Języki Sztucznej Inteligencji

Ontologia a współdzielenie informacji.

W poniższej pracy oparliśmy się głównie na pracy Adam Farquhar, Richard Fikes, Wanda Pratt, James Rice „Collaborative Ontology Construction for Information Integration", stworzonej na Stanford University oraz materiałów Knowledge Systems Laboratory dostępnych na stronie http://www-ksl.stanford.edu. Wykorzystaliśmy także mnogość źródeł dostępnych w sieci Internet.

Grudzień 2002

AGH AiR IV

Paweł Czarny, Radosław Sztando, Jarosław Wetula, Łukasz Zbroszczyk

1. Wprowadzenie

1.1 Zapotrzebowanie na ontologie

Stwierdzenie zapytania do podmiotu wymaga sformułowania projektu koncepcyjnego tego podmiotu. Koncept nazywa i opisuje jednostkę, która może istnieć w podmiocie oraz relacje miedzy jednostkami. Określa ona również słownictwo reprezentujące i komunikujące wiedze w podmiocie.

Jasno określona specyfikacja podmiotu, zwana ontologią jest podstawą dla rozwoju i użycia inteligentnych systemów tak samo jak interoperacje dla systemów heterogenicznych Ułatwia on ludziom rozwijającym system słownictwa do reprezentowania wiedzy zawartej w podmiocie jak również rdzeń samego podmiotu. Ontologie informują użytkownika systemu o słownictwie dostępnym do współdziałania z systemem, oraz znaczenia, jakie system przypisuje określeniu. Ontologie są również decydujące do umożliwienia przedstawicielom działającym na poziomie wiedzy operacji wewnętrznych. Ostatecznie ontologie są pożyteczne w rozumieniu i współdziałaniu człowieka z systemem, np. mogą służyć jako ucieleśnienie jednomyślności uzyskanej przez profesjonalną społeczność przy rozumieniu słownictwa technicznego użytego przez nich.

Konstruowanie ontologii jest trudne i zajmujące dużo czasu. Ten duży koszt jest główną bariera przed tworzeniem dużych „inteligentnych systemów", oraz przed współdziałaniem komputerowych przedstawicieli systemów, działających na poziomie wiedzy. Odkąd ilość projektów koncepcyjnych miała być pożyteczną dla dużej ilości żądań, trudność jest w rozkodowaniu ontologii do użytecznych form i przedstawieniu ich dla aplikacji i zebraniu ich w większe grupy.

Pracowaliśmy nad wymyśleniem oraz rozpowszechnieniem łatwego do użycia narzędzia do tworzenia, oceniania, używania użytecznych ontologii (Fikes, Cutkosky, Gruber& Van Baalen,1991). Stworzyliśmy zestaw narzędzi i usług do wsparcia nie tylko rozwijaniu ontologii, ale archiwizowaniu jednomyślnych i popularnych ontologii. Narzędzia te używają ogólnoświatowej sieci do tworzenia, edytowania i przeglądania ontologii zgromadzonych na serwerze ontologii. Narzędzia wprowadzają udogodnienia, które są kluczowe w promowaniu ontologii przedstawicieli operujących na poziomie wiedzy, wliczając:

- Pól formalny język, który wspiera opis definicji grupujących je w komputerowo interpretowany język. Użyliśmy rozszerzonej wersji języka Ontolingua (Gruber 1992).
- Przeglądanie i odzyskiwanie ontologii ze zbioru. Przeglądanie wymaga prezentacje formalnych opisów w zrozumiałym formacie. Zrobiliśmy to w najłatwiejszy sposób do wyrażenia. Wierzymy ze rosnąca popularność systemów obiektowych (języków, baz danych, syst. CORBA itd.) poszerzy grono korzystających z nich ludzi.
- Zgromadzenie, przetworzenie oraz rozszerzenie ontologii. Wymaga to możliwości identyfikacji i rozwiązywania błędów nazw. Poszerzyliśmy język Ontolingua wiec użytkownicy mogą gromadzić nowe ontologie z starych bibliotek.
- Tłumaczenie ontologii z bibliotek do klasycznych środowisk programistycznych. Napisaliśmy programy tłumaczące ontologie do środowisk: CORBA's IDL, Prolog, CLIPS, LOOM, Epikit, KIF.
- Zdefiniowaliśmy protokół sieciowy i program aplikacji (API) do umożliwienia zdalnym aplikacjom dostępu do serwera ontologii, umożliwiając pobieranie słownictwa, oraz przykładów relacji miedzy terminami.

- Pomoc dla rozpowszechnionych oraz ujednoliconych ontologii. Stworzyliśmy środowisko do rozwijania ontologii zdalnie z duża ilością możliwości do rozwijania, nowych ontologii.

Rozwijanie i używanie technologii powiedzie się, jeśli zostanie ona rozpowszechniona na dla szerokiej ilość użytkowników. Inne wskaźniki sukcesu powinny się pojawić, jeśli technologia pójdzie na przód i zyski uzyskane przez używanie ontologii będą znacząco większe niż koszt wynalezienia ich. W drugiej części dokumentu będziemy próbować omówić nowe możliwości języka Ontolingua, który pozwala zbieranie oraz używanie ontologii zebranych w zbiorze. Sekcja 3 adresuje zaprojektowane narzędzie do edytowania ontologii, w sekcji 4 pokażemy infrastrukturę, która wymyśliliśmy, aby narzędzia mogły trafić do szerokiego grona odbiorców. Sekcja 5 przedstawia strukturę, zastosowanie serwera ontologii.

1.2 Struktura serwera ontologii

Schemat 1 przedstawia schemat serwera ontologii. Najbardziej z lewej strony okienko, przestawia główne zastosowanie i edytowanie serwera ontologii, serwer ten daje nam dostęp do bibliotek ontologii, umożliwia tworzenie nowych ontologii, oraz edytowanie istniejących. Są trzy moduły interakcji z serwerem ontologii wskazywany przez trzy okienka z prawej strony schematu 1.

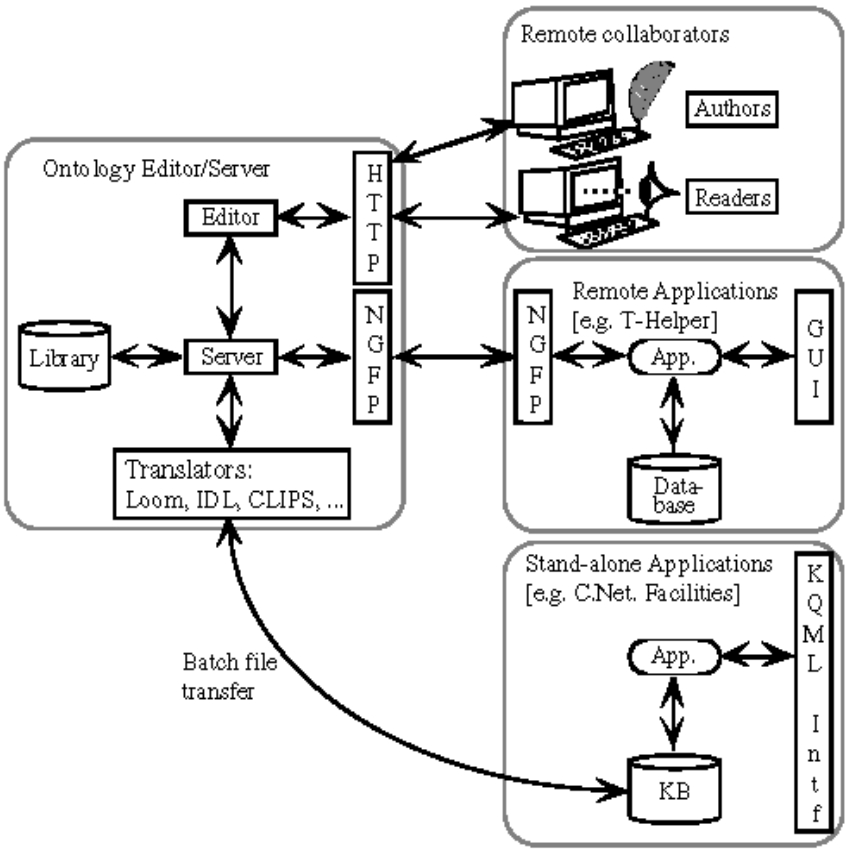


Figure 1: Architecture of the Ontology Server.

Po pierwsze, zdalnie udostępnione grupy mogą używać swoich przeglądarek Internetowych, do otwierania, tworzenia ontologii na serwerze. Serwer współdziała z nimi poprzez ogólnodostępny protokół *HTTP (ang. hypertext transfer protocol)* i język skryptowy *HTML (ang. hypertext language)* używanych w przeglądarkach. To pozwala na dostępność serwera,

do szerokiego grona odbiorców. Każdy użytkownik zaznajomiony z przeglądarka taka jak NetScape Navigator albo NCSA'a Mosaic może otwierać, budować oraz składować ontologie na serwerze. Serwer pozwala na wieloseryjny dostęp, czyli wielu użytkowników, może być zalogowanych na raz do serwera ontologii. Serwer udostępnia wiele narzędzi do współdziałania użytkowników zalogowanych.

Po drugie zdalne aplikacje mogą przeglądać, modyfikować ontologie zgromadzone na serwerze, poprzez Internet, te programy używają sieciowego API, który rozszerza Generic Frame Protocol (Karp, Myers, &Gruber 1995) poprzez sieciowy wygląd. Sieciowy wygląd umożliwia przeglądanie, jak również unowocześnianie ontologii.

Po trzecie użytkownik może tłumaczyć ontologie na format używany przez odpowiednie aplikacje (Grube 1995). Wynik tłumaczenia może by użyty na wiele sposobów. Na przykład tłumaczenie CLIPS daje nam zbiór definicji klas oraz wygląd, który umożliwia uruchomienie go prosto, bez modyfikacji pod aplikacjami CLIPS. Tłumaczenie IDL produkuje pliki nagłówkowe IDL, które kompilator CORBA, może użyć do współdziałania z ORB. Tłumaczenie KIF produkuje, pliki z zadaniami logicznymi, które można użyć przez programy oparte na schematach logicznych, takich jak CommerceNet, do rysowania schematów.

