STUDIA INFORMATICA 2003
Volume 24 Number 3 (55)

Krzysztof KADUK, Franciszek GRABOWSKI
Politechnika Rzeszowska, Zakład Systemów Rozproszonych

ZASTOSOWANIE TECHNOLOGII INTERNETOWYCH AGENTÓW W PRZETWARZANIU KRATOWYM

Streszczenie. Opracowanie to jest teoretycznym rozważaniem praktycznego zastosowania technologii Systemów Wieloagentowych (Multi Agent System) do aplikacji w środowisku systemów otwartych. Podejmujemy próbę eliminacji złożoności systemu poprzez zastosowanie sprawdzonych metod adaptacyjnych – ekonomii wolnego rynku, wyrażanej w cechach socjalnych autonomicznych komponentów oprogramowania.

Słowa kluczowe: systemy rozproszone, internetowy agent.

MULTI AGENT SYSTEM TECHNOLOGY APPLIED TO GRID COMPUTING

Summary. This paper is theoretical investigation of practical possibility to use Multi Agent System technology in grid computing. We try to eliminate complexity by use so called Computational Economy realised by agent's social capability.

Keywords: distributed systems, multi agent.

1. Wprowadzenie

Kratownica obliczeniowa (computational grid) to system rozproszony oparty na sieci rozległej (Internet) dedykowany do współdzielenia na dużą skalę zasobów informatycznych i nakierowany na wysoką wydajność obliczeniową. Środowisko to różni się od systemu klastrowego tym, że jest systemem otwartym, składającym się z autonomicznych zasobów, udostępnianych na dowolnych zasadach określanych przez dostawców serwisów. W ostatnim okresie powstało szereg niekomercyjnych projektów, takich jak np. SETI@Home, Condor,

DAS, eGrid.[1] Również komercyjne przedsięwzięcia (Entropia, ProcessTree, Parabon) korzystają z możliwości zagospodarowania dostępnych a niewykorzystanych zasobów sieci (np. cykle procesora w biurowych komputerach PC). Oczywisty staje się fakt, że do szerszego, powszechnego i otwartego zastosowania tej koncepcji niezbędne jest ustalenie zasad umożliwiających osiągnięcie korzyści zarówno dla dostawcy, jak i dla konsumenta usług. Realizacja takiego systemu stawia szereg wyzwań związanych ze skalowalnością, otwartością, bezpieczeństwem i optymalizacją systemu.

2. Czym są Internetowi agenci (Multi - Agent Systems)

Przyjąć można, że istnieje zgoda co do definicji, czym są internetowi agenci, choć badacze i praktycy, w zależności od tego, z jakiej dziedziny naukowej się wywodzą, podkreślają, uwypuklają różne cechy. Badania nad internetowymi agentami czerpią z takich obszarów nauki, jak: sztuczna inteligencja, robotyka, systemy rozproszone, ekonomia, nauki społeczne. W sensie technologicznym jest to dalszy ciąg ewolucji metodologii tworzenia oprogramowania, stanowiący wyższy poziom abstrakcji, bardziej pasujący do opisu dziedzin modelowanego świata rzeczywistego. Podkreślić należy jednak, że nie jest to narzędzie uniwersalne, lecz stosowane do rozwiązywania pewnej klasy problemów, gdzie trudno wykorzystać metody tradycyjne. Na nasz użytek zdefiniujemy agenta oprogramowania (ang. *software agent*) jako komponent oprogramowania charakteryzujący się następującymi cechami:

Autonomia. Komponent posiada wewnętrzny stan i możliwość wyboru podejmowanych działań bez zewnętrznej interwencji użytkownika lub innych elementów systemu. Nie jest zobowiązany do bezwarunkowego wykonania żądań nadchodzących od innych elementów systemu.

Współpraca. Komponent posiada możliwość wymiany informacji (np. zadawania pytań i otrzymywania odpowiedzi) od innych komponentów na pewnym wysokim poziomie abstrakcji, opisującym środowisko (*świat*), w którym funkcjonuje.

Mobilność. Komponent może przemieszczać się pomiędzy jednostkami obliczeniowymi sieci lub klonować się. W tym celu posiada również *świadomość* położenia w przestrzeni sieci.

Proaktywność. Samodzielnie, bez zewnętrznych stymulacji podejmuje działania zmierzające do realizacji zadanego celu.

[1] Jednym z najbardziej popularnych zastosowań przetwarzania rozproszonego jest projekt SETI, którego celem jest zaangażowanie tysięcy komputerów w sieci Internet do znalezienia śladów cywilizacji w sygnałach nadchodzących z Kosmosu.

Reaktywność. Komponent odpowiada na *bodźce* napływające ze środowiska i podejmuje zgodne ze swoim celem działania.

Celowość. Komponent realizuje samodzielnie wyznaczony cel poprzez podejmowanie działań.

Z przedstawionych cech wynikają inne właściwości, np. takie jak *adaptacyjność* czy też *komunikatywność* komponentu. Chyba najbardziej znaną cechą komputerowego agenta jest przepis na działanie sprowadzający się do: 1) ustalenia wiedzy o środowisku, 2) określenia planu działań, 3) przeprowadzenia określonych akcji (ang. *Belive, Desire, Intention*).

Systemy oparte na MAS (Multi Agent System) możemy rozpatrywać z punktu widzenia pojedynczego agenta (ang. *intra-agent viewpoint*) lub jako strukturę całości systemu (ang. *inter-agent viewpoint*). Ten drugi punkt widzenia ujawnia szczególne cechy tej technologii w zastosowaniach praktycznych.

W projektach takich jak *grid computing* wydaje się korzystne podejście oparte na komputerowych agentach.

