

**Prace Naukowe
Instytutu Informatyki Stosowanej
Politechniki Wrocławskiej**

8

BAZY DANYCH

Redakcja naukowa

Zygmunt Mazur

**Oficyna Wydawnicza
Politechniki Wrocławskiej**



Prace Naukowe
Instytutu Informatyki Stosowanej
Politechniki Wrocławskiej

8

BAZY DANYCH

Redakcja naukowa

Zygmunt Mazur



Oficyna Wydawnicza Politechniki Wrocławskiej
Wrocław 2007



Politechnika Wrocławska

Instytut Informatyki Stosowanej

Recenzent

Andrzej KWIECIEN

Opracowanie redakcyjne materiałów

Hanna MAZUR

Projekt okładki

Zygmunt MAZUR

Wszelkie prawa zastrzeżone. Żadna część niniejszej książki, zarówno w całości, jak i we fragmentach, nie może być reprodukowana w sposób elektroniczny, fotograficzny i inny bez zgody właściciela praw autorskich.

© Copyright by Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2007

OFICYNĄ WYDAWNICZĄ POLITECHNIKI WROCŁAWSKIEJ

Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

<http://www.oficyna.pwr.wroc.pl>

e-mail: oficwyd@pwr.wroc.pl

ISBN 978-83-7493-376-6

Drukarnia Oficyny Wydawniczej Politechniki Wrocławskiej

SPIS TREŚCI

<i>Wstęp</i>	5
W. Guszpit, A. Wilczek, <i>Języki zapytań strumieniowych baz danych</i>	9
P. Sokołowski, A. Wilczek, <i>Metody przetwarzania danych strumieniowych</i>	29
P. Sokołowski, A. Wilczek, <i>Techniki adaptacji i aproksymacji w przetwarzaniu strumieni danych</i>	43
Z. Mazur, P. Pawluk, <i>Problem analizy danych w sieciach społecznych</i>	59
Z. Mazur, S. Pasieczny, <i>Programowanie sieciowe w Symbianie – protokół HTTP</i>	71
Z. Mazur, S. Pasieczny, <i>Aspekty sieci GSM/GPRS w kontekście programowania w Symbianie</i>	83
J. Gruber, T. Brzozowski, <i>Modelowanie i wytwarzanie systemów o architekturze zorientowanej na usługi</i>	105
D. Szepielak, <i>Semantyczna integracja systemów informatycznych przy użyciu Web Oriented Integration Architecture</i>	121
Z. Mazur, T. Mendyk-Krajewska, <i>Walka o bezpieczeństwo systemu informatycznego</i>	149
T. Mendyk-Krajewska, Z. Mazur, <i>Bezpieczeństwo rozproszonych systemów bazodanowych w sieciach bezprzewodowych</i>	155

CONTENTS

<i>Introduction</i>	5
W. Guszpit, A. Wilczek, <i>Stream Database Query Languages</i>	9
P. Sokołowski, A. Wilczek, <i>Stream Data Processing Methods</i>	29
P. Sokołowski, A. Wilczek, <i>Adaptation and Approximation Stream Data Processing Methods</i>	43
Z. Mazur, P. Pawluk, <i>The Data Analysis Issue in the Social Network</i>	59
Z. Mazur, S. Pasieczny, <i>Network Programming in Symbian – the HTTP Protocol</i>	71
Z. Mazur, S. Pasieczny, <i>The Aspects of the GSM/GPRS Based Networks in the Symbian Programming Context</i>	83
J. Gruber, T. Brzozowski, <i>Service Oriented Modeling and Implementing of Business Processes on the Sybase Platform</i>	105
D. Szepielak, <i>Web Oriented Integration Architecture for Semantic Integration of Information Systems</i>	121
Z. Mazur, T. Mendyk-Krajewska, <i>The Battle for Information System Security</i>	149
T. Mendyk-Krajewska, Z. Mazur, <i>Security of Distributed Database Systems in Wireless Networks</i>	155

*Wielu życiowych rozbitków to ludzie,
którzy nie zdawali sobie sprawy,
jak bliscy są sukcesu,
kiedy się poddali.*

☛ Thomas Alva Edison

WSTĘP

Przekazujemy Państwu ósmy numer Zeszytu Naukowego serii „*Bazy danych*” *Prace Naukowe Instytutu Informatyki Stosowanej Politechniki Wrocławskiej*. Zeszyty tej serii wydawane są cyklicznie, co rok od 2000 roku począwszy. W niniejszym numerze zamieszczone są prace, których tematyka dotyczy:

- języków zapytań dla strumieniowych baz danych, metod przetwarzania danych strumieniowych oraz technik adaptacji i aproksymacji w przetwarzaniu strumieni danych,
- analizy danych w sieciach społecznych,
- programowania sieciowego w systemie Symbian,
- modelowania i wytwarzania systemów o architekturze zorientowanej na usługi (SOA),
- integracji systemów informatycznych za pomocą nowatorskiej architektury zwanej *Web Oriented Integration Architecture* (WOIA),
- bezpieczeństwa systemów informatycznych oraz rozproszonych systemów bazodanowych w sieciach bezprzewodowych.

System Zarządzania Strumieniową Bazą Danych (SZSBD) pożądanym jest w sytuacjach, w których nie może być użyty pasywny model przetwarzania, charakterystyczny dla tradycyjnych Systemów Zarządzania Bazą Danych (SZBD). Celem SZSBD jest przetwarzanie strumieniowej bazy danych czyli bazy, w której dane są przedstawione w postaci zbioru strumieni danych. Strumieniowe bazy danych z reguły implementują języki ciągłych zapytań opartych na SQL-u (istnieją jednak wyjątki od tej reguły, na przykład rozwiązania graficzne). Trzy pierwsze artykuły Zeszytu 8. są związane z tematyką języków zapytań dla strumieniowych baz danych, metod przetwarzania danych strumieniowych oraz technik adaptacji i aproksymacji w przetwarzaniu strumieni danych.

Analiza Sieci Społecznych (ang. *Social Network Analysis* – *SNA*) oparta jest na matematycznej teorii grafów, która dostarcza formalnych sposobów opisywania społecznych struktur sieciowych oraz sieciowych cech jednostek. Jej przedmiotem są sieci tworzone przez różnego rodzaju „obiekty” społeczne (np. ludzi, grupy ludzi,

firmy, państwa), które stanowią węzły sieci i są ze sobą połączone relacjami (między innymi o charakterze społecznym). Relacje mogą być symetryczne (na przykład komunikowanie się obiektów lub ich współpraca) albo skierowane (na przykład zapytanie do obiektu), dychotomiczne (stwierdza się tylko istnienie lub brak relacji) albo wartościowane (określa się siłę relacji – np. jak często dwie osoby ze sobą rozmawiają). Analiza sieci społecznej, w celu ekstrakcji z niej informacji, wiąże się z przetwarzaniem dużej liczby danych. To powoduje konieczność poszukiwania nowych, bardziej efektywnych metod przetwarzania i budowania struktur dostosowanych do specyfiki zagadnienia. Autorzy czwartego artykułu przedstawiają teoretyczne podstawy budowy sieci społecznych oraz metody ich analizy ze szczególnym zwróceniem uwagi na zagadnienie przetwarzania danych, omówienie struktur stosowanych do przechowywania danych oraz problemów występujących w analizie sieci społecznych.

Kolejne dwie prace związane są z programowaniem w systemie Symbian. W pracy piątej omówione są podstawy protokołu HTTP oraz sposób wykorzystania HTTP API systemu operacyjnego Symbian w urządzeniach mobilnych działających z tym właśnie systemem operacyjnym. W pracy szóstej przedstawiono i omówiono elementy architektury sieci GSM/GPRS oraz procesy zachodzące w tych sieciach wraz z przykładami kodu, które obrazują jak procesy takie powinny być przeprowadzane oraz jakie problemy są napotykanne w trakcie przeprowadzania takich procesów.

Od pewnego czasu można zauważyć coraz większy udział firmy Sybase w rozwoju metodyk i narzędzi do modelowania procesów biznesowych w systemach IT. Zbudowano zintegrowaną infrastrukturę do modelowania procesów biznesowych w skali przedsiębiorstwa lub organizacji. Dzięki wbudowaniu w tę infrastrukturę architektury zorientowanej na usługi SOA, Web serwisów oraz języków przetwarzania procesów biznesowych, Sybase znacząco rozszerzył modelowanie biznesowe w prawie wszystkich kierunkach i aspektach. W pracy siódmej niniejszego Zeszytu przedstawiono podejście firmy Sybase do zagadnień modelowania biznesowego oraz ocenę użyteczności tego podejścia.

Ostatnie lata w dziedzinie integracji systemów informatycznych przyniosły dominację rozwiązań bazowanych na *Service Oriented Architecture* (SOA) i SOAP Web Services (SOAP – ang. *Simple Object Access Protocol* – to prosty protokół komunikacyjny oparty na języku XML, umożliwiający przekazywanie wywołań zdalnych komponentów Web Services). Coraz więcej uwagi poświęca się odmianie SOA zwanej *Web Oriented Architecture* (WOA), która upraszcza tradycyjny sposób interakcji pomiędzy rozproszonymi aplikacjami w sieci poprzez wprowadzenie jednolitego interfejsu funkcjonalnego, który ułatwia proces integracji. W artykule ósmym opisano bazowaną na WOA nowatorską architekturę integracji systemów informatycznych zwaną *Web Oriented Integration Architecture* (WOIA), która w znaczący sposób rozszerza możliwości WOA jednocześnie zachowując jej najważniejszą cechę – prostotę użycia przez konsumentów usług sieciowych.

Artykuł dziewiąty dotyczy zagadnień bezpieczeństwa systemów informatycznych, przy czym autorzy kładą nacisk na złożoność problemu ochrony oraz wynikające z tego trudności w osiągnięciu wysokiego poziomu bezpieczeństwa. W pracy opisano przyczyny zagrożeń wynikające z połączenia systemu z siecią oraz poruszono wybrane problemy związane z zabezpieczaniem systemu informatycznego przed niepożądanym działaniem.

Bezprzewodowa technologia transmisji danych wykorzystywana w systemach teleinformatycznych niesie dodatkowe zagrożenia, nieznane w rozwiązaniach konwencjonalnych stosujących połączenia kablowe. Zagadnieniom tym poświęcona jest ostatnia praca zamieszczona w Zeszycie 8.

Autorzy składają podziękowanie Recenzentowi za uwagi zamieszczone w recenzji, które zostały uwzględnione w końcowych wersjach artykułów.

Zygmunt Mazur

*rumieniowa baza danych,
język zapytań,
Aurora/Borealis,
STREAM*

Wojciech GUSZPIT
Artur WILCZEK*

JĘZYKI ZAPYTAŃ STRUMIENIOWYCH BAZ DANYCH

Systemy Zarządzania Strumieniową Bazą Danych (SZSBD) stanowią alternatywę dla tradycyjnych relacyjnych bądź obiektowych Systemów Zarządzania Bazą Danych (SZBD). Tworzy się je na potrzeby aplikacji, w których nie może być użyty pasywny model przetwarzania, charakterystyczny dla tradycyjnych SZBD. System Zarządzania Strumieniową Bazą Danych stanowi aktywny element aplikacji, który na bieżąco generuje wyniki, wraz z napływem danych wejściowych. Możliwe jest to dzięki mechanizmowi rejestracji zapytań ciągłych (ang. *continuous queries*), które, w odróżnieniu od zapytań jednorazowych typowych dla tradycyjnych SZBD, wykonywane są w SZSBD stale przez określony przedział czasu. Taki sposób przetwarzania jest źródłem wymagań dla języka służącego do wyrażania zapytań ciągłych. W niniejszym artykule przedstawiono te wymagania oraz zaprezentowano wybrane, istniejące obecnie języki strumieniowe.

1. WPROWADZENIE

Mechanizm rejestracji ciągłych zapytań, jak i konstrukcja pozostałych elementów Systemu Zarządzania Strumieniową Bazą Danych (SZSBD), wynikają bezpośrednio z postaci, oraz sposobu napływu danych, na których operuje taki system. Sekwencję danych wejściowych niosącą informację o zmieniającym się stanie rzeczywistości określa się mianem *strumienia*, definiowanego jako nieskończony zbiór elementów (s, t) , gdzie s jest krotką należącą do schematu strumienia, a t należące do T jest stemplem czasowym elementu [WWW3]. Według innej definicji strumień jest sekwencją elementów $x_1, \dots, x_i, \dots, x_n$, gdzie każdy z elementów jest odczytywany tylko raz, wraz ze zwiększającym się indeksem i [WWW3], co, przy założeniu, iż indeksem jest stempel czasowy porządkujący zbiór elementów (s, t) , stanowi uzupełnienie definicji przytoczonej wcześniej.

* Instytut Informatyki Stosowanej, Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27, artur.wilczek@pwr.wroc.pl

Ze względu na sposób definiowania zapytań, języki dzielą się na deklaratywne, za pomocą których wyrażane jest jedynie co ma być wykonane bez określania sposobu wykonania, oraz proceduralne, które pozwalają definiować sposób wykonania zapytania. Wśród tradycyjnych SZBD powszechnie używanym językiem jest deklaratywny SQL, którego dialekty tworzone są również na potrzeby wielu Systemów Zarządzania Strumieniową Bazą Danych. Podstawę języka zapytań, zarówno proceduralnego, jak i deklaratywnego, stanowi zbiór operatorów możliwych do wyrażenia przy jego pomocy. Sprawą kluczową, w przypadku operatorów działających na strumieniach jest fakt, iż nie mogą one być blokujące, czyli takie, że do wygenerowania jakiegokolwiek wyniku na wyjściu, potrzebują wszystkich danych z wejścia [Babcock 2002]. Wynika to z samej własności strumienia, który z definicji jest nieskończony, co przekreśla możliwość operowania na wszystkich jego elementach. Przykładem konstrukcji o blokującym charakterze w języku SQL są funkcje agregujące takie jak SUM czy AVG pozwalające odpowiednio policzyć sumę oraz średnią z wartości krotek relacji dla pewnego atrybutu jej schematu. Naturalnie, aby wyliczyć sumę czy wartość średnią elementów konieczna jest znajomość ich wszystkich.

Adaptacja operatorów blokujących do działania na danych w postaci strumieni jest możliwa dzięki *oknu strumieni danych*, stanowiącemu kluczowe pojęcie SZSBD, które musi być wyrażalne w jego języku zapytań. Pozwala ono wybrać spośród danych napływających do strumienia pewien ich podzbiór, będący podstawą dalszych obliczeń [Golab 2003]. O konieczności uwzględnienia go podczas tworzenia języka, jak i samego SZSBD, przesadzają wymagania narzucane przez obszary zastosowań, z których każdy oczekuje od SZSBD możliwości operowania na danych z pewnego przedziału czasu, bądź też na określonej ich ilości, wyrażanej liczbą elementów strumienia. Ponadto okno stanowi środek służący aproksymacji wyników poprzez ograniczanie ilości danych, dla których są one wyznaczane [Babcock 2002].

W definicjach okien stosuje się różne sposoby określania kryteriów wyboru elementów strumienia a najczęściej stosowane to [Golab 2003]:

- *Kierunek ruchu krawędzi okna.* Okno tworzy przedział kolejnych elementów strumienia. W miarę pojawiania się nowych elementów w strumieniu każda z krawędzi przedziału może przesuwać się w kierunku coraz nowszych elementów lub przesuwać się w kierunku coraz starszych elementów ewentualnie nie zmieniać swojej pozycji. W przypadku, gdy obie krawędzie się przesuwały, okno nazywamy *ruchomym* (ang. *sliding window*), gdy tylko jedna z krawędzi się przesuwa – *jednostronnie-ruchomym* (ang. *landmark window*), gdy obie krawędzie nie zmieniają swojej pozycji – *statym* (ang. *fixed*).
- *Typ okna.* Przedział tworzący okno jest określany ze względu na liczbę elementów, jaka musi się w nim znaleźć (ang. *tuple-based window*), bądź ze względu na czas, jaki musi obejmować (ang. *time-based window*).
- *Sposób przesuwania krawędzi okna.* Określa, w jaki sposób wyznaczane są krawędzie nowego okna na podstawie poprzedniego, czyli o ile jednostek w zależ-

ności od typu okna ma się przesunąć jego krawędź, w zadanym kierunku. Dotyczy to oczywiście jedynie ruchomych krawędzi okna.

Najczęściej wykorzystywanym jest okno ruchome, o stałej szerokości, którego krawędzie przesuwają się w kierunku coraz nowszych elementów. Stała szerokość świadczy o tym, że liczba jednostek, o które przesuwane są obie krawędzie okna, jest jednokowa. W zależności od liczby tych jednostek, kolejne wyznaczane okna mogą zachodzić na siebie, przylegać do siebie, bądź też może dzielić je pewna liczba elementów.

2. PRZYKŁADOWE ZAPYTANIA CIĄGŁE

Własności wybranych języków strumieniowych zostaną omówione w kolejnym rozdziale w oparciu o przedstawiony niżej zestaw przykładowych zapytań strumieniowych. Zapytania dobrano tak, by dotyczyły aplikacji, na potrzeby których powstają Systemy Zarządzania Strumieniową Bazą Danych. W literaturze wyróżnia się najczęściej cztery obszary zastosowań SZSBD [Golab 2003]:

- sieci czujników,
- systemy analizujące ruch w sieci,
- systemy analizujące dane finansowe,
- systemy analizujące dzienniki transakcji.

W dalszej części rozdziału zaproponowano i omówiono przykładowe zapytania dla każdego z wyżej wymienionych obszarów zastosowań. W dalszej części pracy zapytania te wykorzystano do przedstawienia języków zapytań wybranych SZSBD – systemów STREAM oraz Aurora/Borealis.

2.1. ZAPYTANIE 1

Obszarem zastosowań, którego dotyczy zapytanie, są sieci czujników stosowane w systemach wspierających logistykę wojskową. Zapytanie wyrażone w języku naturalnym brzmi następująco:

Odnotowuj każdą sytuację, w której przynajmniej m żołnierzy jednocześnie przekroczy granicę k.

Źródło danych dla tego zapytania stanowi strumień `ObjectPositionReport` o następującym schemacie:

```
ObjectPositionReport(objectId, timestamp, position)
```

Do strumienia `ObjectPositionReport` trafiają dane z czujników jednostek wojskowych informujące o aktualnie zajmowanej pozycji. W powyższym schemacie `objectId` określa unikalny identyfikator jednostki (żołnierza), `timestamp` czas

dokonania pomiaru, a `position` odnotowaną pozycję. Dla uproszczenia zarówno `timestamp` jak i `position` są liczbami całkowitymi większymi od zera. W związku z tym fraza *jednocześnie przekroczy* z powyższego zapytania oznacza, iż elementy strumienia muszą mieć jednakową wartości pola `timestamp`, natomiast fraza *przekroczy granicę k*, że wartość pola `position` musi być większa bądź równa k [Abadi 2003].

2.2. ZAPYTANIE 2

Zapytanie dotyczy systemów analizujących ruch w sieci:

Dla każdego źródłowego adresu IP odnotowuj, w pięciominutowych przedziałach czasowych, liczbę pakietów oraz sumę bajtów będących wynikiem żądań protokołu HTTP.

Dane dla powyższego zapytania pochodzą ze strumienia `Packets` zawierającego pakiety generowane przez węzły znajdujące się w sieci. Jego schemat to:

```
Packets(srcIP, srcPort, destIP, destPort, len, id,
        timestamp)
```

Zapytanie zaczerpnięto z [WWW2]. Dla uproszczenia, schemat strumienia został uproszczony. Atrybuty `srcIP` i `srcPort` powyższego schematu określają odpowiednio adres IP oraz port nadawcy pakietu, `destIP` i `destPort` adres IP i port odbiorcy pakietu, `len` długość pakietu w bajtach, `id` identyfikator pakietu, a `timestamp` czas, w którym pakiet został zarejestrowany.

Żądania protokołu HTTP identyfikowane są z portem 80 odbiorcy. Oczywiście nie jest to jedyny port, na którym może operować protokół HTTP, jednak ponieważ jest to ustawienie domyślne i w przeważającej liczbie przypadków rezerwowane dla HTTP, to ograniczono się do zliczania pakietów wysyłanych właśnie na ten port.

2.3. ZAPYTANIE 3

Kolejnym obszarem zastosowań, dla którego zaproponowano zapytanie są systemy analizujące dane finansowe. Zapytanie pozwala śledzić poczynania inwestorów dysponujących dużą ilością akcji, w celu wykrycia działań będących podstawą analizy technicznej Hammer discovery [Lerner 2003]:

Na bieżąco wskazuj inwestorów, którzy w przeciągu czasu t wyrazili chęć kupna lub sprzedaży więcej niż m akcji któreś ze spółek.

Strumieniem, z którego czerpane są dane dla powyższego zapytania jest strumień `Ticks` o schemacie:

Ticks(sID, MMID, price, volume, type, timestamp)

Atrybuty powyższego schematu oznaczają kolejno: identyfikator spółki, identyfikator inwestora, cenę, liczbę akcji, typ rekordu, czas rejestracji rekordu. Typ rekordu (atrybut type) może przyjmować trzy różne wartości: 'bid' identyfikującą rekord z ofertą kupna; 'ask' identyfikującą rekord z ofertą sprzedaży oraz 'trade' identyfikującą rekord z dokonaną transakcją kupna. W związku z tym spośród wszystkich elementów strumienia do realizacji zapytania wybierane będą te, których wartością atrybutu type będzie 'bid' bądź 'ask'.

2.4. ZAPYTANIE 4

Kolejne zapytanie dotyczy zastosowań systemów strumieniowych do analizy dzienników transakcji. Zadanie, które należy zrealizować, przedstawia się następująco:

Na bieżąco informuj o stacji przekąźnikowej, przy pomocy której abonent legitymujący się numerem n najczęściej wykonuje połączenia, podając jej udział procentowy wśród wszystkich połączeń realizowanych przez tego abonenta.

Naturalnie chodzi tutaj o abonenta operatora sieci komórkowej. Aby zapytanie nieco skomplikować przyjmujemy, iż operator, oprócz swoich stacji przekąźnikowych, wykorzystuje również stacje konkurenta. W związku z tym dane do obliczeń dla tego zapytania pochodzą z dwóch następujących strumieni:

CallRecords1(subscriberNo, destinationNo, startTime,
endTime, recorderId, timestamp)

CallRecords2(subscriberNo, destinationNo, startTime,
endTime, recorderId, timestamp)

Atrybut subscriberNo oznacza numer abonenta, destinationNo numer jego rozmówcy, startTime moment rozpoczęcia rozmowy, endTime moment zakończenia rozmowy, recorderId identyfikator stacji przekąźnikowej, timestamp moment zarejestrowania rozmowy. W zapytaniu pomijane będą długości rozmów. Dla ustalenia rankingu stacji przekąźnikowych zliczana będzie jedynie liczba rekordów dla każdej z nich.

3. SYSTEM STREAM

System STREAM jest Systemem Zarządzania Strumieniową Bazą Danych, nad którym prace prowadzone były do stycznia 2006 roku na Uniwersytecie Stanford w Stanach Zjednoczonych. Szereg prac, które powstały podczas jego rozwoju stano-

wią bogate źródło wiedzy na temat strumieniowych baz danych. Priorytetem autorów systemu było stworzenie SZSBD ogólnego zastosowania [Motwani 2003], w opozycji do większości projektów prowadzonych w dziedzinie przetwarzania danych strumieniowych, które koncentrują się zazwyczaj na jednym obszarze zastosowań.

Zapytania systemu STREAM definiuje się w języku CQL (ang. *Continuous Query Language*) będącym językiem deklaratywnym zbudowanym na bazie języka SQL [Arasu 2002].

Oprócz strumienia, dopuszczalne źródło danych dla operatorów języka zapytań stanowi również relacja, określana jako odwzorowanie każdej chwili czasu t należącej do T , w skończony, ale nieograniczony wielozbiór krotek należących do schematu relacji T oznacza tę samą dziedzinę czasu, o której wspomniano podając definicję strumienia na początku niniejszej pracy. Takie przedstawienie relacji nie pokrywa się z tym, znanym z relacyjnych baz danych, któremu odpowiada sama wartość odwzorowania – $R(t)$ zwanego relacją chwilową relacji R .

Język CQL pozwala definiować jedynie ruchome okno o stałej szerokości, którego krawędzie przesuwają się w stronę coraz nowszych elementów. Dostępne są zarówno okna ilościowe jak i czasowe. Składnia języka pozwala również określać sposób przesuwania krawędzi okna poprzez parametr – liczbę całkowitą większą od zera dla okna ilościowego lub okres dla okna czasowego.

Ze względu na rodzaj źródeł danych stanowiących wejście i wyjście operatorów, w języku CQL wyróżniono ich rodzaje okien zwracające:

- relację na podstawie strumienia (strumień-relacja),
- relację z relacji (relacja-relacja),
- strumień z relacji (relacja-strumień).

Język CQL korzysta z konstrukcji języka SQL do wyrażania operatorów typu relacja-relacja oraz okien czasowych i ilościowych do wyrażenia operatorów typu strumień-relacja. Specjalny zestaw operacji umożliwia uzyskiwanie strumieni danych z relacji.

Okna języka CQL zdefiniowano w oparciu o pojęcie ruchomego okna ze stałą szerokością okna, oraz przesuwaniem krawędzi w kierunku coraz nowszych elementów. Okna budowane są w oparciu o liczbę (ang. *tuple-based*) bądź znacznik czasu (ang. *time-based*) elementów strumienia. Wprowadzono także okna grupujące (ang. *partitioned*) pozwalające tworzyć okna dla grup elementów strumienia utworzonych na podstawie wartości atrybutów schematu strumienia, analogicznie do operatora GROUP BY w języku SQL.

Język SQL wykorzystywany jest do specyfikacji operacji na relacjach chwilowych. W dostępnych publikacjach autorzy nie znaleźli jaki zakres wyrażań języka SQL jest dostępny w języku CQL, jednak analizując kod źródłowy systemu STREAM [WWW1] można znaleźć informacje o: projekcji, selekcji, agregacji i grupowaniu, różnicy zbiorów, sumie zbiorów, eliminacji duplikatów, iloczynie kartezyjańskim.

Przekształcenia relacja-strumień dostępne są poprzez operatory: *Istream*, *Dstream*, oraz *Rstream*. Operator *Istream* (*insert stream*) umieszcza w strumieniu jedynie nowe elementy czyli takie, które nie występowały w relacji chwilowej poprzedzającej aktualnie analizowaną. Operator *Dstream* (*delete stream*) umieszcza w strumieniu elementy usunięte z relacji, a zatem takie, które nie wchodziły w skład relacji chwilowej aktualnie analizowanej, a występowały w bezpośrednio ją poprzedzającej. Operator *Rstream* (*relation stream*) natomiast umieszcza w strumieniu wszystkie elementy aktualnie analizowanej relacji chwilowej, w efekcie czego dany element może trafiać do strumienia wielokrotnie z różnymi znacznikami czasu.

Poniżej przedstawiono omówione wcześniej przykładowe zapytania, wyrażone w języku CQL, ilustrujące sposób użycia wyżej opisanych operatorów. Ponieważ w systemie STREAM znacznik czasu nie należy do schematu strumienia i zapytanie nie może w sposób bezpośredni się do niego odwoływać, wykluczono atrybut *timestamp* ze schematów strumieni wykorzystywanych w prezentowanych zapytaniach, nadając mu jednocześnie funkcje znacznika czasu.

3.1. ZAPYTANIE 1

Poniżej przedstawiono treść pierwszego przykładowego zapytania ciągłego omówionego w poprzednim rozdziale wyrażoną w języku zapytań systemu STREAM. Przyjęto pewne wartości parametrów zapytania m i k .

Dla $m = 40$, $k = 10$:

```
SELECT Rstream(num)
FROM (SELECT Count(Distinct objectId) as num
      FROM ObjectPositionReport[Now]
      WHERE position >= 10)
WHERE num >= 40
```

Wynikiem podzapytania jest relacja, której każda relacja chwilowa zawiera jedną krotkę z wartością *num*, informującą o liczbie żołnierzy znajdujących się w danym momencie czasu (konstrukcja *Now*) na pozycji przekraczającej $k = 10$. W efekcie wykonania zapytania zewnętrznego, z relacji będącej wynikiem podzapytania, wybrane są krotki, których wartość pola *num* jest większa bądź równa $m = 40$ i wstawione są do strumienia wyjściowego.

3.2. ZAPYTANIE 2

Treść kolejnego przykładowego zapytania ciągłego wyrażona w języku CQL może mieć następującą postać:

```
SELECT Rstream(srcIP, Sum(len), Count(*))
FROM Packets[Range 5 Minutes Slide 5 Minutes]
WHERE destPort = 80
GROUP BY srcIP
```

W klauzuli FROM co 5 minut tworzona jest relacja chwilowa zawierająca krotki reprezentujące elementy strumienia Packets z ostatnich 5 minut. Spośród nich wybierane są krotki, których atrybut destPort przyjmuje wartość 80, co, przy założeniach zapytania, oznacza, iż reprezentują one żądania protokołu HTTP. Dla tak wybranych krotek tworzone są grupy ze względu na wartość atrybutu srcIP, w ramach których obliczane jest Sum(len) i Count(*), reprezentujące odpowiednio łączną sumę bajtów i liczbę pakietów. Krotki tak utworzonej relacji chwilowej, zawierające informacje o adresie IP, sumie bajtów i łącznej liczbie pakietów wysłanych na ten adres IP w wyniku żądań HTTP, trafiają do strumienia wyjściowego.