2. Dołączenie modelu Ontolingua

2.1 Zbieranie ontologii

Chcieliśmy, aby ontologie były praktyczne i użyteczne to znaczy, ze koszt wymagany do skonstruowania nowej ontologii musi być minimalny, oraz ogólny koszt budowania ontologii musi być podzielony na wiele potrzeb i użytkowników. Mówimy tu o wielu krokach, dochodząc do celu. Pierwszym z nich jest tworzenie potężnego środowiska do edytowania oraz sprawdzania. Drugim jest, aby środowisko, które stworzymy, było ogólnie dostępne dla szerokiej rzeszy odbiorców. Trzecim jest zbudowanie pomocy dla tworzenia ontologii. Czwartym krokiem jest umożliwienie pisania, oraz używania istniejących ontologii, w bardzo elastyczny sposób. W tej sekcji pokażemy jak edytor ontologii pozwala użytkownikom na wykorzystanie istniejących ontologii z modularnej struktury bibliotek.

Formalnie ontologie w naszym systemie są specyfikacjami logicznych teorii. Specyfikacja ontologii składa się z słownictwa nie-logicznych znaków i pakietu, KIF, które zawierają akceptowalne znaki zastępujące te symbole. Użytkownik ontologii definiuje te znaki, formy definicji, klasy, zależności, funkcji.

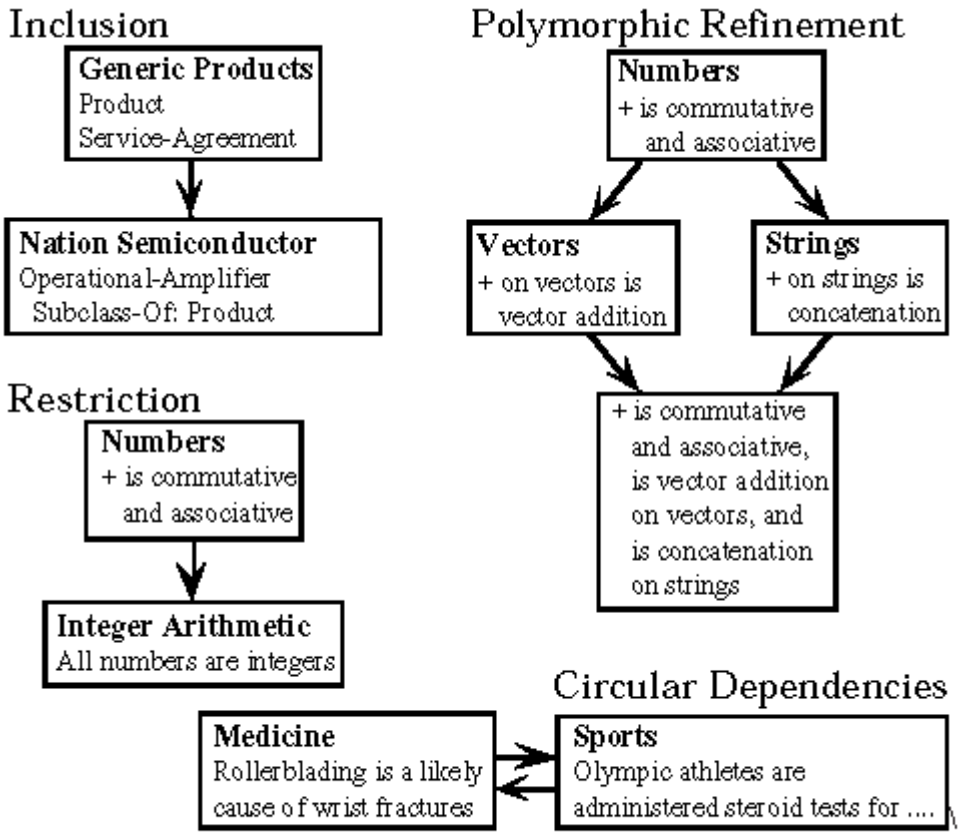


Figure 2 : There are many relationships between ontologies.

Schemat 2 pokazuje kilka motywujących przykładów, które są zaczerpnięte z doświadczenia przy budowaniu ontologii.

Przykład 1 pokazuje najprostszą relacje pomiędzy ontologiami: „zawieranie” Autor National Semiconductor potrzebuje reprezentowanie podstawowych informacji na temat produktu, cen, usług i tak dalej. Chce włączyć całą zawartość ontologii produktu z biblioteki ontologii bez modyfikacji. W przykładzie 2, autor chce rozszerzyć standardowy operator „+” na dwa możliwe sposoby. Biblioteka zawiera dodatkowe informacje w KIF ontologiach. Chce rozszerzyć operator, aby dodawał wektory do listy znaków innej ontologii, odnosimy się do tej operacji jako wielowątkowej . W przykładzie 3 widzimy ze wbudowane relacje pomiędzy ontologiami, mogą się zapętlić. Rozważamy dwie ontologie: jedna dla medycyny, druga dla sportu. Medyczna ontologia musi się odnosić do różnych określeń ze sportowej ontologii i sportowa ontologia, musi się odnosić do medycznej. Musimy sobie poradzić z tego rodzaju zależnościami ostrożnie, ponieważ projektant ontologii nie chce, aby żadna z ontologii nie była pomieszana z nielogicznymi symbolami z innej ontologii. W przykładzie 4 widzimy specjalistyczne ontologie upraszczają założenia zabronionych części. Jako przykład, w wbudowanej arytmetycznej ontologii, wszystkie liczby musza być całkowite. Wiele systemów reprezentacji wiedzy adresuje te problemy w różny sposób. Zanim zaczniemy rozwiązanie, musimy omówić rzeczy, które użyli inni, ilustrujące ich mankamenty, i użyć ich do motywacji naszych nowatorskich wyborów.

Najprostszym i najłatwiejszym podejściem jest dostarczyć pomoc dla modulowanie reprezentowanej wiedzy. Na przykład system THEO (Michell, 1989) używa pojedynczej bazy wiedzy. W niektórych odniesieniach odnosi się to przypadków 1-3 ale posiada dwa minusy: po pierwsze uniemożliwia ograniczyć definicje. Po drugie eliminowanie modularności, wprowadza mylne rozumienie i określenie ontologii. Autor używający systemów tego typu często umiejscawia leksykalne konwencje pomiędzy symbolami. Bez automatycznej pomocy, takie konwekcje jest trudno do wprowadzenia. Jednak wprowadzenie ich może być nawet pożądany. W drugim przykładzie aksjomaty w wektorowej ontologii są takie same jak operatory KIF numeryczne .

Średnio popularne rozszerzenie jest używane do konkretnego grafu (DAG) wbudowanej relacji pomiędzy relacjami takich jak udostępnionymi przez Epikit. Mechanizm umożliwiający modularność, ograniczenia i różna ilość argumentów ma dwa minusy: po pierwsze nie mogą wystąpić cykle pomiędzy teoriami, jak widzieliśmy jest to zarówno łatwiejsza forma, jak i bardziej pożądana mając cykliczne relacje pomiędzy termami w ontologii. Po drugie w łatwiejszej formie w mechanizmie rezultatu nie koniecznie są konflikty nazw, na przykład ontologii dla naukowego kursu może zawierać ontologie chemiczne, akademickie, obie zawierające i definiujące *testy*, ale na różne sposoby. Musza być zasady rozgraniczające **testy** w chemii i **testy** w akademii.

System LOOM pokazuje DAG zawartość relacji, ale rozszerza proste podejście, aby pozwalać referencjom nie-logicznych symboli w ontologii, które nie były zawarte. Odniesienie się do symboli jest nieokreśloną ontologią, ale nie zawiera wszystkich aksjomatów z ontologii, ale minimalną ich ilość.

Są dwa odrębne aspekty naszego rozwiązania problemu: wbudowany model, który definiuje jak aksjomaty i nielogiczne symbole są zawarte w nowej ontologii i model interfejsu, który definiuje relacje pomiędzy ciągami znaków wprowadzonych przez użytkownika i nie-logicznych symboli w ontologie.

2.2 Naturalne wejście i wyjście

Semantyczny model przedstawiony powyżej dostarcza mocnej, prostej drogi dla ontologii do zmontowania i ponownego użycia. Jakkolwiek, w celu eliminacji dwuznaczności wymaga symboli o niezręcznym rozszerzeniu niepowtarzalnych nazw, które mogą być nieznane dla użytkownika. Ponadto nie pozwala użytkownikowi dokonywać ważnych operacji takich jak przemianowanie symboli zawartych w ontologiach lub selektywnej kontroli, której symbole mogą być importowane z dołączonych ontologii lub eksportowane do innych ontologii. Serwer Ontologii rozwiązuje te problemy z dodatkową zdolnością do konwersji odniesień symboli w tekście przesyłanym do i z wewnętrznego odwzorowania symboli. Zdolność ta zostanie opisana poniżej.

Ontolingua wymaga aby, każdy nielogiczny symbol odniesienia w wejściowym lub wyjściowym strumieniu był zdefiniowany w jakiejś ontologii i posiadał nazwę. Ontologia w której symbol jest definiowany wywołuje rodzimą ontologię symbolu. Podobnie każda ontologia mająca nazwę jest odróżnialna od innych ontologii.

Serwer Ontologii interpretuje relacje symbolu w wejściowym strumieniu lub wytwarza referencje w strumieniu wyjściowym z perspektywy danej ontologii. Na przykład, jeżeli symbol S jest zdefiniowany w ontologii A oraz zdefiniowany w ontologii B, wówczas z perspektywy A, wejściowy tekst „S" jest interpretowany jako „symbol o nazwie S zdefiniowany w ontologii A", podczas gdy z perspektywy B, wejściowy tekst „S" jest interpretowany jako „symbol o nazwie S zdefiniowany w ontologii B".

Perspektywa z której dane odwołanie do symbolu jest interpretowana może być wyraźnie wyspecyfikowana poprzez znak @. Tak więc do symbolu S interpretowanego z perspektywy ontologii A można się odwołać poprzez „S@A". Odwołanie do symbolu zawierające „@«nazwa_ontologii»" świadczy o całkowicie wykwalifikowanym odwołaniu, umożliwiającym odwołanie do zdefiniowanych symboli z innych ontologii.