3. Opis problemu

Rozwój globalnej sieci Internet przyczynił się do powstania szeregu technologii, np. takich jak: integracja przedsiębiorstw [2] (*enterprise integration*), model ASP dostawcy usług internetowych (*application service provider, storage service provider*) oraz coraz szerszego rozpowszechnienia sieci *peer-to-peer*. Nowe modele biznesowe, możliwe dzięki rozwijanym technologiom, są jednocześnie katalizatorami nowych rozwiązań. To dzięki temu obserwujemy w ostatnim okresie tak imponujący rozkwit zastosowań i badań nad oprogramowaniem wspomagającym komunikację i współpracę. Z drugiej strony szybki rozwój techniki komunikacyjnej (np. technologie bezprzewodowe) umożliwia podłączenie do sieci Internet praktycznie wszystkiego, co stanowi skład technicznego otoczenia współczesnego człowieka. Trudno przecenić skutki społeczne i biznesowe tego stanu rzeczy. Wprowadzane technologie rozwiązują pewną klasę problemów, stwarzając jednocześnie inne wyzwania (np. przeładowanie informacją, trudności koordynacyjne, bariery przepustowości sieci itp.). Prowadzonych jest wiele badań, wdrażanych wiele rozwiązań praktycznych, których celem jest zmierzenie się z nowymi wyzwaniami. Niektóre z nich to np.:

Otwartość polegająca na tym, że do obszaru aplikacji mogą w dowolnym momencie zostać dołączane lub odłączane kolejne zasoby w sposób nieprzewidziany w momencie projektowania aplikacji.

---

[2] Integracja zachodzi najczęściej w ramach jednego „łańcucha dostaw", przyczyniając się do lepszego wykorzystania wspólnych zasobów (np. obniżenie stanów zapasów).

Skalowalność umożliwiająca praktycznie nieograniczony wzrost komponentów systemu (zasobów) bez znaczących kosztów organizacyjnych podczas realizacji zadań.

Rozliczenie wzajemne poszczególnych komponentów z użycia zasobów.

Optymalność działania objawiająca się najkorzystniejszym zachowaniem całości systemu.

Bezpieczeństwo polegające na zapewnieniu wzajemnej weryfikacji intencji komponentów oraz ich identyfikacji i autoryzacji.

Wszystkie te cechy są niezbędnym składnikiem istnienia struktur sieciowych biznesu. Przenosząc do sieci coraz większą część procesów biznesowych, potrzebujemy nowego modelu, opartego na wyższym poziomie abstrakcji niż protokoły, gniazda, porty itp. Elementy te muszą mieć podobną strukturę semantyczną do obiektów biznesowych istniejących w świecie rzeczywistym[3]. Gospodarka staje się bardziej uzależniona od informacji, widzimy wyraźny dryft w kierunku większego udziału usług w globalnym obrocie. Organizacje gospodarcze stają się coraz bardziej wirtualne, a wraz z tym zachodzi konieczność przenoszenia do sieci większej ilości instytucji współczesnej ekonomii (np. banki, usługi prawne, medyczne itp.). Niezbędne będą narzędzia pozwalające opisywać lepiej model organizacyjny na bardzo wysokim poziomie abstrakcji.

Grid Computing traktować można jako pewien ogólniejszy model dzielenia zasobów i współpracy globalnej. Idee te są szczególnie aktualne obecnie.

4. Stan dzisiejszy – modele koordynacji wykorzystania zasobów i integracji aplikacji

Przetwarzanie rozproszone i integracja aplikacji w systemach otwartych realizowane są przez szereg ogólnie akceptowanych i cieszących się coraz większą popularnością standardów. Usługi Webowe[4] (*Web Services*) [29], których głównym celem jest umożliwienie niezależnym podmiotom biznesowym nawiązanie kontaktów biznesowych poprzez odkrywanie i korzystanie z publikowanych usług. Technika ta wykorzystuje model komponentowy oprogramowania i umożliwia również integrację aplikacji. Poprzez standaryzację szeregu interfejsów i protokołów (UDDI - *Universal Description, Discovery and Integration*, WSDL - *Web Services Description Language*, SOAP - *Simple Object Access Protocol*, ebXML - *Electronic Business XML*) pozwala na odnajdywanie szukanych usług (serwisów), odczytanie sposobu ich wykorzystania (poziom syntaktyczny) i w fazie

[3]Metodologia jest tym lepsza, im jej model abstrakcyjny jest bardziej zbliżony do modelowanych procesów rzeczywistych

[4]Wcześniej taką próbę podjął Hewllet Packard, wprowadzając technologię e-speek.

końcowej wywołanie usługi. Choć umożliwia ona wzajemne wykorzystanie zasobów, nie zajmuje się problemami koordynacji na poziomie globalnym. Nie stanowi to problemu, gdy usługą webową jest proste zapytanie np. o cenę produktu lub uzyskanie innych informacji. Co natomiast, gdy usługą webową jest udostępnienie np. pewnych ograniczonych zasobów obliczeniowych, o które współzawodniczą procesy kilku niezależnych klientów ?

Problemy wzajemnej koordynacji i współdzielenia zasobów podejmuje szereg projektów. Jedne z nich koncentrują się na udostępnianiu cykli procesora [30], inne na wymianie zasobów plików [33], niektóre z nich starają się rozwiązać zagadnienie kompleksowo poprzez udostępnienie odpowiednich protokołów zapewniających otwartość, skalowalność i bezpieczeństwo całości systemu.

Najpopularniejszym modelem takiego stylu przetwarzania jest *Grid computing*[5]. Jednym z bardziej znanych jest projekt *Globus* [27]. Specyfikacja [19] oparta jest na technologii *Web Services* poprzez odpowiednie jej rozszerzenie (zgodnie ze standardem WSDL - *extensibility elements*) o zagadnienia niezbędne do realizacji projektu [18]. Całość koncepcji (*framework*) rozwiązuje problem otwartości systemu poprzez wykorzystanie podstawowej cechy Usług Webowych - możliwości odnajdywania usług (komponentów) i *odkrywania* interfejsów (*UDI*). Wykorzystując specyfikację WSDL 1.2, która definiuje element *serviceTyp* dla agregacji typu portu (*portType*) wprowadza (ograniczoną) możliwość dziedziczenia typu serwisu. Wywołanie usług na zdalnych komponentach, nazywane ogólnie zakontraktowaniem (*binding*), realizowane jest poprzez mechanizmy Usług Webowych w sposób asynchroniczny (*document centric*) lub synchroniczny (*RPC*). Dokładniejszy opis tej technologii wychodzi poza ramy opracowania, a jej specyfikację jak i opis architektury można znaleźć w opracowaniach [19, 35].