3.3. ZAPYTANIE 3

Zapytanie dotyczące analizy danych finansowych wyrażone w języku CQL, przy wskazanych wartościach parametrów, może mieć następującą postać:

Dla $t = 2$ Hours (2 godziny), $m = 1000$:

```
SELECT Istream(MMID, sId, type)
FROM Ticks[Range 2 Hours Slide 1 Minute]
WHERE type = 'bid' OR
       type = 'ask'
GROUP BY MMID, sId, type
HAVING Sum(volume) > 1000
```

W zapytaniu tym ze strumienia wejściowego Ticks tworzona jest relacja, której relacje chwilowe, tworzone co minutę, zawierają elementy, które napłynęły do tego strumienia w przeciągu ostatnich dwóch godzin. Z każdej relacji chwilowej wybierane są krotki zawierające wartość 'bid' lub 'ask' dla atrybutu type. W ramach tak utworzonego wielozbioru tworzone są grupy, z których każda zawiera krotki o jednakowych wartościach atrybutów MMID, sId, type. Sumując wartości atrybutu volume w każdej z grup uzyskuje się informacje: (1) ile akcji danej spółki chciał kupić dany inwestor; (2) ile akcji danej spółki chciał sprzedać dany inwestor. Każda taka informacja reprezentowana jest przez krotkę. Spośród nich, w strumieniu wyjściowym, umieszczane są te, w których wspomniana suma przekracza $m = 1000$. Wykorzystanie operatora Istream sprawia, iż informacja o przekroczeniu sumy $m = 1000$ dla określonych MMID, sId, type trafi do strumienia wyjściowego i nie

będzie powielana przez okres, w którym suma utrzyma się na poziomie przewyższającym $m = 1000$.

3.4. ZAPYTANIE 4

Kolejne przykładowe zapytanie ciągłe, przy wskazanych wartościach parametrów można w systemie STREAM wyrazić następująco:

Dla $n = '+48600601602'$:

```
RecordCount(recorderId, recCount):
  SELECT recorderId, Count(*)
  FROM (SELECT recorderId
        FROM CallRecords1[Rows Unbounded]
        WHERE subscriberNo = '+48600601602'
        UNION
        SELECT recorderId
        FROM CallRecords2[Rows Unbounded]
        WHERE subscriberNo = '+48600601602')
  GROUP BY recorderId)

SELECT Istream(rc.recorderId,
               rc.recCount,
               (rc.recCount / rAll.total) * 100)
FROM RecordCount rc,
   (SELECT Sum(recCount) AS "total"
    FROM RecCount) rAll
WHERE rc.recCount = (SELECT Max(recCount)
                     FROM RecordCount)
```

Do realizacji tego zapytania wykorzystano możliwość definiowania w języku CQL zapytań nazwanych. Wynikiem działania RecordCount jest relacja zawierająca sumę rekordów dla każdej stacji przekąźnikowej, przy pomocy której abonent o numerze $n = '+48600601602'$ wykonywał połączenia odnotowane w strumieniach CallRecords1 i CallRecords2. Należy zwrócić uwagę, że do utworzenia każdej relacji chwilowej relacji będącej wynikiem RecordCount brane są wszystkie dotychczasowe elementy strumieni, w wyniku użycia konstrukcji Rows Unbounded do definicji okna.

We właściwym zapytaniu, wskazującym poszukiwaną stację przekąźnikową, RecordCount jest wykorzystywane trzykrotnie: (1) w klauzuli WHERE wybierana jest maksymalna występująca wartość dla ilości połączeń; (2) w klauzuli FROM obli-

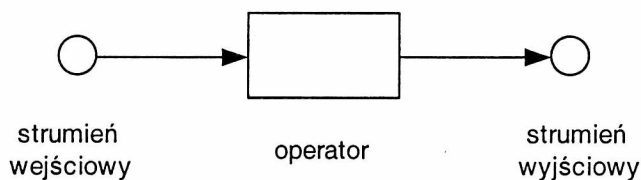
czana jest łączna liczba połączeń realizowana za pośrednictwem wszystkich stacji przekaźnikowych – r_{All} ; (3) w klauzuli WHERE z relacji rc , zdefiniowanej w klauzuli FROM, wybierane są stacje przekaźnikowe o maksymalnej ilości połączeń. Ponieważ w klauzuli FROM podane są dwie relacje (rc i r_{All}), obliczany jest ich iloczyn kartezjański, w efekcie czego w klauzuli SELECT możliwe jest określenie udziału procentowego wybranej stacji wśród wszystkich połączeń realizowanych przez abonenta. Użycie operatora Istream powoduje, iż w strumieniu wyjściowym wystąpi element tylko wtedy, jeśli jakaś stacja wyprzedzi tę wcześniej podaną, bądź zmienia się statystyki dotyczące ilości połączeń lub udziału procentowego.

4. SYSTEM AURORA/BOREALIS

Aurora/Borealis to projekt Systemu Zarządzania Strumieniową Bazą Danych realizowany wspólnie przez Uniwersytet Brandeis, Uniwersytet Brown oraz Instytut Technologii w Massachusetts (MIT). Obecnie właściwą nazwą tworzonego systemu jest Borealis. Stanowi on rozproszoną wersję wcześniej rozwijanego systemu Aurora, w ramach którego opracowano między innymi proceduralny język zapytań strumieniowych.

Obszarem zastosowań, dla którego twórcy projektowali Aurorę/Borealis są sieci czujników [Golab 2003], jednak efekt ich pracy z powodzeniem służyć może jako system ogólnego zastosowania [Jin 2006].

Język zapytań systemu Aurora/Borealis jest proceduralny. Zapytanie tworzy skierowany graf bez cykli. Wierzchołki grafu nie będące końcem żadnej krawędzi reprezentują strumień wejściowy, natomiast wierzchołki nie będące początkiem żadnej krawędzi to strumień wyjściowy zapytania. Wierzchołki będące zarówno początkiem jak i końcem jakiejś krawędzi reprezentują operatory. Krawędzie grafu przedstawiają przepływ danych pomiędzy wierzchołkami. Definiowanie zapytania polega na budowie takiego grafu przy pomocy interfejsu graficznego. Możliwie najprostsze zapytanie zostało przedstawione na rysunku 1. Reprezentacja grafu zapisywana jest w postaci dokumentu XML [Borealis 2006].



Rys. 1. Proste zapytanie w języku systemu Aurora/Borealis

Podobnie jak w systemie STREAM, również w systemie Aurora/Borealis możliwe jest definiowanie jedynie ruchomego okna o stałej szerokości, którego krawędzie

przesuwają się w stronę coraz nowszych elementów. Podstawą budowy okna może być liczba elementów lub dowolny atrybut schematu strumienia, na przykład jawnie uwzględniony w nim znacznik czasu.

Operatory języka zapytań operują wyłącznie na strumieniach danych zgodnych z definicją podaną we wstępie opracowania. Wprowadzono także mechanizm punktów dołączeniowych (ang. *connection point*), pod które można podstawiać tabele reprezentujące relacje, definiowane jak w relacyjnych bazach danych [Abadi 2003]. Twórcy systemu Aurora/Borealis wyróżnili dwa typy operatorów:

- operatory niewrażliwe na uporządkowanie elementów (ang. *order-agnostic*),
- operatory wrażliwe na uporządkowanie elementów (ang. *order-sensitive*).

Wyodrębnienie powyższych typów operatorów jest bezpośrednią konsekwencją: (1) rezygnacji z założenia, że elementy napływają do operatorów w odpowiedniej kolejności; (2) umożliwienia budowy okna na podstawie dowolnego atrybutu schematu strumienia.

Operatory niewrażliwe na uporządkowanie elementów mogą zawsze przetworzyć elementy w takiej kolejności w jakiej napływają [Abadi 2003]. Wyróżniono trzy operatory tego typu: *Filter*, *Map* oraz *Union*.

Operator *Filter* realizuje zadania operatora selekcji znanego z algebry relacji [Ramakrishnan 2003]. Operator *Map* jest generalizacją operatora projekcji algebry relacji. Wartości atrybutów elementu strumienia wyjściowego obliczane są przez funkcje, których parametrem jest odwzorowywany element strumienia wejściowego. Operator *Union* łączy dwa lub więcej strumieni o jednakowych schematach w jeden, będąc tym samym odpowiednikiem operatora obliczającego sumę zbiorów w algebrze relacji.

Operatory wrażliwe na uporządkowanie elementów wymagają podania *specyfikacji uporządkowania* (ang. *order specification*) określającej oczekiwaną kolejność, w jakiej napływać będą elementy strumienia. Definiowana jest ona jako: $\text{Order}(\text{On } A, \text{Slack } n, \text{GroupBy } B_1, \dots, B_m)$. Parametr *On* określa atrybut *A* schematu strumienia, według którego rozpatrywane jest uporządkowanie, natomiast *GroupBy* zawiera listę $B_1, \dots, B_m$ atrybutów schematu strumienia, których zestawy wartości wyznaczają podział elementów na grupy (elementy strumienia posiadające jednakowe wartości dla wszystkich tych atrybutów znajdują się w tej samej grupie). W ramach grupy dopuszczalne jest, aby kolejny trafiający do niej element *t* poprzedzało co najwyżej *n* (parametr *Slack*) elementów z wartością atrybutu *A* większą od *t.A*. Niespełnienie tego warunku sprawia, iż element *t* jest ignorowany i nie bierze udziału w dalszym przetwarzaniu¹. Oczywiście w przypadku, gdy lista atrybutów *GroupBy* jest pusta, tworzona jest tylko jedna grupa, do której trafiają wszystkie elementy strumienia. W przeciwnym razie tworzonych jest tyle grup, ile różnych zestawów wartości atrybutów $B_1, \dots, B_m$ znajduje się wśród napływających elementów. Podstawienie pod pa-

¹ Wyjątek stanowi operator *BSort*. W jego przypadku element nie jest ignorowany i bierze udział w dalszym przetwarzaniu.

parametr Slack wartości zero, pociąga za sobą ignorowanie wszystkich elementów, których wartość atrybutu A jest mniejsza od wartości tego atrybutu dla niezignorowanego poprzednika.

W języku zapytań systemu Aurora/Borealis występują cztery operatory wrażliwe na uporządkowanie elementów: BSort, Aggregate, Join, Resample.

Operator BSort daje na wyjściu strumień zawierający posortowany ciąg elementów strumienia wejściowego. Jedynym parametrem operatora jest specyfikacja uporządkowania, która jednak nie jest stosowana zgodnie z opisem przedstawionym wcześniej, ponieważ elementy strumienia wejściowego w żadnym przypadku nie są ignorowane. Określa ona jedynie rozmiar bufora (parametr Slack), jaki zostanie utworzony dla każdej z grup elementów (parametr GroupBy), w ramach którego sortowane będą elementy. I tak dla $\text{Slack}=n$, rozmiar bufora wynosi $n+1$.

Operator Aggregate umożliwia wykonanie funkcji agregującej na ruchomym oknie danych strumienia wejściowego. Funkcja ta może być predefiniowaną funkcją agregującą znaną z języka SQL [Ramakrishnan 2003] – Avg, Count, Sum, Max, Min, bądź też *funkcją zdefiniowaną przez użytkownika* (ang. *User Defined Function*) podczas dodawania operatora do zapytania. Na UDF składają się trzy metody: `init()`, `incr(t)` oraz `final()` napisane w języku przypominającym język programowania C. Pierwsza z nich wywoływana jest przy inicjalizacji okna, kiedy określany jest jego stan początkowy. Druga – `incr(t)` – wywoływana jest w celu aktualizacji stanu okna, dla każdej krotki t do niego trafiającej. Trzecia natomiast ma za zadanie zwrócić wynik funkcji na podstawie stanu okna, gdy jest ono zamykane.

Operator Join pozwala wykonać złączenie dwóch strumieni wejściowych w jeden strumień wyjściowy, o schemacie zawierającym wszystkie atrybuty schematów strumieni wejściowych.

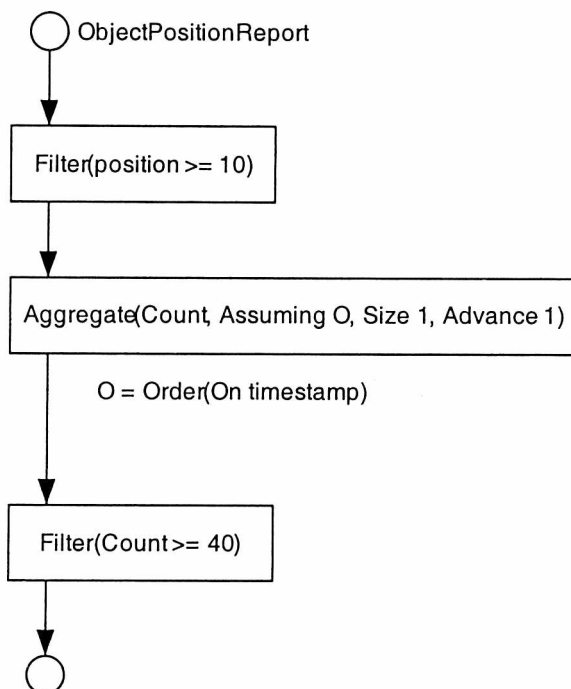
Operator Resample dla każdego elementu ze strumienia wejściowego S1 tworzy element strumienia wyjściowego, na podstawie pewnego okna strumieni danych drugiego strumienia wejściowego S2. Jest to operator, który, w odróżnieniu od wcześniej opisanych, nie posiada swojego odpowiednika w języku SQL. Pozwala on jawnie realizować zadanie synchronizacji strumieni danych, szczególnie przydatne, gdy różnią się one znacznie tempem napływu elementów [Abadi 2003].

Kolejne podrozdziały przedstawiają zapis w języku systemu Aurora/Borealis, przykładowych zapytań, omówionych w podrozdz. 2.1–2.4.

4.1. ZAPYTANIE 1

Na rysunku 2 przedstawiono graf dla pierwszego przykładowego zapytania ciągłego omówionego w podrozdziale 3.1 wyrażony w języku zapytań systemu Aurora. Przyjęto pewne wartości parametrów zapytania m i k .

Dla $m = 40$, $k = 10$:



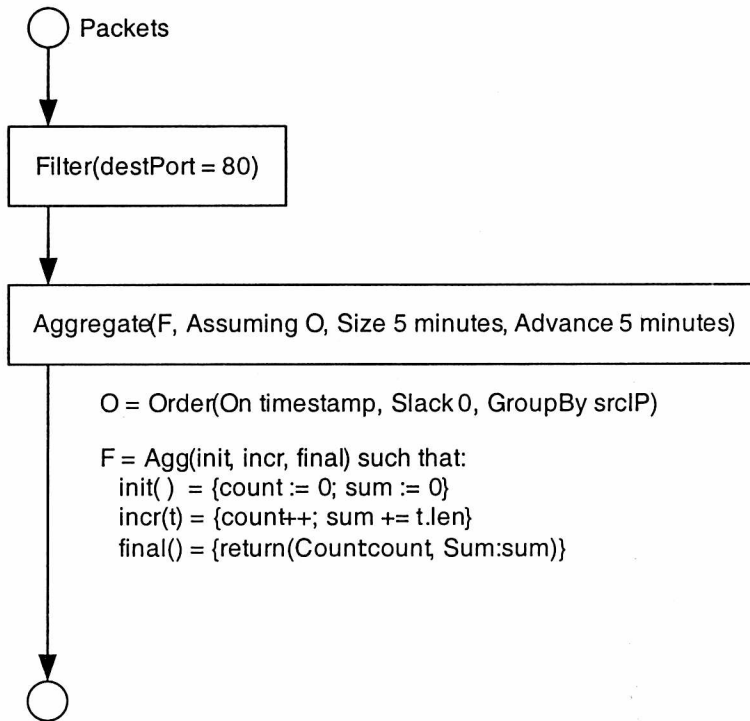
Rys. 2. Graf dla zapytania 1

Operator *Filter* wybiera żołnierzy, których pozycja jest większa, bądź równa 10. Następnie, dla tak wybranych żołnierzy, operator *Aggregate* tworzy okno danych zawierające elementy o jednakowych wartościach atrybutu *timestamp* i zwraca ich liczbę wraz z wartością *timestamp*, dla której liczba została obliczona. Na koniec operator *Filter*, wybiera spośród elementów strumienia wyjściowego operatora *Aggregate* te, których wartość pola *Count*, zawierającego liczbę elementów dla danego *timestamp* jest większa lub równa 40.

Parametr *Slack* specyfikacji uporządkowania operatora *Aggregate* wynosi zero, ponieważ oczekuje się, iż elementy będą napływać ze strumienia wejściowego *ObjectPositionReport* uporządkowane według atrybutu *timestamp*.

4.2. ZAPYTANIE 2

Graf kolejnego przykładowego zapytania ciągłego (podrozdz. 3.2) wyrażony w języku systemu Aurora może mieć postać przedstawioną na rysunku 3:



Rys. 3. Graf zapytania 2

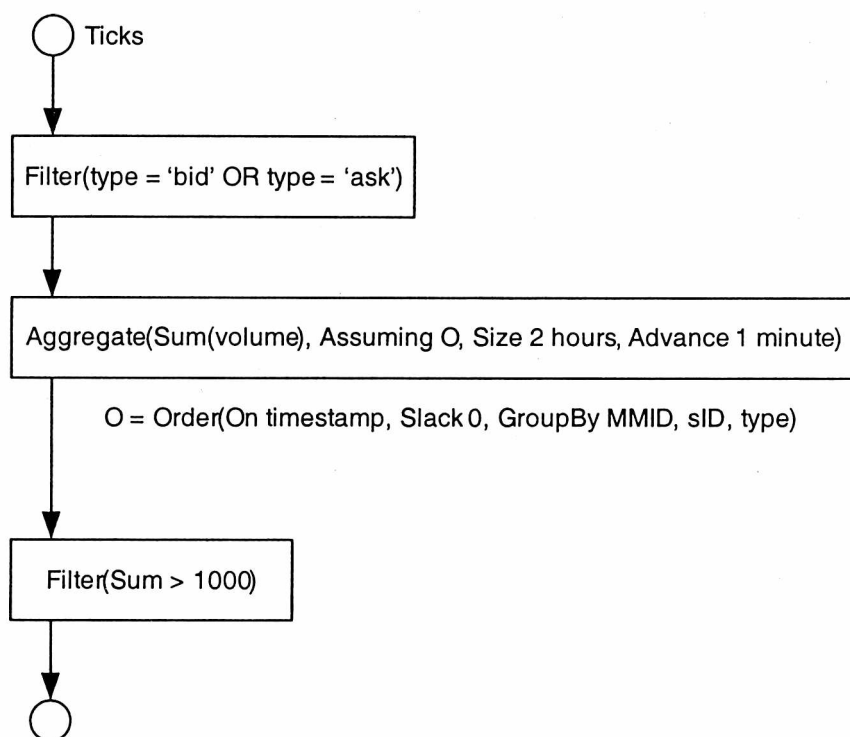
Operator `Filter` wybiera ze strumienia wejściowego `Packets` elementy o wartości pola `destPort` równej 80. Strumień zawierający te elementy stanowi wejście operatora `Aggregate`, w którym funkcja agregująca zdefiniowana przez użytkownika oblicza sumę wartości oraz liczbę elementów o jednakowych wartościach atrybutu `srcIP`, wchodzących w skład pięciominutowego okna utworzonego na podstawie wartości atrybutu `timestamp`.

Podobnie jak w przypadku poprzedniego zapytania parametrowi `Slack` przypisane jest zero, co oznacza, iż oczekiwany jest napływ elementów o niemalejących wartościach atrybut `timestamp`.

4.3. ZAPYTANIE 3

Zapytanie dotyczące analizy danych finansowych wyrażone w języku systemu Aurora, przy wskazanych wartościach parametrów, może mieć następującą postać:

Dla $t = 2$ Hours (2 godziny), $m = 1000$:



Rys. 4. Graf zapytania 3

Przy pomocy operatora `Filter` wybierane są ze strumienia wejściowego elementy o wartości "bid", lub "ask" atrybutu `type`. Następnie operator `Aggregate` sumuje wartości atrybutu `volume` w grupach elementów o jednakowych wartościach atrybutów: `MMID`, `sId` oraz `type`, w ramach okna o szerokości dwóch godzin, utworzonego w oparciu o atrybut `timestamp`. Z wartości parametru `Advance` wynika, że kolejne okno jest przesunięte w stosunku do poprzedniego o jedną minutę do przodu – najmniejsza wartość atrybutu `timestamp` w oknie następującym jest o minutę większa od najmniejszej wartości w oknie poprzedzającym. Ostatni operator zapytania – `Filter` – wybiera ze strumienia wyjściowego `Aggregate` te elementy, których wartość atrybutu `Sum`, zawierająca zsumowaną liczbę akcji dla elementów z grupy reprezentowanej przez pozostałe atrybuty, jest większa od 1000.

4.4. ZAPYTANIE 4

Kolejne przykładowe zapytanie ciągłe (podrozdz. 3.4) można w systemie Aurora, dla $n = \text{'+48600601602'}$, wyrazić tak jak na rysunku 5.

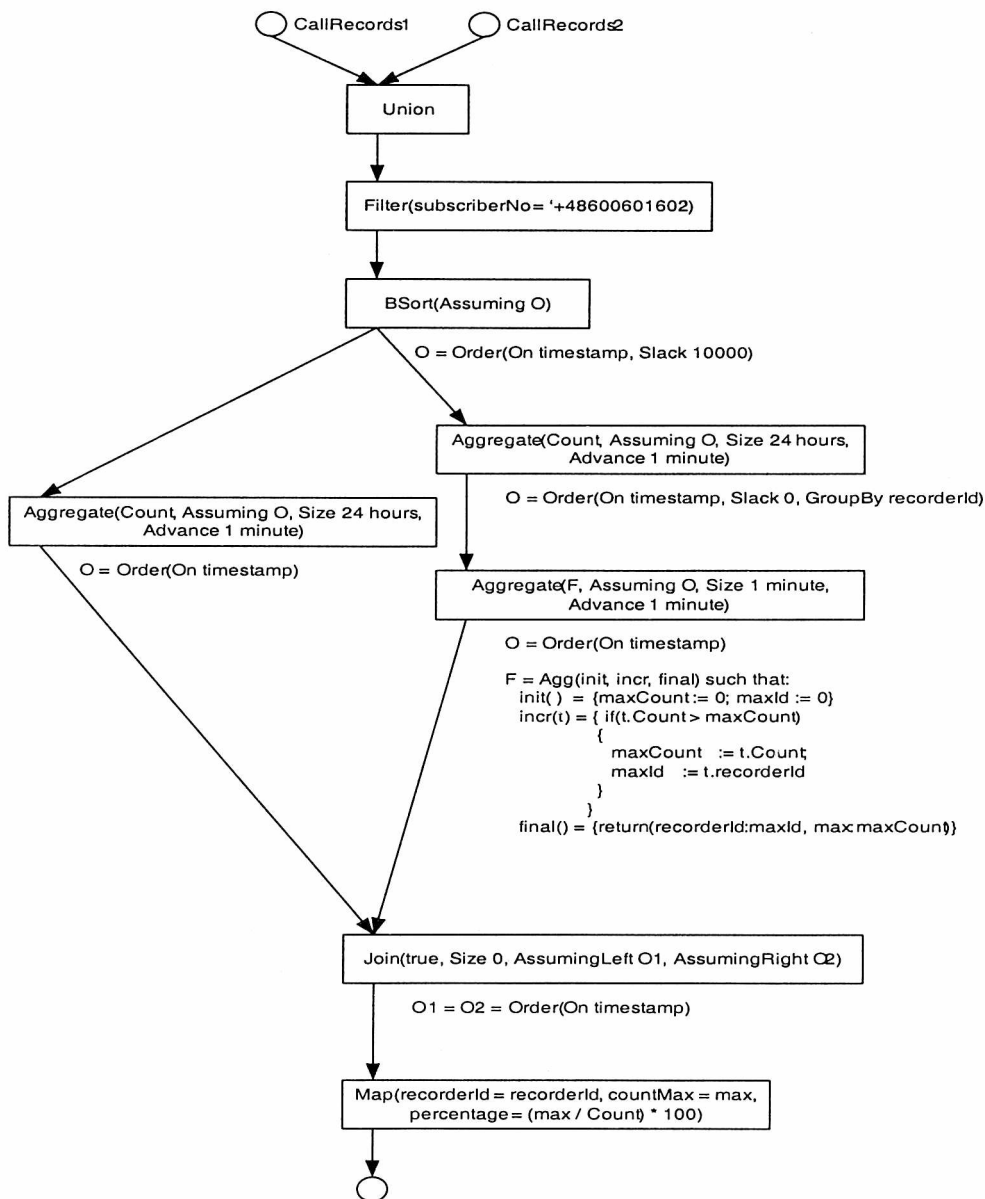
Ponieważ język zapytań systemu Aurora/Borealis nie pozwala zdefiniować nieograniczonego okna, zapytanie to nie może być zrealizowane w oryginalnej postaci. Poniższy zapis zakłada, iż należy rozpatrywać jedynie połączenia z ostatnich 24 godzin.

Strumienie `CallRecords1` i `CallRecords2` są łączone w jeden przez operator `Union`. Z tak utworzonego strumienia operator `Filter` wybiera jedynie te elementy, które reprezentują rozmowy wykonywane przez abonenta o numerze "+48600601602". Ponieważ dane o rozmowach mogą nie napływać do strumieni wejściowych synchronicznie, operator `BSort` porządkuje je według wartości atrybutu `timestamp` w obrębie bufora mieszczącego 10000 elementów. Uporządkowany strumień stanowi wejście dla dwóch operatorów `Aggregate`. Pierwszy od lewej oblicza łączną liczbę rozmów wykonywanych przez abonenta, w ramach okna obejmującego ostatnie 24 godziny. Drugi natomiast, oblicza liczbę rozmów wykonywanych przez abonenta za pośrednictwem poszczególnych stacji przekąźnikowych, spośród których operator `Aggregate` po nim następujący wybiera tę, z największą ilością realizowanych połączeń. Strumień będący wynikiem tego operatora jest łączony, przy pomocy operatora `Join`, ze strumieniem wynikowym operatora `Aggregate` obliczającego łączną liczbę wykonywanych rozmów przez abonenta. Jedynym warunkiem złączenia jest identyczna wartość atrybutu `timestamp`, co wyraża: wartość 0 dla parametru `Size` oraz atrybut `timestamp` podstawiony pod parametr `On` specyfikacji uporządkowania obu strumieni.

Element będący wynikiem złączenia posiada schemat

```
(Count, timestamp, timestamp, recorderId, max)
```

Na jego podstawie ostatni w zapytaniu operator `Map` wybiera identyfikator stacji przekąźnikowej (`recorderId`), liczbę połączeń wykonanych za jej pośrednictwem (`max`), oraz oblicza procentowy udział tych połączeń wśród wszystkich realizowanych przez abonenta $((\text{max}/\text{Count}) * 100)$.



Rys. 5. Graf zapytania 4

4.5. ZESTAWIENIE CECH JĘZYKÓW ZAPYTAŃ SYSTEMU STREAM I AURORA/BOREALIS

W tabeli 1 przedstawiono zbiorcze zestawienie cech języków zapytań systemu STREAM i Aurora/Borealis.

Tabela 1. Zestawienie cech języków zapytań systemów STREAM i Aurora/Borealis

Cecha	System STREAM	System Aurora/Borealis
Sposób definiowania zapytań	Deklaratywny	proceduralny
Obsługiwane źródła danych	strumienie i relacje	strumienie
Dostępne okna strumieni danych		
kierunek ruchu krawędzi	ruchome (ang. <i>sliding</i>)	ruchome (ang. <i>sliding</i>)
podstawa budowy okna	liczba elementów (ang. <i>tuple-based</i>), znacznik czasu (ang. <i>time-based</i>)	liczba elementów (ang. <i>tuple-based</i>), znacznik czasu (ang. <i>time-based</i>), dowolny atrybut schematu strumienia
sposób przesuwania krawędzi	dowolny	dowolny
Uwzględnione operatory	relacyjne, znane z języka SQL: projekcja, selekcja, agregacja i grupowanie, różnica zbiorów, suma zbiorów, eliminacja duplikatów, iloczyn kartezjański; tworzące okna (strumień-relacja): <i>tuple-based window</i> , <i>time-based window</i> , <i>partitioned window</i> ; tworzące strumień (relacja-strumień): <i>Istream</i> , <i>Dstream</i> , <i>Rstream</i>	projekcja (<i>Map</i>), selekcja (<i>Filter</i>), agregacja i grupowanie (<i>Aggregate</i>), suma zbiorów (<i>Union</i>), łączenie strumieni (<i>Join</i>), sortowanie (<i>BSort</i>), <i>Resample</i>

5. PODSUMOWANIE

W pracy przedstawiono dwa wybrane języki zapytań strumieniowych, z których każdy pozwala definiować typowe zadania dla obszarów zastosowań Systemów Zarządzania Strumieniowych Baz Danych. Sposoby definiowania zadań w każdym

z nich są jednak odmienne. Pomijając różnice wynikające z deklaratywnego bądź proceduralnego charakteru języka, należy zwrócić uwagę na rozbieżności w specyfikacjach odpowiadających sobie operatorów. Dla przykładu operator `Aggregate` z języka zapytań systemu Aurora/Borealis samodzielnie tworzy okno danych, dla którego obliczana jest zagregowana wartość, podczas gdy w języku systemu STREAM wyróżnione są osobne operatory tworzące okna, których wyniki stanowią wejścia dla operatorów dokonujących agregacji.

Rozpatrując system Aurora/Borealis, zauważyć można, iż pozwala on na definiowanie sposobu uporządkowania elementów strumienia, natomiast w systemie STREAM zakłada się, że elementy strumienia są uporządkowane, a ewentualne zaburzenia porządku niwelowane są na poziomie implementacji struktur danych reprezentujących strumienie. Wiąże się z tym bezpośrednio fakt, iż w systemie Aurora/Borealis podstawą budowy okna może być dowolny atrybut schematu strumienia, tymczasem w systemie STREAM może to być jedynie znacznik czasu lub liczba elementów.

Oprócz rozbieżności w definicjach poszczególnych operatorów omówione języki zapytań posiadają specyficzne operatory, nie występujące w innych językach. Przykładem jest operator `Resample` z systemu Aurora/Borealis.