System wejścia/wyjścia Serwera Ontologii dostarcza narzędzi do przemianowania symboli, które umożliwiają użytkownikowi przypisanie nazwy symbolowi lokalnemu z perspektywy danej ontologii. Przemianowanie jest sprecyzowane przez regułę zawierającą nazwę ontologii referencję symbolu, i nazwę która zostanie nadana jako nazwa lokalna danego symbolu z perspektywy danej ontologii. Wykorzystując regułę przemianowania, system rozpozna lokalną nazwę jako referencję do danego symbolu gdy proces wchodzi w daną perspektywę i użyje lokalnej nazwy do odwołań do danego symbolu wytwarzając wyjście z danej perspektywy. Przykładowo w ontologii A reguła przemianowania może przypisać lokalną nazwę `car` dla `auto@vehicles`. To narzędzie umożliwia środowisku ontologii odwołanie do symboli z innych używanych ontologii używając odpowiedniej nazwy w danej ontologii i określa jak mają zostać rozwiązane konflikty między nazwami symboli z wielokrotnymi ontologiami.

Dla wygody wejścia i składni wyjścia, „@«nazwa_ontologii»" może zostać ominięty z referencji symbolu, gdy nazwa symbolu jest jednoznaczna z zamierzoną perspektywą. W szczególności do przekazywanego symbolu z perspektywy ontologii A można się odwoływać prosto jako S wtedy i tylko wtedy gdy symbol:

- jest zdefiniowany w ontologii A pod nazwą S
- został przemianowany na S w ontologii A
- został zaimportowany do ontologii A

Nazwa używana do odnoszenia się do definiowanego symbolu musi być rozpoznawana w danej ontologii z jej perspektywy. Tak, więc przedstawiona powyżej konwencja opuszczania przedrostka „@«nazwa_ontologii»" z symbolu odwołania stwierdza, że nazwa S jest rozpoznawana w ontologii A wtedy i tylko wtedy jeżeli S jest nazwą symbolu zdefiniowanego w A, lub S jest nazwą symbolu zaimportowanego do A bez zmiany nazwy w A, lub S jest lokalną nazwą dla symbolu w A.

Poniżej zostanie opisany mechanizm importowania symboli do ontologii.

Każdy symbol w ontologii, w której jest zdefiniowany jest wyznaczony jako publiczny lub prywatny. Standardowo symbole są rozważane jako publiczne.

Serwer Ontologii związuje z każdą ontologią zestaw takich ontologii których publiczne symbole są do niej importowane. Polecenia użytkownika są dostępne (przydatne) dla grupy ontologii. Jednakże, w celu ułatwienia mechanizmu importu symboli, Serwer Ontologii automatycznie dodaje do tej grupy każdą ontologię jednoznacznie dołączoną.

W celu zapewnienia kontroli nad importowaniem indywidualnych symboli, Serwer Ontologii kojarzy z każdą ontologią zbiór śledzonych symboli, zablokowanych przed zaimportowaniem do ontologii.

Serwer Ontologii używa mechanizmu śledzenia do ochrony przed dwuznacznościami występującymi w formach tekstowych odniesień symbolicznych przez automatyczne blokowanie importu do ontologii symboli które mogą mieć taką samą postać tekstową z perspektywy ontologii. W ten sposób jeżeli do symbolu można się odwołać przez S z perspektywy ontologii A, a do innego symbolu odwołujemy się przez S z perspektywy

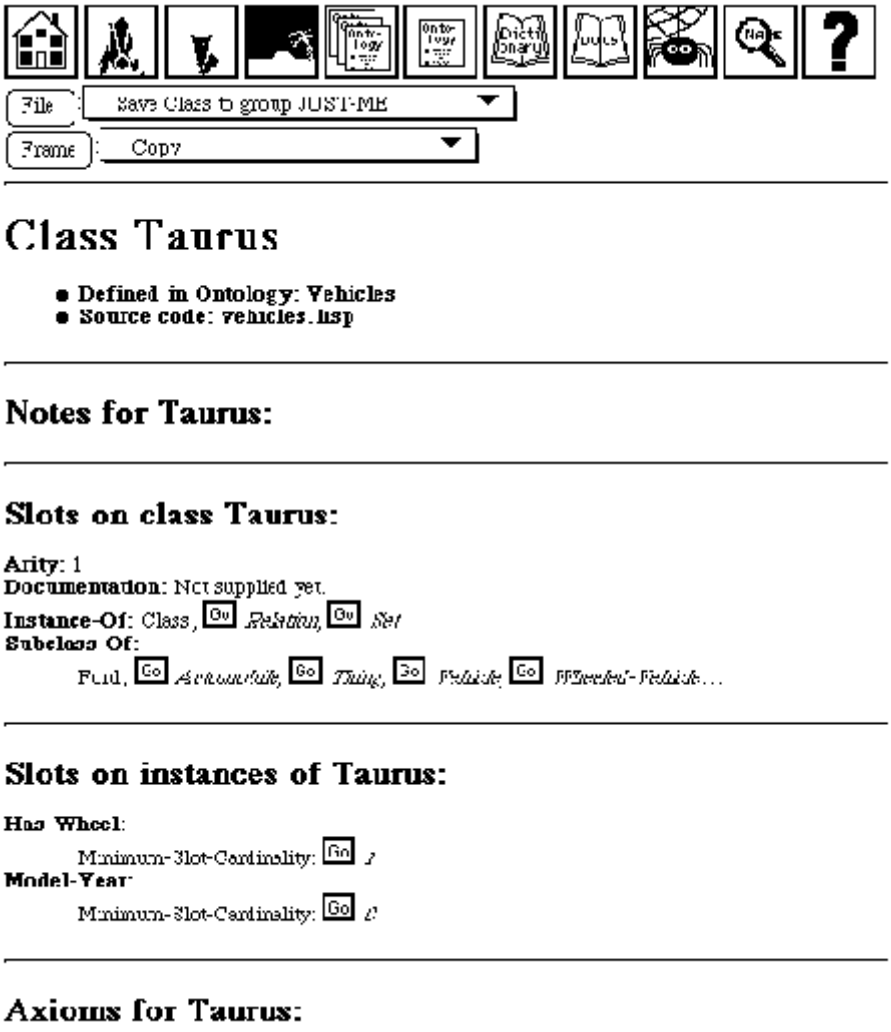
ontologii B i publiczne symbole z obu ontologii A i B są importowane do ontologii C, wówczas Serwer Ontologi automatycznie doda S@A i S@B do listy śledzonych symboli C w celu ochrony „S" przed dwuznacznością odwołań z perspektywy C. To automatyczne śledzenie występuje niezależnie od rozkazów w których definicje i związki są sprecyzowane.

3. Środowisko rozwojowe ontologii

Celem było stworzenie ogólnego środowiska ułatwiającego rozwój i używanie ontologii. Środowisko musi wspierać użytkownika w bazowym tworzeniu zadań przeglądania, tworzenia, zachowywania, rozwoju i użycia ontologii. My także realizujemy wiele życzeń użytkowników rozwijając ontologie w procesie porozumienia, dlatego musimy dostarczać narzędzi pomagających ludziom współpracować podczas rozwoju ich ontologii. W tej sekcji będziemy rozważać opisane na naszej stronie podstawowe środowisko ontologii które dostarcza wsparcia z bazowym rozwojem zadań, ułatwiającym współpracę i użycie.

3.1 Przeglądanie ontologii

Istotnym komponentem środowiska jest możliwość szybkiego skoku z jednego termu w ontologii do innego używając odnośników. Wybierając nazwę termu użytkownik przenosi się do strony wyświetlającej definicje termu. Definicja zawiera nieformalną dokumentacje i formalne instrukcje o termie. Częściej niż wyświetlanie tych informacji w czystej formie opartej na logice, wyświetlamy tą informację w ukierunkowanej wynikowo lub w formie *frame-based*. W ramce, sloty są wyświetlane wzdłuż lewej strony ekranu , a wartości korespondujące z slotami są wyświetlane po nazwach slotów. Informacje nie mogące być wyświetlone jako slot i wartości aspektów ukazują się na stronie jako aksjomaty. Aksjomaty są dzielone na 3 kategorie: aksjomaty równoważności, implikacji i inne spokrewnione aksjomaty.



Kiedy użytkownicy badają całe ontologie, często chcą zobaczyć informację o termie o którym dowiedzieli się z definicji innego termu. Do utrzymania tej cechy, serwer ontologii dokonuje ograniczonej grupy wnioskowań pozwalającej wnioskować informacje, co jest typowe dla ramkowo-bazowej reprezentacji systemu. Serwer ontologii utrzymuje klasy, podklasy, dziedziczenie i odwracanie slotów. Dla odróżnienia między wnioskowaniem i natychmiastowymi danymi, bezpośrednie informacje ukazują się w zwykłej czcionce, wnioskowane informacje w czcionce *Italic*.

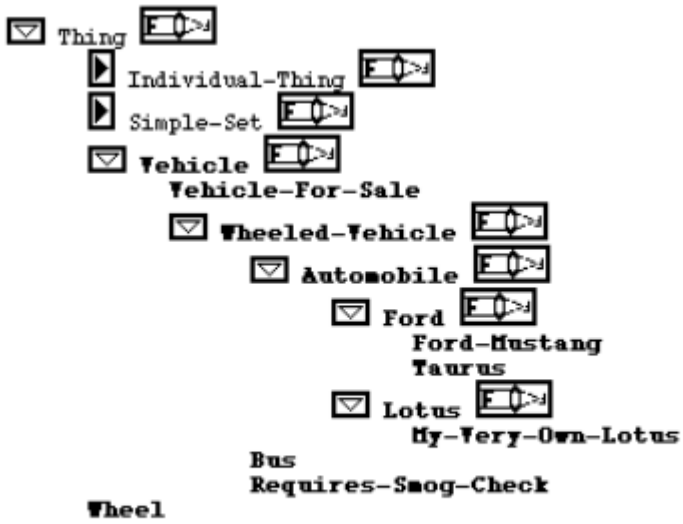


Figure 4: The class browser provides an overview of the class hierarchy.

Kiedy użytkownik nie jest zainteresowany szczegółami, może uzyskać widok generalnej hierarchii klas. Dla podtrzymania funkcjonalności zaimplementowano cechę, gdzie użytkownik może zobaczyć wszystkie klasy w ontologii i relacje hierarchiczne w bibliotece ontologii. Można również wyświetlić wszystkie podklasy klas, które je posiadają.

3.2 Tworzenie i edytowanie ontologii

Środowisko edytowania ontologii zawiera podobny wygląd do środowiska przeglądania ontologii więc użytkownik nie musi się uczyć dwóch różnych interfejsów. Różnica między interfejsami to dwa nowe typy ikon dodane do środowiska edycji: pisak i ikona wstawiana. Pióro edycji ukazuje informacje, które użytkownik może modyfikować. Jeżeli użytkownik pragnie zmienić informacje wybiera odpowiedni `edit pen` i pole do modyfikacji aktualnych informacji. Naciskając jedną z ikon do wstawiania, wyświetli się pole do wprowadzenia nowych informacji. Zawartość tej formy zależy od typu który zostanie dodany lub edytowany.