Większość prac koncentruje się na rozwijaniu protokołów, które definiują serwisy sieciowe (protokół + zachowanie(interfejsy) = serwisy sieciowe), wykorzystując poziom abstrakcji proceduralnych języków oprogramowania. Protokoły definiują wzajemne relacje pomiędzy infrastrukturą niezbędną do utrzymania całości systemu, jak również sposoby wzajemnej koordynacji wykorzystania zasobów. Koncentrują się jednak na zależnościach bilateralnych pomiędzy komponentami pozostawiając otwarte kwestie związane z optymalnością globalną systemu.

Najbardziej odpowiednim rozwiązaniem wydaje się zastosowanie technologii systemów wieloagentowych (*Multi Agent System - MAS*), choć podobnie jak w każdej rozwijającej się metodologii powstaje szereg pułapek związanych z przewartościowaniem lub niewłaściwą implementacją [24].

[5]Nazwa pochodzi od analogii do sieci elektrycznej, gdzie jej zasoby (energia elektryczna) mogą być dowolnie konsumowane, niezależnie od miejsca fizycznego podłączenia do sieci. System zapewnia jednak dokładne rozliczenie ich zużycia.

Z publikowanych pracy i praktycznych zastosowań wyróżniają się dwa nurty związane ze sposobem podziału zadań pomiędzy samodzielnymi komponentami (agentami). Jeden z nich stara się przyporządkować poszczególne organizacyjne role do wyspecjalizowanych agentów, których zadaniem jest realizacja globalnej strategii systemu [9], inne pozostawiają to zagadnienie sprawdzonym mechanizmom adaptacyjnym, takim jak np. ekonomia wolnego rynku [6, 10, 11]. To drugie podejście wydaje się najbardziej odpowiednie nie tylko do rozwiązywania problemów współdzielenia zasobów, ale również jako pewna ogólna metoda.

Tradycyjne metodologie nie są odpowiednie do projektowania systemów opartych na systemach wieloagentowych (*MAS*), chociażby z powodu innego poziomu abstrakcji, jakim operują (całkowity brak możliwości wyrażenia modelu socjalnego i organizacyjnego agentów). Propozycje wywodzące się z dziedziny inżynierii wiedzy również podlegają podobnym ograniczeniom.

Jak na razie nieliczne, dedykowane do tego celu metodologie rzadko obejmują pełny cykl życia produktu lub posiadają inne ograniczenia, jak np. nie radzą sobie z systemami otwartymi (Gaia) [1] lub są dedykowane do specyficznego rodzaju zadań (Zeus) [22].

5. Teoretyczny opis systemu realizującego przetwarzanie kratowe w oparciu o samodzielne komponenty oprogramowania

Zadaniem systemu jest realizacja algorytmu obliczeniowego w dynamicznym środowisku rozproszonym, w sieci Internet. Proces (zadanie) realizowany jest za pomocą proceduralnego języka oprogramowania, operując na prostym modelu pojęciowym *zasobów*. Zasób obliczeniowy (cykle procesora, baza danych, specjalizowane urządzenia, serwisy poczty, itp.) reprezentowany jest przez autonomicznego agenta. Agent ten w pewnym przybliżeniu jest podobny obiektowi reprezentowanemu przez instancję klasy w obiektowym języku programowania. Każdy zasób realizuje swój cel najczęściej poprzez sprzedaż lub zakup usług. Usługą jest zakontraktowane działanie na rzecz kupującego, którego efektem jest dostarczenie określonej informacji lub wykonanie pewnego działania. Mogą zatem występować zasoby, które nie udostępniają (sprzedają) żadnych usług, lecz realizują tylko swój cel, jak również zasoby, które zarówno sprzedają, jak i kupują usługi. Zakłada się, że celem wszystkich agentów jest racjonalna maksymalizacja zysku na operacjach handlowych. Jak zatem zabezpieczyć się przed agentami, które zaczną realizować całkiem inny cel, np. kontraktując jednocześnie dużą liczbę usług po zniżonej cenie i następnie nie wywiązują się z przyjętych zobowiązań? Działanie takie mogłoby doprowadzić do nieoptymalnego zachowania systemu lub innych poważnych zakłóceń, które łatwo w tym przypadku przewidzieć. Musi istnieć zatem mechanizm nie dopuszczający do takich przypadków. Jest

nim *Reputacja* każdego zasobu, zdefiniowana jako średnia wszystkich ocen otrzymanych za realizację kontraktów od wszystkich agentów w pewnym otoczeniu (o czym w dalszej części). Reputacja przypisana jest do *właściciela* agenta (zasobu), ponieważ musi posiadać pewną trwałość rozciągającą się znacznie poza cykl życia agenta. W przeciwnym przpadku można by było sobie wyobrazić atak na środowisko poprzez generację dużej liczby szkodliwych agentów, które po zakontraktowaniu usługi byłyby zabijane.

Zasób reprezentowany przez agenta posiada zatem *Właściciela* oraz jest zmuszony do maksymalizacji *Reputacji*, aby móc uczestniczyć w wymianie usług, a dopiero potem może optymalizować swój cel. Zatem jeśli *Reputacja* może być miernikiem zgodności celu agenta z celem pożądanym przez daną społeczność (zysk na wolnym rynku), to posiadamy mechanizm gwarantujący optymalne działanie takiego systemu w środowisku otwartym.