LITERATURA

- [Abadi 2003] Abadi D., Carney D., Cetintemel U., Cherniack M., Convey C., Lee S., Stonebraker M., Tatbul N., Zdonik S., *Aurora: A New Model and Architecture for Data Stream Management*, 2003.
- [Arasu 2002] Arasu A., Babu S., Widom J., *The CQL Continuous Query Language: Semantic Foundations and Query Execution*, 2002.
- [Babcock 2002] Babcock B., Babu S., Datar M., Motwani R., Widom J., *Models and Issues in Data Stream Systems*, 2002.
- [Golab 2003] Golab L., Tamer Özsu M., *Data Stream Management Issues – A Survey*, 2003.
- [Jin 2006] Jin C., Carbonell J., *ARGUS: Efficient Scalable Continuous Query Optimization for Large-Volume Data Streams*, 2006.
- [Lerner 2003] Lerner A., Shasha D., *The Virtues and Challenges of Ad Hoc + Streams Querying in Finance*, 2003.
- [Motwani 2003] Motwani R., Widom J., Arasu A., Babcock B., Babu S., Datar M., Manku G., Olston C., Rosenstein J., Varma R., *Query Processing, Approximation, and Resource Management in a Data Stream Management System*, 2003.
- [Ramakrishnan 2003] Ramakrishnan R., Gehrke J., *Database Management Systems*. McGraw-Hill, 2003.
- [Borealis 2006] *Borealis Application Programmer's Guide*, 2006.
- [WWW1] <http://infolab.stanford.edu/stream/code/>
- [WWW2] <http://infolab.stanford.edu/stream/sqr/>
- [WWW3] http://pl.wikipedia.org/wiki/Strumien%C5%84_danych

STREAM DATABASE QUERY LANGUAGES

Data Stream Management Systems (DSMS) are an alternative to traditional – relational or object oriented – Database Management Systems (DBMS). They are built for the purpose of the applications, in which passive processing model, characteristic for traditional DBMS, can not be applied due to constantly varying input parameters. Data Stream Management System is an active part of the application that outputs results in response to input parameters changes, represented by the incoming data sequences. This is possible since Data Stream Management Systems allow defining continuous queries to be performed for a period of time, contrary to one-time queries typical for traditional DBMS. This way of processing, for which data source come in form of sequences, defines set of features that language created for specifying continuous queries has to have. This document presents those features, and gives examples of query languages that own them.

Paweł SOKOŁOWSKI
Artur WILCZEK*

METODY PRZETWARZANIA DANYCH STRUMIENIOWYCH

Szybki rozwój technologii informatycznych sprawia, że w coraz większej liczbie dziedzin dane przybierają postać nieograniczonego strumienia. Powstają rozwiązania, takie jak Systemy Zarządzania Strumieniową Bazą Danych (SZSBD), umożliwiające osiągnięcie większej niezawodności i efektywności działania aplikacji korzystających z danych strumieniowych.

W niniejszej pracy przedstawiono metody przetwarzania strumieni danych zaproponowane w wybranych SZSBD, w systemie STREAM i AURORA.

1. WPROWADZENIE

Wraz z rozwojem technologii informatycznych projektanci aplikacji stają często przed zadaniem wytworzenia rozwiązania służącego przetwarzaniu w czasie rzeczywistym znacznej liczby danych. W niektórych dziedzinach aplikacje wykonujące podobne obliczenia istniały już od dawna, ale do tej pory nie musiały sobie one radzić z problemami pracy w czasie rzeczywistym i przetwarzania ogromnej liczby danych jednocześnie. Zazwyczaj albo pracowały one w trybie wsadowym przetwarzając zapisane wcześniej partie danych albo liczba danych, jakie należało przeanalizować w jednostce czasu nie była wielka w stosunku do możliwości obliczeniowych maszyn, na których działały. Z nową klasą aplikacji mamy zatem do czynienia w momencie połączenia obu wspomnianych wymagań. Dane w takich aplikacjach przybierają formę nieograniczonych sekwencji danych zwanych strumieniami danych.

Tradycyjne Systemy Zarządzania Bazami Danych (SZBD), opracowane z myślą o przetwarzaniu ograniczonych zbiorów danych, często nie są w stanie sprostać wymaganiom aplikacji monitorujących strumień danych. Skłania to, projektantów aplikacji strumieniowych, do rezygnacji z wykorzystania gotowego SZBD i budowy swojego rozwiązania od podstaw. Niewątpliwą zaletą takiego podejścia jest ściśle

* Instytut Informatyki Stosowanej, Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27, artur.wilczek@pwr.wroc.pl

dostosowanie używanych algorytmów do dziedziny problemu, co pozwala maksymalnie zoptymalizować przetwarzanie i zapewnić wymaganą dokładność obliczeń czy opóźnienie (ang. *latency*). Ma ono jednak poważne wady, jakimi są małe szanse na ponowne wykorzystanie kodu, długi czas powstawania rozwiązania oraz zwiększona awaryjność.

Celem niniejszej pracy jest przedstawienie kilku technik wykorzystywanych w aplikacjach przetwarzających dane strumieniowe. Część z nich powstała podczas prac nad prototypami SZSBD, inne w dużej mierze są jeszcze na etapie rozważań teoretycznych.

2. SYSTEM AURORA

System Aurora jest jednym z najbardziej zaawansowanych systemów zarządzania strumieniami danych. Prace nad nim zostały niedawno wstrzymane, ale stał się on integralną częścią rozproszonego systemu zarządzania strumieniami danych Borealis [Abadi 2003].

System Aurora proponuje wykorzystanie proceduralnego języka zapytań. Zapytanie jest reprezentowane poprzez skierowany graf nie zawierający cykli, w którym węzłami są operatory podobne do tych występujących w algebrze relacyjnej. Przetwarzanie danych jest realizowane przez zestaw komunikujących się ze sobą komponentów. Każdy z nich odpowiada za inny aspekt obliczeń. Wspólnym ich zadaniem jest zapewnienie środowiska, w którym możliwe jest ciągle zwracanie wyników zapytań reprezentowanych przez graf operatorów. Zarządca danych (ang. *storage manager*) odpowiada za przechowywanie danych historycznych oraz zarządzanie pulą kolejek przechowujących krotki pomiędzy kolejnymi etapami przetwarzania. Planista (ang. *scheduler*) zarządza wykonywaniem obliczeń operatorów w wątkach z uwzględnieniem priorytetów wyznaczonych przez monitor jakości usług (ang. *quality of service monitor*).

2.1. OPERACJE NA STRUMIENIACH DANYCH

Operacje, które można realizować na strumieniach danych są analogiczne do tych wykonywanych przez operatory algebry relacyjnej. Występują tu jednak zasadnicze różnice ze względu na konieczność sprostania specyficznym wymaganiom przetwarzania strumieni danych.

Strumień jest sekwencją krotek, która zmienia się tylko i wyłącznie przez dodawanie nowych krotek na jej końcu. Wszystkie krotki należące do tego samego strumienia mają jednakowy schemat. Poza wartościami atrybutów określonych w schemacie, każda krotka niesie również znacznik czasu, który określa jej czas pojawienia się

w systemie. W momencie, gdy podczas przetwarzania zapytania tworzona jest nowa krotka, zostaje ona oznaczona czasem nadejścia najstarszej krotki, której dane wykorzystano do jej stworzenia.

Model przetwarzania Aurory nie zakłada żadnego porządku napływających krotek. W szczególności nie zakłada się, iż krotki dostarczane do operatora są uszeregowane zgodnie z czasem nadejścia do systemu. Przeciwnie podejście byłoby naturalne i należy wyjaśnić motywy postępowania projektantów systemu. Doszli oni mianowicie do wniosku, że z punktu widzenia użyteczności wyniku, często bardziej opłacalne jest przetwarzanie krotek o większym znaczeniu dla końcowego wyniku przed krotkami, które nadeszły wcześniej, lecz mają dużo mniejszy wpływ na wynik obliczeń. Operatory możemy podzielić ze względu na ich wrażliwość na uszeregowanie krotek w strumieniu.

2.2. OPERACJE NIEWRAŻLIWE NA KOLEJNOŚĆ KROTEK

Jednym z operatorów niewrażliwych na kolejność krotek jest operator `Filter`. Jego działanie przypomina działanie relacyjnego operatora selekcji. Pełni on również funkcję „nawigatora” krotek kierując je do odpowiednich strumieni wyjściowych w zależności od tego, które predykaty spełniają.

Operator `Map` jest podobny do relacyjnej projekcji. Zwraca on krotki o schemacie relacji, który zazwyczaj różni się od schematu krotek, jakie operator otrzymuje na wejściu. Dla każdego atrybutu nowego schematu określona jest funkcja, która zwraca wynik w oparciu o parametr, jakim jest krotka z wejścia operatora.

Operator `Union` ma za zadanie łączenie krotek z wielu strumieni wejściowych w jeden strumień wyjściowy. Wszystkie strumienie wejściowe muszą zawierać krotki o takim samym schemacie.

2.3. OPERACJE WRAŻLIWE NA KOLEJNOŚĆ KROTEK

Każdy z operatorów wrażliwych na kolejność krotek może zachowywać się przewidywalnie tylko zakładając jakiś porządek nadchodzenia krotek. Sprecyzowanie tego porządku należy do administratora budującego graf operatorów. W klauzuli `Order`, definiującej spodziewany porządek krotek, należy podać atrybut porządkujący. Jeżeli zostanie pominięty jako atrybut porządkujący przyjmowany jest czas.

Krotki niespełniające określonego w danym operatorze porządku zostaną pominięte. Można złagodzić ten rygor precyzując dwa kolejne argumenty klauzuli `Order`. Argument `Slack`, gdy jest większy od zera, oznacza liczbę krotek bezpośrednio poprzedzających krotkę, które musiałyby mieć wartość porządkującego atrybutu większą od krotki, która jest rozpatrywana. Co oznacza, że krotka t zostanie odrzucona wtedy,

gdy poprzedza ją więcej niż n krotek z wartością atrybutu porządkującego większą niż w krotce t , gdzie n oznacza wartość parametru *Slack*.

Ostatnim argumentem jest sekwencja atrybutów grupujących *GroupBy*, której wystąpienie oznacza, iż porządek krotek rozpatrywany jest niezależnie w powstałych grupach.

Pierwszym z grupy operatorów wrażliwych na kolejność krotek jest operator *BSort*, który jako argument przyjmuje tylko specyfikacje porządku nadchodzących krotek a wynik zwraca w postaci strumienia nieco lepiej uporządkowanego niż ten wejściowy. Gdyby traktować sekwencje krotek wejściowych i wyjściowych jako tablice, to tablica zwracanych krotek odpowiada tablicy krotek odczytanych, nad którą wykonano n iteracji sortowania bąbelkowego, gdzie n jest równe wartości argumentu *Slack* podanego podczas specyfikacji porządku.

Operator *Aggregate* służy wykonywaniu obliczeń na fragmentach strumienia wyznaczanych poprzez okna. Jego sygnatura wygląda następująco:

$$\text{Aggregate}(F, O, s, i)(S) \quad (1)$$

gdzie S oznacza strumień wejściowy operatora.

Parametr O oznacza zakładaną kolejność krotek w strumieniu źródłowym, która określona jest z pomocą wspomnianej wyżej klauzuli *Order*. Okna przesuwne (ang. *sliding window*) definiuje się w oparciu o atrybut porządkujący, określając szerokość okna jako parametr s . Sposób przemieszczania się okna zależy od wartości parametru i . W najprostszym przypadku powinien być on wartością (w jednostkach atrybutu porządkującego) określającą przesunięcie początku okna. Możliwe jest również określenie grupy funkcji *init*, *incr* oraz *final*, które system wywołuje odpowiednio w momencie rozpoczynania nowego okna, gdy nadejdzie nowa krotka lub w chwili zamykania okna. Funkcje te utrzymują stan, dzięki któremu w momencie zamykania okna możliwe jest określenie miejsca początku kolejnego.

Parametr F jest operatorem agregującym lub funkcją agregującą, która jako parametr przyjmuje podzbiór krotek ze strumienia źródłowego wyznaczonego poprzez szerokość okna oraz parametr *GroupBy* klauzuli *Order*. Jako wynik zwraca natomiast krotkę zawierającą często tylko jeden atrybut, którą operator *Aggregate* konkatenuje z krotką zawierającą wartości atrybutów grupujących oraz wartość atrybutu porządkowego pierwszej krotki aktualnego okna. Otrzymana krotka jest wstawiana do kolejki wyjściowej operatora.

Parametr *Slack* klauzuli określającej oczekiwany porządek krotek, ma tutaj wpływ na moment zamknięcia okna. Dla wartości równej zero, okno zostanie zamknięte w momencie pojawienia się w strumieniu źródłowym krotki z wartością atrybutu porządkującego wykraczającą poza szerokość bieżącego okna. Jeżeli rozluźniony zostanie rygor porządku, wówczas będą uwzględniane krotki niespełniające zakładanego porządku. Może to prowadzić do sytuacji zbyt długiego oczekiwania na wynik działania operatora, który jest operatorem blokującym i do wyznaczenia wyniku, dla

danego okna, wymaga jego wszystkich krotek. W przypadku aplikacji czasu rzeczywistego należy, więc, umożliwić przerwanie oczekiwania i zwrócenie wyniku przybliżonego. Do określenia ograniczeń czasowych służy opcjonalny parametr t , który wskazuje, w jednostkach czasowych, maksymalną długość kompletowania danych dla jednego okna. Po przekroczeniu tego czasu, operator zwraca wynik, a kolejne krotki należące do tego okna zostaną zignorowane.

Właściwością operatora `Aggregate`, wartą zauważenia, jest możliwość określenia oczekiwanego porządku w dwojaki sposób. Poza podanym wyżej możliwe jest wstawienie przed operator `Aggregate` operatora `SortBy`. Klauzula porządku w obu operatorach powinna różnić się tylko wartością parametru `Slack`, który powinien być określony tylko w operatorze porządkującym. Wynik działania takiego zestawu będzie identyczny z działaniem samego operatora agregującego, z tym samym porządkiem. Sposób ten może się jednak okazać korzystny, gdy wynik przetwarzania operatora sortującego będzie przekazywany do kilku operatorów agregujących, o tym samym porządku. W przypadku mało rygorystycznego porządku i braku ograniczeń czasowych, taki zabieg pozwoli obniżyć ilość pamięci używanej na przechowywanie krotek w poszczególnych oknach.

Kolejny operator, `Join`, łączy dwa strumienie podobnie jak jego relacyjny odpowiednik łączy tabele. Dla obu strumieni określa się osobno spodziewany porządek krotek, przy czym atrybuty, względem których strumienie mają być uporządkowane powinny przyjmować wartości z tej samej dziedziny. Złączenie krotek z obu strumieni dokonywane jest pod warunkiem spełniania przez nie warunku złączenia, określonego w predykanie przekazywanym jako kolejny parametr operatora. Nie jest to jednak jedyny warunek, jaki spełniać muszą łączone krotki. W przeciwieństwie do złączenia występującego w algebrze relacyjnej operator nie dysponuje pełnym zbiorem danych dla żadnego ze strumieni. Wynika to bezpośrednio z definicji strumienia, która mówi, że jest on nieograniczony. Z tego względu możliwe jest jedynie dokonywanie złączeń fragmentów strumieni. Na oba strumienie nakładane jest ograniczenie w postaci okna zdefiniowanego za pomocą szerokości w tych samych jednostkach, co wartości atrybutów porządkujących w klauzulach porządkujących. Stąd też dodatkowym warunkiem złączenia jest odległość od siebie wartości atrybutów porządkujących dla obu krotek nie większa niż szerokość okna złączenia. Tak jak ma to miejsce dla operatora agregującego, deklaracja porządków obu strumieni może uwzględniać krotki, które nadejdą z opóźnieniem. Analogicznie możliwe jest też połączenie z operatorami porządkującymi, zapewniające niezmiennione działanie oraz dające dodatkowe możliwości optymalizacyjne.

Ostatnim operatorem wrażliwym na kolejność krotek, pojawiających się w jego kolejce wejściowej, jest operator `Resample`. Podobnie jak operator `Join`, jest to operator binarny, tym razem jednak kolejność strumieni wejściowych ma wpływ na wynik. Stanowi on niejako połączenie dwu opisywanych wyżej operatorów. Wykonuje on operację złączenia strumieni, podobnie jak `Join`, w jego sygnaturze jednak

zamiast argumentu określającego warunek złączenia występuje deklaracja funkcji F , od której oczekuje się, że będzie zachowywać się podobnie jak funkcja podawana jako parametr dla operatora *Aggregate*. Wynikiem działania operatora, dla każdej krotki t ze strumienia S_1 , jest krotka zawierająca wszystkie atrybuty grupujące oraz atrybut porządkowy, skonkatenowana z krotką powstałą w wyniku działania funkcji agregującej, na wszystkie krotki u ze strumienia S_2 , znajdujące się w tym samym oknie co krotka t . Jednym z zastosowań tego operatora jest synchronizacja strumieni. Jest on również przydatny w przypadku strumienia, w którym dane pojawiają się nieregularnie a aplikacja kliencka oczekuje na nowe dane w regularnych odstępach czasu. Łącząc „nieregularny” strumień operatorem *Resample* ze strumieniem zawierającym elementy w regularnych odstępach czasu można generować odpowiedź zgodnie z oczekiwaniami aplikacji wynikowej.

3. SYSTEM STREAM

System *STREAM* jest prototypem SZSBD opracowanym na *Stanford University* i rozwijanym do stycznia 2006 roku [AB 2003]. Obecnie prace nad nim zostały przerwane. Pomimo tego pozostaje on jednym z najbardziej rozbudowanych i najlepiej udokumentowanych prototypów, jakie istnieją. Prace badawcze obejmowały swoim zakresem wszystkie aspekty działania systemu. Od badań nad językiem deklaratywnym CQL, jakim można się posługiwać budując zapytania po opracowywanie algorytmów adaptacji parametrów działania systemu w zależności od liczbie dostępnych zasobów oraz natężenia napływających danych. Tylko część z nich została zaimplementowana w prototypie.

Strumień danych w systemie *STREAM* ma postać wielozbioru par $\langle t, s \rangle$, gdzie t oznacza czas pojawienia się w strumieniu krotki s . W trakcie przetwarzania danych przez operatory zakłada się, że kolejne krotki nadchodzą uporządkowane rosnąco względem znacznika czasu nadawanego im przez system w momencie zarejestrowania lub przez systemy źródłowe. Fizyczna reprezentacja strumienia jest tutaj identyczna z reprezentacją relacji. System traktuje strumienie i relacje jako sekwencje krotek z dodatkowymi znacznikami. Jednym z takich znaczników jest oznaczenie czasowe, które dla każdej ze struktur ma inną interpretację. W przypadku strumienia oznacza ono czas nadejścia krotki do systemu. W przypadku relacji, zbiór krotek z tym samym znacznikiem czasowym oznacza stan relacji w danym momencie. Zapytania rejestrowane w systemie poprzez wykorzystanie okna czasowego wyznaczają krotki z konkretnego przedziału czasowego.

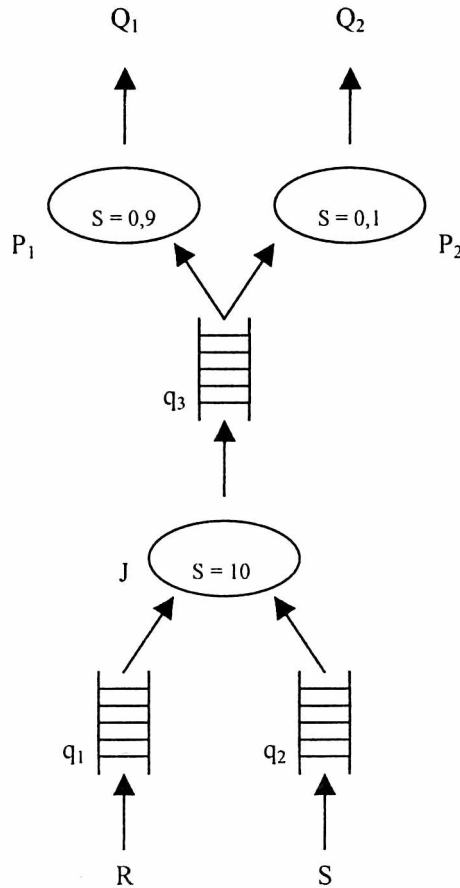
Formułowanie zapytań w systemie *STREAM* może odbywać się na dwa sposoby. Przy użyciu deklaratywnego języka CQL [ABW 2003], który stanowi rozszerzenie języka SQL o elementy pozwalające na łączenie relacji ze strumieniami lub poprzez określenie konkretnego planu wykonania zapytania.

Elementami składającymi się na plan wykonania zapytania mogą być operacje, kolejki lub streszczenia. W implementacji systemu reprezentują je trzy klasy dziedziczące po klasie bazowej *Entity*, klasy: *Operator*, *Queue*, *Synopsis*. Reprezentują one odpowiednio operatory podobne do operatorów algebry relacji, kolejki krotek oczekujących na przetworzenie oraz struktury danych zawierające streszczenie krotek zaobserwowanych w pewnym miejscu planu. System STREAM umożliwia aby operatory mogły być współdzielone pomiędzy różnymi zapytaniem. Ponadto współdzielone mogą być także kolejki i streszczenia. Pozwala to w pewnych sytuacjach zwiększyć oszczędność zasobów systemowych.

Każdy element planu wykonania zapytania opatrzony jest zbiorem parametrów, których wartości wpływają na sposób jego działania, najczęściej determinując dokładność obliczeń oraz poziom zużycia zasobów systemowych. W zamyśle autorów systemu budowa planu zapytania na podstawie wyrażenia w języku CQL miała, poza tradycyjną optymalizacją spotykaną w klasycznych SZBD, również dostosowywać niektóre z tych parametrów. Implementacja tej funkcjonalności nie została jednak ukończona. Podobnie stało się z zamysłem zbudowania algorytmu, który w trakcie przetwarzania danych, reagując na zmiany zachodzące w środowisku działania dostosowywałby wybrane parametry dynamicznie, realizując tym samym funkcje adaptacyjne i balansując działanie systemu pomiędzy dokładnością przetwarzania a poziomem wykorzystania zasobów.

Inaczej niż w przypadku systemu Aurora, w systemie STREAM, plan wykonania zapytania budowany jest dla każdego zapytania osobno i dopiero w kolejnym kroku plany zarejestrowanych zapytań podlegają analizie w celu znalezienia elementów, których współdzielenie przyniesie korzyści. Istnieje bowiem zagrożenie, iż współdzielenie niektórych poddrzew nie doprowadzi do optymalizacji przetwarzania. Znajduje tutaj zastosowanie algorytm planowania łańcuchowego (ang. *chain-scheduling*) [Babcock 2004], którego zadaniem jest minimalizacja zajętości pamięci operacyjnej wymaganej do przetworzenia zbioru zapytań.

Współdzielenie fragmentu planu wykonania zapytania może być nieoptymalne, na przykład w sytuacji przedstawionej na rysunku 1, gdy współdzielony operator J , wykonujący operację złączenia będzie zwracał liczbę krotek zbliżoną do tej w iloczynie kartezjańskim zbiorów na swoich wejściach. Gdy dodatkowo operator P_1 , pochodzący z planu pierwszego zapytania będzie szybciej pobierał krotki zwracane przez J niż operator P_2 pochodzący z planu zapytania drugiego, wówczas planista stanie przed dylematem. Z jednej strony wykonanie operatora J często zapewnia krotki, które natychmiast są przetwarzane przez operator P_1 ale z drugiej strony, wykonanie J zwiększa rozmiar kolejki na jego wyjściu, która rośnie szybko ze względu na to, że operator P_2 rzadko pobiera z niej krotki do przetworzenia.



Rys. 1. Diagram przedstawiający plan wykonania zapytania ze współdzielonym operatorem

Implementacja funkcjonalności łączenia planów zapytań w prototypie ogranicza się do znajdowania poddrzew planów, które są ze sobą identyczne w obu planach. Autorzy systemu STREAM przewidują jednak możliwość modyfikacji algorytmu łączącego plany zapytań, polegającą na wykorzystaniu aproksymacji wyniku zapytań w celu zmniejszenia zużycia zasobów. Przykładem takiego działania byłaby decyzja, aby współdzielić operatory o tej samej semantyce działające na oknach o różnych rozmiarach i wymuszone zawężanie okna. W zależności od rodzaju obliczeń, jakie wykonują oba operatory, takie działanie mogłoby prowadzić do zmniejszenia dokładności prezentowanego wyniku w jednym z operatorów i jednocześnie korzyści płynącej z zaoszczędzenia dodatkowych cykli procesora, a poprzez współdzielenie streszczeń danych, również z zaoszczędzenia pamięci. Decyzja o podjęciu takich kroków, musiałaby być jednak w jakiś sposób zaakceptowana przez użytkownika, np. przez okre-

ślenie wymaganej dokładności wyniku przetwarzania. Takiej możliwości system STREAM nie oferuje. Rozważania na ten temat przewidywano jako jeden z kierunków dalszego rozwoju systemu.

Z każdym zapytaniem związane są dwie właściwości. Jedna z nich określa czy dane zapytanie jest aktywne, a więc czy planista powinien uwzględniać operatory wchodzące w jego skład w trakcie planowania kolejności wykonania. Nieaktywność zapytania nie jest tożsama z jego usunięciem z systemu albowiem może ono zostać w każdej chwili uaktywnione i dlatego pewne czynności związane z danymi, jakie ma ono przetwarzać muszą być wykonywane. Druga właściwość zapytania określa priorytet zapytania. System będzie starał się zapewnić większą dokładność zapytaniom o wyższym priorytecie w sytuacji niedoboru zasobów systemowych. W zamyśle autorów systemu priorytety zapytań mogłyby także wpływać na decyzje planisty dotyczące kolejkwania wykonania operatorów. Taka możliwość nie została jednak zaimplementowana w prototypie.

3.1. OPERACJE NA STRUMIENIACH DANYCH

Zaimplementowane w systemie STREAM operatory dzielą się na dwie podstawowe grupy, operatorów języka CQL i operatorów systemowych. Do pierwszej z nich, należą operatory reprezentujące operacje języka CQL, takie jak selekcja lub projekcja. Operatory systemowe są odpowiedzialne za wykonywanie zadań niższego poziomu, umożliwiających poprawne działanie systemu jak importowanie danych z zewnętrznych źródeł, eksport wyników do aplikacji klienckich czy redukcja obciążenia.

Wykonywanie operacji na strumieniach danych zachodzi zawsze w trzech fazach. W pierwszej fazie dochodzi do konwersji fragmentów strumieni na relacje. Do tego celu służą operatory z grupy strumień-relacja. W drugiej fazie, przy użyciu operatorów relacja-relacja, dane przetwarzane są z wykorzystaniem operacji z algebry relacyjnej. Następnie dane wynikowe przybierają z powrotem formę strumienia za pomocą operatorów grupy relacja-strumień. Korzyścią z takiego podejścia jest możliwość wykorzystania ogromnej liczbie algorytmów i technik stworzonych na potrzeby systemów relacyjnych. Utrudnieniem jest fakt, iż jak powiedziano relacja w systemie STREAM różni się od tradycyjnego modelu relacji tym, że zachowuje swoje stany historyczne. Przez to użyteczne stają się tu również techniki opracowane w badaniach nad bazami temporalnymi. Ten trójfazowy model przetwarzania może budzić wątpliwości co do swojej optymalności, z uwagi na konieczność ciągłego wykonywania dodatkowych operacji, oprócz tych związanych z samymi obliczeniami. Dodatkowy nakład pracy redukuje jednak fakt, że dane strumieniowe i relacyjne reprezentowane są w systemie w ten sam sposób.

4. PLANOWANIE WYKONANIA ZAPYTAŃ

Korzyści płynące ze zrównoleglenia obliczeń oraz architektura współczesnych komputerów, wspomagająca przetwarzanie wielowątkowe, skłaniają do projektowania każdego rozwiązania programistycznego z myślą o współbieżności. Także przetwarzanie strumieni danych odbywa się zazwyczaj wielowątkowo. W tym wypadku jest to tym bardziej uzasadnione, iż zachodzi konieczność obsługi wielu strumieni jednocześnie. Dodatkowo zawsze w systemie zarejestrowanych jest jednocześnie wiele zapytań.

Niestety, podczas projektowania systemu zarządzania strumieniami danych nie można sobie pozwolić na przyjęcie założenia mówiącego, że każdy z operatorów planu wykonania zapytania, będzie obsługiwany przez osobny procesor. Rozwiązanie takie byłoby zbyt drogie, poza tym nadmiarowe, ponieważ czas każdego z procesorów byłby wykorzystany tylko w bardzo małym stopniu. O wiele efektywniejsze, tańsze i bardziej skalowalne okazują się systemy z jednym lub najwyżej kilkoma procesorami, w których system operacyjny posiada zaimplementowane odpowiednie rozwiązania kolejujące zadania, zgłoszone do wykonania. Badania nad tym tematem prowadzone są już od wielu lat, co zaowocowało szeregiem dostępnych algorytmów poczynając od prostych, jak algorytm karuzelowy (ang. *Round Robin*) i FIFO (ang. *First In First Out*) po bardziej skomplikowane, jak np. algorytm EDF (ang. *Earliest Deadline First*). Są one zaimplementowane w popularnych systemach operacyjnych, które dzięki ich wykorzystaniu pracują bardzo wydajnie. Kusząca, zatem może wydawać się perspektywa zaprojektowania systemu, który każdy operator umieści w osobnym wątku, a zarządzanie wątkami zostawi w gestii systemu operacyjnego. Projektant nie jest tu też całkiem pozbawiony wpływu na kolejność wykonywania obliczeń, ponieważ system operacyjny zazwyczaj pozwala na manipulowanie priorytetami poszczególnych wątków.