Reszta komend jest dostępna jako menu opcji u góry ekranu np. polecenia dla stworzenia nowego termu jest wyświetlane w tym menu. Użytkownik może wybrać typ tworzonego termu i zostanie zachęcony do wprowadzenia informacji o tym termie.

3.3 Zachowywanie ontologii

Serwer Ontologii dostarcza kilku cech pomagających w zachowywaniu ontologii. Np. użytkownik może porównać ontologię z wcześniejszą wersją jej samej. To jest bardzo użyteczne gdy użytkownik pragnie monitorować zmiany w ontologii lub czasami cofać dokonane zmiany. Ontologie mogą być także analizowane wykrywając niezgodności przy użyciu polecenia analizy ontologii. Wszystkie sloty i ich wartości, oraz aspekty są sprawdzane w celu potwierdzenia spełniania zastosowanych ograniczeń.

Własna dokumentacja jest ważną częścią tworzenia (zachowywania) ontologii. Serwer Ontologii przetrzymuje specjalne słowa kluczowe wewnątrz prywatnej dokumentacji znanej jako *notes*. Użytkownik przydziela słowa kluczowe jak np. *verified-by, see-also, modification-by* rozszerzając strukturę swojej dokumentacji.

3.4 Używanie ontologii

Mimo iż Serwer Ontologii aktualnie nie dostarcza dużej mocy wnioskowania, to stara się ułatwiać używanie ontologii. Jednym sposobem wykorzystania rozwiniętych ontologii z Serwerem Ontologii jest adaptacja ontologii do reprezentacji języka innego systemu jak CLIPS, LOOM lub Prologa. Aktualnie Serwer Ontologii umożliwia przełożenie do innej reprezentacji, które użytkownik później może przesłać wykorzystując email. Użytkownicy mają także możliwość przesłania ich ontologii jako kodu źródłowego lub formatowanego tekstu.

4. Infrastruktura

W tej części, omówimy infrastrukturę konieczną do zapewnienia narzędzia, które wspierać będzie rozproszoną współpracę na ontologiach Jednym z naszych najważniejszych celów jest zapewnienie tak łatwego i szerokiego dostępu jak to tylko możliwe. Oznacza to, że musi ono wspierać wiele platform (np.: PC, Mac i Unix) dla szerokiego zakresu zastosowań (np.: przemysłowe, akademickie, badania). Narzuca to silne wymagania na obecnie wykorzystywane narzędzia. Dla osiągnięcia tego celu przy dopuszczalnych kosztach, zdecydowaliśmy się na wybór istniejącej (i rozwijającej się) struktury sieci WWW.

Sieć WWW dostarcza rozproszony dostęp do multimedialnych dokumentów hypertekstowych. Jej struktura zbudowana jest z czterech elementów: klienta działającego na każdej maszynie użytkownika, serwera, który działa na maszynie dostarczającej informacje, protokołu HTTP i języka HTML. Klienci to przykładowo Netscape Navigator, Mosaic, tekstowy Lynx czy inne. Działają na wszystkich platformach, są już zainstalowane i dziennie korzysta z nich wiele milionów potencjalnych użytkowników. HTTP określa żądania klientów i jak serwer powinien zwrócić im informację. HTML przekazuje informacje na temat tego jak sformatowany ma być ten tekst podobnie jak widzi to większość edytorów tekstu (pogrubiona czcionka, kursywa , nowy akapit itp.) a także odnośniki związane z wybraną porcją tekstu. Odnośniki wskazują na nowy dokument który ma być przetworzony kiedy użytkownik wybierze odpowiedni tekst. Kluczowym elementem HTTP/HTML jest jednolity lokalizator zasobu (URL – *uniform resorce locator*) który specyfikuje położenie dokumentu hypertekstowego. URL zawiera typ protokołu dostępu, nazwę maszyny, port i informację o położeniu dokumentu na danej maszynie.

Dostosowanie struktury WWW polegało na: (1) udostępnieniu użytkownikom oglądania ontologii jako dokumentu HTML w ich przeglądarkach, (2) stworzeniu specjalnego serwera zapewniającego im możliwość edycji. Osiągnęliśmy ten cel dzięki temu że klienci WWW są rozproszeni, działają na wielu różnych platformach, interfejsach i systemach operacyjnych. Poza tym bardzo wielu użytkowników jest zaznajomionych z programami klienckimi WWW więc łatwo mogą połączyć nasze narzędzia w codziennym pracy.

Jest jednak w tym małe „ale”. HTTP było początkowo zaprojektowane do żądania i opisywania „statycznych” dokumentów hypertekstowych. Społeczność WWW rozszerzyła HTTP/HTML aby zawierało więcej możliwości interaktywnych i lepiej wspierało dynamiczne tworzenie dokumentów co umożliwia pracę takich wyszukanych aplikacji jak Serwer Ontologii. Niestety wiele problemów związanych z HTTP/HTML stale wymaga rozwiązania. Poniższa część skupi się na rozwiązywaniu niektórych ograniczeń w HTTP.

HTTP jest *bezstanowym, opartym na transakcji* protokołem. Przez *bezstanowość* rozumiemy, że serwer nie wymaga posiadania informacji na temat klienta czyjego żądań. Oznacza to że każde żądanie jest całkowicie samodzielne; zawiera wszystkie informacje potrzebne

serwerowi do zwrócenia na nie odpowiedzi. *Oparte na transakcji* oznacza że podstawowym elementem interakcji pomiędzy serwerem a klientem jest otwarcie przez ostatniego połączenia, wysłanie pojedynczego żądania, otrzymanie odpowiedzi i zamknięcie - zakończenie połączenia. Może to kontrastować z protokołami sesyjnymi gdzie połączenie istnieje, trwa przy wielu transakcjach. Decyzja stworzenia HTTP protokołem bezstanowym opartym na transakcji była słuszna. Serwery mogą być bardziej niezawodne i wydajniejsze – nie muszą pamiętać informacji o klientach lub połączeń do nich. Przetwarzając setki tysięcy żądań od dziesiątków tysięcy klientów na dzień mogłyby inaczej mieć duże problemy z poprawnym działaniem.

Jakkolwiek wymagania protokołu dla rozproszonej współpracy, edycji są całkiem różne. W pewnej części, środowisko do rozproszonego „współprzetwarzania" musi zapewnić pełno funkcjonalny połączeniowy protokół. Jednym z naszych wyzwań jest stworzenie takiego protokołu który byłby położony ponad HTTP w sposób przezroczysty dla istniejących przeglądarek.

4.1 Wykorzystanie poziomu HTTP

Są trzy obszerne sposoby podejścia do przedstawienia stanu w bezstanowym transakcyjnym protokole HTTP:

Zakodować stan w każdym żądaniu i odpowiedzi. Przy takim podejściu, klient jest odpowiedzialny za zachowanie informacji o stanie; serwer potrzebuje jedynie zakodować bieżący stan w odpowiedzi i zdekodować go gdy przetwarza żądanie.

Zawrzeć znacznik czasu w każdej transakcji wskazujący na stan kiedy dana transakcja wystąpiła. Serwer zachowuje informacje o stanie wraz ze znacznikiem. Jeśli serwer zapamiętuje stary stan (np.: przez zapisanie w tablicy stanów lub zachowanie odtwarzalnego zapisu transakcji) metoda ta umożliwia coś na kształt podróży w czasie. Przez ponowne wykonanie starego żądania możesz wrócić do stanu, który oryginalnie był rezultatem z jego wykonania. Ponadto wykonanie tego samego żądanie wielokrotnie nie przynosi dodatkowego efektu.

Zawarcie w żądaniu wystarczających informacji do określenia porcji stanów co jest sztucznym rozwiązaniem. Żądania zawsze rozwijają się z bieżącego stanu, który jest zachowany na serwerze. To podejście nie umożliwia „podróży w czasie". Wykonanie tego samego żądania wielokrotnie może dać różne efekty.

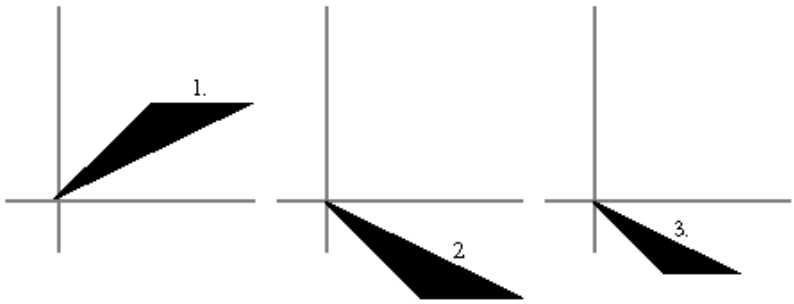


Figure 5 : A triangle after being flipped and scaled as a result of client requests.

(1) Client State	(2) Server Tagged State	(3) Server State
obj=[(0,0)(1,1)(2,1)], command=flip-vertical	obj=1, command=flip-vertical	obj=1, command=flip-vertical
obj=[(0,0)(1,-1)(,2,-1)]	obj=2	obj=1
obj=[(0,0)(1,-1)(,2,-1)], command=scale75	obj=2, command=scale75	obj=1, command=scale75
obj=[(0,0)(,75,-,75)(1,5,-,75)]	obj=3	obj=1

Tabela 1 Różnice w podejściu trzech metod

Tabela 1 i rysunek 5 przedstawiają różnice pomiędzy tymi trzema podejściami. Pokazane jest jak trzy różne serwery mogą poradzić sobie z problemem symetrii względem osi i skalowaniem trójkąta. W podejściu 1 pole `obj` zawiera listę punktów które opisują wielokąt. Wysłanie żądania definiującego trójkąt wraz z komendą wykonania symetrii osiowej zwraca nową reprezentację odbitego elementu. Następne wysłane żądania zawiera współrzędne obiektu wraz z komendą skalowania i współczynnikiem skali jako dodatkowym argumentem. Serwer zwraca współrzędne opisujące obiekt po transformacji. Wiele serwerów WWW wykorzystuje takie podejście. Kiedy stan jest duży lub nie przyjmuje prostej reprezentacji należy znaleźć bardziej dogodne podejście i łatwiej przekazywalną reprezentację danych (np. przetwarzany wielokąt może być opisany za pomocą wielu tysięcy punktów). Dwie najpowszedniejsze metody to (1) zastosować algorytm kompresji dla opisu obiektu, i (2) użyć sekwencji przekształceń zastosowanych do obiektu odniesienia (np. Xerox PARC Map Server). Jednak dla wielu aplikacji to podejście jest niewykonalne. Wyobraźmy sobie edytor tekstu skonstruowany zgodnie z tą metodą – każda zmiana tekstu wymagałaby przesłania dwukrotnie całości dokumentu.