Pozostaje do przeanalizowania problem skalowalności. W takim środowisku założyć należy, że dodanie nowego zasobu nie może zwiększyć więcej niż liniowo narzutów związanych z zarządzaniem środowiskiem. Ekonomia wolnego rynku udowodniła już w praktyce swoją efektywność oraz pokazała, jak bardzo optymalność systemu zależy od przepływu informacji. Wolny rynek idealnie rozwiązuje problem optymalizacji przydziału ograniczonych zasobów pod warunkiem, że strony biorące udział w transakcjach posiadają pełną informację o ofercie na całym rynku. Takie idealne sytuacje nie istnieją w praktyce, ponieważ pozyskanie informacji jest kosztowne (czasami nawet niemożliwe). Koszt bezpośrednich negocjacji pomiędzy agentami zasobów będzie wzrastać bardziej niż liniowo w przypadku zwiększania skali systemu. Tutaj też przychodzą z pomocą realne instytucje wolnego rynku. Jedną z nich jest giełda, która organizuje zawieranie transakcji dla grupy uczestników zainteresowanych pewnym typem towarów (giełda spożywcza, budowlana, pieniężna itp.). Jak zatem powołać do życia takie instytucje w abstrakcyjnym świecie agentów ? Instytucją zapewniającą możliwość ustalenia ceny za pomocą mechanizmu popytu i podaży jest *Giełda*.

Do realizacji systemu może zostać wybrana platforma zgodna ze specyfikacją FIPA[6]. Zgodnie z tym standardem musi istnieć wyspecjalizowany agent zwany *Directory Facilator*, umożliwiający rejestrowanie usług, jakie oferują inni agenci. Przyjmuje się zatem, że w katalogu tym zostaną zarejestrowane tylko zasoby odpowiedzialne za realizację *Giełdy*. Otrzymujemy zatem prosty mechanizm startu agenta (*bootstrap*), który po zarejestrowaniu się w dowolnym kontenerze platformy (mechanizmy standardowe FIPA) poszuka tylko najbliższej usługi *Giełdy* i już może realizować swoje zadanie.

Instytucje *Giełdy* muszą tworzyć jakąś strukturę umożliwiającą globalne oddziaływanie mechanizmu popytu i podaży. Może być to np. struktura hierarchiczna, maksymalizująca

[6] Organizacja, której celem jest standaryzacja protokołów wymiany komunikatów pomiędzy samodzielnymi komponentami oprogramowania - agentami . www.fipa.org

szansę na uczestnictwo w danej licytacji możliwie największej liczby zainteresowanych partnerów. Proces *odnajdywania* oraz *zakontraktowania* zasobu musi być na tyle prosty, aby narzut czasowy na jego realizację nie przewyższał kosztu wykonania samej usługi.

Zakreślony obraz systemu wymaga jeszcze doprecyzowania, czym jest realizowane zadanie i jak wyglądać ma to praktycznie. Podstawową cechą systemu Grid jest możliwość uruchamiania zadań, czyli realizacji algorytmu programu zapisanego w języku proceduralnym. Program ten korzysta z zasobów Grid i w wyniku jego wykonania (wykonywania) otrzymujemy konkretną wyrażoną w algorytmie korzyść (np. wynik obliczeń, poszukiwane dokumenty). Korzyść tę otrzymuje właściciel uruchamiający zadanie. Sposób dostarczenia tej korzyści musi być zawarty w samym algorytmie (np. przesłanie wyników pocztą, zapis transakcji do bazy danych). Aby umożliwić łatwe kodowanie takich zadań, musimy stworzyć pewien szkielet (*framework*), o czym w dalszej części opracowania.

Do realizacji obliczeń zostanie wykorzystana koncepcja agenta zwana *head agent*, który z jednej strony realizuje funkcje na wysokim poziomie abstrakcji (socjalne, organizacyjne), czyli głowa (*head*), a z drugiej jest nośnikiem kodu proceduralnego (*ciało*), który może być wykonywany przez innych agentów poprzez wywołanie jego funkcji (np. metod w języku obiektowym). W tym sensie jest to zachowanie reaktywne, ale poprzedzone fazą negocjacji i zawarcia kontraktu na wykonanie usługi (wywołanie zdalne danej metody na obiekcie). Wykonywana w wyniku tego funkcja sama może zarządzać również zewnętrznymi, zdalnymi zasobami. W takim przypadku zostanie to zgłoszone do *głowy* agenta, który zajmie się odszukaniem zasobu, wynegocjowaniem kontraktu, zleceniem wykonania, odebraniem wyniku i przekazaniem go do *ciała* (callback). W tym przypadku zadanie wykonuje jeden fizyczny agent (zasób). Można jednak założyć możliwość, podczas realizacji proceduralnego algorytmu w ciele agenta, żądania stworzenia nowego zasobu poprzez powołanie do życia innego agenta i załadowanie go odpowiednim kodem proceduralnym. Ten nowy agent będzie kontynuował równolegle dalszą część obliczeń (lub dublował je dla zwiększenia niezawodności zadania). Wyraźnie widać, że pewne zasoby będą stanowiły prywatną część zadania, a inne będą publiczne. Tutaj występuje analogia do obiektów ze stanem i bezstanowych lub do klas statycznych i tych tworzonych na stercie pamięci w językach obiektowych.

Ciało agenta to załadowany, wykonywalny kod programu, którego funkcje (metody) są uaktywniane przez samego agenta (głowę) na zlecenie innych agentów lub zgodnie z celem działania i na potrzeby samego agenta. Podczas wykonywania kodu ciała agenta nie ma on możliwości przerwania zleconej akcji (wszystkie czynności agenta w jednym wątku). Powstaje problem pomiaru zużytych przez agenta cykli procesora na wykonanie zadania obliczeniowego oraz wcześniejszych negocjacji kontraktu, które wymagają wiedzy agenta o przewidywanych obciążeniach procesora (przynajmniej szacunkowych wielkości). Jedną z

możliwości jest takie skonstruowanie interfejsów ciała agenta, aby mogły one być odpytywane o przewidywany poziom obciążania funkcji (minimum, średnie, maksymalne) wyrażone np. miernikiem cykli procesora (lub innym wspólnym wskaźnikiem dla wszystkich platform obliczeniowych).