Eksperymenty dowodzą jednak, że postępowanie w taki sposób daje bardzo słabe wyniki w porównaniu z rozwiązaniami zaimplementowanymi nie dla przypadku ogólnego, a z myślą o kolejkowaniu wykonania operatorów planu zapytania systemu przetwarzającego strumień danych.

4.1. ALGORYTM SUPERBOX-SCHEDULING

Algorytm został opracowany w trakcie prac nad systemem Aurora. Przetwarzanie danych przepływających przez graf operatorów odbywa się wielowątkowo a operatory szeregowane są ze względu na przypisane im priorytety. Głównym kryterium, jakie brane jest pod uwagę przy wyznaczaniu priorytetów jest utrzymanie wymaganej użyteczności wyników zwracanych przez zapytania, których częściami są operatory. Jest

ona mierzona przy użyciu szeregu czynników takich jak świeżość danych czy przedział wartości, w jakim się znajdują.

Użyteczność otrzymanego wyniku obliczana jest w momencie eksportu krotek z systemu. W wypadku obniżania się użyteczności danych należy podnieść priorytety operatorom wchodzącym w skład zapytania zwracającego dane wyniki. W ten sposób, w skrócie, można by opisać podstawową strategię szeregowania operatorów w systemie Aurora. Autorzy systemu zauważają jednak, iż duże korzyści daje przetwarzanie krotek partiami, a także łączenie operatorów w bloki.

4.1.1. ŁĄCZENIE OPERATORÓW

Łączenie operatorów w bloki (ang. *superbox*), które wykonywane są kolejno, daje korzyści w szybkości przetwarzania poprzez zmniejszenie czasu, przez jaki proces zajmuje się operacjami planowania wykonania operatorów czy przełączaniem kontekstu pomiędzy wątkami, a także w wyniku pominięcia zarządcy pamięci. Zmniejsza się też prawdopodobieństwo konieczności skorzystania z mechanizmu stronicowania danych i czasochłonnego dostępu do pamięci masowej.

Teoretycznie możliwe byłoby połączenie w nierozdzielny blok dowolnego poddrzewa w grafie operatorów. System Aurora nakłada jednak ograniczenie wymagające, iż każdy taki blok musi kończyć się operatorem eksportującym dane poza system (ang. *output box*). Głównie dlatego, że tylko odnośnie tych operatorów istnieją informacje dotyczące użyteczności.

Problem wyboru odpowiedniego poddrzewa rozwiązywany jest na dwa sposoby. Po pierwsze poprzez statyczny dobór poddrzew zanim rozpoczęte zostanie przetwarzanie oraz w czasie przetwarzania, na podstawie priorytetów nadanych poszczególnym operatorom.

Budowa bloków w sposób statyczny grupuje wszystkie operatory należące do jednego zapytania. W ten sposób liczba jednostek, jakich wykonanie należy zaplanować jest równa liczbie wyjść z systemu, a kolejne bloki wykonywane są na przemian.

W kwestii kolejności wykonywania obliczeń wewnątrz bloków, czyli uszeregowania składających się na nie operatorów, system AURORA przewiduje kilka możliwości doboru strategii. Każda z nich nastawiona jest na uzyskanie innego rodzaju korzyści.

Strategia nastawiona na minimalizowanie średniego kosztu przetwarzania szereguje operatory w kolejności przechodzenia drzewa post-order, czyli przed wykonaniem obliczeń dowolnego operatora wykonywane są obliczenia wszystkich operatorów w jego poddrzewach.

W przypadku, gdy najistotniejsza jest „świeżość” danych w prezentowanym wyniku, wykorzystuje się strategię minimalnego opóźnienia danych (ang. *minimal latency*). System nastawiony jest na rozpoczęcie zwracania wyniku najszybciej jak to jest możliwe. Aby to osiągnąć, szereguje operatory zgodnie z kosztem zwrócenia wyniku (ang.

output cost). Jest to średni czas procesora potrzebny na zwrócenie pojedynczej krotki wyniku poprzez przetworzenie krotek znajdujących się w kolejce, z której czerpie dane wejściowe operator. Należy również zaznaczyć, że po zakończeniu przetwarzania każdego operatora koszt obliczany jest na nowo i najczęściej dochodzi do zmiany porządku.

Ostatnią dostępną strategią jest strategia zorientowana na zapewnienie minimalnego zużycia pamięci poprzez jak najszybsze zwalnianie pamięci. Dla każdego operatora wyliczana jest oczekiwana ilość pamięci, jaka będzie zwolniona w jednostce czasu, mierzona jako iloczyn rozmiaru krotek znajdujących się w kolejce źródłowej operatora i stosunku konsumowanych krotek do zwracanych podzielona przez szacowany koszt wytworzenia jednej krotki wyniku. Podobnie jak w poprzedniej strategii kolejność uszeregowania operatorów jest weryfikowana w każdym cyklu.

4.1.2. ŁĄCZENIE KROTEK

Dodatkową optymalizacją algorytmu jest, oprócz łączenia operatorów w grupy, także planowanie przetwarzania krotek w grupach. Ma to przynieść korzyści w postaci zmniejszenia całkowitych nakładów na przetwarzanie poprzez rzadsze wykonywanie czynności związanych z planowaniem wykonania, przełączaniem kontekstu czy odwołaniami do kodu operatora. Autorzy systemu wspominają również o możliwości wydajniejszego zaimplementowania operacji wykonywanych przez operatory [Carney 2003].

Maksymalna liczba krotek, jaka może być przetworzona w jednym kroku jest modyfikowana przez administratora systemu poprzez ustalenie wartości zmiennej systemowej. W trakcie przetwarzania planista zleca operatorowi przetworzenie jednorazowo grupy krotek z kolejki wejściowej.

5. PODSUMOWANIE

W pracy omówiono techniki stosowane w wybranych Systemach Zarządzania Strumieniową Bazą Danych w celu zapewnienia efektywnego przetwarzania nieograniczonych strumieni danych. Przedstawiono realizowane w systemach STREAM i Aurora operacje na strumieniach danych a także techniki planowania i wykonywania zapytań.

Systemy STREAM i Aurora reprezentują różne podejście do budowy systemów przetwarzających strumienie danych. System STREAM proponuje wykorzystanie deklaratywnego języka zapytań CQL. Zapytania wyrażone w języku CQL podlegają procesom podobnym do stosowanych w tradycyjnych SZBD polegającym na analizie treści zapytania, budowie planu wykonania i jego optymalizacji. System Aurora pro-

ponuje wykorzystanie języka proceduralnego. Treść zapytania ma, w systemie Aurora, postać grafu będącego jednocześnie wstępnym planem wykonania zapytania, który następnie podlega optymalizacji.

Celem budowy Systemów Zarządzania Strumieniową Baza Danych jest poprawa efektywności przetwarzania danych strumieniowych w stosunku do rozwiązań wykorzystujących tradycyjne SZBD. Zastosowanie przedstawionych w pracy metod planowania i optymalizacji wykonania zapytań ciągłych pozwala osiągnąć poprawę efektywności przetwarzania strumieni danych.

Badania nad systemami przetwarzającymi strumienie danych są stosunkowo nową dziedziną. Pozostało jeszcze wiele otwartych problemów badawczych a szereg istniejących systemów jest nadal w fazie prototypu. Pomimo tego powstały już rozwiązania z powodzeniem konkurujące, pod względem efektywności przetwarzania danych strumieniowych, z systemami wykorzystującymi tradycyjne SZBD.

6. LITERATURA

- [Abadi 2003] Abadi D., Carney D., Cetintemel U., Cherniack M., Convey C., Lee S., Stonebraker M., Tatbul N., Zdonik S., *Aurora: A New Model and Architecture for Data Stream Management*, 2003.
- [AB 2003] Arasu A., Babcock B., Babu S., Datar M., Ito K., Nishizawa I., Rosenstein J., Widom J., *STREAM: The Stanford Stream Data Manager*. In ACM SIGMOD, 2003.
- [ABW 2003] Arasu A., Babu S., Widom J.: *The CQL Continuous Query Language: Semantic Foundations and Query Execution*. Technical Report, Stanford University, 2003.
- [Babcock 2004] Babcock B., Babu S., Datar M., Motwani R., Thomas D., *Operator Scheduling in Data Stream Systems*, The VLDB Journal The International Journal on Very Large Data Bases, 2004.
- [Carney 2003] Carney D., Cetintemel U., Rasin A., Zdonik S., Cherniack M., Stonebraker M., *Operator Scheduling in a Data Stream Manager*. VLDB Conference, 2003.

STREAM DATA PROCESSING METHODS

Rapid development of computer technologies induces the constant increase in the number of domains that utilize unbounded streams of data, thus it is becoming more and more important for the applications making use of that data to run efficiently and reliably.

This paper introduces the reader to a set of techniques introduced in Data Stream Management Systems (DSMS) STREAM and AURORA.

*przetwarzanie danych strumieniowych,
strumień danych,
adaptacja, aproksymacja*

Paweł SOKOŁOWSKI
Artur WILCZEK*

TECHNIKI ADAPTACJI I APROKSYMACJI W PRZETWARZANIU STRUMIENI DANYCH

Efektywne przetwarzanie nieograniczonych strumieni danych wymaga zastosowania odmiennych technik niż stosowane w tradycyjnych Systemach Zarządzania Bazami Danych (SZBD). Prowadzone są badania związane z poszukiwaniem metod pozwalających na osiągnięcie większej niezawodności i efektywności działania aplikacji przetwarzających dane strumieniowe.

W niniejszej pracy zaprezentowano kilka technik, które mogą być wykorzystane przy budowie Systemu Zarządzania Strumieniową Bazą Danych (SZSBD) i pozwalają przezwyciężyć trudności płynące z konieczności przetwarzania nieograniczonych strumieni danych w czasie rzeczywistym, oferując jednocześnie możliwie największą efektywność.

1. WPROWADZENIE

Przetwarzanie danych strumieniowych wymaga stosowania nowych technik w celu osiągnięcia podobnej efektywności, jaką oferują tradycyjne SZBD operujące na skończonych zbiorach danych. W systemach strumieniowych brak jest możliwości przechowywania wszystkich danych pochodzących ze strumieni, w związku z tym znajdują tutaj zastosowanie różne techniki aproksymacji wyników oraz adaptacji planów wykonywania zapytań, pozwalające uzyskać przybliżone wyniki zapytań na ograniczonym zestawie danych.

W niniejszej pracy omówiono techniki adaptacji i aproksymacji stosowane w wybranych Systemach Zarządzania Strumieniową Bazą Danych (SZSBD) systemie STREAM [AB 2003] i Aurora [Abadi 2003].

* Instytut Informatyki Stosowanej, Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27, artur.wilczek@pwr.wroc.pl

2. WŁASNOŚCI DANYCH W STRUMIENIACH

Jedną z zaimplementowanych w systemie STREAM technik adaptacyjnych jest metoda ograniczania zasobów przeznaczanych do realizacji obliczeń przez operatory na podstawie zaobserwowanych charakterystycznych cech danych nadchodzących do systemu. Wyselekcjonowano kilka typów ograniczeń, pod których kątem badane są nadchodzące dane. Jeżeli można stwierdzić, iż dane zawsze spełniają ograniczenia z pewną dokładnością, jest to podstawą do poczynienia założeń, które pozwalają ograniczyć rozmiary struktur przechowujących streszczenia danych przy wejściach do operatorów. Poniższy przykład obrazuje sposób działania algorytmu.

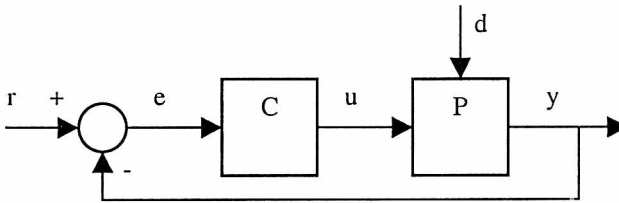
Rozważmy dwa strumienie *Zamówienia* (*Z*) oraz *Realizacje* (*R*), które badamy w celu ustalenia średnich opóźnień w realizacji zamówień. Operator złączenia, w realizującym to zadanie zapytaniu, jest zmuszony utrzymywać po jednej strukturze streszczającej dane dla każdego strumienia. Obie struktury będą stale rosnąć, co w efekcie doprowadzi do wyczerpania zasobów systemowych. Jeżeli jednak poczynimy pewne założenia dotyczące danych w obu strumieniach, możliwe jest określenie stałego rozmiaru wykorzystywanych struktur. Wiedząc, że wszystkie krotki o danym *IdZamówienia* i *IdProduktu* pojawiają się w strumieniu *Z*, zanim pojawiają się w strumieniu *R*, nie potrzebna staje się struktura streszczająca dane ze strumienia *R*. Dodatkowo, jeżeli wiadomo, że krotki ze strumienia *R* nadchodzą skupione w grupy z jednakowym *IdZamówienia*, to w strukturze streszczającej dla *Z* wystarczy przechowywać krotki, aż do momentu zaobserwowania zmiany wartości *IdZamówienia* w strumieniu *R*.

W rzeczywistości nie możemy poczynić aż tak daleko idących założeń. Z tego powodu opisywany algorytm wprowadza parametr tolerancji. Definiuje on, w jakich ramach obserwowane dane mogą nie stosować się do założonych ograniczeń. Jest on wyrażony liczbą krotek niespełniających założenia, jakie mogą się pojawić pomiędzy kolejnymi krotkami z oczekiwanymi danymi. Definicja ta nasuwa słuszne skojarzenia z parametrem *Slack*, definiowanym w klauzulach szeregujących operatorów systemu Aurora.

Właściwości, pod kątem których system analizuje dane, to zależności występujące pomiędzy dwoma strumieniami: złączenie wiele-do-jednego (ang. *many-to-one join*), integralność referencyjna (ang. *referential integrity*) oraz właściwości danych w pojedynczym strumieniu: grupowanie krotek (ang. *clustered-arrival*) i uporządkowanie krotek (ang. *ordered-arrival*). Warty podkreślenia jest fakt, że wykorzystywanie tego algorytmu nie powoduje obniżenia dokładności wyniku przetwarzania.

3. TEORIA STEROWANIA W REDUKCJI OBCIĄŻENIA

Metody redukcji obciążenia stosowane w przetwarzaniu strumieni danych wykorzystują techniki stosowane w teorii sterowania. Teoria sterowania jest nauką zajmującą się zachowaniem układów dynamicznych w czasie. Skupia się głównie na problemie sterowania układem w taki sposób, aby zachowywał się on zgodnie z oczekiwaniami. Ustala się pożądaną wartość wyjścia układu a następnie manipuluje jego wejściem tak, aby na wyjściu uzyskać zadaną wartość. Rozróżnia się dwa podstawowe sposoby sterowania układem: sterowanie w otwartej pętli oraz sterowanie ze sprzężeniem zwrotnym. Druga z metod, jest zdecydowanie bardziej interesująca w kontekście wykorzystania w technikach adaptacji systemów zarządzania strumieniami danych, ze względu na fakt, iż bierze pod uwagę wpływ dynamiki układu na wartość wyjściową.



Rys. 1. Schemat układu automatyki ze sprzężeniem zwrotnym

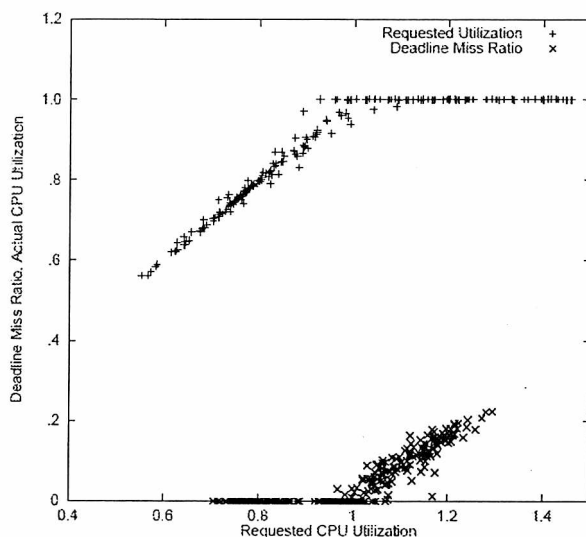
Na rysunku 1 przedstawiono schemat sprzężenia zwrotnego. Wyjście układu y jest porównywane z wartością zadaną r a różnica tych wielkości, zwana uchybem e , jest podawana na wejście regulatora C . Dzięki temu regulator posiada aktualną informację o błędzie sterowania i może tak zmieniać wartość wejściową u obiektu P , aby zmniejszyć uchyb do zera. Co za tym idzie, wielkość wyjściowa jest stale utrzymywana możliwie blisko wartości zadanej. Ważną częścią sprzężenia zwrotnego jest funkcja kontrolna odwzorowująca wartość uchybu na wartość wejściową obiektu P . Jej dobór oraz w ogóle możliwość zastosowania sterowania ze sprzężeniem zwrotnym wymagają od systemu, którym chcemy sterować spełniania pewnych założeń, które zostaną dalej przedstawione.

3.1. SPRZĘŻENIE ZWROTNE

W pracy [Tu 2004] autorzy przedstawili model SZSBD, podporządkowany wymogom związanym z wykorzystaniem sterowania ze sprzężeniem zwrotnym, jako metody adaptacji systemu w warunkach nieprzewidywalnego oraz potencjalnie nieograni-

czonego natężenia napływu danych do przetworzenia. Celem adaptacyjnym jest zwracanie wyników zapytań w określonym czasie. Wartością, na podstawie której można oceniać działanie układu, jest liczbie przekroczonych terminów (ang. *deadline misses*). Wykonanie zadań przetwarzania planowane jest z wykorzystaniem algorytmu EDF (ang. *Earliest Deadline First*), który poza tym, że, bezpośrednio wspomaga osiągnięcie celu adaptacyjnego, poprzez dążenie do wykonania zadań z najbliższym terminem, gwarantuje również dążenie liczbie przekroczonych terminów do zera przy pełnym obciążeniu procesora.

Jako wartość wejściową układu obrano żądane obciążenie procesora U_r . Można je wyznaczyć sumując szacowane obciążenie procesora przez każde z zaplanowanych zadań, co w przypadku przetwarzania danych przez łańcuch operatorów oznacza obciążenie, jakie wywoła przetwarzanie przez nie krotek napływających do systemu. Aby móc stosować metodę sterowania ze sprzężeniem zwrotnym, wartość wyjściowa układu powinna liniowo zależeć od wartości zadanej. Tutaj będą to dwie wartości albowiem w różnych przedziałach wartości U_r inna wartość spełnia to założenie. Dla przedziału poniżej wartości 1,0 jest to U_r , czyli faktyczne obciążenie procesora. Dla wartości większych od 1,0 proponuje się zastosowanie wspomnianej liczbie przekroczonych terminów wykonania M . Nie podano dowodu na fakt liniowej zależności wartości M od U_r . W pewnym ograniczonym zakresie powyżej stuprocentowego obciążenia procesora wydaje się ona jednak być do zaakceptowania, co więcej, jak wynika z rysunku 2, przypuszczenie to potwierdzają doświadczenia. Można tu zaobserwować, że założenie to jest spełnione w zakresie $[1,0; 1,3]$ co jest w zupełności wystarczające.



Rys. 2. Weryfikacja modelu liniowego [Tu 2004]

Obciążenie procesora redukuje się poprzez odrzucanie krotek ze strumieni wejściowych. Ponieważ spowoduje to obniżenie dokładności obliczeń, użytkownikowi należy udostępnić możliwość określenia wymaganej jakości usług (ang. *quality of service*) i pośrednio decydowania o tym, które krotki odrzucić. Także w opisywanej technice przewidziano taką możliwość.

Poziom redukcji obciążenia zostanie wyznaczony przez funkcję kontrolną, której dobór zależy od właściwości układu, jakim steruje. W prezentowanym rozwiązaniu obrana została funkcja regulatora proporcjonalno-całkująco-różniczkującego, która charakteryzuje się wrażliwością na sterowanie oraz dobrze sprawdza się w zakresie utrzymywania stabilności pracy w układach liniowych. Funkcje określające zależności liniowe pomiędzy wartościami wyjściowymi a zadanymi układu zostały określone następująco [Tu 2004]:

$$U(t+1) = U(t) + G_a(t)U_d(t) + D(t) \quad (1)$$

$$M(t+1) = M(t) + mG_a(t)U_d(t) + D(t) \quad (2)$$

Wartości w momencie $t+1$ wyznaczane są na podstawie pomiarów wykonanych podczas poprzedzającego cyklu sterowania. G_a określa wariancję zmiany obciążenia. Jest dodawana w celu uchwycenia błędu wynikłego z niedokładności oszacowania kosztu wykonania zadania, która pociągnęła za sobą nieprecyzyjne określenie żadanego obciążenia procesora. $U_d(t)$ określa wynik funkcji kontrolnej z poprzedniego cyklu, a funkcja określająca zakłócenia układu $D(t)$ odwzorowuje nieregularności w natężeniu danych napływających do systemu. Stała m w funkcji $M(t)$ określa poziom zależności M (liczbie przekroczonych terminów wykonania zadań) od U_r (żadanego obciążenia procesora), czyli nachylenie funkcji $M(U_r)$ widocznej na wykresie na rysunku 2 do osi odciętych. W pracy [Tu 2004], z której zaczerpnięto rysunek 2, wyznaczono tę stałą korzystając z regresji liniowej i otrzymano wartość 0,64.

Jako funkcję kontrolną obrano [Tu 04]:

$$U_d(t) = \alpha E(t) + \beta \sum_{i=t-W}^{t-1} E(i) \quad (3)$$

$$E(t) = \begin{cases} U_s - U(t), & \text{gdy } U(t) \leq 1 \\ m(M_s - M(t)), & \text{gdy } U(t) > 1 \end{cases} \quad (4)$$

gdzie W jest szerokością okna czasu, w jakim obliczamy część całkującą a stałe α i β reprezentują wagi odpowiednio członu proporcjonalnego i całkującego regulatora. Człon różniczkujący został pominięty ze względu na powodowanie przeszacowań.

Prezentowany model dostosowany jest do wymagań systemów czasu rzeczywistego z ostrymi wymaganiami czasowymi (ang. *hard real-time*). Przekroczenie terminu wykonania jest tu równoznaczne z usunięciem zadania z kolejki. W systemach z ła-

godnymi wymaganiami czasowymi (ang. *soft real-time*) stosowana wartość U_r nie jest już właściwa, ponieważ muszą zostać wzięte pod uwagę krotki już obecne w systemie. Dlatego też w takim wypadku należy stosować wartość tzw. całkowitego zakumulowanego obciążenia oznaczaną przez U_c . Można dokonać takiej zamiany bez innych konsekwencji, ponieważ związki liniowe z obranymi wartościami wyjściowymi układu także zachodzą.

3.2. REDUKCJA OBCIĄŻENIA

Wykorzystanie sprzężenia zwrotnego pozwala określić moment, w którym należy zredukować obciążenie oraz procentowo określa wielkość obciążenia, jakie należy zredukować. Do rozwiązania pozostaje kwestia „krotek-ofiar” redukcji obciążenia.

Aby znaleźć „ofiary” redukcji obciążenia należy wskazać strumień, z którego ma pochodzić krotka, wybrać konkretną instancję (w oparciu o określone kryteria) oraz wyznaczyć etap przetwarzania, na którym zostanie ona odrzucona, ponieważ w pewnych okolicznościach istotne jest przetworzenie krotki przez część operatorów przed jej odrzuceniem. W naszym przypadku operacje na jednej ścieżce przetwarzania są planowane do wykonania jako nierozłączna całość. Z tego względu częściowe przetwarzanie nie ma sensu i krotki odrzuca się zaraz po nadejściu do systemu. Dla porównania, w opisywanym dalej sposobie rozwiązania tego problemu w systemie Aurora, krotki mogą być odrzucone na każdym etapie przetwarzania. Kolejnym podobieństwem jest wykorzystanie funkcji użyteczności. W pracy [Tu 2004] jest ona zdefiniowana ogólnie dla każdego rodzaju właściwości wyniku przetwarzania. Użyteczność konkretnej krotki określana jest w następujący sposób:

$$U(\vec{q}) = \sum_{i=1}^n w_i u_i(q_i) \quad (5)$$

gdzie wektor $\vec{q}$ jest wektorem wartości parametrów krotki, q_i wartością jednego z tych parametrów, u_i określa funkcję użyteczności dla tego parametru, co do której zakłada się, że przyjmuje wartości z zakresu $[0, 1]$, jest rosnąca oraz wypukła [Tu 2004]. Użytkownik systemu określa swoje preferencje dotyczące jakości usług (ang. *quality of service*) poprzez określenie wektora wag $\vec{w}$, którego składowe pojawiają się w przytoczonej zależności, dla n oznaczającego liczbie branych pod uwagę parametrów mamy również $\sum_{i=1}^n w_i = 1$. Pomimo faktu, iż sposób wyznaczania użyteczności krotki został zdefiniowany generycznie, autorzy określają kilka najistotniejszych parametrów:

- **Opóźnienia krotek.** Oznacza ilość czasu, jaka może upłynąć pomiędzy utworzeniem krotki w systemie, a ukończeniem przetwarzania niesionych przez nią danych. Wartość ta posłuży do wyznaczania terminów wykonania zadań, któ-

rymi kierował się będzie planista EDF. Ze względu na możliwość zoptymalizowanej implementacji kolejki priorytetowej zadań z zestawem priorytetów z ograniczonej puli, autorzy proponują obieranie maksymalnych opóźnień również z zamkniętego zbioru. Jak wspomniano, wartość parametru dla funkcji użyteczności ma należeć do przedziału $[0, 1]$, a ona sama ma być rosnąca. W takim wypadku właściwą wartością tego parametru dla danej krotki będzie stosunek ilości czasu, jaki pozostał do przekroczenia terminu do wybranego maksymalnego opóźnienia. W ten sposób w miarę zbliżania się terminu zakończenia przetwarzania wartość parametru, a tym samym i użyteczność krotki będzie malała. Będzie ona natomiast maksymalna w chwili pojawienia się w systemie.

- **Częstotliwość prezentowania wyniku przetwarzania.** Pokazuje jak często na wyjściu powinna pojawiać się nowa wartość. Może chodzić o powiadamianie, co ustalony okres czasu lub też o generowanie ostrzeżenia w oparciu o wartości danych w strumieniu (na przykład: powiadamianie w wypadku, gdy cena akcji ulegnie zmianie o 1%). W takim wypadku warto określić parametr związany z wartością lub ze zmianą wartości. Funkcja użyteczności dla tego parametru w przykładzie podanym powyżej mogłaby być bliska zeru dla całego przedziału reprezentującego zmianę wartości atrybutu krotki poniżej 1%. Pozwoliłoby to zredukować obciążenie systemu bez żadnej straty użyteczności wyniku.
- **Procent utraty danych.** Ten parametr pozwala pośrednio określić wymaganą dokładność przetwarzania. Wartość tego parametru będzie jednakowa dla każdej krotki pochodzącej z danego strumienia.

Najprostszym sposobem wyznaczenia ofiar redukcji obciążenia jest losowe dobieranie krotek spośród wszystkich dostarczonych do systemu. Posiadając jednak wskazania użytkownika odnośnie użyteczności danych zdefiniowane w wyżej opisany sposób można podejść do tego zadania w sposób gwarantujący efekt najbardziej pożądanym przez użytkownika systemu. Cała procedura staje się też nieporównanie bardziej skomplikowana. Ze względu na fakt, iż obrana metoda powinna również w jak najmniejszym stopniu absorbować procesor konieczne może się stać wprowadzenie technik heurystycznych tak jak to zaproponowano w [Tu 2004].

4. FUNKCJE UŻYTECZNOŚCI W REDUKCJI OBCIĄŻENIA

Jedną z podstawowych cech strumieni danych jest fakt, iż są one nieograniczone. W praktyce oznacza to, że liczba danych napływających do systemu w jednostce czasu jest ograniczona tylko możliwościami przesyłowymi łącza. Jako, że możliwości obciążeniowe każdej maszyny są ograniczone, prawdopodobny jest scenariusz, w którym nie będzie ona sobie w stanie poradzić z przetworzeniem napływających danych. System zarządzania strumieniami danych musi radzić sobie z takimi problemami.

Pomocne są oczywiście metody zmniejszania złożoności obliczeniowej zapytań poprzez przekształcenia czy też planowanie wykonania zapytań z myślą o zminimalizowaniu czasu procesora przeznaczonego na czynności niezwiązane bezpośrednio z wykonywaniem obliczeń wykorzystywanych w zapytaniu jak przełączanie kontekstu, zapisywanie lub odczytywanie danych z nośników danych czy też budowanie struktur przechowujących dane w pamięci operacyjnej komputera. Żadna z powyższych metod jednak, o ile usprawnia proces przetwarzania danych to nie gwarantuje utrzymania stabilnej pracy systemu w czasie, gdy liczba napływających danych w jednostce czasu przewyższa możliwości obliczeniowe komputera.

Rozwiązaniem umożliwiającym kontynuowanie pracy systemu jest redukcja obciążenia w strumieniu danych (ang. *load shedding*). Podjęcie tego kroku obniża dokładność prezentowanego wyniku i dlatego metoda ta musi być stosowana w połączeniu z wykorzystaniem wytycznych określających wymagane przez aplikację docelową parametry wyniku. System powinien dążyć do sytuacji, w której rezygnuje z przetwarzania odpowiedniej liczby krotek wybierając na ofiary te, których brak w najmniejszym stopniu przyczyni się do obniżenia się przydatności wyniku. Dlatego też dla każdej aplikacji administrator systemu powinien za pomocą szeregu wskaźników określić właściwości wyniku, jakie są najbardziej istotne, np. wymaganą jakość usług (ang. *quality of service*).

