nich zadania związane z rozliczeniami wykorzystania zasobów oraz opłat za usługi dostarczane przez system. Przewiduje się bowiem, że użytkownicy powinni być obciążani za korzystanie z określonych usług, a nie za użyte pasmo transmisji



Rysunek 5: Opcje: nawigacji w dodatkowym oknie, selekcji i wyświetlenia atrybutów monitorowanej jednostki

danych. System MODIMOS posiada wiele elementów pozwalających na wykorzystanie go w tego typu zastosowaniach. Jest skalowalny, posiada rozbudowane mechanizmy wizulizacji, selekcji i filtracji operujące na strumieniach danych.

Implementacja systemu MODIMOS w środowisku CORBA oraz wyposażenie go w warstwę współdziałania zapewnia, iż system ten jest w pełni dostosowanych do nowoczesnych hetrogenicznych, obiektowych aplikacji rozproszonych, które w niedługiej perspektywie powinny zdominować oprogramowanie użytkowe dostarczające usług w sieciach komputerowych.

Bibliografia

[1] ANSAware 4.0 – Application Programmer's Manual, APM Ltd. Cambridge, 1992.

[2] Andrews G.R., at el.: An overview of the SR language and implementation, *ACM Trans. on Progr. Languages and Systems*, 10(1):51–86, January 1988.

[3] Draft Common Object Request Broker Architecture Revision 1.1, OMG Report 91-12-1, OMG Inc., 1991.

[4] Campbell R., Islam N.: A Technique for Documenting the Framework of an Object-Oriented System, *Proc. IWOOS'92*, IEEE Computer Society Press, Los Alamitos, CA, September 1992.

[5] Gamma E., Helm R., Johnson R., Vlissides J.: *Design Patterns: Elements of Object-Oriented Software Architecture*, Addison-Wesley, 1994.

[6] Nierstrasz O., Gibbs S., and Tsichritzis D.: Component-Oriented Software Development, *Comm. of the ACM*, 35(9):160–165, September 1992.

[7] Udell J., Componentware, *Byte* vol. 19 no.5, pp. 46-56, May 1994.

[8] Czajkowski G., Uszok A., Zieliński K.: Distributed Declarative Systems as Parts of Cooperating Software Environments, *Proc. of the ICLP'94 Post-Conference Workshop on Integration of Declarative Paradigms*, Santa Margerita, Italy, June 1994.

[9] The Orbix Architecture, IONA Technologies Ltd., 1993.

[10] Jerding D.F. and Stasko J.T.: Using Visualization to Foster Object-Oriented Program Understanding, Georgia Institute of Technology, TR GIT-GVU-94-33.

[11] Reed D.A., Aydt R.A., Noe R.J., Roth P.C., Shields K.A., Schwartz B., Tavera L.F.: Scalable Performance Analysis: The Pablo Performance Analysis Environment, In *Proc. of the Scalable Parallel Libraries Conference*, IEEE Computer Society Press, 1993.

[12] Jamrozik H., Roisin C., Santana M.: A Graphical Debugger for Object-Oriented Distributed Programs, *Proc. TOOLS'91*, pp. 117-128, July 1991.

[13] Pauw W.De, Helm R., Kimelman D. Vlissides J.: Visualizing the Behavior of Object-Oriented Systems, *Proc. OOPSLA '93*, ACM Press pp. 326-337, SIGPLAN, Washington D.C, October 1993.

[14] Miller B.P., Clark M., Hollingsworth J., Kierstead S., Lim S., Torzewski T.: IPS-2: The Second Generation of a Parallel Program Measurement System, *IEEE Transactions on Parallel and Distributed Systems* 1, 2 (April 1990), pp. 206-217.

[15] Szymaszek J., Laurentowski A., Zieliński K., Instrumentation of CORBA-compliant Applications for Monitoring Purposes, Proceedings 2nd European Research Seminar on Advances in Distributed Systems, Zinnal, Szwajcaria, marzec 1997 (w druku).

[16] Steinder M., Uszok A., Zieliński K., A Framework for Inter-ORB Request Level Bridge Construction. ,,Distributed Platforms" - Proceedings of the ICDP'96 - IFIP/IEEE International Conference on Distributed Platforms, Drezno, Niemcy, Chapman & Hall, luty 1996, str. 86-99,