W podejściu 2, obiekt nie jest bezpośrednio reprezentowany w żądaniu czy odpowiedzi. Zamiast tego każda wiadomość zawiera znacznik określający serwerowi, która wersja przestrzeni operacji jest zastosowana. `Obj=1` odnosi się do oryginalnego trójkąta, `Obj=2` do odbitego a `Obj=3` do przeskalowanego, odbitego trójkąta. Jedną miłą własnością tej reprezentacji jest możliwość „podróży w czasie”. Dla przykładu, użytkownik mógł wykonać polecenie `scale50` do obiektu `obj=2` rezultatem czego byłby obiekt `obj=4` w którym trójkąt jest odbity a później przeskalowany 50%. Jeśli użytkownik odświeży stronę w swojej przeglądarce – zawsze uzyska ten sam rezultat. Związane są z tym dwa problemy: pierwszy, może być zbyt kosztownym dla serwera aby zachowywać informację o wszystkich stanach. Jeśli natomiast stare stany nie zostaną zapisane, odświeżenie strony może spowodować błąd mówiący, że stan odniesienia nie jest już dostępny. Drugi problem: dla niektórych użytkowników zjawisko „podróży w czasie” może być nieco zaskakujący. Dla przykładu

przypuśćmy, że nasz użytkownik z przeglądarką powtórnie odwiedza stronę z oryginalnym trójkątem a następnie ją odświeża, obraz pozostaje niezmieniony. Wielu użytkowników może uznać że ich zmiany nie zostały zachowane, co może być dość wkurzające.

W podejściu 3, zapytania są zawsze wykonywane w odniesieniu do bieżącego stanu. Zamiast znacznika stanu, wykorzystywany jest znacznik dla każdego argumentu, który konieczny jest do wykonania polecenia. Zauważmy, że odpowiedź serwera może być inna po wykonaniu każdej komendy. Jeśli użytkownicy wykorzystywali przeglądarkę, zachowywali stare widoki trójkąta i odświeżali stronę po przekształceniach to mogą w rezultacie zobaczyć obraz po transformacji. To podejście wymaga mniejszego nakładu pracy po stronie serwera ponieważ tylko bieżący stan jest zachowywany. Jest on także bardziej intuicyjny dla wielu użytkowników.

Serwer Ontologii wykorzystuje to ostatnie podejście. Każde żądanie jest rozwijane z bieżącego stanu. Każde żądanie posiada zakodowane polecenie i jego argumenty.

Stałość stanu użytkownika po stronie serwera jest konieczna do zaimplementowania tego schematu (konieczne jest istnienie stałego stanu). Zdecydowaliśmy się wykorzystać stały serwera. Protokół HTTP jest ciągle wystarczająco prosty by stworzyć praktyczną tego realizację.

Nasza strategia jest inna od wykorzystywanej przez większość serwerów HTTP jak NCSA'owy HTTPD. Typowy serwer HTTP nasłuchuje oczekując połączeń i w razie jego wystąpienia tworzy efemeryczny proces do obsługi żądania. Te procesy posiadają własną przestrzeń adresową i nie mają dostępnych stanów innych niż argumenty podawane im w żądaniu HTTP. Nasz serwer, w przeciwieństwie, posiada pojedynczy stały proces nasłuchujący połączeń. Gdy pojawia się żądanie tworzony jest krótkotrwały *wątek* który zajmuje się obsługą żądania. Wątek ten ma dostęp nie tylko do argumentów z żądania, ale także do wszystkich stanów we wspólnej przestrzeni adresowej serwera.

Sesja jest podstawowym stanem reprezentującym obiekt. Pojedyncze strony również mogą posiadać stan. Stan użytkownika (preferencje) jest zgodny z sesjami. Oddzielenie sesji od użytkownika umożliwia zwielokrotnienie sesji dla użytkowników. Konsekwencją jest blokada odczytu-zapisu. Nie mamy tu jednak impasu ponieważ brak tu sprawdzania (tak problematycznego w protokołach transakcyjnych) dla stron. Zapis zdarza się sekwencyjnie. Porównania, cofanie i ponawianie zmian przez użytkownika pomaga uniknąć problemów. Nasza aplikacja nie obsługuje wyników wielokrotnej równoczesnej edycji tego same obiektu.

4.2 Dodawanie sesji, użytkowników i grup.

Sam stan nie jest wystarczający do zapewnienia wspólnego edytowania. Potrzebujemy również reprezentacji użytkowników i sesji więc każdy użytkownik może dobrać indywidualnie środowisko i grupę użytkowników którzy mogą pracować wspólnie na danym obiekcie (np. ontologii). Sesja zapewnia więcej niż grupowanie transakcji. Wielu użytkowników może dzielić stan w obrębie sesji. Serwer musi jednak umożliwiać pracę ze stanami i sesjami oraz utrzymywać zawartość każdej sesji oddzielnie i nienaruszalnie dla pozostałych sesji.

Istnieje kilka możliwości zapewnienia określanych stanów sesji:

- Tworzyć procesy z ich własną niezależną przestrzenią adresową dla każdej sesji
- Wykorzystywać ciągłe zachowywanie wszystkich informacji o każdej sesji, przywracanych w trakcie obsługi odpowiedniego żądania.

- Wykorzystywać sesyjne struktury danych dla zachowania informacji o stanie i uniknięcia niewłaściwych odwołań z jednej sesji do innej.

Serwer Ontologii korzysta z ostatniej z tych metod z trzech powodów. Po pierwsze jest wiele dużych bibliotek tylko odczytywanych które zawarte są w większości ontologii użytkowników. Wykorzystanie pojedynczej współdzielonej przestrzeni adresowej pozawala na łatwiejsze wykorzystanie ich pomiędzy sesjami. Drugim powodem jest to że kompletne środowisko zawierające edytor, tłumacza, narzędzia analizujące i inne zawierające się w takim systemie mogłyby się niepotrzebnie powielać w każdym tworzonym procesie. Po trzecie, używanie pojedynczego procesu serwera umożliwia łatwe poprawianie, modyfikację czy rozwijanie go podczas działania. W rzeczywistości większość tych zmian jest wprowadzana bez przerywania obsługi klientów.

Gdy tworzona jest sesja, serwer buduje obiekt w którym zachowywany jest specyficzny dla niej stan i generowany jest unikalny identyfikator sesji. Po utworzeniu sesji wszystkie żądania i działania zmieniające stan sesji musza posiadać identyfikator sesji.

Ostatnim elementem jest użytkownik. Serwer utrzymuje indywidualnych użytkowników z bazą ich preferencji. Kiedy użytkownik wykonuje operacje korzystając ze swojego chronionego hasłem loginu może, on stworzyć nową sesję lub dołączyć się do istniejącej jeśli została stworzona z możliwością dostępu grupowego.

4.3 Kodownie żądań i rezultatów w URL.

W poprzednim rozdziale opisaliśmy trzy główne metody wykorzystania stanów w bezstanowym protokole jakim jest HTTP. W tym opiszemy precyzyjnie mechanizm, który wykorzystuje nasz serwer. Ten mechanizm jest szeroko wykorzystywany w systemach które zapewniają pełno funkcjonalną interakcję poprzez WWW.

Każdy URL reprezentuje kompletne wywołanie procedury – procedurę która ma zostać wykonana wraz z koniecznymi jej argumentami. Te procedury generują strony HTML, które mogą zawierać osadzone URL jako rezultaty dodatkowych wywołanych procedur. Zamiast kodowania procedur i wszystkich ich argumentów w URL, możemy użyć unikalnego znacznika. Jest on używany wówczas jako indeks w tabeli która zawarta jest na serwerze. Ponieważ serwer posiada również sesje i stan użytkownika, URL będzie składał się z trzech części: identyfikatora sesji, identyfikatora użytkownika, i identyfikatora polecenia strony.

Skupmy się na problemie dostępu do relacyjnej bazy danych z informacją biblioteczną. Pierwsza otwierana strona, którą zobaczy użytkownik zawierać będzie pole do wpisania zapytania oraz przycisk do wykonania go. Tytuły kilku pierwszych pozycji odpowiadających temu zapytaniu będą widoczne na drugiej stronie. Wybranie odpowiedniego tytułu powoduje pojawienie się pełnej informacji. Na dole drugiej strony znajduje się przycisk pobierania dalszych tytułów.

Zobaczmy jak jest to przedstawione w tablicy UID. Tablica musi zawierać wszystkie informacje konieczne do jednoznacznego określenia zawartości każdej ze stron.

UID	Command	DB	Start	Query
UID0	get-query	—	—	—
UID1	execute-query	db1	0	select title from bib where author = "Fikes"
UID2	execute-query	db1	2	select title from bib where author = "Fikes"
UID3	get-citation	db1	—	select citation from bib where no = 25
UID4	get-citation	db1	—	select citation from bib where no = 37

Tablica 2 Tablica UID – wiązanie komend ich parametrów z URL

URL jest zbudowany z kombinacji UID z identyfikatora sesji i użytkownika. Dla przykładu URL dla żądania inicjacji bazy danych może wyglądać następująco `/sid27/uid1&user=farquhar`. Zauważ, że jeśli inny użytkownik doda napis do „Fikes" i pierwszy użytkownik przeładuje stronę z tak dodanym napisem uzyska nowe dane. Ponadto serwer nie potrzebuje zachowywać wszystkich informacji zwracanych na żądania – tylko polecenia dające dostęp do dodatkowych informacji, których może potrzebować użytkownik. Jeśli polecenie `execute-query` rozpozna preferencje użytkownika określające co powinno być zwracane – tylko tytuły czy pełne informacje o pozycjach, wtedy zmiana preferencji i odświeżenie strony powinno zakończyć się poprawnym zwracaniem od odpowiedzi.