Użyteczność danych zwracanych przez system Aurora mierzona jest w wartościach z przedziału $[0..1]$, gdzie 1 oznacza użyteczność maksymalną a 0 dane bezużyteczne. Istnieje kilka sposobów na jej obliczenie dla każdego wyjścia z systemu. Dla każdej aplikacji korzystającej z danych zwracanych przez system, co innego stanowiło będzie o użyteczności danych. Na przykład dla aplikacji czasu rzeczywistego, dla której czas otrzymania wyniku jest rzeczą najistotniejszą, użyteczność będzie funkcją świeżości danych. Inna aplikacja może tolerować opóźnienie w przekazaniu wyniku, ale oczekiwać wyniku dostarczonego z określoną dokładnością. W tym wypadku miarą użyteczności danych będzie stosunek liczby krotek, jakie napłynęły do systemu do liczby krotek przetworzonych w procesie obliczania wyniku. Jak wspomniano wcześniej, nie zawsze możliwe jest przetworzenie wszystkich dostarczonych danych z uwagi na ryzyko destabilizacji pracy systemu.

W każdym momencie działania systemu Aurora znana jest dokładna wartość użyteczności danych wpływających z każdego wyjścia systemu. Wartościami tymi kieruje się system, w celu podjęcia najkorzystniejszych działań, w momencie pojawienia się zagrożenia przeciążenia systemu. Właściwie posługuje się nimi przede wszystkim po to, aby zdecydować czy system zagrożony jest destabilizacją. Do tego, żeby o tym zdecydować potrzeba jednak określenia warunków destabilizacji. Jednym z nich byłoby przepełnienie buforów pamięci napływającymi danymi ale ponieważ system Aurora korzysta, w trakcie przetwarzania, oprócz pamięci operacyjnej również z pamięci dyskowej sytuacja taka jest mało prawdopodobna.

Warunki destabilizacji pracy systemu określane są przez specyfikację funkcji użyteczności danych przez administratora systemu. Dla każdego wyjścia określa on, w jaki sposób zmienia się użyteczność danych w zależności od zmiany wartości parametru za pomocą trzech funkcji. Są to mianowicie: funkcja użyteczności w zależności od świeżości danych oraz funkcja użyteczności w zależności od procentowego stosunku krotek wykorzystanych przy produkcji wyniku do wszystkich jakie napłynęły. Trzecia funkcja pozwala określić na ile użyteczność zależy od wartości zwracanych krotek. Można sobie wyobrazić, iż korzyści przynosi możliwość określenia użyteczności poszczególnych wartości. Na przykład, dla aplikacji odbierającej dane z systemu SZSBD, mogą być istotne jedynie wartości z pewnego zakresu ponad tzw. „punktem alarmowym”, dla których użyteczność będzie określona jako wartość 1 a dla pozostałych jako wartość 0.

W dalszej części pracy przedstawiono, w jaki sposób funkcje użyteczności wykorzystywane są w procesie redukcji obciążenia.

4.1. REDUKCJA OBCIĄŻENIA

Odrzucanie krotek odbywa się przy pomocy dodatkowych operatorów wstawianych do istniejącego grafu. Są to operatory unarne zwracające mniejszą liczbę krotek niż otrzymują. Istnieją dwa takie operatory różniące się sposobem wyboru odrzucanych krotek.

W prostszym przypadku, wybór dokonywany jest poprzez losowanie z prawdopodobieństwem odrzucenia równym procentowo określonej liczby krotek, jakie mają zostać pominięte w dalszym przetwarzaniu.

Drugą możliwością jest tzw. odrzucanie semantyczne (ang. *semantic drop*), które dyskryminuje krotki ze względu na wartości atrybutów. W tym wypadku odrzucane są krotki niespełniające określonego predykatu.

Dla każdego ze wspomnianych operatorów istnieje osobny wariant algorytmu, który ma za zadanie wybór optymalnych dla nich miejsc w grafie operatorów. Każdy z nich analizuje istniejącą sieć operatorów zanim rozpocznie się przetwarzanie. Jako wynik działania podawana jest lista możliwych kombinacji uporządkowana według „agresywności” scenariusza. W ten sposób, w momencie nadmiernego obciążenia, system może zastosować rozwiązanie odpowiednie do aktualnego natężenia danych i odrzucić jedynie taką liczbę krotek, jaka jest konieczna.

System Aurora reaguje na zmiany w natężeniu danych. W regularnych odstępach czasu bada stosunek aktualnego zapotrzebowania na cykle procesora do maksymalnej wydolności systemu pomniejszonej o kilka procent dla zachowania stabilności. Aktualne zapotrzebowanie stanowi suma cykli, jakie są wymagane do przetworzenia danych napływających do systemu i cykli, jakie muszą zostać poświęcone na przetworzenie krotek już w systemie obecnych.

Jeżeli całkowite zapotrzebowanie na pracę procesora przekroczy wyznaczone maksimum, system musi rozpocząć proces odrzucania krotek poprzez wstawienie do grafu dodatkowych operatorów za tę czynność odpowiedzialnych. Z drugiej strony, jeżeli natężenie danych spadnie, możliwe jest usunięcie części wcześniej dodanych operatorów.

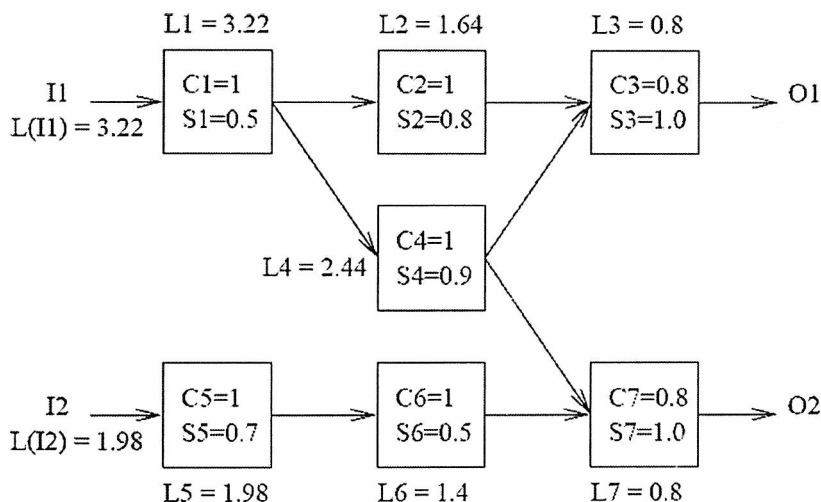
Wstawianie i usuwanie operatorów zmniejszających natężenie danych odbywa się za pomocą planu składającego się z listy wcześniej przygotowanych scenariuszy (ang. *load shedding road map*). W każdym cyklu system znajduje się na określonej pozycji we wspomnianej liście i, w zależności od tego czy potrzebne jest zmniejszenie natężenia danych czy też możliwe jest przetworzenie dodatkowych krotek, przesuwa wskaźnik w odpowiednim kierunku. Dodatkowo wiadomo dokładnie o ile cykli procesora zmniejszy się zapotrzebowanie w wypadku użycia danego scenariusza. Dlatego możliwe jest poruszanie się o kilka pozycji, do razu wybierając rozwiązanie, które przywróci system do stanu stabilnego. Pojawia się tu jednak pewna trudność, która wynika z faktu, iż liczba oszczędzonych cykli nie daje się wyznaczyć w prosty sposób, zanim system zacznie swoje działanie, ponieważ jest zależna od natężenia ruchu na poszczególnych wejściach, a to znane jest dopiero w czasie przetwarzania. Z drugiej strony, przygotowanie gotowej listy rozwiązań przed startem działania systemu jest konieczne, z tego względu, że poszukiwanie optymalnego scenariusza odrzucania krotek odbywa się zazwyczaj w momencie bardzo dużego obciążenia systemu i sam proces wyboru powinien, w jak najmniejszym stopniu, absorbować procesor. Rozwiązanie tego dylematu, jakie proponują autorzy Aurory polega na wyliczaniu tzw. współczynników obciążenia (ang. *load coefficients*). Dla każdego operatora w grafie, możliwe jest obliczenie takiego współczynnika w sposób statyczny. Określa on koszt w cyklach, jaki jest związany z przetworzeniem danych „niesionych” przez jedną krotkę.

4.2. WSPÓŁCZYNNIKI OBCIĄŻENIA

Właściwościami każdego operatora w grafie są jego koszt wyrażony w cyklach procesora oraz selektywność (ang. *selectivity*). Pierwsza z tych właściwości określa liczbę cykli procesora potrzebnych do przetworzenia jednej krotki. Należy tutaj zwrócić uwagę na to, że chodzi o koszt przetworzenia jednej krotki przeczytanej na wejściu. Selektywność natomiast to liczba krotek, jaka jest zwracana na wyjściu operatora z jednej krotki odczytanej na wejściu. Korzystając z tych właściwości można obliczyć współczynnik obciążenia L dla każdego punktu w grafie przy pomocy poniższej zależności [Tatbul 2003]:

$$L = \sum_{i=1}^n \left(\prod_{j=1}^{i-1} S_j \right) * C_i \quad (6)$$

gdzie S_j oznacza selektywność operatora j , a C_i koszt operatora i . Oznacza to, że dla pojedynczego operatora współczynnik obciążenia otrzymujemy poprzez przemnożenie kosztu danego operatora z iloczynem selektywności operatorów poprzedzających go. Przy rozgałęzieniach współczynniki poddrzew sumuje się pamiętając o tym, że koszt liczony jest dla krotek przeczytanych na wejściu każdego operatora, selektywność korzenia wlicza się przy obliczeniach dla każdego poddrzewa, ale koszt korzenia bierzemy pod uwagę tylko raz.



Rys. 3. Przykładowy graf operatorów wraz ze współczynnikami obciążenia [TATBUL 03]

Na rysunku 3 przedstawiono przykładowy graf operatorów z zaznaczonymi selektywnością oraz kosztem przetwarzania, dla którego zostały wyznaczone współczynniki obciążenia przy każdym operatorze.

W czasie działania systemu możliwe jest obliczenie rzeczywistego obciążenia jako iloczynu współczynnika i zaobserwowanego natężenia danych. Jak już wspomniano, obciążenie całkowite systemu to suma zapotrzebowania na cykle nowych danych i danych znajdujących się w systemie. Pierwszy składnik jest sumą rzeczywistych obciążeń dla wszystkich wejść systemu. Do obliczenia drugiego składnika, stosowana jest stała systemowa oznaczająca minimalny procent, o jaki skracają się, w jednostce czasu, kolejki danych na wejściach operatorów.

4.3. WYZNACZANIE SCENARIUSZY

Każdy scenariusz składa się z zestawu operatorów odrzucających krotki, listy wejść, z których krotki zostaną odrzucone wraz z współczynnikami zaoszczędzonych

cykli (ang. *cycle saving's coefficients*) oraz listy wyjść z systemu wraz z przypisanymi im procentowymi wskaźnikami dostarczonych danych. Każdy z zestawu operatorów odrzucających krotki ma określone miejsce, w które zostanie wstawiony oraz procent krotek, jaki ma za zadanie odrzucić. Współczynniki zaoszczędzonych cykli przemnożone przez aktualne natężenie danych na odpowiednich wejściach dają w sumie oszczędność w cyklach w wypadku wprowadzenia w życie danego scenariusza. Ostatni element scenariusza służy do określenia jak zmieni się jakość wyniku po wprowadzeniu w życie scenariusza na każdym z wyjść.

Do tej pory omówiono głównie zagadnienie wyboru momentu rozpoczęcia przetwarzania z odrzucaniem krotek i zwrócenia aproksymowanego wyniku. Zadaniem poniżej opisywanych algorytmów jest stwierdzenie, w którym miejscu w grafie operatorów umieścić operatory odrzucenia oraz ile i jakie krotki odrzucać. Jak zostało to już powiedziane, w systemie Aurora istnieją dwa takie algorytmy, inny dla każdego rodzaju operatorów odrzucenia – losowego i z użyciem filtra.

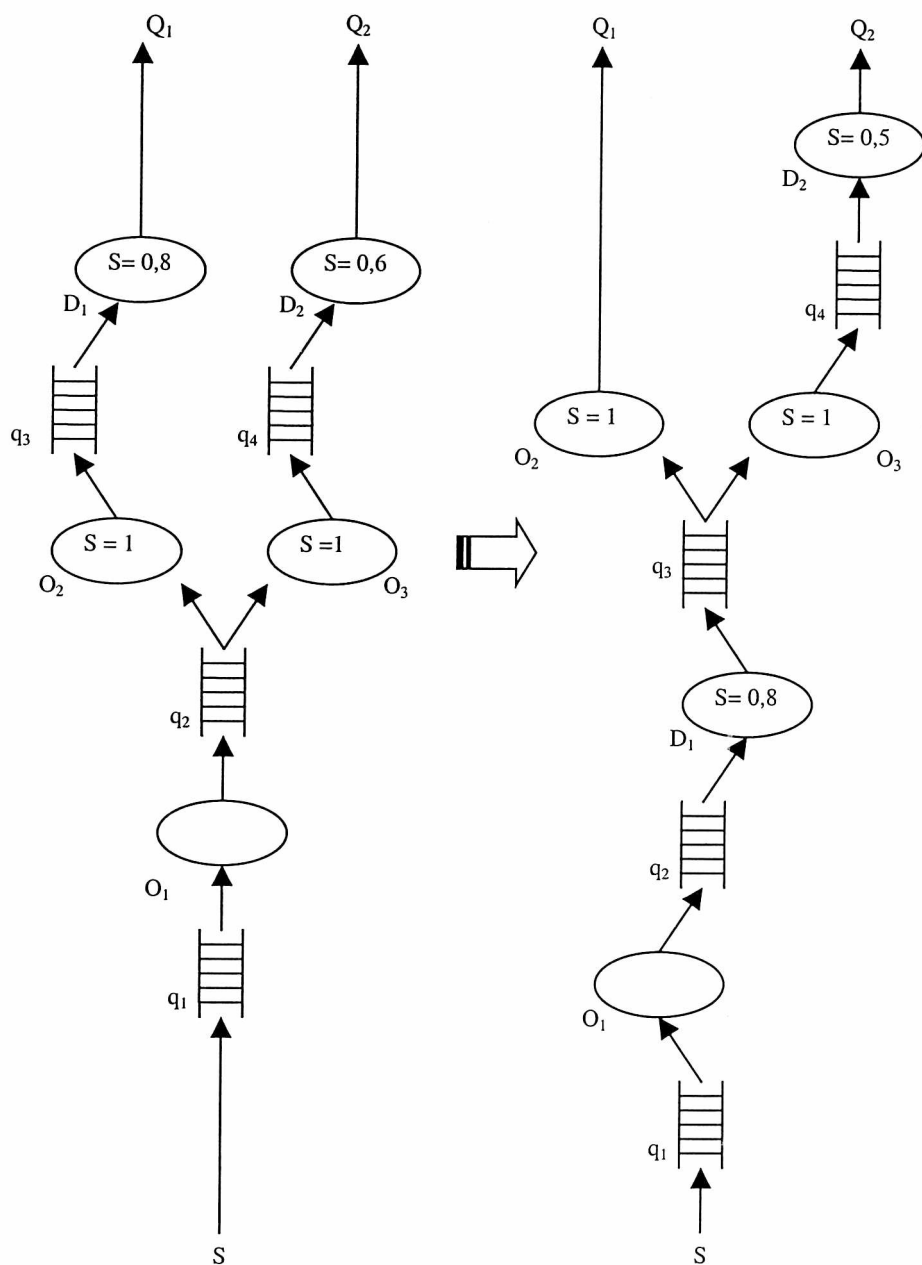
Zasadą działania obu algorytmów jest dodawanie operatorów przed kolejne wyjścia z systemu i przesuwanie ich w stronę wejść, w ten sposób zwiększany jest zasięg działania operatora oraz liczbę zaoszczędzonych cykli. Każdy uzyskany w ten sposób graf jest kolejnym scenariuszem w generowanym planie. Z każdym krokiem obliczane są współczynniki zaoszczędzonych cykli dla każdego z wejść oraz wskaźniki procentowe danych wziętych pod uwagę przy wyznaczaniu wyniku dla każdego z wyjść. Algorytm kończy działanie w momencie, gdy ostatni wskaźnik dla każdego wyjścia z systemu jest równy 0.

4.4. ALGORYTM ODRZUCANIA LOSOWEGO

Działanie algorytmu rozpoczyna się od wyznaczenia wyjścia, za które wstawimy dodatkowe operatory. Będzie to wyjście, dla którego funkcja użyteczności wyniku maleje najwolniej wraz ze wzrostem liczby odrzuconych krotek. Warunek ten ma na celu zagwarantowanie, że odrzucanie krotek w trakcie przetwarzania, przyczynia się do spadku użyteczności wyniku w jak najmniejszym stopniu.

W systemie zdefiniowana jest stała `STEP_SIZE`. Określa ona procentowo maksymalną liczbę krotek, jakie można odrzucić z jednego wyjścia w każdym kroku działania algorytmu. Jest ona szczególnie użyteczna w wypadku, gdy istnieje więcej niż jedno wyjście, w którym nachylenie wykresu użyteczności do osi rzędnych jest minimalne w określonym przedziale. Przyjmuje się wówczas liczbę odrzucanych krotek równą długości przedziału, w którym owo nachylenie jest minimalne, bądź też równą wartości stałej `STEP_SIZE`. Wybór zależy od tego, która z wartości jest mniejsza.

W każdym kolejnym kroku algorytmu operatory wstawione przy każdym z wyjść z systemu są przesuwane wzdłuż ścieżek prowadzących do wejść. Jako że dzięki temu operatory odrzucania obejmują swoim działaniem z każdym krokiem więcej operatorów przetwarzających dane to zyskuje się coraz więcej cykli procesora.



Rys. 4. Przesuwanie operatorów losowo odrzucających krotki przez rozgałęzienie w grafie operatorów

Postępowanie takie nie wymaga żadnych dodatkowych działań o ile na działanie operatora, z którym zamieniamy operator odrzucania podczas operacji przesunięcia nie ma wpływu zamiana miejsc. Do tej grupy należą operatory takie jak `Filter` czy `Map`. W przypadku, gdy napotkany operator ma bardziej skomplikowane działanie, jak np. `Aggregate` przesunięcie nie jest dozwolone, ponieważ taka operacja mogłaby w nieprzewidywalny sposób zniekształcić działanie systemu.

Dodatkowe działania są zawarte w algorytmie postępowania dla operatorów, dla których liczba wejść różni się od liczby wyjść. W wypadku rozdzielania wejścia na kilka wyjść sieć operatorów zmienia się w sposób podobny do tego zaprezentowanego na rysunku 4. Operator O_1 , O_2 i O_3 są zwykłymi operatorami przetwarzającymi dane, operatory D_1 i D_2 dokonują odrzucenia krotek dobierając ofiary losowo z początkowym prawdopodobieństwem, odpowiednio, 20% i 40%.

Zanim może zostać dokonana modyfikacja układu wymagane jest, aby przy wszystkich wyjściach znajdował się operator odrzucania. Gdy ten warunek jest spełniony wybierany jest operator z najwyższym współczynnikiem selektywności, czyli odrzucający najmniej krotek i zostaje przeniesiony przed operator rozdzielający. Zmianie ulega parametr określający procent odrzucanych krotek dla pozostałych operatorów odrzucania. Przyjmuj on nową wartość daną jako iloraz wartości tego parametru dla operatora, który został przesunięty i starej wartości parametru. Warto zauważyć, że tak przeprowadzona operacja przesunięcia operatora odrzucenia nie zmieni użyteczności wyniku tylko w wypadku, gdy wszystkie operatory przetwarzające dane, umieszczone za rozgałęzieniem będą miały selektywność równą jeden, tak jak to ma miejsce na rysunku 4. W przeciwnym wypadku selektywność operatora odrzucającego krotki, który został przesunięty przed rozgałęzienie należy zmodyfikować tak, aby selektywność całego układu nie uległa zmianie.

W wypadku, gdy wszystkie operatory odrzucenia miałyby taką samą wartość parametru mogą one zostać usunięte, gdyż ich selektywność i tak wynosiłaby 1. W przypadku napotkania operatora z wieloma wejściami, jak `Union`, przesuwany operator odrzucenia może być skopiowany na wszystkie wejścia.

4.5. ALGORYTM ODRZUCANIA Z FILTREM

Algorytm odrzucania z filtrem jest podobny do algorytmu odrzucania losowego. Zasadnicza różnica polega na rodzaju operatorów, jakie wstawiane są do grafu reprezentującego zapytania. O ile poprzednio wstawiano operatory odrzucania losowego, które nie brały pod uwagę właściwości krotek podczas podejmowania decyzji o odrzuceniu, teraz wartości atrybutów krotki decydują o tym czy krotka zostanie przetworzona przez system bądź odrzucona. Dla szeregu aplikacji najistotniejsze są bowiem wyniki z określonego zakresu wartości, np. w aplikacji medycznej monitorującej sygnały pochodzące z czujników najistotniejsze będą wartości w przedziałach alarmo-

wych, które mają informować o pogorszeniu sytuacji i dlatego krotki zawierające taką informację, nie powinny zostać odrzucone. O ile wartości z pozostałych przedziałów także mogą być interesujące to pominięcie ich zmniejszy użyteczność wyznaczanego wyniku w mniejszym stopniu. Autorzy systemu doszli do wniosku, że biorąc pod uwagę informacje, jaką dostarczają niektóre wartości atrybutów krotki, podczas procesu odrzucania krotek możliwe jest, z jednej strony zaoszczędzenie większej liczby cykli przy jednoczesnym mniejszym obniżeniu użyteczności zwracanego wyniku a z drugiej strony zapewnienie dokładnego wyniku w wymagających tego przypadkach.

Podstawową różnicą, w porównaniu z omówionym wcześniej algorytmem, jest konieczność określenia predykatu, jaki spełniać powinny krotki, aby zostać odrzucone. Kolejną konsekwencją różnicy w rodzaju użytych operatorów, jest różnica w postępowaniu z rozgałęzieniami, na ścieżce prowadzącej w stronę wejść systemu. Także tutaj dochodzi do połączenia wszystkich operatorów odrzucenia w jeden. Aby zachować spójność w działaniu grafu nowo powstały operator musi zawierać predykat będący sumą logiczną predykatów składowych.

Definiowanie predykatów dla operatorów odrzucających poprzez filtrowanie opiera się na funkcji określającej przydatność wyniku w zależności od wartości atrybutu krotki. Na jego podstawie oraz na podstawie znajomości histogramu wartości danego atrybutu dla krotek wyznaczanych przez system, Aurora generuje funkcję używaną przez algorytm z operatorami odrzucającymi krotki losowo. Znajomość tej funkcji pozwala na określanie w kolejnych krokach procentowej liczbie krotek, jakie każdy z operatorów musi odrzucić tak jak to zostało już wcześniej opisane. Przy filtrującym operatorze odrzucania musi być określony predykat dyskryminujący krotki ze względu na wartość atrybutu. Dlatego też następnym krokiem będzie określenie przedziałów, z jakich należy odrzucać krotki, aby z jednej strony uzyskać wymagany procent odrzuconych krotek a z drugiej jak najmniej obniżyć użyteczność. Aby zobrazować ten proces najprościej będzie posłużyć się przykładem. Założmy, że w pewnym momencie działania algorytmu zdecydował on o konieczności odrzucenia 20% krotek. Funkcja użyteczności wynosi 0,2 w przedziale wartości atrybutu A [0..50] i 1,0 w przedziale [50..100]. Dodatkowo wiadomo, że wartość atrybutu A dla 40% krotek znajduje się w przedziale [0..50]. Stąd wiadomo, że wystarczy odrzucić połowę krotek z przedziału o najniższej użyteczności, aby zapewnić stabilne działanie systemu. Autorzy systemu w pracy [TATBUL 03] proponują, aby użyć filtra z predykatem ustawionym na $A > 25$, który odrzuci wszystkie krotki z wartością atrybutu A z przedziału [0..25]. Takie rozwiązanie jest poprawne tylko wtedy, jeżeli wiadomo, że rozkład liczby krotek pomiędzy przedziałami [0..25] i [25..50] jest równy. W przeciwnym przypadku należałoby zastosować filtr z predykatem $A > 50$ oraz wstawić za nim operator odrzucenia losowego z prawdopodobieństwem odrzucenia 50%.

5. PODSUMOWANIE

W pracy omówiono techniki adaptacyjne stosowane w aplikacjach przetwarzających strumienie danych. Część z nich pozwala na zmniejszanie zużycia zasobów systemowych bez strat dokładności wyniku obliczeń. Przykładem może być wykorzystywanie wykrytych cech charakterystycznych dla danych w strumieniu. Potrzeba przetwarzania nieograniczonych strumieni danych wymusza uwzględnienie pewnej tolerancji dla niedokładności wyniku przetwarzania, wynikającej z odrzucenia części napływających do systemu krotek w celu zredukowania obciążenia systemu. Przedstawiono kilka technik pozwalających określić ile krotek odrzucić, na jakim etapie przetwarzania je odrzucić oraz również, które z nich odrzucić, aby osiągnąć mniejsze zapotrzebowania na zasoby systemowe, jednocześnie niewiele tracąc na dokładności prezentowanego wyniku obliczeń.

LITERATURA

- [Abadi 2003] Abadi D., Carney D., Cetintemel U., Cherniack M., Convey C., Lee S., Stonebraker M., Tatbul N., Zdonik S., *Aurora: A New Model and Architecture for Data Stream Management*, 2003.
- [AB 2003] Arasu A., Babcock B., Babu S., Datar M., Ito K., Nishizawa I., Rosenstein J., Widom J., *STREAM: The Stanford Stream Data Manager*. In: ACM SIGMOD, 2003.
- [Tu 2004] Tu Y.C., Xia Y., Prabhakar S., *Quality of Service Adaptation in Data Stream Management Systems: A Control-Based Approach*, Proceedings of the 30th VLDB Conference, 2004.
- [Tatbul 2003] Tatbul N., Cetintemel U., Zdonik S., Cherniack M., Stonebraker M., *Load shedding in a data stream manager*, VLDB Conference, 2003.

ADAPTATION AND APPROXIMATION STREAM DATA PROCESSING METHODS

Effective methods of processing unbounded streams of data require different techniques than used in traditional Database Management Systems (DSMS). Research are conducted toward finding new methods ensuring data stream applications to run efficiently and reliably.

This paper introduces the reader to a set of techniques designed to serve as a means of creating a data stream processing system ready to tackle difficulties encountered when processing unbounded streams of data in real time offering maximum efficiency and a flexibility level high enough to allow use in development of applications handling data from any domain.

Zygmunt MAZUR*
Przemysław PAWLUK

PROBLEM ANALIZY DANYCH W SIECIACH SPOŁECZNYCH

Praca prezentuje zastosowanie baz danych w dziedzinie z pogranicza informatyki i psychologii czyli sieci społeczne (ang. *Social Networks*). Ze specyfiką sieci społecznych wiąże się wiele problemów w tzw. warstwie danych, a więc problemów związanych ze składowaniem oraz przetwarzaniem danych. Analiza sieci społecznej (SNA), w celu ekstrakcji z niej informacji, wiąże się z przetwarzaniem dużej liczby danych. To powoduje konieczność poszukiwania nowych, bardziej efektywnych metod przetwarzania i budowania struktur dostosowanych do specyfiki zagadnienia.

1. WSTĘP

Sieć jest zbiorem węzłów połączonych krawędziami. Podobnie sieć społeczna (ang. *Social Network*) jest zbiorem czy też mapą węzłów. Węzły sieci społecznej nazywane są aktorami, krawędzie natomiast zwane są w tym przypadku relacjami [HannRidd 2005]. Taka terminologia bliższa jest wycinkowi rzeczywistości, modelowanemu przez sieć społeczną, bowiem modeluje ona relacje pomiędzy osobami lub organizacjami. Dzięki temu możemy za jej pomocą modelować czy też badać zarówno związki między ludźmi w sieci, współpracę pomiędzy autorami publikacji czy drzewa genealogiczne i wiele podobnych.

W literaturze występują także byty zwane wirtualnymi sieciami społecznymi (ang. *Virtual Social Network*). Wirtualna sieć społeczna jest to sieć społeczna przeniesiona w środowisko Internetu, w której do komunikacji wykorzystywane są środki elektroniczne np. różnego typu komunikatory czy też poczta elektroniczna. Takie właśnie sieci społeczne są podstawą rozważań w tej pracy. Wybór taki spowodowany jest tym, że sieci społeczne są zbyt ogólnym i szerokim tematem. Sieci wirtualne pozwalają na zawężenie dziedziny, a dodatkowo, ze względu na swój charakter, silnie związane są z technologiami informatycznymi. Aktorzy w wirtualnych sieciach społecznych identyfikowani są przez unikalne identyfikatory (adresy internetowe lub numery identyfi-

* Instytut Informatyki Stosowanej, Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27, zygmunt.mazur@pwr.wroc.pl

kacyjne komunikatorów). Relacje odpowiadają komunikacji między tak identyfikowanymi adresami. Jednakże napotyka się wiele problemów związanych np. z jednoznacznością identyfikacją aktorów spowodowanych tzw. adresami grupowymi, które pozwalają na komunikację grup aktorów. Przykładem takich adresów grupowych mogą być adresy kontaktowe w firmach postaci kontakt@firma.com czy adresy grup mailingowych. Zagadnienia te jednak nie stanowią istoty tej pracy, dlatego nie są one szczegółowo omawiane.

Informacje związane z komunikacją w Internecie (ale nie tylko tam) stanowią cenny towar. Pozwalają one nie tylko na śledzenie kanałów komunikacyjnych, ale także na stymulację komunikacji w sieci. Stanowi to istotny bodziec napędzający rozwój tej dziedziny funkcjonującej na pograniczu socjologii, psychologii i informatyki. W Internecie pojawia się coraz więcej portali bazujących na mechanizmach sieci społecznych. Przykładami mogą być LinkedIn (www.linkedin.com) czy polski Spinacz (www.spinacz.pl). Cele działania tego rodzaju witryn są najróżniejsze, od czysto rozrywkowych, gdzie można nawiązać znajomości i znaleźć przyjaciół, po portale profilowane, na których można odszukać specjalistów z rozmaitych dziedzin.