6. Potencjalne korzyści zastosowania technologii samodzielnych komponentów oprogramowania

Olbrzymi narzut obliczeniowy związany z samą organizacją populacji systemów wieloagentowych musi w warunkach praktycznych zostać zrównoważony innymi korzyściami, aby technologia ta miała praktyczny sens zastosowania. Jakie to korzyści ?

Przede wszystkim rozwiązanie oparte na sprawdzonych regułach wolnego rynku zapewniają najlepszą skalowalność i optymalność przydziału zasobów w środowisku otwartym. Poprzez odpowiednią standaryzację interfejsów i pojęć (ontologia) opisujących zasoby możemy wprowadzić hierarchię specjalizacji (analogicznie do modelu dziedziczenia klas w programowaniu obiektowym)[7]. Tak więc zapytanie o parametry zasobu obliczeniowego będzie miało taką samą semantykę ogólną obowiązującą dla wszystkich kategorii typów, natomiast agent zainteresowany pewnymi szczególnymi, wyspecjalizowanymi funkcjami otrzyma również interesującą go odpowiedź.

Wprowadzenie pojęcia reputacji podmiotu (właściciela) agenta (zasobu) zmusza go do przestrzegania reguł środowiska, w ramach którego jest uruchamiany lub z którym oddziałuje. Poprzez odpowiedni protokół wymiany informacji o reputacji właścicieli agentów oraz wyceny jej wartości przez każdego agenta zapewniana jest dystrybucja *lokalna* tej informacji[8].

Mobilność kodu agenta może zapewnić również mobilność procesów (podobnie jak *Telescript*) poprzez odpowiednią serializację kodu i stanu programu (ciało agenta) i przemieszczenie się wraz z agentem do innego miejsca[9]. Agent może podczas swojego cyklu życia realizować różny kod wykonawczy (ciało) poprzez załadowanie na żądanie właściciela odpowiedniego kodu[10] z podanego adresu (podpisanego przez właściciela - sygnatura).

---

[7]W podobnym kierunku zdaje się zmierzać standaryzacja Usług Webowych - agregacja typów portów.

[8]Można sobie wyobrazić, że przed zawarciem transakcji, po zaakceptowaniu jej kryteriów rynkowych (cena), agent odpyta pewną ilość sąsiednich agentów o poziom reputacji kooperanta i dopiero wtedy podejmie decyzję o ewentualnym zakontraktowaniu usługi.

[9]Mechanizm serializacji klas w języku Java.

[10]Kod ten może być np. wyrażony w pewnym języku skryptowym, którego interpreter znany jest danej populacji agentów.

Dla *świata zewnętrznego* postrzeganie agenta (wnioskowanie o jego celach, przewidywanie zachowań) może zachodzić tylko poprzez *obserwację* jego zewnętrznego zachowania. Dlatego też algorytmy działania agenta (głowa) również mogą być dowolnie wymieniane (np. mechanizm *plugin*) podczas cyklu jego życia lub kolejnych instancji. Możemy nawet mówić o pewnej bibliotece zachowań (*bechavior*) dedykowanych do konkretnych środowisk. W przypadku konieczności oddziaływania (poszukiwania zasobów) w modelu aukcji angielskiej pobrany zostanie inny kod niż np. w przypadku, gdy agent napotka aukcję typu niemieckiego.

Postrzec można olbrzymi obszar powtórnego wykorzystania już raz napisanego kodu. Każdy zasób sieci wykorzystuje ten sam kod związany z organizacją środowiska - kod głowy (*head*) agenta.

Nie piszemy żadnego kodu (ponieważ nie ma fizycznych elementów odpowiedzialnych za ten aspekt) związanego z optymalnym działaniem całości systemu (przydziałem zasobów), który w alternatywnej technologii stanowić może niemały zakres prac i poważne obciążenie obliczeniowe sieci i hostów.

Wymuszamy poprzez mechanizm *Reputacji* prawidłowe, czyli zgodne z mechanizmem adaptacyjnym (np. kształtowanie ceny za pomocą mechanizmu popytu i podaży) zachowanie się elementów systemu. Cecha ta jest nieodzowna w środowiskach otwartych, a jej implementacja w technologii tradycyjnej może sprawić wiele kłopotów, szczególnie tych związanych ze skalowalnością systemu.

Wprowadzamy podział pracy i kompetencji pomiędzy twórcami systemu. Tak pewna grupa może zajmować się rozwijaniem ontologii i protokołów współdziałania (współzawodnictwa) komponentów, inna optymalizacją działania samego agenta (*plugin bechaviors*), jeszcze inna rozwojem aplikacji rezydujących w ciele (*body*) agenta.

Objętość kodu agenta związanego z zarządzaniem na wyższym poziomie abstrakcji powinna być możliwie mała, dlatego też istotny jest mechanizm wtyczek pozwalający na żądanie (zależnie od zmieniającego się dynamicznie środowiska agenta) załadować odpowiedni kod (zachowanie).

Można sobie wyobrazić, że mechanizm ten wraz z mechanizmem reputacji umożliwiałby rozpoznanie przez agenta reguł, jakimi kieruje się dana społeczność, do której ma zamiar dołączyć[11]

Zaproponowane rozwiązanie zapewnia łatwą integrację z istniejącymi technikami. Klasyczne usługi mogą być obudowane przez odpowiednich agentów pośredniczących, którzy w ich imieniu będą udostępniać zasoby.

[11]Być może potrzebny będzie tu bardziej rozbudowany mechanizm reputacji, taki który pozwalałby na rozróżnienie agentów szkodliwych (tych o złych zamiarach) od agentów *uczących* się zasad nowego dla nich środowiska.