Nasz obecny serwer implementuje to podejście. Istnieje globalna tablica sesji, każda sesja zawiera tablice użytkowników którzy dołączeni są do sesji, stan sesji i dwie tablice zawierające UID. Pierwsza z nich odnosi UID do określonej pozycji danych zawierających polecenie i argumenty a także wskaźnik powrotny do UID. Druga tablica odnosi polecenie i jego argumenty do pozycji a w końcu do UID. Każda z nich jest zaimplementowana jako rozdzielona tablica dla zapewnienia natychmiastowego dostępu, choć pierwsza tablica mogłaby być również zaimplementowana jako rozszerzalna macierz indeksowana przez UID.

Inne podejście to alokować różne UID dla każdego wystąpienia odniesienia na stronie niezależnie czy miało ono miejsce wcześniej. To oczywiście zapobiega konieczności posiadania tablicy z odwrotnym odwzorowaniem, ale w praktyce mogłoby wybitnie zwiększyć rozmiar tablicy UID. W wypadku odwrotnej tablicy odwzorowań każdy odnośnik oznacza że podane polecenie może zmienić kolor po wybraniu prze użytkownika. W drugim wypadku mogłoby to nie wystąpić co pozostawiłoby użytkownika zmieszanego, myślącego „Mógłbym przysiąc że byłem tam, ale on nie zmienił się na czerwony".

Na poniższym przykładzie zobaczymy jak te tablice i struktury danych są używane do przetwarzania pewnych żądań. Kiedy użytkownik wyśle zapytanie jego przeglądarka prześle taki URL:

```
/sid27/uid0
&user=farquhar&db=db1&command=executequery
&query=select+title+from+bib+where+author=fikes
```

Serwer wydobędzie z tego identyfikatory sesji, unikalny i użytkownika. Identyfikator sesji jest używany do pobierania sesji i ustalenia każdej z globalnych (lub dynamicznych)

zmiennych obowiązujących w niej. To samo jest dokonywane z identyfikatorem użytkownika. Następnie z tablicy UID-pozycja_sesji pobierana jest pozycja odpowiadająca UID0. Odpowiednio polecenie `get-query` jest wykonywana z `db` i argumentami `query` które pobierane są z URL. Polecenie `get-query` wykorzystuje argumenty `db` i `query` do konstrukcji klucza dla drugiej tabeli by sprawdzić czy znajduje się tam już UID dla takiego żądania. Ponieważ nie ma tam takiego pola, tworzony jest więc identyfikator UID1, wykonywane jest żądanie oraz powstają UID2, UID3 i UID4 w trakcie budowania strony HTML dla tej odpowiedzi. Wynikowa strona będzie zawierała URL `/sid27/uid3&user=farquhar`, który wybrany spowoduje wykonanie polecenia pobierającego opis do pozycji 25. Możemy myśleć, że ten URL jest związany z tym opisem, pomimo że powstał on w wyniku `polecenia get-citation` i opis ten może nie istnieć jako dokument HTML. Strona związana z tym URL równie dobrze może być dynamicznie konstruowana przez serwer. Jeśli użytkownik wybierze ten URL serwer zbuduje informacje o użytkowniku i sesji analogicznie jak poprzednio. Następnie po prostu pobierze pozycje dla uid3 z tabeli UID-pozycja i wykona zapisaną komendę `get-citation`.

Niezawodne serwery muszą oczywiście sprawdzać sesję, użytkownika i unikalny identyfikator a także raportować o każdym błędzie wykonywania poleceń. Nasz serwer jest zaimplementowany przy użyciu Common Lisp. Oznacza to że wiele z wymaganych do implementacji tego podejścia mechanizmów było już stworzonych - w tym tablice pracujące na samodzielnych strukturach, dynamiczne powiązania i silny system warunkowy.

Koszty tego podejścia okazały się być bardzo rozsądnymi dla naszej aplikacji. W praktyce tablica UID jest niezwykle rzadka. Jej rozmiar jest ograniczony przez liczbę URL'i które są w rzeczywistości przedstawiane użytkownikowi. Z jednej strony, w Serwerze Ontologii każda strona jest mocno „odnośnikowa" (prawie każde ze słów na stronie jest odnośnikiem). Sugeruje to istnienie wielu identyfikatorów UID. Z drugiej jednak strony duża liczba z nich powtarza się wielokrotnie i powszechnie pojawia się na innych stronach. W wyniku tego liczba pozycji w tablicy UID jest mała – rzędu setek albo kilku tysięcy pozycji. Nasz serwer obecnie obsługuje wiele setek ludzi i przetwarza tysiące żądań dziennie bez żadnych opóźnień czy problemów z wydajnością ze strony serwera. Serwer zazwyczaj pracuje bez przerwy przez dwa tygodnie pomiędzy zaplanowanymi wyłączeniami dla aktualizacji oprogramowania bez zbytniego zużycia pamięci.

4.4 Podsumowanie

Pomimo wielu wymienionych braków w HTML/HTTP, pokazaliśmy że można użyć do konstrukcji środowiska interaktywnej edycji, którego możliwości są faktycznie dużo większe niż można było się spodziewać z prostej strony i interfejsu opartego na transakcji.

Pokazaliśmy że:

- bezstanowy HTTP może być wykorzystany do interkacji z systemami pełnostanowymi.
- prosty mechanizm (UID) oraz zastosowanie stanów zapewnia podstawę dla sesji, użytkowników i grup
- warstwa stanów umożliwia systemom wyjście poza rozproszony dostęp do współpracy pomiędzy użytkownikami
- tablica UID może być używana do przedstawienia rzadkiej grupy stron które użytkownik rzeczywiście otwiera (lub tworzy) pośród prawie nieskończonej przestrzeni dynamicznie generowanych stron.

Mechanizmy, które opisaliśmy mogą być łatwo zaadoptowane przez każdego, kto chce zapewnić dostęp do pełnofunkcjonalnych, współdzielonych aplikacji w sieci.

5. Wyniki

W tej części omówimy doświadczenia jakie zdobyliśmy korzystając i udostępniając Serwer Ontologii w sieci Internet.

Generalnie nasza próba udostępnienia naukowego oprogramowania w sieci wydaje się być bardzo udana. Właściwie przekroczyła nasze oczekiwania w niemal każdej mierze.

Byliśmy w stanie zdobyć szeroką publiczność w efektywny sposób z punktu widzenia kosztów, oraz dostarczyć jej wysokiej jakość narzędzia, dużo bardziej funkcjonalnego niż byłoby to możliwe w przypadku rozprowadzania kodu źródłowego.

Instytucje naukowe takie jak *KSL* (*Knowledge Systems Laboratory Stanford University*) mają ograniczony czas przeznaczony na pielęgnacje, dokumentowanie i rozprowadzanie oprogramowania. Generalizując, wolimy raczej inwestować nasz wysiłek w polepszanie funkcjonalności niż adaptowanie kolejnej platformy, czy specyficznego sprzętu i oprogramowania.

Ponadto użytkownicy takiego oprogramowania woleliby spożytkować czas i energie na korzystanie i ocenę oprogramowania niż na jego ściąganie, konfigurację i kompilacje. Chronimy ich także przed zakupem potrzebnego licencjonowanego oprogramowania oraz przesiadania się na platformy sprzętowe niezbędne dla naszego oprogramowania.

Kiedy rozpoczynaliśmy pracę nad Serwerem Ontologii mieliśmy nadzieję że będziemy w stanie pokonać dwa problemy:

- poprawić funkcjonalność naszego oprogramowania przy jednoczesnym zmniejszeniu kosztów utrzymania i dystrybucji,
- zwiększyć liczby użytkowników przez zmniejszenie kosztów oprogramowania,

Ograniczony interfejs utworzony w HTML już w tej chwili ma pozytywny wpływ na swój własny rozwój – jest dużo bardziej przejrzysty i wygodniejszy niż mógłby być utworzony za pomocą jakiejś innej techniki.

Użyteczna interaktywna dokumentacja jest dostępna tą samą drogą dla każdego. Serwer HTTP został łatwo rozbudowany w celu obsługi innych protokołów sieciowych takich jak np. Network GFP. Organizacja sesji została łatwo rozszerzone, by wspierać kilka innych usług sieciowych które nie są tutaj opisane (opis dostępny na http://www-ksl.stanford.edu)

Ocena wpływu jaki Serwer Ontologii miał na konkretne projekty jest trudna. Nie rozpoczęliśmy jeszcze typowej analizy danych, a same informacje w logach nie mówią dużo. Wynika to częściowo z konstrukcji, a częściowo z natury samego systemu. Chcemy aby użytkownicy czuli się pewnie jeżeli chodzi o ich prywatne dane i współdziałania z serwerem. Jest to wyjątkowo ważne dla użytkowników przemysłowych.

Nie jest też jeszcze do końca jasne jaki rodzaj informacji jest użyteczny i powinien być rejestrowany. Większość serwerów sieciowych przetrzymuje kompletny rejestr URL jakie były pobierane. Pozwala to na określenie schematów użycia, takich jak np. stwierdzenie jak często dane dokumenty są przeglądane. W przypadku Serwera Ontologii to nie jest takie proste.

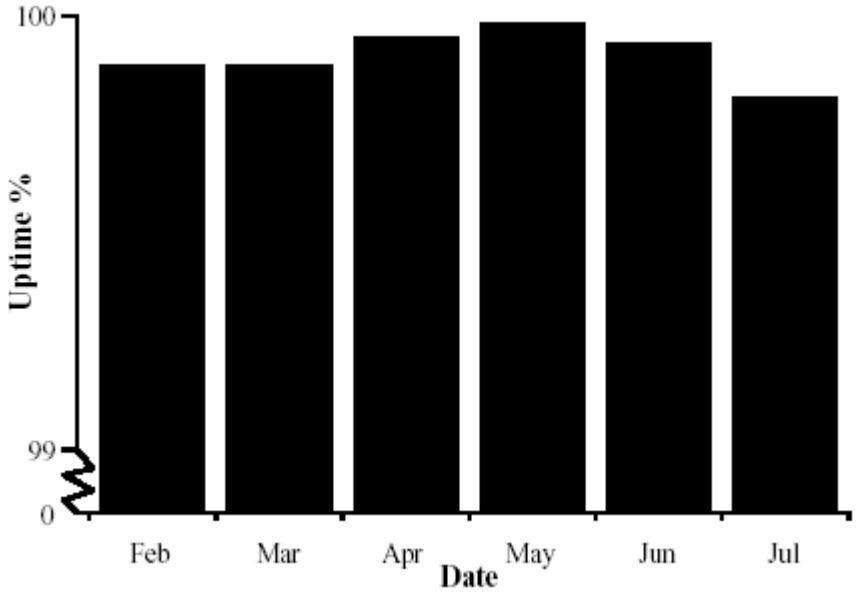
Przypominając dyskusję z sekcji 3. URLe które wchodzą i wychodzą z Serwera Ontologii kodują każde zapytanie z identyfikatorem który jest unikalny dla polecenia, użytkownika i sesji.