Analiza sieci społecznych (ang. *Social Network Analysis* – SNA) jest metodą badania struktury społecznej i wzorów relacji pomiędzy ludźmi i grupami, pozwala również na badanie społecznych konsekwencji tych struktur np. alokacji zasobów, przepływów informacji, zmiany poglądów itd.

W dalszej części pracy przedstawione zostaną teoretyczne podstawy budowy sieci społecznych oraz metody ich analizy. Szczególna uwaga zwrócona zostanie na zagadnienie przetwarzania danych, struktury stosowane do przechowywania danych oraz problemy występujące w analizie sieci społecznych.

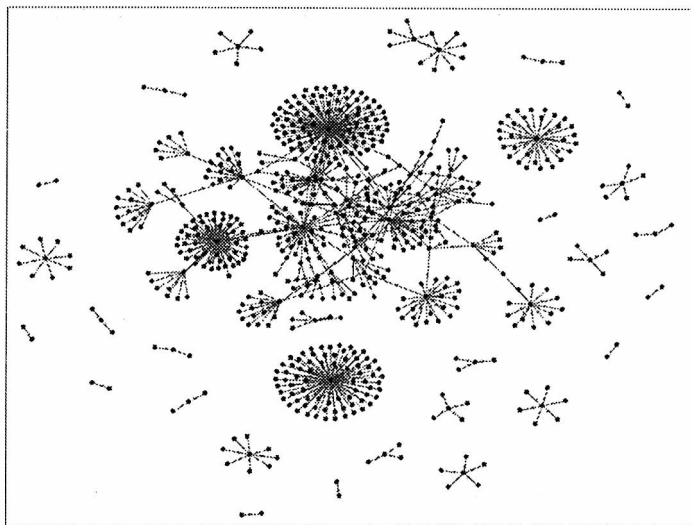
2. IDEA SIECI SPOŁECZNYCH

Sieć społeczna definiowana jest jako zbiór lub mapa relacji między osobami. Opisuje ona zarówno występowanie tych relacji jak i ich siłę. Pojedyncza osoba jest zatem węzłem sieci, krawędzie stanowią natomiast połączenia czy też „linki” stanowiące reprezentację relacji łączących ludzi w sieci [Jensen 2004]. Z sieciami społecznymi wiążą się różne miary określające ilość połączeń, spójność, jednostki centralne czy najbardziej aktywne w sieci.

Sieci społeczne tworzą rozbudowane i skomplikowane struktury [rysunek 1]. Połączenia tworzone są na podstawie własnych doświadczeń, wiedzy i zaufania. Zaufanie to oraz podobieństwo zainteresowań między ludźmi okazuje się być istotną cechą sieci społecznych [Goldbeck 2005, GoldHend 2004, Jensen 2004].

Zazwyczaj sieci społeczne, takie jak Friendster czy LinkedIn, są utworzone i utrzymywane przez komercyjne instytucje. Istnieją jednak sieci nieformalne związane z różnymi usługami dostępnymi za pośrednictwem Internetu (np. komunikatorów czy

poczty elektronicznej). Także autorzy powiązanych ze sobą hiperłączami stron internetowych stanowią swoistą sieć społeczną. Pojęcie to jest na tyle ogólne, że może z powodzeniem pomieścić w sobie powiązania genealogiczne danej rodziny, komunikację w organizacji czy nawet posłużyć do rozwiązywania problemów komunikacyjnych w miastach.



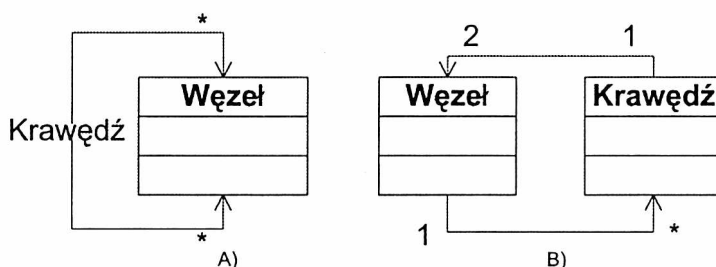
Rys. 1. Wizualizacja sieci społecznej
[źródło: <http://trust.mindswap.org/trustViz.shtml>]

W literaturze opisane są teoretyczne metody analizy sieci społecznej od metod polegających na całkowitej analizie sieci (ang. *full network method*), metodę kuli śnieżnej (ang. *snowball method*) czy też metody typu ego-centric. Metody te stanowią teoretyczne podejście do problemu analizy danych zawartych w sieci społecznej. Są to metody czasochłonne, szczególnie w sytuacjach, kiedy dochodzi do całkowitej analizy sieci społecznej, a więc do rozważenia wszystkich istniejących połączeń między węzłami. Dodatkowo, tego typu metody działające na grafach, nie są wspierane przez obecnie proponowane bazy danych. „Wspieranie” oznacza tu możliwość wygodnego, a więc możliwie łatwego, bezpiecznego oraz efektywnego, wykorzystania danych metod [HannRidd 2005].

3. STRUKTURY DANYCH W SIECIACH SPOŁECZNYCH

Sieci społeczne są przykładem struktur o charakterze grafowym. W pracy [SanGut 2006] autorzy przedstawili dwa podejścia do tego typu struktur. Pierwsze z nich to proste związki węzłów (typu wiele-do-wiele) połączonych krawędziami jednego

typu. Drugie podejście umożliwia połączenie węzłów różnymi typami relacji. Wprowadza się więc relacje heterogeniczne. Podejścia te zostało zilustrowane na rysunku 2.



Rys. 2. Ilustracja podejścia homogenicznego (A) i heterogenicznego (B) do sieci społecznych (oprac. własne na podstawie [TsDiCa 2005])

Podejście pierwsze jest najczęściej stosowane w implementacjach wykorzystujących relacyjne bazy danych. Należy zwrócić uwagę na fakt, iż w podejściu heterogenicznym (rysunek 2B) krawędzie z prostych połączeń między węzłami zmieniają się w klasy, które mogą posiadać dodatkowe atrybuty.

Przy tak widzianych modelach danych konieczne okazuje się wykorzystanie operatora, który w pracy [SanGut 2006] nazwany jest *recursive JOIN* (RJOIN), co przetłumaczyć można jako operator złączenia rekursywnego. Standard języka SQL nie dopuszcza takiego złączenia przez co utrudnia zastosowanie baz relacyjnych w dziedzinie sieci społecznych. Jednakże wydaje się, że rozszerzanie istniejących i dobrze zbadanych standardów może być drogą do zbudowania skalowalnego rozwiązania pozwalającego na przechowywanie i manipulowanie danymi o charakterze grafowym. Pytanie jakie należałoby postawić podejmując próby implementacji takiego rozszerzenia dotyczy możliwości efektywnej implementacji mechanizmów i algorytmów działających na grafach.

Przykładem sytuacji, w której zastosowanie operatora RJOIN znacznie ułatwiłoby pozyskiwanie danych jest zapytanie o odległość węzłów w sieci. Niech zatem omawianym wycinkiem rzeczywistości będzie zbiór danych dotyczących publikacji (np. DBLP). Autorzy publikacji tworzą swoistą sieć społeczną, której krawędzie są abstrakcją współpracy (współautorstwa artykułów). Odległość pomiędzy tak opisanymi węzłami, zwana czasami z języka angielskiego *collaborative distance*, można zdefiniować jako najkrótszą ścieżkę w tym grafie łączącą dwóch autorów. Przykładem tak zdefiniowanej odległości może być „liczba Erdősa” (LE). Oblicza się ją w sieci społecznej, której centralnym punktem jest sławny matematyk Paul Erdős. On sam ma $LE = 0$, jego bezpośredni współpracownicy 1, i tak dalej. Osoby, które nie są powiązane w żaden sposób z Erdősem mają $LE = \infty$. Dlaczego obliczenie liczby Erdősa stanowi problem, jeśli wykorzystywana jest baza relacyjna do składowania danych?

Otóż uzyskanie informacji na ten temat wymagałoby wielokrotnego (zależnego od odległości między węzłami) łączenia tabel.

Oczywiście można stwierdzić, że systemy zarządzania bazami danych firmy Oracle w swoim standardzie zapytań oferują klauzulę *connect by priori* umożliwiającą przedstawienie danych w postaci drzewa oraz uzyskanie informacji na temat poziomu węzła w drzewie. Nie umożliwia on jednak obliczenia długości ścieżki do węzła, który nie jest korzeniem drzewa.

Istnieją jednakże systemy, w których bazę relacyjną zastąpiono przez zbudowaną od podstaw bazę opartą o model grafowy lub bazę obiektową z dodanym grafowym językiem zapytań. Przykładami takich systemów są LorelQL [Abite 1997] i QGraph [Jensen 2004]. Podejście takie zmusza badającego do ingerencji w dane na niskim poziomie przez przystosowanie ich formatu do wymagań autorskiego systemu. Ingerencja na niskim poziomie i konieczność ręcznych operacji na danych zmusza badacza do ograniczenia się do małego zbioru danych, a przez to ma istotny wpływ na wiarygodność wniosków wyciąganych na podstawie tak przeprowadzonych eksperymentów. Konieczność transformacji danych wynika najczęściej z faktu, że składowanie danych wykonywane jest z wykorzystaniem baz danych „niegrafowych” czyli relacyjnych lub obiektowych.

Efektywny system wspomagający gromadzenie i analizę danych dla sieci społecznych powinien zapewniać możliwie dużą kompatybilność z innymi systemami. Przez kompatybilność rozumiana jest możliwość wprowadzania czy też „rozumienia” przez system różnych formatów danych wprowadzanych. Dodatkowo struktura bazy danych powinna być opisana w sposób umożliwiający utrzymanie spójności danych oraz umożliwiający łatwe dodawanie nowych węzłów czy typów relacji, a także nowych źródeł danych.

4. DANE W SIECIACH SPOŁECZNYCH

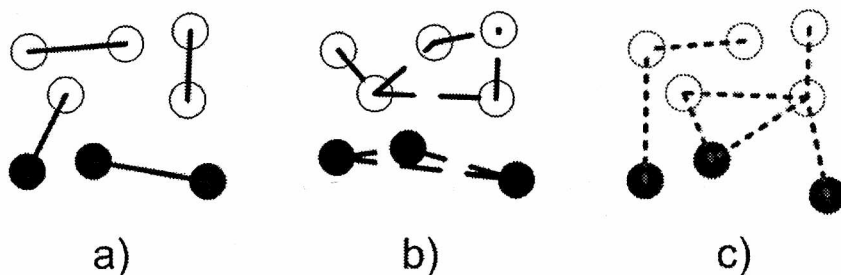
W rozdziale przedstawione zostaną problemy, jakie pojawiają się w warstwie danych sieci społecznych. Omówione zostaną także korzyści ze zmiany podejścia do przechowywania danych. Zaprezentowane zostaną też dwa przykładowe języki zapytań grafowych.

4.1. PODSTAWOWE PROBLEMY WARSTWY DANYCH

Podstawowym problemem występującym w procesie analizy sieci społecznych jest niedostosowanie istniejących, popularnych modeli danych do specyfiki sieci. Niedostosowanie to objawia się przez brak metod wspierających często wykonywane operacje jak np. znajdowanie ścieżki, dystansu (np. odległość Erdős między autorami), czy

znaczenia węzła (np. najczęściej cytowany artykuł). Wykonanie tego typu zapytań na bazie działającej w modelu relacyjnym stanowi problem przede wszystkim wydajnościowy. Wymaga to nieokreślonej liczby złączeń między tabelami, zależnej od odległości między węzłami. Wiadomo, że złączenie jest operacją najmocniej wpływającą na wydajność systemu, szczególnie mocno uzależnioną od wielkości relacji, na których działa.

Kolejnym problemem jest heterogeniczność danych występujących w sieci społecznej. Modele relacyjne i sieciowe wymuszają poprzez swoje założenia homogeniczność danych. Struktura relacji jest ustalona i narzucona odgórnie. W sieciach społecznych spotykamy się natomiast z modelem heterogenicznym. W szczególności heterogeniczność dotyczy może relacji. Problem ten znany jest z życia codziennego, gdzie spotkać można relacje biznesowe i osobiste [Cai 2005]. Różne relacje mogą mieć różną wartość w zależności od tego, jakie informacje użytkownik chce pozyskać z sieci. Problem ten przedstawiono na rysunku 3, na którym zaprezentowano trzy rodzaje połączeń, przedstawione na oddzielnych grafach (dla zwiększenia czytelności).



Rys. 3. Przykład sieci o połączeniach heterogenicznych

Zgodnie z pytaniem użytkownika, węzły oznaczone wypełnionymi kołami, należą do jednej grupy. Z tego możemy wnioskować, że najbardziej wartościowe będą w takim przypadku połączenia z rysunku 3b. W przedstawionej sytuacji łatwo jest zauważyć, która klasa relacji jest najbardziej wartościowa dla danego zapytania. Problem komplikuje się w sytuacji, gdy połączenia mogą być jednokierunkowe (graf skierowany), a dodatkowo do krawędzi przypisane są wagi określające siłę relacji (graf ważony). W tak postawionej sytuacji odnalezienie optymalnej klasy relacji dla wymagań użytkownika nie jest prostym do rozwiązania problemem.

Problemy te spowodowały poszukiwania nowych rozwiązań, które byłyby bardziej dopasowane do określonych potrzeb. Modelem danych, który szczególnie wydaje się pasować do specyfiki sieci społecznych jest Resource Description Framework (RDF) [HannRidd 2005, Hayes 2004]. RDF stanowi model lub bazę danych, która jest reprezentowana w formie grafu. Model ten może on być rozpatrywany z dwóch perspektyw. Z jednej strony jako minimalny zbiór narzędzi stanowiący język opisu metadanych, czy też powtarzając za W3C, „język twierdzeń pomyślany tak, aby mógł być

używany do wyrażania propozycji, przy użyciu precyzyjnego, formalnego słownictwa” [AnGut 2005].

W pracy sformułowanie model danych rozumiane jest w sposób zaproponowany przez Codd’a [Codd 1980]. Zawiera on zatem trzy podstawowe komponenty:

1. Zbiór typów struktur danych.
2. Zbiór operatorów lub reguł, które mogą zostać zastosowane do instancji typów.
3. Zbiór reguł poprawności, definiujących spójność i możliwe zmiany stanów.

4.2. KORZYŚCI ZE ZMIANY PODEJŚCIA

Zmiana podejścia do przechowywania i manipulowania danymi w sieciach społecznych na bliższe grafowemu wspomogłaby analizę danych opartą o algorytmy działające na grafach. Dodatkowo, odpowiednia implementacja umożliwiłaby dostęp do tych danych na poziomie abstrakcji zbliżonym do tego, na jakim dane będą prezentowane. Można porównać tę sytuację do tej, która istniała do niedawna w systemach typu GIS. Przez długi czas w systemach tych brak było wspomagania dla geometrycznych zagadnień, brak typów i operacji odpowiadających specyfice zagadnienia. Wprowadzenie odpowiedniej typizacji pozwala na znaczne przyspieszenie prac, a także obniżenie ich kosztów gdyż nie jest już konieczne oprogramowanie obsługi wymaganych typów od podstaw.

Hayes, przedstawiając motywację i cele leżące u podstaw RDF, koncentruje się szczególnie na opisie stron internetowych [Hayes 2004]. Jako cele wymienia, między innymi, prostotę modelu danych oraz wsparcie dla standardu XML. Na możliwość zastosowania RDF w sieciach społecznych zwracają uwagę autorzy prac [AnGut 2005, SanGut 2006]. Szczegółowe porównanie możliwości i właściwości różnych modeli danych wyróżnia modele semistrukturalne i RDF jako kandydatów do odegrania szczególnej roli w dziedzinie przechowywania i analizy danych w sieciach społecznych. Jednak RDF nie jest idealnym kandydatem i wymaga rozstrzygnięcia kwestii związanych np. z ograniczeniami czy też językiem zapytań. RDF nie precyzuje pojęcia ograniczenia. Istnieją języki zapytań zaprojektowane dla RDF, nie wspierają one jednak często zapytań występujących w systemach grafowych. Wspomniane problemy wynikają z faktu, iż RDF zaprojektowany został raczej jako narzędzie opisu i przechowywania informacji, a nie jako w pełni funkcjonalny model danych w przytaczanym już rozumieniu Codd’a. Z dostępnej literatury wynika, że trwają prace mające na celu wytworzenie nowego systemu bazodanowego, dostosowanego do wymagań sieci społecznych [AnGut 2005, SanGut 2006]. Innym pomysłem jest rozszerzenie istniejących rozwiązań opartych o SQL w taki sposób, aby wspierały algorytmy analizy sieci społecznej [TsDiCa 2005].

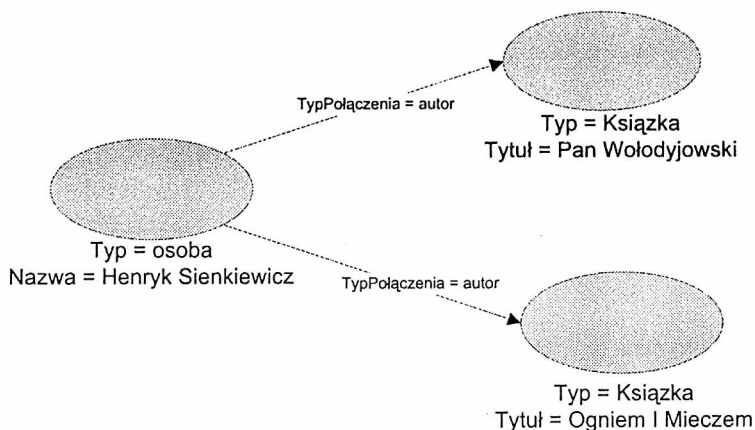
5. PRZYKŁADY JĘZYKÓW ZAPYTAŃ W BAZACH GRAFOWYCH

W rozdziale przedstawimy dwa języki zapytań zaprojektowane z myślą o bazach grafowych: QGraph oraz LorelQL.

5.1. QGRAPH

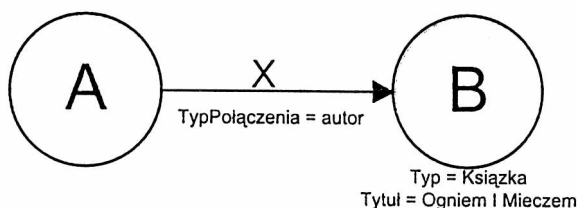
QGraph [10] jest językiem graficznym zaprojektowanym do użytku w dużych bazach danych grafowych. W odróżnieniu od języków działających na bazach relacyjnych (np. SQL), które jako wynik swojego działania zwracają zbiór rekordów, zapytania QGraph jako wynik zwracają zbiór podgrafów (ang. *subgraph*). Dzięki temu użytkownik otrzymuje jednocześnie informacje o zawartości jak i o strukturze danych.

Język ten jest wykorzystywany w otwartym projekcie Proximity, służącym do ekstrakcji wiedzy z danych relacyjnych. Operuje on na bazach grafowych, w których relacje pomiędzy węzłami są skierowane, a zarówno węzły jak i relacje mogą posiadać wiele atrybutów. Rysunek 4 przedstawia przykładowy, bardzo prosty graf stanowiący wycinek bazy danych o książkach. Owale stanowią węzły grafu, do których przypisane są typy. Relacje łączące węzły także mają przypisany atrybut TypPołączenia.



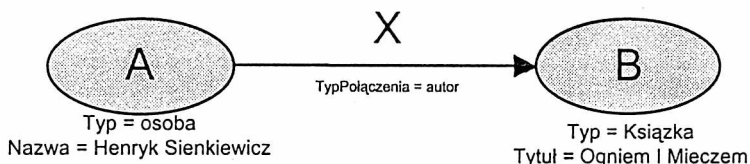
Rys. 4. Przykładowy wycinek bazy danych o książkach
(oprac. wł. na podst. [Jensen 2004])

Rysunek 5 przedstawia przykładowe pytanie, którego wynikiem jest zbiór podgrafów. Pytamy o autora/autorów książki „Ogniem i mieczem”.



Rys. 5. Przykładowe zapytanie w języku Qgraph
(oprac. wł. na podst. [Jensen 2004])

Wynikiem wykonania zapytania z rysunku 5. na grafie przedstawionym na rysunku 4. będzie podgraf zaprezentowany na rysunku 6.



Rys. 6. Przykład odpowiedzi w języku Qgraph
(oprac. wł. na podst. [Jensen 2004])

Konstruując zapytania pamiętać należy, że wielokrotne połączenia między węzłami traktowane są jako różne, co może powodować wielokrotne pojawianie się tych samych węzłów w odpowiedzi. Każda etykieta krawędzi traktowana jest jako różna, mimo posiadany identycznych wartości atrybutów. Może to także powodować powtórzenia grafów wyglądających na pierwszy rzut oka identycznie.

5.2. LOREL QUERY LANGUAGE (LORELQL)

LorelQL [Abite 1997] jest językiem dla danych semistrukturalnych. Język ten został zaprojektowany tak, aby zachowywał styl SQL. Jest on zaprojektowany jako rozszerzenie języka OQL (ang. *Object Query Language*). Model danych wykorzystany w projekcie Lorel jest rozszerzeniem modelu danych ODMG.

Język skonstruowany jest tak, aby umożliwiał analizę danych pochodzących z różnych źródeł, o niejednorodnej strukturze (różne typy pól o tych samych nazwach) lub takich, w których brakuje pewnych pól (właściwości). Język wprowadza znaki wieloznaczne (ang. *wildcards*) pozwalające na dopasowywanie wzorców.

Poniższy przykład, zaczerpnięty z pracy [Abite 1997], wykorzystuje bazę danych opisującą restauracje. Dane mogą pochodzić z różnych źródeł np. gazet, przewodników czy stron internetowych. W przykładzie pytamy o wszystkie restauracje oraz opcjonalnie o ich adresy, które w jakimkolwiek polu zawierają frazę „tani”:

```
Select Guide.restaurant.name,  
       Guide.restaurant(.address)?.zipcode  
Where Guide.restaurant.% grep "tani"
```

Znak "?" oznacza, że pole address jest opcjonalne. Symbol „%” jest symbolem wieloznacznym pozwalającym na dopasowanie jakiejkolwiek właściwości obiektu restaurant. Operator porównania grep zwraca wartość „prawda”, jeśli ciąg „tani” pojawi się w jakimkolwiek polu obiektu restaurant.

Język LorelQL udostępnia szereg interesujących konstrukcji, jednakże ich prezentacja nie jest celem tej pracy. Szczegółowa specyfikacja języka opisana jest w pracy [Abite 1997].

Wymienione w tej części rozdziału języki zapytań są zaledwie przedstawicielami większej grupy eksperymentalnych języków zapytań. Szersze omówienie ich składni oraz przykłady użycia zawierają dostępne publikacje [Abite 1997, Jensen 2004].

6. PODSUMOWANIE

Sieci społeczne intensywnie rozwijają się i cieszą się olbrzymim zainteresowaniem zarówno ze strony użytkowników indywidualnych jak i firm. Tym bardziej, że działające obecnie sieci społeczne stają się miejscem nie tylko rozrywki, ale także wymiany informacji, poszukiwania pracy i pracowników, nawiązywania kontaktów prywatnych i biznesowych. Duże zainteresowanie portalami pozwalającymi na budowę własnej sieci kontaktów powoduje, że opiekujące się nimi firmy i organizacje poszukują nowych, efektywnych metod składowania, przetwarzania oraz analizy olbrzymich ilości danych.

Charakter sieci powoduje problemy z przetwarzaniem i analizą sieci społecznych z zastosowaniem tradycyjnych metod. Obecnie systemy zarządzania bazami danych nie oferują dedykowanych dla sieci społecznych typów danych ani metod analitycznych.

Z drugiej strony istnieje wiele prób implementacji systemu zarządzania wspomagającego sieci społeczne przez np. grafowe języki zapytań. Implementacje te opierają się jednak najczęściej na rozszerzaniu istniejących narzędzi.

W tym kontekście, nakreślona przez Gutierrez drogą implementacji nowego, dostosowanego do potrzeb sieci społecznych i opartego o RDF systemu zarządzania bazą danych wydaje się być interesująca [AnGut 2005, SanGut 2006]. Powodzenie tego typu przedsięwzięcia przyniesie oczywiste korzyści w postaci efektywnych metod przetwarzania danych, odpowiednią typizację oraz reguły poprawności, język zapytań dostosowany do specyfiki dziedziny. Nasuwa się jednak pytanie o sensowność tworzenia tak wyspecjalizowanego oprogramowania. Być może lepiej wybrać drogę zaproponowaną między innymi w pracy [TsDiCa 2005], rozszerzania istniejących, dobrze znanych i przetestowanych rozwiązań, dodając do nich jedynie brakujące mechanizmy wspomagające analizę sieci społecznych?

Także w dziedzinie nowych języków baz danych sieci społeczne pobudzają poszukiwania, zarówno w zakresie języków „pochodnych” SQL, jak też innych języków, np. graficznych. Badania te mają na celu udostępnienie użytkownikom wygodnego i efektywnego narzędzia eksploracji danych. Jaki będzie kierunek rozwoju języków baz grafowych i semistrukturalnych?

Przy obecnym stanie wiedzy nie jest możliwe udzielenie jednoznacznej odpowiedzi na postawione pytania. Konieczne są implementacja i badania. Otwiera się jednak nowy kierunek badań nad bazami danych oraz nowy obszar ich zastosowania.

LITERATURA

- [Abite 1997] Abiteboul S., Quasi D., McHugh J., Widom J., Wierne J.L., *The Lorel Query Language for Semistructured Data*. International Journal on Digital Libraries, 1(1), 1997, pp. 68–88,
www-db.stanford.edu/pub/papers/lore96.ps
- [Adler 2002] Adler P.S., *Social Capital: Prospects for a New Concept*. Academy of Management Review, 27(1), 2002, pp. 17–40.
- [AnGut 2005] Angles, R., Gutierrez, C., *Querying RDF data from a graph database perspective*. In: ESWC. 2005, pp. 346–360.
- [Cai 2005] Cai D., Shao Z., He X., Yan X., Han J., *Mining Hidden Community in Heterogeneous Social Networks* LinkKDD'05, 2005, pp. 58–65.
- [Codd 1980] Codd E., *Data models in database management*. In: Workshop on Data abstraction, databases and conceptual modeling, 1980, pp. 112–115.
- [Goldbeck 2007] Goldbeck J., *Trust in Web Based Social Networks*.
<http://trust.mindswap.org/>
- [GoldHend 2004] Golbeck J., Hendler J.A., *Accuracy of Metrics for Inferring Trust and Reputation in Semantic Web-Based Social Networks*. EKAW, LNCS 3257, 2004, pp. 116–131.
- [Goldbeck 2005] Golbeck J., *Computing and Applying Trust in Web-Based Social Networks*. Ph.D. Thesis, University of Maryland, College Park, 2005.
- [HannRidd 2005] Hanneman R.A., Riddle M., *Introduction to social network methods*. CA: University of California, Riverside, 2005,
<http://faculty.ucr.edu/~hanneman/>
- [Jensen 2004] Jensen D., *Proximity 4.2 QGraph Guide*. The Knowledge Discovery Laboratory, 2004,
<http://kdl.cs.umass.edu/software/documentation/QGraphGuide.pdf>
- [KazMus 2005] Kazienko P., Musiał K., *Social Networks, Prepared for European Network of Excellence: Nature-inspired Smart Information Systems*, 2005,
www.zsi.pwr.wroc.pl/~kazienko/projekty.htm
- [KlyCar 2004] Klyne G., Carroll J. J., *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation, 2004,
www.w3.org/TR/2004/REC-rdf-concepts-20040210/
- [Hayes 2004] Hayes, P., *RDF Semantics*. 2004,
www.w3.org/TR/2004/REC-rdf-mt-20040210/

- [SanGut 2006] San Martín M., Gutierrez C., *A Database Perspective of Social Network Analysis Data Processing*. Sunbelt XXVI, 2006.
- [Strou 1988] Stroustrup B., *What Is Object-Oriented Programming?* IEEE Softw. 5, 1988, pp. 10–20.
- [TsDiCa 2005] Tsvetovat M., Diesner J., Carley K. M., *NetIntel: A Database for Manipulation of Rich Social Network Data*. CMU-ISRI-04-135, Pittsburgh 2005.

THE DATA ANALYSIS ISSUE IN THE SOCIAL NETWORK

This chapter presents the application of the databases in which the computer science and psychology work together. Those area are Social Networks. With this specific area many issues in the data layer are connected. Those are issues with data storage and processing. The analysis to extract the information from the social network requires processing of large amount of data. That is the reason of searching for new methods and data structures adapted to the social network character.

*protokół HTTP,
programowanie sieciowe,
system operacyjny Symbian*

Zygmunt MAZUR*
Sławomir PASIECZNY

PROGRAMOWANIE SIECIOWE W SYMBIANIE – PROTOKÓŁ HTTP

W pracy przedstawiono podstawy protokołu HTTP oraz sposób wykorzystania HTTP API systemu operacyjnego Symbian w urządzeniach mobilnych działających z tym właśnie systemem operacyjnym rozpoczynając od podstawowych definicji i pojęć a kończąc na bardziej złożonych zagadnieniach.

1. WSTĘP

Wykorzystanie protokołu HTTP do komunikacji urządzeń w sieci GSM/GPRS jest korzystniejsze i bardziej niezależne od operatora sieci komórkowej niż wykorzystanie protokołu TCP/IP, który to protokół często jest różnie traktowany przez operatorów. Spotkać można między innymi blokowanie portów TCP/IP innych niż porty 80 i 443.

Po rozpoznaniu problemów związanych z komunikacją przez protokół TCP/IP wielokrotnie w badaniach testowano i wykorzystywano protokół HTTP, który nie jest obciążony takimi ograniczeniami jak protokół TCP/IP (w kontekście operatorów sieci GSM/GRPS). Protokół HTTP sprawdził się wielokrotnie w wielu projektach, pozwalając na przesyłania różnego rodzaju danych, w tym zdjęć fotograficznych.