Bardzo ważnym elementem tego typu systemu jest sprawny mechanizm rozliczenia kosztów świadczonych usług (płatności). Zapewnia on korzyści podmiotom świadczącym usługi, zatem zachęca do udostępniania zasobów komputerowych. Ta informacja jest niejako łącznikiem wirtualnego świata agentów ze światem rzeczywistym.

Może on również pełnić inną funkcję. Poprzez wyposażenie agenta w informację o przewidywanym koszcie swoich działań zawartych w ciele agenta (np. koszt wywołania metody klasy) oraz w pewną ograniczoną pulę możliwych do wydania wirtualnych pieniędzy (budżet) algorytm procesu działania agenta (głowa) może podejmować decyzję o wyborze z dostępnych zasobów tych, które najlepiej spełniają zadane ograniczenia budżetowe, zapewniając jednocześnie realizację zadania (np. wybór pomiędzy szybkością realizacji a kosztem obliczeń).

7. Możliwy schemat realizacji praktycznej przetwarzania rozproszonego opartego na cechach socjalnych samodzielnych komponentów oprogramowania

Realizacja praktyczna zakreślonego powyżej systemu informatycznego musi zostać podzielona na szereg faz, które realizować będzie pewien wybrany podzbiór proponowanych mechanizmów, jednocześnie na tyle elastyczne, aby można je rozszerzać o inne elementy (funkcje organizacyjne świata agentów).

- Sprawdzenie funkcjonowania mechanizmu *reputacji*, w środowisku narażonym na oddziaływanie agentów, nie przestrzegających przyjętych norm socjalnych.
- Stworzenie bibliotek (*plugin*) dla podstawowych zachowań agenta (uczestnictwo w giełdach, negocjacja jeden-do-jeden, rozpoznawanie zasad socjalnych środowiska, itp.).
- Wykonanie szkieletu (*framework*) pozwalającego na funkcjonowanie ciała agenta (odkrywanie interfejsów, wywołanie zakontraktowanej usługi, itp.).
- Wykonanie systemu rozliczeń zużycia zasobów i systemu płatności.

Możliwość wymiany informacji pomiędzy agentami aplikacji zapewnić może odpowiednio zaprojektowana *ontologia*. Jest ona niezależna od platformy agentów, jednak bardzo pomocne może okazać się narzędzie wspomagające proces projektowy oraz pozwalające na automatyczną generację kodu dla platformy docelowej. Takim narzędziem jest *Protege-2000* [36]. Jedno z rozszerzeń pozwala na automatyczną generację kodu Java dla platformy JADE.

7.1. Ontologia dziedziny

Ontologia jest konceptualizacją rzeczywistości, służy do zdefiniowania formalnej reprezentacji wiedzy z danej dziedziny[12]. Reprezentacja ta jest niezależna od wewnętrznie używanych przez agenta symboli i oparta na *logice predykatów pierwszego rzędu*. Niezwykle istotne dla możliwości funkcjonowania agenta w sztucznej społeczności jest posiadanie pewnego obrazu swego otoczenia. Rozstrzygnąć musimy zatem następujące kwestie:

- W jaki sposób agent gromadzi informacje o stanie otoczenia ?
- W jakiej reprezentacji wewnętrznej tę wiedzę przechowuje ?
- W jaki sposób wykorzystuje ją do realizacji swego celu ?

Założyć musimy, że obraz ten prawie nigdy nie jest kompletny, bo nie potrafimy zebrać w rozsądnym czasie i po rozsądnym koszcie wszystkich możliwych faktów. Przyjmujemy również jego *względną* prawdziwość, bo wynika ona z faktu, że system *wnioskowania* agenta jest o wiele szybszy od szybkości zmian w otoczeniu.

7.2. Protokoły wysokiego poziomu

Wymiana informacji pomiędzy agentami systemu, umożliwiająca realizacje cech socjalnych środowiska, to protokoły wysokiego poziomu[13].

Jednym z najważniejszych protokołów jest protokół pozwalający agentowi wynegocjowanie ceny kontraktu. Czynność ta jest wspomagana wymianą informacji dotyczącej poziomu *Reputacji* drugiej strony kontraktu (właściciela oferowanego zasobu). Po każdym wykonaniu zdalnej usługi (bez względu na wynik) jest ona oceniana w pewnej skali punktowej, która jest tym wyższa, im lepiej otrzymane parametry wykonania są zgodne z tymi zakontraktowanymi. Agent posiada listę właścicieli zasobów, w której pamięta przyporządkowany im poziom reputacji. Poziom ten może być uaktualniany poprzez odpytywanie sąsiednich agentów o ich *doświadczenia* z danym właścicielem. Podczas podejmowania decyzji o zakupie danej usługi w każdej iteracji licytacji stosunek ceny do korzyści jest modyfikowany współczynnikiem reputacji. Agent może nie być zainteresowany w ogóle transakcjami z właścicielami, których poziom reputacji jest poniżej pewnego minimum. Podstawowym mechanizmem wyceny transakcji jest giełda. Giełda jest zasobem jak każdy inny. Agent dostarczający serwisów giełdy musi komunikować się z otoczeniem zgodnie z ontologią oraz protokołami zdefiniowanymi globalnie dla tej usługi. Serwisy te

[12]Termin ten zapożyczony został z filozofii, gdzie oznacza systematyczny opis istnienia. Do tej pory stworzono wiele ontologii szczegółowych opisujących dane dziedziny zainteresowań, które mogą być wykorzystywane w takich projektach.

oceniane są według takich samych zasad, jak każdy kontrakt (element reputacji) oraz agent pobiera opłatę od każdej transakcji. Giełda dostarcza również katalogu oferowanych usług wraz ze średnią ceną transakcji (oraz inne informacje).