Rejestr tych identyfikatorów jest bezużyteczny. Ponadto, strony do których użytkownicy mają dostęp nie pochodzą z jakiegokolwiek ustalonego zbioru plików na serwerze. Użytkownicy tworzą, modyfikują i usuwają nowe obiekty w czasie pracy. W takiej nieskończonej przestrzeni obiektów, jest względnie mało powiązań między użytkownikami. Tym samym rejestracja tych obiektów jest mało interesująca.

Chociaż ciekawym mogłoby się okazać rejestrowanie wykonanych poleceń, to Serwer Ontologii wspiera na chwilę obecną ponad 500 odrębnych poleceń, a aktualna architektura tychże poleceń sprawia trudnym uzyskanie jakiegokolwiek spójnego obrazu aktywności użytkowników z takich rejestrów. Ważnym krokiem jest stworzenie pomocnego narzędzia, które pozwoli na ocen wpływu edytora ontologii na współpracę działania.

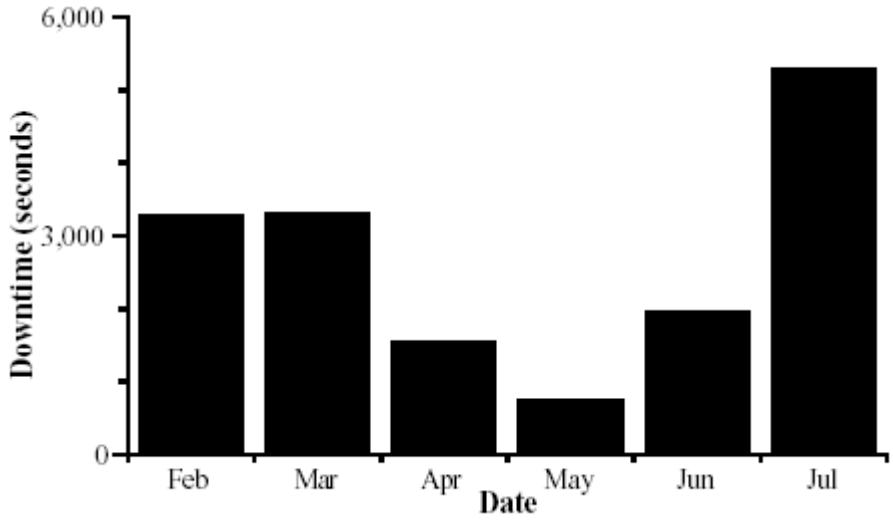
Pomimo to jesteśmy w stanie określić podstawową informację o schematach użycia i aktywności.

Serwera Ontologii jest niezawodny. Od czasu oficjalnego startu w lutym 1995 serwer był dostępny przez 99.89% czasu, jak widać na rysunku 6.



Rysunek 6. Czas kiedy serwer działał, od momentu wejścia w etap alpha.

Wliczając planowane wyłączenia poświęcone na modernizację sprzętu i oprogramowania, całkowity czas wyłączenia typowo wynosił poniżej 1 godziny na miesiąc (rysunek 7).



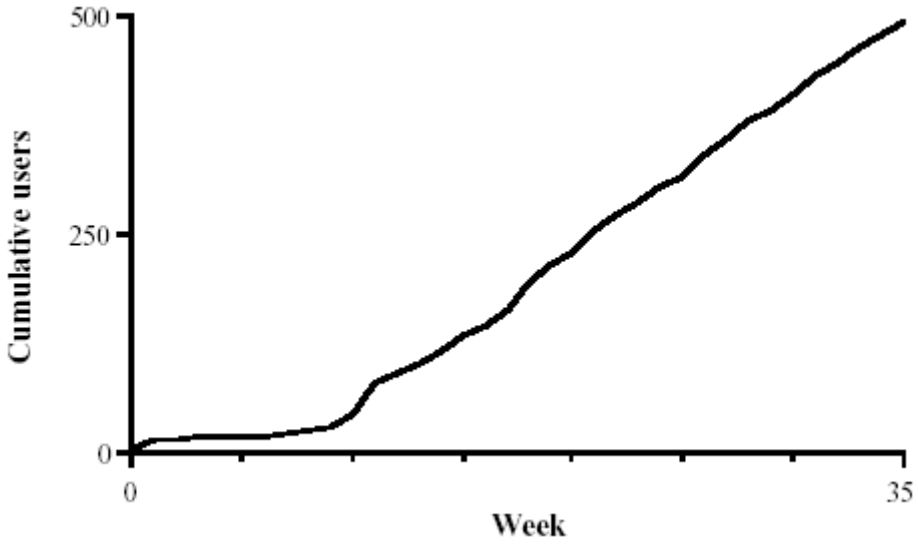
Rysunek 7. Czas kiedy serwer był wyłączony (w sekundach na miesiąc)

Wysoka niezawodność jest bardzo ważna jeżeli chcemy by użytkownicy korzystali z naszych narzędzi i usług.

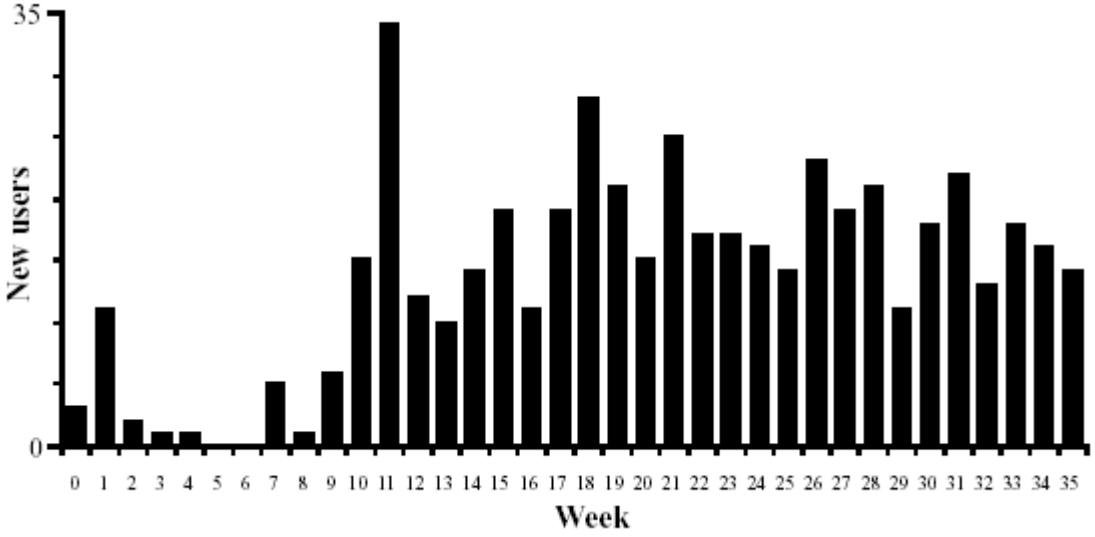
Serwer Ontologii zyskał szeroką publiczność z akceptowalnym poziomem kontynuacji stosowania. Chociaż wierzymy że ontologie stają się użyteczne w wielu dziedzinach, to jednak Serwer Ontologii nadal jest znany wśród wąskiego grona ludzi. Tym samym nie powinniśmy spodziewać się poziomu użycia podobnego do stworzonych do ogólniejszych zastosowań serwisów sieciowych jak np. Lycos.

Serwer Ontologii został ogłoszony tylko na listach dyskusyjnych osób zainteresowanych Ontolingua, Shareable Reusable Knowledge Bases, i Qualitative Reasoning.

Pomimo to, jak widać na rys 8, liczba użytkowników stale rosła. Rysunek 9 pokazuje ilość rejestrowań tygodniowo.



Rysunek 8. Liczba zarejestrowanych użytkowników stale rosła od pierwszego ogłoszenia.



Rysunek 9. Liczba nowych użytkowników wskazuje na rosnące zainteresowanie

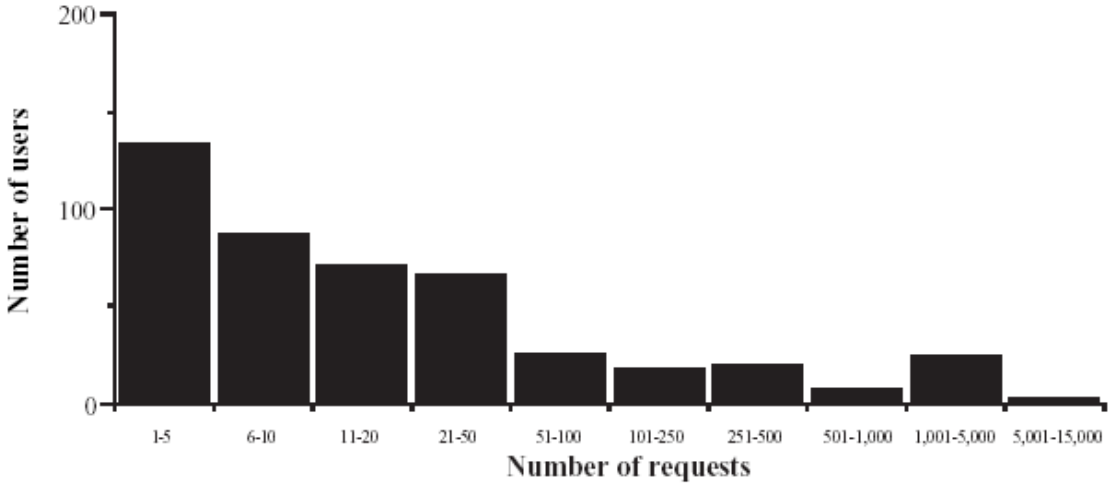
Szybki wzrost ok. 11 tygodnia jest odpowiedzią na wcześniejsze powiadomienie na listach dyskusyjnych o wejściu projektu w fazę alfa. Interesujący jest fakt, że liczba użytkowników wzrasta równym tempem od ok. 15 tygodnia. Uważamy że użytkownicy ci nie zostali przyciągnięci przez nasze ogłoszenie, raczej dowiedzieli się o nas od współpracowników lub znaleźli nasz serwer przeszukując sieć za pomocą którejś z wyszukiwarek internetowych. Ale liczba użytkowników nie pokazuje wszystkiego.