Dodatkowym atutem tego protokołu jest jego prostota i łatwość wykorzystania na urządzeniach mobilnych działających z systemem operacyjnym Symbian OS [WWW1].

W pracy przedstawiono podstawy protokołu HTTP oraz sposób wykorzystania HTTP API systemu Symbian w urządzeniach mobilnych, działających z tym właśnie

* Instytut Informatyki Stosowanej, Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27, zygmunt.mazur@pwr.wroc.pl

systemem operacyjnym, rozpoczynając od podstawowych definicji i pojęć a kończąc na bardziej złożonych zagadnieniach.

2. WPROWADZENIE DO HTTP API

HTTP API dostarcza interfejsy i mechanizmy do komunikacji z serwerami przez protokół HTTP. Architektura dostarcza generycznego mechanizmu dla HTTP, który pozwala programiście na wybór odpowiedniego protokołu HTTP, kodowania itp.

Wewnętrzna implementacja HTTP API jest asynchroniczna i komunikuje się z klientem poprzez mechanizm funkcji zwrotnych. HTTP API opiera się na pięciu podstawowych pojęciach:

- Sesja – reprezentowana jest przez klasę `RHTTPSession`. Sesja reprezentuje całą komunikację między aplikacją klienta a serwerem. W ramach sesji odbywają się transakcje, czyli obsługa pojedynczego żądania lub odpowiedzi. Sesja definiuje ogólne własności dla wszystkich transakcji wchodzących w jej skład takie jak kodowanie, protokół itp. Dodatkowo może mieć skojarzony ze sobą zbiór filtrów, na przykład filtr autentyfikacji.
- Transakcja – reprezentowana jest przez klasę `RHTTPTransaction`. Transakcja odbywa się w ramach określonej sesji i oznacza interakcje pomiędzy klientem a serwerem – może to być zarówno żądanie jak i odpowiedź. Transakcja składa się z nagłówka i ciała. Transakcje wykonują się asynchronicznie i używają mechanizmu wywołań zwrotnych w celu przekazania poszczególnych zdarzeń i informacji.
- Nagłówek – reprezentowany jest przez klasę `RHTTPHeaders` i reprezentuje nagłówek żądania lub odpowiedzi.
- Dostawca danych – klasa implementująca interfejs `MHTTPDataSupplier`; enkapsuluje ciało transakcji (żądania lub odpowiedzi).
- Filtry – klasa implementująca interfejs `MHTTPFilter`. Filtry to dodatkowe moduły, które pozwalają na dodanie do sesji specyficznego zachowania obejmującego wszystkie transakcje wchodzące w skład sesji. Przykładem takiego filtra może być mechanizm autentykacji, która jest realizowana przez kilka transakcji przenoszących dane do i od serwera.

3. SESJE I TRANSAKcje HTTP

Zanim użyta zostanie jakakolwiek transakcja, konieczne jest utworzenie i otwarcie nowej sesji (każda transakcja dotyczy jednej sesji, natomiast z daną sesją może być związanych wiele transakcji).

3.1. TWORZENIE SESJI

Sesja musi być otwarta przez cały czas życia transakcji (zamknięcie sesji, gdy posiada ona otwarte transakcje, jest bezpieczne – powoduje to zamknięcie wszystkich aktywnych transakcji).

Typowy sposób otwarcia sesji przedstawia poniższy kod:

```
private: RHTTPSession iSession;
.
.
.
void CClientEngine::ConstructL()
{
iSession.OpenL();
}
CClientEngine::~CClientEngine()
{
iSession.Close();
}
```

Po otwarciu sesji, uruchomienie pierwszej transakcji spowoduje pojawienie się pytania o punkt dostępu do internetu (APN – ang. *Access Point Name*). Jeśli chcemy uniknąć pytania o APN (np. posiadamy już otwarte RConnection korzystające z pewnego APN), wówczas można związać sesję z istniejącym APN/RConnection – dokonuje się tego poprzez własności sesji. Prezentuje to poniższy kod:

```
RHTTPConnectionInfo connInfo = iSess.ConnectionInfo();
RStringPool pool = iSess.StringPool();
connInfo.SetPropertyL(pool.StringF(HTTP::EHttpSocketServ,
                                     RHTTPSession::GetTable()),

THTTPHdrVal(iSocketServ.Handle()));

TInt connPtr= REINTERPRET_CAST(TInt, &iConnection);
connfo.SetPropertyL(pool.StringF(HTTP::EHttpSocketConnection,
RHTTPSession::GetTable()),
THTTPHdrVal(connPtr));
```

W powyższym przykładzie zakłada się, iż sesja reprezentowana przez obiekt iSess została otwarta, ale żadna transakcja nie została jeszcze uruchomiona oraz, że obiekty RConnection i Connection są utworzone i poprawne. Użycie powyższego zapisu pozwala uniknąć pytania o APN, w przypadku, gdy już jakiś APN mamy przydzielony, wykorzystujemy go i chcemy, aby sesja i jej transakcje używały tego samego.

3.2. TWORZENIE TRANSAKCJI

W celu utworzenia nowej transakcji konieczne jest wykonanie trzech podstawowych kroków:

- Otwarcie nowej transakcji (RHTTPTransaction) oraz związanie jej z wcześniejszym przygotowanym obiektem sesji. Przedstawia to poniższy kod:

```
TUriParser8 uri;
uri.Parse(aUri);
RStringF method=iSession.StringPool().StringF(HTTP::EGET,
RHTTPSession::GetTable());
iTransaction = iSession.OpenTransactionL(uri, *this, method);
```

Klasa TUriParser8 pozwala na sprawdzenie, czy przekazany URI jest poprawny. Kolejnym krokiem jest pobranie metody GET w formacie RStringF – dokonuje się tego dzięki wykorzystaniu tablicy „string pool” należącej do otwartej sesji. „String pool” to tablica definiująca nazwy metod, nazwy nagłówków, własności sesji i transakcji. Ostatnim krokiem jest otwarcie obiektu nowej transakcji – należy tu zaznaczyć, iż drugi parametr metody OpenTransactionL to wskaźnik do obiektu klasy implementującej interfejs MHTTPTransactionCallback (czyli metody MHFRunError i MHFRunL). Ponadto metoda OpenTransactionL nie zwraca błędów, gdy dostanie niepoprawne parametry – błędy te zostaną zgłoszone dopiero w metodzie MHFRunError.

- Ustawienie nagłówka transakcji

Kolejnym krokiem jest zmiana domyślnych ustawień dla wartości nagłówka transakcji. Odpowiedzialny za to jest poniższy kod:

```
LIT8(KUserAgent, "SimpleClient 1.0");
_LIT8(KAccept, "*/*");
RHTTPHeaders hdr = iTransaction.Request().GetHeaderCollection();
SetHeaderL(hdr, HTTP::EUserAgent, KUserAgent);
SetHeaderL(hdr, HTTP::EAccept, KAccept);
```

Metoda SetHeaderL zdefiniowana jest następująco:

```
Void SetHeaderL(RHTTPHeaders aHeaders,
                TInt aHdrField,
                const TDesC8& aHdrValue)
{
    RStringF valStr =
    iSession.StringPool().OpenFStringL(aHdrValue);
    CleanupClosePushL(valStr);
    THTTPHdrVal val(valStr);
```



```

aHeaders.SetFieldL(iSession.StringPool().StringF(aHdrField,
    RHTTPSession::GetTable()), val);
CleanupStack::PopAndDestroy(); // valStr
}

```

- Zatwierdzenie transakcji

Kiedy transakcja jest otwarta oraz ustawione zostały wszystkie wymagane nagłówki, możliwe jest zatwierdzenie transakcji:

```
iTransaction.SubmitL();
```

Po wywołaniu tej metody, żądanie zostanie przekazane do serwera, a wszelkie informacje o tej transakcji będzie można uzyskać w obiekcie nasłuchującym przez wywołanie metody MHFRunError oraz MHFRunL.

3.3. OBSŁUGA ZDARZEŃ TRANSAKCJI

Podczas otwierania nowej transakcji konieczne jest przekazanie do konstruktora klasy RHTTPTransaction obiektu implementującego interfejs MHTTPTransactionCallback. Kiedy transakcja jest uruchomiona, system informuje obiekt nasłuchujący za pomocą metod MHFRunL i MHFRunError interfejsu MHTTPTransactionCallback o błędach i zdarzeniach (odbiór ciała i nagłówka), które wystąpiły podczas obsługi transakcji.

Podstawowa postać metody MHFRunL:

```

void CClientEngine::MHFRunL(RHTTPTransaction aTransaction,
    const THTTPEvent& aEvent)
{
    switch (aEvent.iStatus)
    {
        case THTTPEvent::EGotResponseHeaders:
            // do something with received header
            break;
        case THTTPEvent::EGotResponseBodyData:
            // do something with received part of the body
            break;
        case THTTPEvent::EResponseComplete:
            // response completely received
            break;
        case THTTPEvent::ESucceeded:
            // transaction successful
            break;
        case THTTPEvent::EFailed:
            // transaction failed
            break;
    }
}

```

```

default:
if(aEvent.iStatus < 0) {
// Handle errors
}
}
}

```

Do użytecznych zdarzeń, które powinny być obsługiwane należą:

- `EGotResponseBodyData` – oznacza odbiór fragmentu ciała odpowiedzi; w celu uzyskania dostępu do ciała wiadomości należy wywołać następującą metodę:

```
aTransaction.Response().Body()->GetNextDataPart()
```

W celu zwolnienia odczytanych danych należy wykonać polecenie:

```
aTransaction.Response().Body()->ReleaseData()
```

Dane odbierane są w postaci małych paczek, dlatego zdarzenie to może być odebrane kilkakrotnie, ponadto o tym, że całe ciało wiadomości zostało odebrane świadczy dopiero wystąpienie zdarzenia `EResponseComplete`.

- `EGotResponseHeaders` – nagłówki odpowiedzi serwera; dostępne poprzez wywołanie następującej linii kodu:

```
aTransaction.Response().GetHeaderCollection()
```

- `EresponseComplete` – odpowiedź serwera została skompletowana (nagłówek plus ciało wiadomości). Po tym zdarzeniu może wystąpić zdarzenie `ESucceeded` lub `EFailed`.
- `ESucceeded` – informacja o zakończeniu transakcji sukcesem; po tym zdarzeniu transakcja nie jest kontynuowana, dlatego zdarzenie to może być użyte do zwolnienia zasobów związanych z transakcją.
- `EFailed` – informacja o tym, że transakcja zakończyła się niepowodzeniem; po tym zdarzeniu transakcja nie jest kontynuowana, dlatego zdarzenie to może być użyte do zwolnienia zasobów związanych z transakcją.
- Każde zdarzenie, dla którego `iStatus < 0` – kod statusu mniejszy od zera oznacza wystąpienie błędu.

3.4. ZATRZYMYWANIE TRANSAKCJI

W celu zatrzymania transakcji można użyć metody `iTransaction.Cancel()` – metoda ta nie zwalnia zasobów używanych przez transakcje. Innym sposobem zatrzymania transakcji, jest wywołanie metody `iTransaction.Close()` – po wywołaniu tej metody transakcja jest zamykana, zasoby używane przez transakcję są zwalniane. Po wywołaniu tej metody transakcja nie może być więcej używana.

4. DOSTAWCA DANYCH – HTTP POST

Dostawca danych jest wykorzystywany w celu dostępu do ciała żądania lub odpowiedzi HTTP. Implementacja transakcji HTTP POST wymaga użycia specjalnego dostawcy danych, który jest tworzony poprzez implementację interfejsu `MHTTPDataSupplier`.

Interfejs ten definiuje następujące metody:

- `TBool GetnextDataPart (TPtrC8& aDataPart)` – metoda wywoływana, gdy potrzebna jest kolejna porcja danych, zwraca `ETrue` – jeśli zwrócona porcja danych jest ostatnia;
- `void ReleaseData()` – metoda wywoływana, gdy dane pobrane przez metodę `GetNextDataPart` są już nie potrzebne; w metodzie tej można wywołać metodę

`RHTTPTransaction::NotifyNewRequestBodyPartL`,
aby poinformować system o kolejnej porcji danych;

- `TInt Reset()` – metoda wywoływana, aby restartować dostawcę danych; metoda ta powinna zwracać wartość `KErrNone`, gdy możliwe jest ponowne dostarczenie danych do systemu, lub wartość `KErrNotSupported`, gdy dane nie mogą być ponownie przesłane; zwrócenie innej wartości spowoduje zakończenie transakcji błędem;
- `OverallDataSize` – metoda wywoływana przez system w celu pobrania całkowitego rozmiaru ciała wiadomości; jeśli rozmiar ten jest nieznany, to metoda zwraca wartość `KErrNotFound` lub `KErrNotSupported` – wówczas transakcja będzie kontynuowana ale nagłówek `Content-Length` nie będzie ustawiony w żądaniu.

W celu wysłania żądania metodą POST konieczne jest wprowadzenie kilku zmian w stosunku do używania metody GET (prezentowanej w poprzednich podrozdziałach):

- W nagłówku wiadomości należy ustawić wartość pola `content-type` na jeden z typów MIME (ang. *Multipurpose Internet Mail Extensions* – standard stosowany przy przesyłaniu poczty elektronicznej, definiujący budowę komunikatu e-mail), np. „text/plain”.
- Należy stworzyć klasę implementującą interfejs `MHTTPDataSupplier` – klasa ta pełni rolę dostawcy danych tzn. jej zadaniem jest dostarczenie faktycznej treści wiadomości. Obiekt stworzonej klasy musi być przekazany do obiektu reprezentującego żądanie poprzez wywołanie metody `RHTTPRequest::SetBody()`. System operacyjny wykorzystuje metody interfejsu `MHTTPDataSupplier` w celu pobrania i wysłania danych.

Przykład kodu prezentującego wykorzystanie metody HTTP POST:

```
SetHeaderL(hdr, HTTP::EContentType, aContentType);
MHTTPDataSupplier* dataSupplier = this;
iTransaction.Request().SetBody(*dataSupplier);
```

Poniżej przedstawiamy kod klasy implementującej interfejs

MHTTPDataSupplier:

```
TBool CDataSupplier::GetNextDataPart(TPtrC8& aDataPart)
{
    if(iPostData)
    {
        aDataPart.Set(iPostData->Des());
    }
    return ETrue;
}

void CDataSupplier::ReleaseData()
{
    delete iPostData;
    iPostData = NULL;
}

TInt CDataSupplier::Reset()
{
    return KErrNone;
}

TInt CDataSupplier::OverallDataSize()
{
    if(iPostData)
    {
        return iPostData->Length();
    }
    else
    {
        return KErrNotFound;
    }
}
```

5. FILTR AUTENTYFIKACJI

Filtr autentykacji pozwala w prosty sposób wspierać autentykację zdefiniowaną w RFC 2617. Filtr autentykacji wspiera autentykacje Base(Base64) i Digest(MD5).

5.1. IMPLEMENTACJA FILTRU AUTENTYFIKACJI

Aby zaimplementować i uruchomić filtr autentykacji konieczne jest utworzenie klasy implementującej interfejs `MHTTPOAuthenticationCallback` i zaimplementowanie metody `GetCredentialsL`.

Interfejs `MHTTPOAuthenticationCallback` dostarcza prostą metodę `InstallAuthenticationL` do instalacji filtru. W konstruktorze klasy implementującej interfejs `MHTTPOAuthenticationCallback` wystarczy dodać następujący kod w celu instalacji filtru:

```
iSession.OpenL();
InstallAuthenticationL(iSession);
```

Powyższy zapis oznacza, że klasa implementująca interfejs `MHTTPOAuthenticationCallback` nasłuchuje zdarzeń związanych z autentyfikacją – w praktyce oznacza to, iż system wywołuje metodę `GetCredentialsL` w celu pobrania nazwy użytkownika oraz hasła.

5.2. OBSŁUGA ZDARZEŃ AUTENTYFIKACJI

W momencie, gdy transakcja próbuje otworzyć stronę, która wymaga autentyfikacji, serwer wysyła żądanie autentyfikacji, które jest parsowane przez system Symbian OS a następnie wywoływana jest metoda `GetCredentialsL`, jeśli zainstalowany jest filtr autentyfikacji.

Metoda `GetCredentialsL` powinna zachować się według następujących scenariuszy:

- Metoda powinna wymusić na użytkowniku podanie nazwy konta oraz hasła poprzez wyświetlenie odpowiednich kontrolek pozwalających na wprowadzenie tych danych, a następnie zwrócić wartość `ETrue`, aby poinformować, że odpowiednio dane zostały pobrane. Dane te są przesyłane do serwera i, jeśli nie są poprawne, metoda `GetCredentialsL` zostanie wywołana ponownie.
- Metoda powinna zwrócić `EFalse`, aby poinformować o braku nazwy użytkownika i hasła. W większości przypadków spowoduje to uruchomienie metody `MHFRUNL` z kodem błędu 401.

Typowa postać metody `GetCredentialsL` jest następująca:

```
TBool CClientEngine::GetCredentialsL(const TUriC8& aURI,
    RString aRealm,
    RStringF aAuthenticationType,
    RString& aUsername,
    RString& aPassword)
```

```

{
TBuf8<256> userName;
TBuf8<256> password;
// pobranie nazwy użytkownika i hasła od użytkownika
// np. Poprzez wyświetlenie odpowiedniego dialogu

// po pobraniu nazwy użytkownika i hasła należy odpowiednio
// ustawić parametry aUsername i aPassword
TRAPD(err, aUsername = aRealm.Pool().OpenStringL(userName));
if (!err)
{
TRAP(err, aPassword = aRealm.Pool().OpenStringL(password));
if (!err) return ETrue;
}
return EFalse;
}

```

System Symbian wspiera także dostęp do HTTPS. Oznacza to, iż system zajmuje się obsługą certyfikatów i wyświetlaniem odpowiednich informacji – nie jest wymagana żadna dodatkowa implementacja. W przypadku pojawienia się błędów podczas używania SSL, przekazywane są one do metody MHFRunL podobnie jak w przypadku zwykłych błędów HTTP.

6. ZAKOŃCZENIE

W kontekście programowania sieciowego z wykorzystaniem urządzeń mobilnych, protokół HTTP wydaje się bardzo interesującym, a niekiedy jedynym rozwiązaniem. Aby zrozumieć istotę tego stwierdzenia należy zapoznać się z podstawami komunikacji w sieci GSM/GPRS.

W sieci GSM/GPRS urządzenie mobilne nie posiada własnego adresu IP. Wynika to z faktu, iż po dołączeniu do sieci GPRS, komunikacja TCP/IP oparta jest na numerze IMSI (ang. *International Mobile Subscriber Identity*) danego urządzenia mobilnego i to bramy operatora komórkowego zarządzają pakietami TCP/IP opierając się właśnie na numerze IMSI. IMSI to unikalny numer przypisany do każdej karty SIM w sieci GSM lub UMTS, jednoznacznie ją identyfikujący.

Ponadto, z przeprowadzonych doświadczeń wynika, iż operator komórkowy Plus GSM nie blokuje żadnych portów komunikacyjnych, jednak już w przypadku operatora Era dostępne są tylko dwa takie porty: 80 i 443 – pozostałe są zablokowane. W związku z tym, w przypadku operatora Era, nie jest możliwa komunikacja urządzeń mobilnych z wykorzystaniem gniazd i protokołu TCP/IP – co było powodem bezpośrednich problemów w jednym z projektów, w którym założono, iż urządzenia będą się komunikowały poprzez protokół TCP/IP. W takiej sytuacji bezpieczniejszym wyj-

ściem, które jednocześnie jest rozwiązaniem bardziej niezależnym od operatora, jego infrastruktury i polityki bezpieczeństwa, jest wykorzystanie komunikacji poprzez protokół HTTP. Komunikacja tego typu jest wspierana przez wszystkich operatorów komórkowych w Polsce.

W kontekście przedstawionego w tym rozdziale API, wykorzystanie protokołu HTTP do komunikacji na urządzeniach z systemem Symbian jest łatwe i nie stwarza problemów.

Po pierwszych problemach z komunikacją poprzez protokół TCP/IP i sprawdzeniu, z czego wynikają te problemy, wielokrotnie potem wykorzystywano protokół HTTP do komunikacji. Protokół ten i jego wykorzystanie w aplikacjach sprawowały się bez problemu, pozwalając na wysyłanie dowolnego typu danych poprzez umieszczenie ich w żądaniu HTTP. W jednym z realizowanych projektów protokół HTTP pomyślnie wykorzystano nawet do przesyłania zdjęć fotograficznych z urządzenia mobilnego do serwera, który zajmował się magazynowaniem tych zdjęć.

LITERATURA

- [WWW1] <http://www.symbian.com/about/index.html>
 [EbeVoBe 2001] Eberspaecher J., Voegel H., Bettstetter Ch., *GSM Switching, Services and Protocol*, Wiley, 2001.

NETWORK PROGRAMMING IN SYMBIAN – THE HTTP PROTOCOL

In this paper we present the basics of the HTTP protocol and the methods for using the HTTP API of the Symbian on mobile devices with this operating system installed.

Zygmunt MAZUR*
Sławomir PASIECZNY

ASPEKTY SIECI GSM/GPRS W KONTEKŚCIE PROGRAMOWANIA W SYMBIANIE

W pracy przedstawiono i omówiono elementy architektury sieci GSM/GPRS oraz procesy zachodzące w tych sieciach wraz z przykładami kodu, które obrazują jak procesy takie powinny być przeprowadzane oraz jakie problemy są napotymane w trakcie przeprowadzania takich procesów.

1. WPROWADZENIE

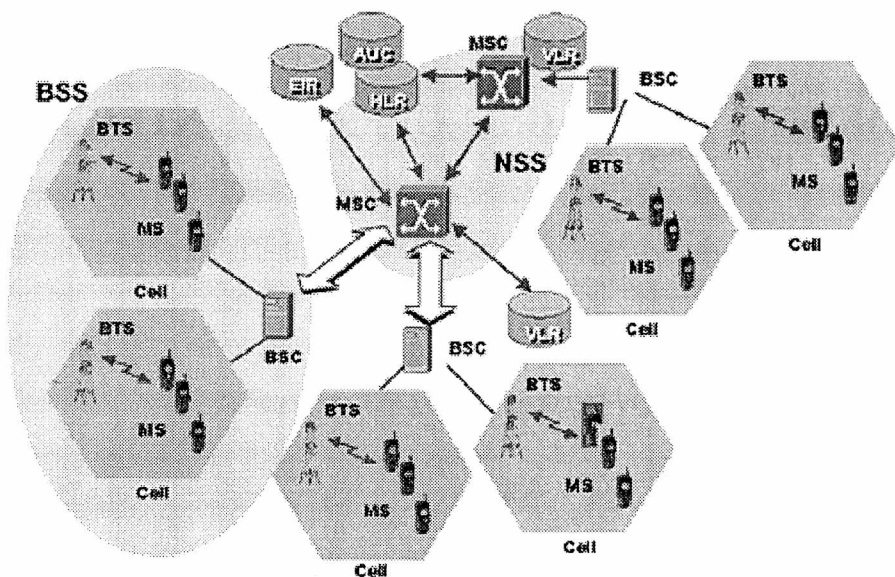
W części projektów, dla których grupą docelową urządzeń są urządzenia mobilne, konieczne jest wykorzystanie aspektów programowania sieciowego. Większość projektów wymaga prostej komunikacji poprzez gniazda TCP/IP lub z wykorzystaniem protokołu HTTP. Jednak w przypadku części projektów konieczne jest wykorzystanie elementów zaawansowanego programowania sieciowego na urządzenia mobilne. Chodzi tu o takie aspekty jak zarządzanie kontekstami PDP tzn. tworzenie nowych kontekstów, przełączanie się między kontekstami, zamykanie kontekstów oraz zarządzanie ich własnościami. Część projektów wymaga również zarządzania takimi elementami jak np. QoS (ang. *Quality of Service*). W przypadku takich projektów, które zakładają wykorzystanie zaawansowanych elementów programowania sieciowego, konieczna jest wiedza na temat architektury sieci GSM i GPRS, w których to sieciach działają urządzenia mobilne.

W niniejszej pracy zostaną przedstawione i omówione elementy architektury sieci GSM/GPRS oraz procesy zachodzące w tych sieciach wraz z przykładami kodu, które obrazują jak procesy takie powinny być przeprowadzane oraz jakie problemy mogą być napotkane w trakcie przeprowadzania takich procesów.

* Instytut Informatyki Stosowanej, Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27, zygmunt.mazur@pwr.wroc.pl

2. ARCHITEKTURA SIECI GSM OPIS PODSTAWOWYCH ELEMENTÓW

Przedstawienie aspektów sieci GSM/GPRS w kontekście programowania urządzeń mobilnych działających z systemem Symbian OS rozpoczniemy od przedstawienia podstawowych elementów wchodzących w skład sieci GSM.



Rys. 1. Elementy sieci GSM [Cisco 2002]

W skład architektury sieci GSM wchodzi następujące elementy (rysunek 1):

- MS (ang. *Mobile Station*) – MS to punkt startowy sieci GSM; jest to po prostu urządzenie mobilne takie jak telefon komórkowy lub palmtop (PDA – ang. *Personal Digital Assistant*) czyli mały, przenośny komputer osobisty.
- BTS (ang. *Base Transceiver Station*) – kiedy użytkownik korzysta z punktu startowego MS w celu przeprowadzenia rozmowy, wówczas MS wysyła żądanie wykonania rozmowy do BTS. BTS uwzględnia wszelkie elementy potrzebne do transmisji sygnału radiowego (anteny, urządzenia przetwarzające sygnał itp.) w pewnym obszarze geograficznym oznaczonym na rysunku 1 jako Cell. BTS jest odpowiedzialny za ustanowienie połączenia do MS po odebraniu od niego żądania rozmowy oraz za modulowanie i demodulowanie sygnału radiowego pomiędzy MS i BTS.
- BSC (ang. *Base Station Controller*) – jest odpowiedzialny za kontrolowanie elementów sieci radiowej oraz zarządzanie stacjami BTS. Uczestniczy w komunikacji, gdy MS łączy się z urządzeniem, które znajduje się w innej loka-

lizacji geograficznej niż obsługiwana przez dany BSC. BSC jest również odpowiedzialny za informowanie MS o nadchodzących rozmowach (odpowiada za przekazanie informacji o nadchodzącej rozmowie do prawidłowego MS).

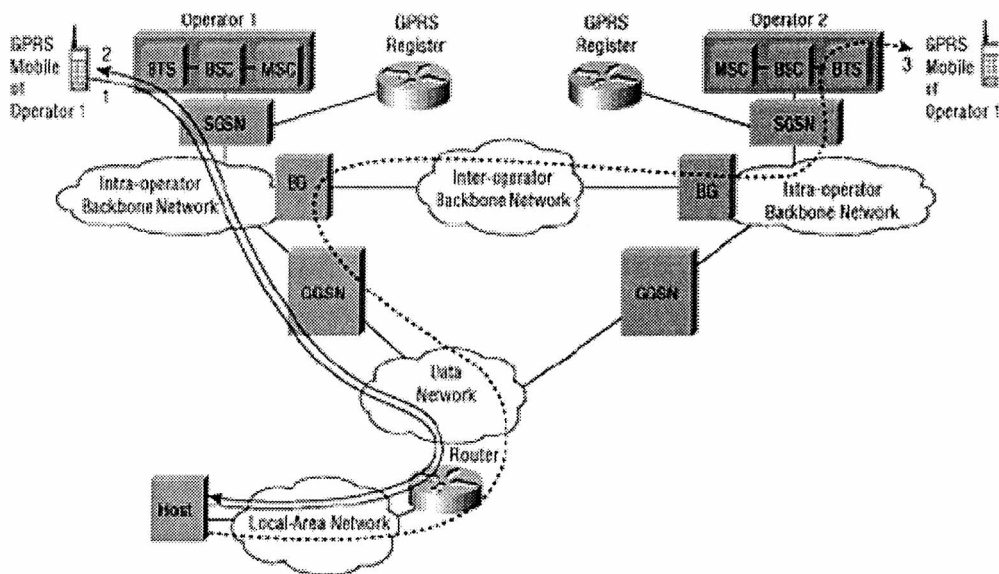
- BSS (ang. *Base Station Subsystem*) – sieć GSM jest podzielona na wiele podsystemów zwanych BSS. Każdy taki podsystem jest kontrolowany przez jeden kontroler BSC. BSS może zawierać wiele stacji BTS.
- MSC (ang. *Mobile Switching Center*) – MSC to cyfrowy przełącznik sieciowy ISDN, który pozwala na zastanowienie połączenia do innego MSC lub kontrolera BSC. Pozwala również na przekazanie rozmowy do publicznej sieci telekomunikacyjnej PSTN.
- EIR (ang. *Equipment Identity Register*) – EIR to baza danych, która przechowuje dane od numerach IMEI wszystkich urządzeń w sieci. IMEI to numer przypisany do telefonu przez producenta – numery te są unikatowe. Pozwala na zapewnienie pewnych aspektów bezpieczeństwa takich jak na przykład blokowanie rozmów z telefonu, które zostały skradzione.
- HLR (ang. *Home Location Register*) – HLR jest centralną bazą danych wszystkich użytkowników zarejestrowanych w sieci GSM. Przechowuje ona statyczne informacje o użytkownikach takie jak IMSI, serwisy użytkownika, oraz klucze do autentyfikacji użytkownika. Pozwala również na przechowywanie informacji dynamicznych taki jak np. aktualna lokalizacja użytkownika.
- AuC (ang. *Authentication Center*) – baza danych skojarzona z HLR. Baza ta przechowuje dane do algorytmu autentyfikacji użytkownika, klucze itp.
- VLR (ang. *Visitor Location Register*) – VLR to rozproszona baza danych, która przechowuje tymczasowe informacje o MS, które są aktywne w obszarze geograficznym, za który odpowiada dany VLR. VLR jest skojarzony z MSC. Kiedy nowy użytkownik aktywuje się w danej lokalizacji, VLR jest odpowiedzialny za skopiowanie informacji z bazy HLR. Pozwala to na odciążenie bazy HLR.