Założenie o małej złożoności i wysokiej efektywności systemu *wnioskowania* agenta wymusza zastosowanie automatu skończonego (ang. *Finite State Machine*) jako modelu jego zachowania. Wprowadzić możemy zatem pojęcie *roli*, jako pewnego zachowania zgodnego z pewnym protokołem akceptowanym przez członków wirtualnej społeczności. Cechy socjalne agenta wyrażane są za pomocą jednej lub więcej ról. W tym znaczeniu rola (wraz ze wspólną ontologią) jest odpowiednikiem protokołu w systemach tradycyjnych, jest łącznikiem pomiędzy wewnętrznym stanem agenta a otoczeniem. Wyróżniamy dwie role *Kupujący* oraz *Sprzedający* usługi związane z udostępnieniem określonego zasobu. Dzięki temu przenosimy się na wyższy poziom abstrakcji, realizując podobne funkcje jak protokóły, np. smtp, ftp, telnet itp.

7.3. Struktury organizacyjne - instytucje

Instytucją jest zasób sieci, który pełni funkcję usługową związaną z organizacją systemu. Implementowana jest w podobny sposób jak każda inna usługa, z tym że udostępniane usługi nie są przeznaczone dla kodu agenta (ciała), lecz dla jego funkcji socjalnych (głowy). Instytucją jest giełda. W środowisku mogą powstawać nowe instytucje, jednakże ich rola musi zostać wcześniej uzgodniona przez grupę agentów mających zamiar korzystać z jej usług. Tylko niektóre podmioty (właściciele agentów) mogą tworzyć instytucje. Poziom zaufania do instytucji jest zawsze maksymalny i nie podlega ocenie. Każda instytucja definiuje skończony zbiór ról, jakie mogą przyjmować jej członkowie oraz związany z tym zakres funkcji (odpowiedzialności) wyrażony pewnym ustalonym protokołem (wcześniej znanym, publicznym). Instytucja może ograniczać korzystanie z jej zasobów dla pewnej klasy agentów (np. od pewnego poziomu reputacji).

7.4. Protokoły niskiego poziomu

Proces zakontraktowania usługi pozwala na bezwarunkowe wywołanie funkcji (metody) z procesu roboczego agenta (ciała) w procesie roboczym (ciele) innego agenta (zasobu). Może być to dowolne wywołanie RPC, usługa webowa, komunikat asynchroniczny lub przetłumaczony przez głowy agentów dwustronny przepływ komunikatów. Na przykład jeśli kod roboczy agenta poszukuje zasobu bazy danych relacyjnej, to zgłasza to do procesu

[13] Protokoły wysokiego poziomu są odpowiedzialne za funkcjonowanie „społeczności agentów", natomiast niskiego poziomu to wywołanie zakontraktowanych usług pomiędzy komponentami.

socjalnego, który negocjuje warunki transakcji, a następnie przekazuje uzyskany od zdalnego zasobu adres połączenia wraz z autoryzacją i innymi parametrami umożliwiającymi uzyskanie połączenia.

7.5. Zasady tworzenia kodu ciała agenta

Kod wykonawczy agenta, czyli algorytm realizacji serwisu, może być realizowany w dowolnym języku proceduralnym. Powinny być jednak opracowane pewne zasady pozwalające na wywoływanie z treści takich procedur innych zasobów sieci. Wywołanie takie musi odbywać się za pomocą wyższego poziomu abstrakcji reprezentowanego przez głowę (*head*) agenta. Istotna jest również możliwość integracji istniejącego oprogramowania (*legacy*).

8. Podsumowanie

Realizacja projektów opartych na niezbyt dobrze poznanych i szeroko stosowanych w praktyce technologii napotyka wiele problemów, szczególnie natury metodologicznej.

Praktycy zaangażowani w przemysłowe tworzenie systemów informatycznych zauważają szereg barier dotychczasowych technik, które ujawniają się szczególnie w systemach otwartych (internet). Technologia oparta na systemach wieloagentowych (MAS) wydaje się być logiczną konsekwencją dotychczasowego wzrastającego poziomu abstrakcji w językach programowania.

Tak jak w każdych nowo powstałych w nauce (biznesie) trendach zachodzi początkowo zjawisko zbyt dużego przewartościowania zawartych idei, aby później osiągnąć okres stabilizacji i zastosowań na skalę komercyjną. Wydaje się, że obecnie jesteśmy na początku tego drugiego okresu i miejmy nadzieję, że technologia ta nie podzieli losów prac nad sztuczną inteligencją (AI).

Wątpliwości te może jednak rozstrzygnąć idące w ślad za teoretycznymi pracami naukowymi praktyczne ich zastosowanie. W tej dziedzinie szczególną rolę odgrywa praktyka, której wyniki przecierają drogę nowym ideom i teoretycznym modelom.

LITERATURA.

1. Zambonelli F., Jennings N., Omicini A., Wooldrige M.: Agent-Oriented Software Engineering for Internet Applications, (http://lia.deis.unibo.it/~ao/pubs/pdf/2001/coordbook-13.pdf), 2001.