Rys 10 i 11 dostarczają trochę użytecznych informacji o charakterze użycia. Pierwszy z nich pokazuje podział użytkowników wg całkowitej liczby żądań.

Jak mogliśmy się spodziewać największa grupa użytkowników to ci, którzy po prostu „surfując" weszli na naszą stronę, wykonali mała liczbę zapytań po czym przenieśli się gdzieś indziej.

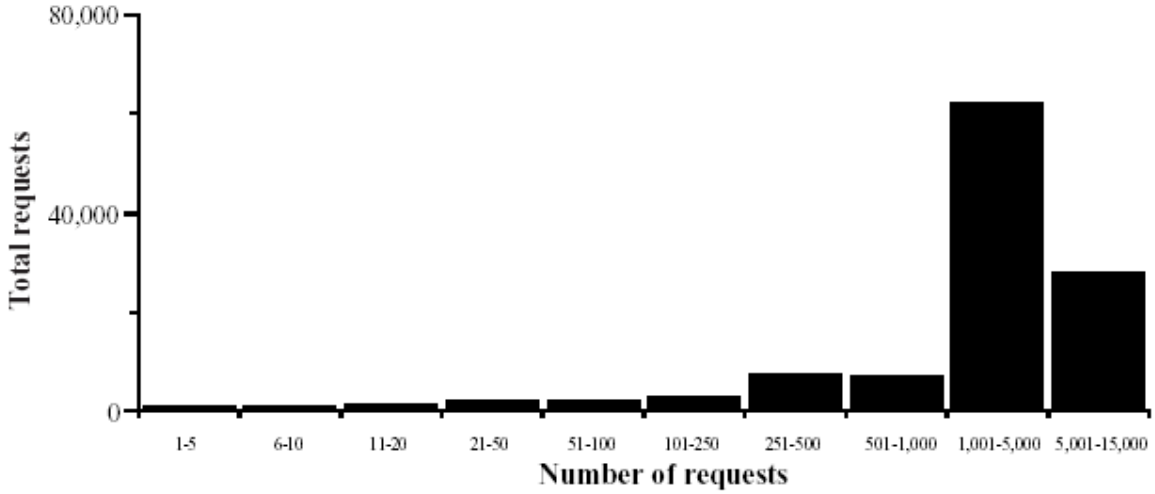
Z powodu ogólnego zastosowania serwera oraz potrzeby rejestrowania się w systemie i stworzenia konta, nowi użytkownicy muszą wykonać kilka zapytań zanim dostaną się do edytora ontologii. Następne kilka grup rozejrzało się trochę na naszej stronie. Obejrzenie ok. 50 podstron, wymaga pewnego poświęcenia czasu i energii. Ci użytkownicy oceniają system i odkrywają część jego możliwości. Ten poziom aktywności może być konsekwencją oglądania ontologii autora przez jego współpracowników i przełożonych. Następny segment to od 50 do 500 żądań. Taka ilość jest wystarczająca do skonstruowania przykładowych ontologii i zrealizowania ćwiczeń akademickich. Ci użytkownicy odkrywają znaczenie systemu i jego możliwości. Ostatecznie, są użytkownicy którzy wykonali tysiące żądań. Są to przeważnie ludzie, z których wielu realizuje poważne projekty oparte na ontologiach.

Uważamy że kształt tego wykresu powinien pozostać mniej więcej taki sam.



Rysunek 10. Użytkownicy pogrupowani wg liczby zapytań

Rysunek 11 ukazuję tą samą informację z innego punktu widzenia. Pokazuję całkowitą ilość zapytań zrealizowanych przez każdą z grup. Okazuje się że 80% żądań jest wykonywana przez 8% użytkowników.

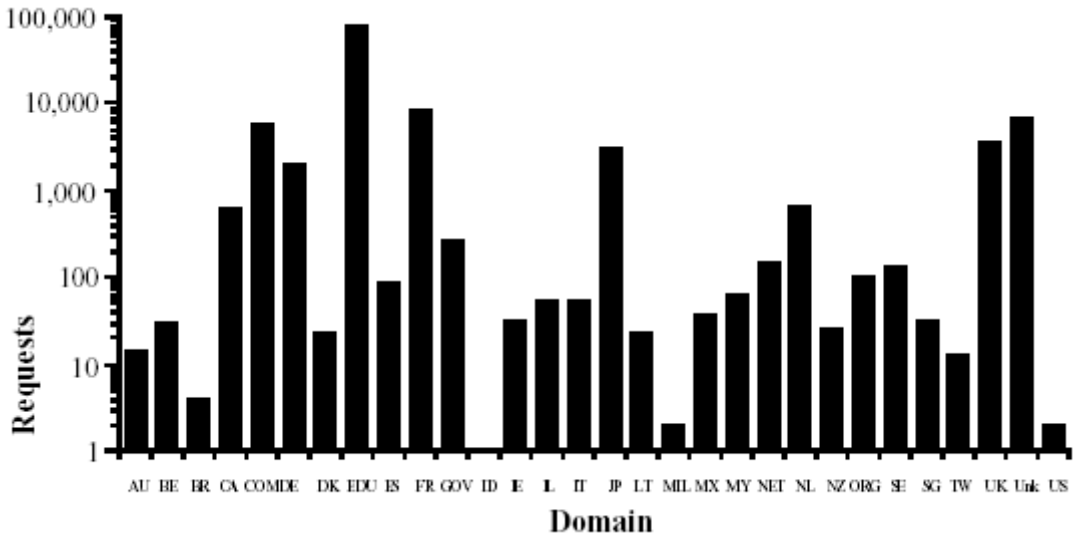


Rysunek 11.

Rysunek 12 ukazuje rozkład zapytań na najważniejsze domeny internetowe. Przytłaczająca większość zapytań pochodzi od amerykańskich domen edukacyjnych, znaczna część pochodzi od amerykańskich domen komercyjnych. Sporą grupę stanowią też domeny europejskie, brytyjskie i japońskie.

Schematy użycia są zgodne z naszymi przewidywaniami a także z przewidywaniami charakterystycznymi dla produkcji oprogramowania.

Tylko mały procent osób które wypróbowały Serwer Ontologii zakończyły swoje próby na podstawach. Reszta wypróbowała, by potem powrócić, kiedy potrzebowali umiejętności Serwera Ontologii.



Rysunek 12.

Ocenienie pracy wykonanej za pomocą Serwera Ontologii i jego wpływu na współdziałanie jest trudne.

Jest jednak jasne, że użytkownicy tworzą zaawansowane ontologie. Tematy tworzonych ontologii są zróżnicowane. Większość dojrzałych ontologii dotyczy: łączenia baz danych genomu, obrazów satelitarnych, integracji przedsięwzięć, katalogów produktów, oscyloskopów, półprzewodników, robotyki, leków, i wielu innych. Niektóre z większych ontologii, jak te dotyczące terminologii medycznej, zawierające ponad dwa tysiące definicji, zostały zaimportowane z istniejących projektów. Inne takie jak ontologia IEEE 1175, która zawiera ponad 140 definicji, została stworzona całkowicie przy użyciu edytorów ontologii.

Dodatkowo Serwera Ontologii był wykorzystywany do co najmniej dwóch projektów wymagających dostępu do ontologii w czasie rzeczywistym.

Medyczna aplikacja T-Helper jest komputerowym systemem rejestracji pojawiających się pacjentów zarażonych wirusem HIV. Aby stwierdzić czy pacjenci nadają się do testów klinicznych system korzysta z ontologii dotyczących rodzajów leków. System korzysta z Serwera Ontologii by określić czy kryteria przydatności związane z lekami zostały napotkane.

Jesteśmy bardzo zadowoleni z poziomu jaki osiągnął Serwer Ontologii. Pokazuje to że system wypełnia pewną ważną nisze zastosowań. Spotkaliśmy się z odzewem szerokiej publiczności której dostarczamy rzetelne i użyteczne narzędzia służące do budowy ontologii.

6. Podsumowanie

W tym opracowaniu zidentyfikowaliśmy kilka przeszkód które musimy pokonać zanim ontologie staną się praktycznym i pożytecznym narzędziem, a także pokazaliśmy kilka sposobów w jakie te przeszkody pokonać.

Przedstawiliśmy dołączone ontologie, umożliwiające użytkownikom błyskawiczne tworzenie nowych ontologii korzystając z już istniejących przechowywanych w repozytorium.

Opisaliśmy nasze środowisko edycyjne oparte na sieci, zintegrowane z Serwera Ontologii i zawierające dołączone ontologie. Dodatkowo aby zapewnić poszczególnym użytkownikom bogate środowisko edycyjne, serwer ten wspiera współpracę pomiędzy grupami

użytkowników i zapewnia dostęp do repozytorium ontologii. Serwer Ontologii dostarcza też bardzo istotne medium do publikacji dokumentacji.

Opisaliśmy także infrastrukturę która umożliwia Serwerowi Ontologii zapewnienie dostępu użytkownikom z całego świata, w taki sposób aby mogli oni tworzyć, modyfikować, przeglądać i eksportować ontologie korzystając z własnych przeglądarek sieciowych.

Infrastruktura którą rozwinęliśmy może być zaadoptowana do innych projektów które chcą wspierać współpracę na szeroką skalę, i rozprowadzać swoje aplikację w jak najtańszy sposób.

Ostatecznie zaprezentowaliśmy empiryczny dowód że sposób stosowania ontologii jest efektywny. Nasze narzędzia i infrastruktura która je wspiera okazała się wyjątkowo niezawodna (99.9% czasu on-line). Infrastruktura działa bardzo dobrze i w tej chwili obsługuje kilkaset użytkowników bez pogorszenia jakości obsługi. Użytkownicy byli w stanie tworzyć poważne ontologie za pomocą sieciowego edytora z wieloma setkami definicji. Kilkanaście z tych konstrukcji powstało z łącznego wysiłku wielu użytkowników z całego świata.

Tworzenie ontologii jest trudnym, czasochłonnym zajęciem. Narzędzia które stworzyliśmy ułatwiły ten proces, i pomnożyły korzyści wielu użytkowników.

Serwera Ontologii jest dostępny do publicznego użytku przez:

`http://www-ksl-svc.stanford.edu:5915/`