2. Zambonelli F., Jennings N., Wooldrige M.: Organisational Abstractions for the Analysis and Design of Multi-Agent Systems, (http://sirio.dsi.unimo.it/Zambonelli/PDF/aose2000.pdf), 2000.
3. Kleijkers S., Wiesman F., Roos N.: A Mobile Multi-Agent System for Distributed Computing, (http://www.cs.unimaas.nl/~roos/publications.html), 2002.
4. Jennings N., Wooldridge M., Wiesman F., Roos N.: Agent – Oriented Software Engineering, (http://www.ecs.soton.ac.uk/~nrj/download-files/agt-handbook.pdf), 2000.
5. d'Inverno M., Kinny D., Luck M., Wooldridge M.: A Formal Specification of dMARS, (http://www.csc.liv.ac.uk/~mjw/pubs), 1998.
6. Buyya R., Abramson D., Giddy J.: An Economy Grid Architecture for Service-Oriented Grid Computing (http://www.ggf1.nl/abstracts/ACCT/ecogrid.pdf), 2001.
7. Overeinder B.J., Wijngaards N.J.E., van Steen M., Brazier F.M.T.: Multi-Agent Support for Internet-Scale Grid Management, (http://www.iids.org/publications/aisb02_aigrid.pdf), 2002.
8. Tveit A.: jfipa - An Architecture for Agent-based Grid Computing, (http://www.jfipa.org/amund/publications/2002/jfipaACC.pdf), 2001.
9. Montresor A., Meling H.: Ozalp Babaouglu, Messor: Load-Balancing throuth a Swarm of Autonomus Agents.
10. Wooldridge M.: Engineering the computational economy, (http://www.csc.liv.ac.uk/~mjw/pubs), 2000.
11. Shen W.L.Y. , Ghenniwa H., Wang C.: Adaptive Negotation for Agent based Grid Computing (http://www.agentcities.org/Challenge02/Proc/Papers/ch02_25_shen.pdf), 2002.
12. Berger M., Bauer B., Watzke M.: A Scalable Agent Infrastructure, (http://www.umcs.maine.edu/~wagner/workshop/04_berger_et_al.pdf), 2001.
13. d'Inverno M., Fisher M., Lomuscio A., Luck M., Rijke M., Ryan M., Wooldridge M.: Formalisms for Multi-Agent Systems, (http://www.ecs.soton.ac.uk/~mml/papers/ker97-2.pdf), 1998.
14. Cao J., Spooner D.P., Turner J.D., Jarvis S.A. , Kerbyson D.J., Saini S., Nudd G.R.: Agent-based Resource Management for Grid Computing, (http://www.dcs.warwick.ac.uk/~hpsg/html/downloads/public/docs/CaoJ.ARMGC.pdf), 2002.
15. Sandholm T., Lesser V.: Issues in Authomated Negotiation and Electronic Commerce: Extending the Contract Net Framework, (http://citeseer.ni.nec.com/57761.html), 1995.
16. Yolum P., Singh M.P.: Locating Trustworthy Services, (http://lsirpeople.epfl.ch/aberer/citations/16_yolum.pdf), 2001.
17. Jennings N.R. : An agent based approach for Building Comples Software Systems, (http://www.insead.fr/CALT/Encyclopedia/ComputerSciences/Agents/).

18. Foster I., Kesselman C., Tiecke S.: The Anatomy of the Grid, (http://www.globus.org/research/papers/anatomy.pdf).
19. Tuecke S., Czajkowski K., Foster I., Frey F., Graham S., Kesselman C.: Grid Services Specification, (http://umbriel.dcs.gla.ac.uk/NeSC/general/papers/GS_Spec_draft02_2002-06-13.pdf), 2002.
20. Frey J., Tannenbaum T., Livny M., Foster I., Tuecke S.: Condor-G: Computational Management Agent for Multi-Institutional Gris, (http://www.cs.wisc.edu/condor/doc/condorg-hpdc10.pdf), 2001.
21. Laszewski G., Foster I., Gawor J., Lane P. , Russell N.R.M.,: Designing Grid-based Problem Solving Environments and Portals.
22. Mangina E., Applied Intelligence (UK) Ltd for AgentLink, June 2002: Review of Software Products for Multi-Agent Systems, (http://www.mcs.anl.gov/~laszewsk/papers/cog-pse-final.pdf), 2002.
23. Luck M., Mc Burney P., Preit C.: Agent Technology: Next Generation Computing, A roadmap for Agent Based Computing, (http://www.agentlink.org),2002.
24. Wooldridge M., Jennings N.R.: Pitfalls of Agent-Oriented Developement, (http://www.csc.liv.ac.uk/~mjw/pubs), 1998.
25. Van Dyke Parunak H.: Practical and Industrial Application of Agent-Basd Systems, (http://www.erim.org/~vparunak/apps98.pdf), 1998.
26. Jennings N.R., Wooldridge M., Mary Q., Westfield College: Application of Intelligent Agents, (http://citeseer.nj.nec.com/jennings98applications.html), 1998.
27. www.globus.org
28. www.fipa.org
29. www.w3.org/2002/ws
30. setiathome.ssl.berkeley.edu
31. www.egrid.org
32. www.entrophy.org
33. www.gnutella.org
34. www.jxta.org
35. Foster I., Kesselman C., Nick J.M., Tuecke S.: The Physiology of the Grid (http://www.gridforum.org/ogsa_drafts/ogsa_roadmap.0.3.doc), 2002.
36. protege.stanford.edu

Recenzent: Prof. dr hab. inż. Andrzej Grzywak

Wpłynęło do Redakcji 7 kwietnia 2003 r.

Abstract

Software engineering is one of main important human activity and a big part of contemporary economy but in practice it is more art then engineering. Practitioner have seen many proclaimed software crises and some receipts to manage with. Despite of this, software products could be find everywhere from simple mechanical device to the area reserved for human mind. For this field seams to be difficult to apply science formalism which lead to simple and concrete answers with practical value come form the word *foreseen* with inherently attribute *proven*.

Starting new software project, we have to cope more often with legacy software and dedicate it more time and effort then to subject filed. We face new barrier, to prove and sometimes even to understand our own product with many complicated interactions with open, non-predictable world (Internet). Our current methodologies are not suitable and have not been applied.

This magnitude interaction led us to find new solutions. It seems that *MAS- Multi Agent System* technology could help. This new technology has many practical working project but not yet widespread in economy. Scientists are concentrating their effort on different subject from *pro-activity* with goal oriented behavior to *reactivity* with social capability.

We start to apply this one technology in hope to realize better system with predictable behavior in this unpredictable open world. As a base we chose some kind of grid computing where autonomous component will share his resource. We will explore social capabilities and so called *computational economy* to achieve stability and optimal behavior.

Adresy

Franciszek GRABOWSKI: Politechnika Rzeszowska, Zakład Systemów Rozproszonych ul. Wincentego Pola 2 , 35-959 Rzeszów, Polska, fgrab@prz.rzeszow.pl .

Krzysztof KADUK: Brzóza Królewska 473, 37-307 Brzóza Królewska, Polska, krzysztof@kaduk.net