Na rysunku 1 zaznaczono również element o nazwie NSS (ang. *Network and Switching Subsystem*), w którego skład wchodzi: MSC, VLR, HLR, AuC, EIR. Podsystem NSS jest sercem sieci GSM – pozwala on na łączenie sieci GSM z publiczną komutowaną siecią telefoniczną (ang. PSTN – *Public Switched Telephone Network*) oraz za dostarczenie usług takich jak naliczanie kosztów, roamingu itp.

3. ARCHITEKTURA SIECI GPRS OPIS PODSTAWOWYCH ELEMENTÓW

GPRS stanowi rozszerzenie sieci GSM drugiej generacji i pozwala na transmisję danych na poziomie od 9.6 do 171 kbps. Ponadto użytkownicy mogą współdzielić te same zasoby wykorzystywane do transmisji danych. GPRS wykorzystuje elementy

sieci GSM (lub dodaje do nich nową funkcjonalność), ale w celu zapewnienia wysokiej efektywności zdecydowano się na wprowadzenie kilku nowych elementów sieci i protokołów. Elementy te zostały przedstawione na rysunku 2.

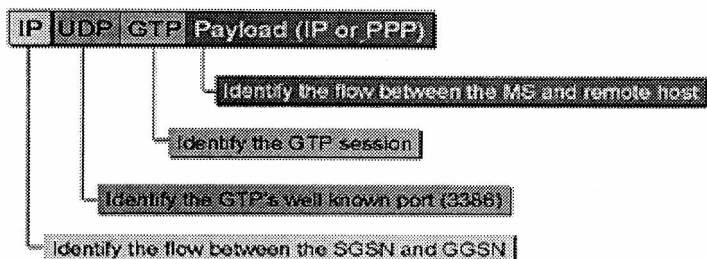


Rys. 2. Elementy sieci GPRS oraz sposób komunikacji [Cisco 2002]

W sieci GPRS poszerzono funkcjonalność elementów takich jak BTS, BSC oraz baz danych (HLR, VLR itp.). Do nowych elementów sieci GPRS należą:

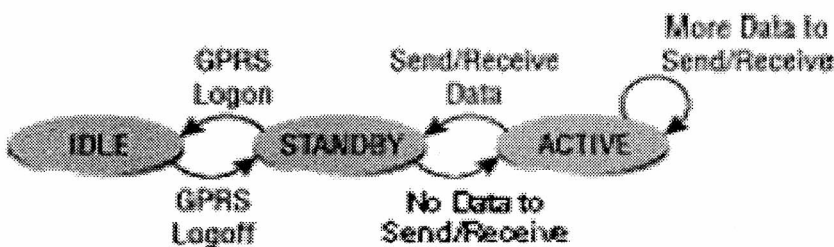
- TE (ang. *Terminal Equipment*) – urządzenia, które pozwalają na wykorzystanie GPRS.
- PCU (ang. *Packet Control Unit*) – element dodany do BSC. PCU zapewnia interfejs pozwalający na transmisję danych w BSS. Z MS dane lub głos są wysyłane do BTS a potem do BSC drogą radiową. Jednak z BSC do MSC głos i dane zostają rozdzielone tzn. głos transmitowany jest w sposób znany z sieci GSM natomiast dane są wysyłane do węzła SGSN przy pomocy PCU, który dostarcza odpowiedniego interfejsu.
- SGSN (ang. *Serving GPRS Support Node*) – SGSN dostarcza pakiety do MS w ramach określonej lokalizacji. Dzięki HLR SGSN pobiera informacje potrzebne do obsługi MS. Ponadto SGSN wykrywa nowe MS w sieci, przeprowadza proces rejestracji itp. SGSN pozwala na obsługę pakietów i komunikacji wewnątrz sieci GPRS.
- GGSN (ang. *Gateway GPRS Support Node*) – GGSN jest interfejsem, przez który MS może komunikować się z zewnętrznymi sieciami IP.

Podczas komunikacji między węzłami SGSN i GGSN wykorzystywany jest wprowadzony w sieci GPRS protokół GTP (ang. *GRPS Tunneling Protocol*). Na rysunku 3 przedstawiono istotę tego protokołu.



Rys. 3. Protokół GTP [Cisco 2002]

Urządzenie MS w sieci GPRS może znajdować się w jednym z trzech stanów co zostało przedstawione na rysunku 4.



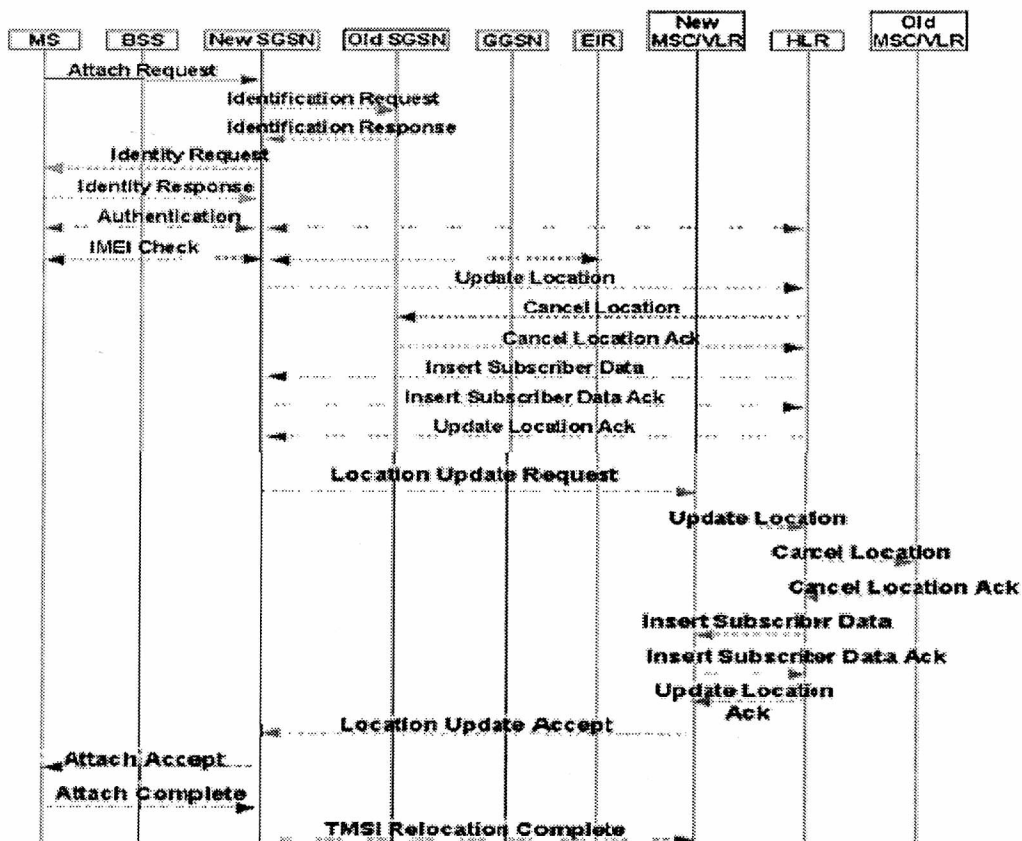
Rys. 4. Stany MS w sieci GPRS [Cisco 2002]

4. PODSTAWOWE PROCESY W SIECI GPRS W KONTEKŚCIE PROGRAMOWANIA W SYMBIANIE

W rozdziale przedstawione zostaną diagramy sekwencji opisujące podstawowe procesy zachodzące w sieci GPRS. Diagramy te są proste i intuicyjne, dlatego nie zostaną opisane krok po kroku (istota tych diagramów polega na przedstawieniu kolejności komunikacji między poszczególnymi węzłami sieci GPRS w celu przeprowadzenia określonego procesu). Przy opisie większości procesów przedstawione zostaną komentarze i fragmenty kodów prezentujące podejście do tych procesów od strony API systemu Symbian.

4.1. PROCES DOŁĄCZANIA URZĄDZENIA DO SIECI GPRS

W systemie Symbian proces dołączania urządzenia do sieci GPRS zachodzi w sposób przezroczysty dla programisty: wystarczy wywołać metodę `Start` dla obiektu klasy `RConnection`. Metoda ta uruchamia proces dołączania urządzenia do sieci GPRS, a następnie kolejne procesy takie jak proces autentykacji oraz proces aktywowania kontekstu PDP.



Rys. 5. Diagram przedstawiający proces dołączania urządzenia do sieci GPRS [Cisco 2002]

Proces dołączania urządzenia do sieci GPRS można również uruchomić ręcznie. Jednak, aby to uczynić potrzebny jest dostęp do klas `RPacketService` oraz `RPacketContext`, której deklaracja znajduje się w pliku `etelpckt.h` (`etelpckt.lib`) – API to należy do nie publicznego API Nokii. Przykładowy fragment kodu, który powinien pozwolić na przeprowadzenie procesu dołączenia do sieci wygląda następująco:

```

RPacketContext::TContextConfigGPRS newConfig;
TPckg<RPacketContext::TContextConfigGPRS> newPacketConfig(newConfig
);

newPacketConfig().iAccessPointName      = DPCKTTSY_ACCESS_POINT1;
newPacketConfig().iAnonymousAccessReqd  = DPCKTTSY_ANONYMOUS_ACCESS1;
newPacketConfig().iPdpCompression      = DPCKTTSY_COMPRESSION1;
newPacketConfig().iPdpType              = DPCKTTSY_PDP_TYPE1;
newPacketConfig().iProtocolConfigOption.iAuthInfo.iUsername =

DPCKTTSY_USER_NAME1;
newPacketConfig().iProtocolConfigOption.iAuthInfo.iPassword =

DPCKTTSY_PASSWORD1;
newPacketConfig().iUseEdge                = DPCKTTSY_EGPRS_REQUIRED;
newPacketConfig().iProtocolConfigOption.iAuthInfo.iProtocol=

DPCKTTSY_PROTOCOL_TYPE1_0;
newPacketConfig().iNWContext              = DPCKTTSY_NWCONTEXT_FALSE;
iPacketServer.SetDefaultContextParams( reqStatus, newPacketConfig );
User::WaitForRequest( reqStatus );
iPacketServer.Attach( reqStatus );
User::WaitForRequest( reqStatus );

```

Z przeprowadzonych testów wynika, iż kod ten faktycznie przeprowadza proces dołączania do sieci – problem polega na ustawieniu poprawnych własności klasy TContextConfigGPRS. Po wywołaniu metod:

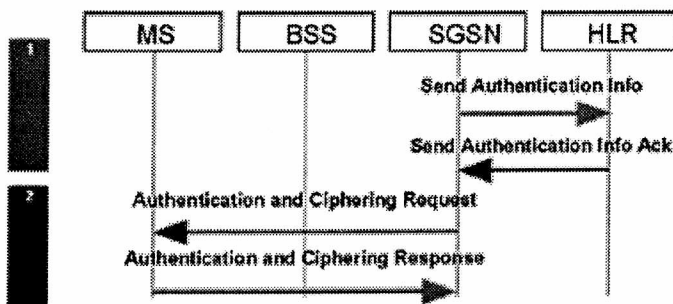
```

iPacketServer.SetDefaultContextParams( reqStatus, newPacketConfig );
User::WaitForRequest( reqStatus );

```

otrzymywano błędy wskazujące, iż ustawienie własności się nie powiodło.

4.2. PROCES AUTENTYFIKACJI W SIECI GPRS



Rys. 6. Proces autentyfikacji w sieci GPRS [Cisco 2002]

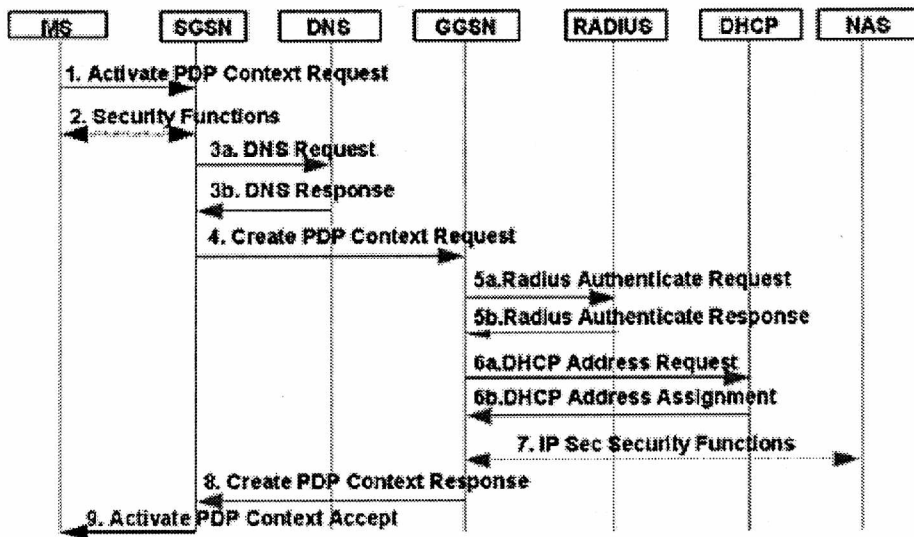
Proces autentyfikacji (rysunek 6) może być przeźroczysty dla programisty, jeśli startowanie połączenia GPRS wykonana się przy pomocy metody `Start` klasy `RConnection`. Metoda ta pobiera potrzebne informacje (czy hasło jest wymagane, jakie jest hasło itp.) z ustawień punktu dostępu (APN), który został wybrany dla danego połączenia.

W przypadku użycia klas `RPacketService` i `RPacketContext`, informacje potrzebne do autentyfikacji można przekazać do klasy `TContextConfigGPRS` przed wywołaniem metody `RPacketService::Attach` w następujący sposób:

```
newPacketConfig().iProtocolConfigOption.iAuthInfo.iUsername=
    DPCKTTSY_USER_NAME1;
newPacketConfig().iProtocolConfigOption.iAuthInfo.iPassword=
    _PASSWORD1;
```

4.3. PROCES AKTYWOWANIA KONTEKSTU PDP

PDP kontekst jest bardzo istotnym elementem użytkowania sieci GPRS, określa on zasady, na których dane urządzenie korzysta z sieci GPRS. Z kontekstem PDP wiąże się również QoS, czyli opis jakości połączenia. Proces aktywowania kontekstu PDP przedstawia rysunek 7.



Rysunek 7. Proces aktywacji kontekstu PDP [Cisco 2002]

W systemie Symbian wywołanie metody `RConnection::Start` powoduje przeprowadzenie również tego procesu – wynika stąd, iż po wywołaniu tej metody na tele-

fonie dostępny będzie nowy kontekst PDP i tak dzieje się w praktyce. Poniżej przedstawiony zostanie kod pozwalający na odczytanie istniejących kontekstów PDP oraz ich własności:

```

RPacketService iPacketServer;
CPdpContext *iPdpContext;
.
.
.
void CGPRSInfo::GetPDPContextsInfos()
{
    TInt count;
    TInt maxAllowed;
    TInt error = 0;
    TRequestStatus reqStatus;

    iPacketServer.EnumerateContexts( reqStatus, count, maxAllowed );
    User::WaitForRequest( reqStatus );

    if ( count > 0 )
    {
        for (TInt i = 0; i < count; i++ )
        {
            RPacketService::TContextInfo info;
            iPacketServer.GetContextInfo(reqStatus, i, info);
            User::WaitForRequest( reqStatus );
            iLogger->LogInt( _L8( "PDP context number = %d\n" ) , i );

            HBufC8 *contextName = HBufC8::NewL(100);
            TPtr8 ptrContextName = contextName->Des();
            ptrContextName.Append( info.iName );
            ptrContextName.Append( _L8( "\n" ) );
            iLogger->Log( ptrContextName );
            delete contextName;

            switch( info.iStatus )
            {
            case RPacketContext::EStatusUnknown:
                iLogger->Log( _L8( "PDP Status = EStatusUnknown\n" ) );
                break;
            case RPacketContext::EStatusInactive:
                iLogger->Log( _L8( "PDP Status = EStatusInactive\n" ) );
                break;
            case RPacketContext::EStatusActivating:
                iLogger->Log( _L8( "PDP Status = EStatusActivating\n" ) );
                break;
            case RPacketContext::EStatusActive:
                iLogger->Log( _L8( "PDP Status = EStatusActive\n" ) );
                break;
            case RPacketContext::EStatusDeactivating:
                iLogger->Log( _L8( "PDP Status = EStatusDeactivating\n" ) );

```

```

        break;
    case RPacketContext::EStatusSuspended:
        iLogger->Log( _L8( "PDP Status = EStatusSuspended\n" ) );
        break;
    case RPacketContext::EStatusDeleted:
        iLogger->Log( _L8( "PDP Status = EStatusDeleted\n" ) );
        break;
}

PDPContextInfo( info.iName );
QoSProfileInfo( info.iName );
}
}

```

Powyższa metoda pobiera listę wszystkich kontekstów PDP i dla każdego z nich pobiera nazwę (konteksty identyfikowane są poprzez unikatową nazwę) oraz status a następnie wywołuje metodę PDPContextInfo (kod poniżej) w celu pobrania szczegółowych informacji o kontekście i metodę QoSProfileInfo (kod poniżej) w celu pobrania informacji o QoS skojarzonym z danym kontekstem.

```

void CGPRSInfo::PDPContextInfo( TName contextName )
{
    iLogger->Log( _L8( "Begin of PDPContextInfo\n" ) );
    TInt error;
    TRequestStatus reqStatus;
    RPacketContext packetContext;

    error = packetContext.OpenExistingContext( iPacketServer, contextName );
    if ( error != KErrNone )
    {
        MessageInt( _L( "OpenExistingContext error = %d\n" ), error );
        return;
    }
    RPacketContext::TContextConfigGPRS contextConfig;
    TPckg<RPacketContext::TContextConfigGPRS> pckgContextConfig( contextConfig );
    iLogger->Log( _L8( "Getting TContextConfigGPRS config...\n" ) );
    packetContext.GetConfig( reqStatus, pckgContextConfig );
    User::WaitForRequest( reqStatus );
    iLogger->LogInt( _L8( "Getting TContextConfigGPRS end=%d\n" ), reqStatus.Int() );

    //pobranie TProtocolType
    iLogger->Log( _L8( "Begin of PDPTYPE\n" ) );
    switch ( contextConfig.iPdpType )
    {
        case RPacketContext::EPdpTypeIPv4:
            iLogger->Log( _L8( "\tEPdpTypeIPv4\n" ) );
            break;
    }
}

```

```

    case RPacketContext::EPdpTypeIPv6:
        iLogger->Log( _L8( "\tEPdpTypeIPv6\n" ) );
        break;
    case RPacketContext::EPdpTypeX25:
        iLogger->Log( _L8( "\tEPdpTypeX25\n" ) );
        break;
    case RPacketContext::EPdpTypePPP:
        iLogger->Log( _L8( "\tEPdpTypePPP\n" ) );
        break;
    case RPacketContext::EPdpTypeCDPD:
        iLogger->Log( _L8( "\tEPdpTypeCDPD\n" ) );
        break;
}
iLogger->Log( _L8( "End of PDPType\n" ) );

//pobranie AccessPointName
iLogger->Log( _L8( "Begin of AccessPointName\n" ) );
TInt state;
TBuf<252> accespointname;
CCnvCharacterSetConverter *converter = CCnvCharacterSetCon-
verter::NewL();
RFS fileServer;
fileServer.Connect();
if(converter-
>PrepareToConvertToOrFromL(KCharacterSetIdentifierAscii,fileServer)
    == CCnvCharacterSetConverter::EAvailable )
{
    converter->ConvertToUnicode( accespointname,
                                contextConfig.iAccessPointName, state );
    HBufC8 *buf = HBufC8::NewL( 253 );
    TPtr8 ptrBuf = buf->Des();
    ptrBuf.Append( accespointname );
    iLogger->Log( ptrBuf );
    iLogger->Log( _L8( "\n" ) );
    delete buf;
}
delete converter;
fileServer.Close();
iLogger->Log( _L8( "End of AccessPointName\n" ) );

//pobranie PtotocolAddress
iLogger->Log( _L8( "Begin of PtotocolAddress\n" ) );
TBuf<252> ptotocoladdress;
converter = CCnvCharacterSetConverter::NewL();
fileServer.Connect();
if(converter-
>PrepareToConvertToOrFromL(KCharacterSetIdentifierAscii,fileServer )
    == CCnvCharacterSetConverter::EAvailable )
{
    converter-
>ConvertToUnicode(ptotocoladdress,contextConfig.iPdpAddress,
state );

```

```

    HBufC8 *buf = HBufC8::NewL( 253 );
    TPtr8 ptrBuf = buf->Des();
    ptrBuf.Append( ptotocoladdress );
    iLogger->Log( ptrBuf );
    iLogger->Log( _L8( "\n" ) );
    delete buf;
}
delete converter;
fileServer.Close();
iLogger->Log( _L8( "End of PtotocolAddress\n" ) );

//pobranie PDPCompression
iLogger->Log( _L8( "Begin of PDPCompression\n" ) );
iLogger->LogInt( _L8("PDPCompression=%d\n" ),
contextConfig.iPdpCompression );
iLogger->Log( _L8( "End of PDPCompression\n" ) );

//pobranie AnonymousAccess
iLogger->Log( _L8( "Begin of AnonymousAccess\n" ) );
switch ( contextConfig.iAnonymousAccessReqd )
{
    case RPacketContext::ENotApplicable:
        iLogger->Log( _L8( "\tENotApplicable\n" ) );
        break;
    case RPacketContext::ERequired:
        iLogger->Log( _L8( "\tERequired\n" ) );
        break;
    case RPacketContext::ENotRequired:
        iLogger->Log( _L8( "\tENotRequired\n" ) );
        break;
}
iLogger->Log( _L8( "End of AnonymousAccess\n" ) );

//pobranie UseEdge
iLogger->Log( _L8( "Begin of UseEdge\n" ) );
if ( contextConfig.iUseEdge == EFalse )
{
    iLogger->Log( _L8( "\tUseEdge = EFalse\n" ) );
}
else
{
    iLogger->Log( _L8( "\tUseEdge = ETrue\n" ) );
}
iLogger->Log( _L8( "End of UseEdge\n" ) );

//pobranie NWIContext
iLogger->Log( _L8( "Begin of NWIContext\n" ) );
if ( contextConfig.iUseEdge == EFalse )
{
    iLogger->Log( _L8( "\tNWIContext = EFalse\n" ) );
}
else

```

```

{
    iLogger->Log( _L8( "\tNWContext = ETrue\n" ) );
}
iLogger->Log( _L8( "End of NWContext\n" ) );

//ProtocolConfigOptionV2 - klasa wchodzaca w sklad klasy
RPacketContext
iLogger->Log( _L8( "-->Begin of ProtocolConfigOptionV2 part of
RPacketContext\n" ) );
//pobranie ProtocolConfigOptionV2::TAuthInfo
iLogger->Log( _L8( "----->Begin of TAuthInfo\n" ) );

//pobranie ProtocolConfigOptionV2::TAuthInfo::TAuthProtocol
iLogger->Log( _L8( "Begin of TAuthProtocol\n" ) );
switch ( contextConfig.iProtocolConfigOption.iAuthInfo.iProtocol )
{
    case RPacketContext::EProtocolNone:
        iLogger->Log( _L8( "\tEProtocolNone\n" ) );
        break;
    case RPacketContext::EProtocolPAP:
        iLogger->Log( _L8( "\tEProtocolPAP\n" ) );
        break;
    case RPacketContext::EProtocolCHAP:
        iLogger->Log( _L8( "\tEProtocolCHAP\n" ) );
        break;
}
iLogger->Log( _L8( "End of TAuthProtocol\n" ) );

//pobranie ProtocolConfigOptionV2::TAuthInfo::Username
iLogger->Log( _L8( "Begin of Username\n" ) );
iLogger->Log( contextConfig.iProtocolConfigOption.iAuthInfo.iUsername
);
iLogger->Log( _L8( "\n" ) );
iLogger->Log( _L8( "End of Username\n" ) );

//pobranie ProtocolConfigOptionV2::TAuthInfo::Password
iLogger->Log( _L8( "Begin of Password\n" ) );
iLogger->Log( contextConfig.iProtocolConfigOption.iAuthInfo.iPassword
);
iLogger->Log( _L8( "\n" ) );
iLogger->Log( _L8( "End of Password\n" ) );
iLogger->Log( _L8( "----->End of TAuthInfo\n" ) );

//pobranie ProtocolConfigOptionV2::Challenge
iLogger->Log( _L8( "Begin of Challenge\n" ) );
iLogger->Log( contextConfig.iProtocolConfigOption.iChallenge );
iLogger->Log( _L8( "\n" ) );
iLogger->Log( _L8( "End of Challenge\n" ) );

//pobranie ProtocolConfigOptionV2::Response
iLogger->Log( _L8( "Begin of Response\n" ) );

```

```

iLogger->Log( contextConfig.iProtocolConfigOption.iResponse );
iLogger->Log( _L8( "\n" ) );
iLogger->Log( _L8( "End of Response\n" ) );

//pobranie ProtocolConfigOptionV2::Id
iLogger->Log( _L8( "Begin of Id\n" ) );
iLogger->LogInt( _L8( "%d" ), contextConfig.iProtocolConfigOption.iId
);
iLogger->Log( _L8( "\n" ) );
iLogger->Log( _L8( "End of Id\n" ) );

// DNSInfo skladowa ProtocolConfigOptionV2
iLogger->Log( _L8( "----->Begin of DNSInfo\n" ) );

//pobranie ProtocolConfigOptionV2::DNSInfo::PrimaryDNS
iLogger->Log( _L8( "Begin of PrimaryDNS\n" ) );
iLogger->Log( contextCon-
fig.iProtocolConfigOption.iDnsAddresses.iPrimaryDns);
iLogger->Log( _L8( "\n" ) );
iLogger->Log( _L8( "End of PrimaryDNS\n" ) );

//pobranie ProtocolConfigOptionV2::DNSInfo::SecondaryDNS
iLogger->Log( _L8( "Begin of SecondaryDNS\n" ) );
iLogger->Log( contextCon-
fig.iProtocolConfigOption.iDnsAddresses.iSecondaryDns);
iLogger->Log( _L8( "\n" ) );
iLogger->Log( _L8( "End of SecondaryDNS\n" ) );
iLogger->Log( _L8( "----->End of DNSInfo\n" ) );

//pobranie ProtocolConfigOptionV2::MiscBuffer
iLogger->Log( _L8( "Begin of MiscBuffer\n" ) );
iLogger->Log( contextConfig.iProtocolConfigOption.iMiscBuffer );
iLogger->Log( _L8( "\n" ) );
iLogger->Log( _L8( "End of MiscBuffer\n" ) );
iLogger->Log( _L8( "----->End of ProtocolConfigOptionV2 part
of
RPacketContext\n" ) );
packetContext.Close(); //wymagane jest zamkniecie kontekstu
iLogger->Log( _L8( "End of PDPCContextInfo\n" ) );
}

void CGPRSInfo::QoSProfileInfo( TName contextName )
{
iLogger->Log( _L8( "Begin of QoSProfileInfo\n" ) );
TInt error;
TRequestStatus reqStatus;
RPacketContext packetContext;

error = packetContext.OpenExistingContext( iPacketServer, contextName
);
if ( error != KErrNone )

```

```

{
MessageInt( _L( "OpenExistingContext error = %d\n" ), error );
return;
}

TName QoSName;
error = packetContext.GetProfileName( QoSName );
if ( error != KErrNone )
{
MessageInt( _L( "GetProfileName error = %d\n" ), error );
packetContext.Close();
return;
}

iLogger->Log( _L8( "QoS name = \n" ) );
HBufC8 *qosName = HBufC8::NewL(100);
TPtr8 ptrQoSName = qosName->Des();
ptrQoSName.Append( QoSName );
ptrQoSName.Append( _L8( "\n" ) );
iLogger->Log( ptrQoSName );
delete qosName;
iLogger->Log( _L8( "\n" ) );

//otwarcie QoS - nowa klasa RPacketQoS
RPacketQoS packetQoS;
packetQoS.OpenExistingQoS( packetContext, QoSName );
RPacketQoS::TQoS_R4Negotiated negotiatedQoS;
TPckg<RPacketQoS::TQoS_R4Negotiated>
pckgNegotiatedQoS(negotiatedQoS);
packetQoS.GetProfileParameters( reqStatus, pckgNegotiatedQoS );
User::WaitForRequest( reqStatus );

//pobranie MaxBitRate
iLogger->Log( _L8( "---->Begin of MaxBitRate\n" ) );
iLogger->Log( _L8( "Begin of Uplink\n" ) );
iLogger->LogInt( _L8( "Uplink = %d kbps\n" ),
                negotiated-
QoS.iMaxRate.iUplinkRate );
iLogger->Log( _L8( "End of Uplink\n" ) );
iLogger->Log( _L8( "Begin of Downlink\n" ) );
iLogger->LogInt( _L8( "Downlink = %d kbps\n" ),
                negotiated-
QoS.iMaxRate.iDownlinkRate );
iLogger->Log( _L8( "End of Downlink\n" ) );
iLogger->Log( _L8( "---->End of MaxBitRate\n" ) );

//pobranie GuaranteedBitRate
iLogger->Log( _L8( "---->Begin of GuaranteedBitRate\n" ) );
iLogger->Log( _L8( "Begin of Uplink\n" ) );
iLogger->LogInt( _L8( "Uplink = %d kbps\n" ),
                negotiatedQoS.iGuaranteedRate.iUplinkRate );

```

```

iLogger->Log( _L8( "End of Uplink\n" ) );
iLogger->Log( _L8( "Begin of Downlink\n" ) );
iLogger->LogInt( _L8( "Downlink=%d kbps\n"),
    negotiatedQoS.iGuaranteedRate.iDownlinkRate);
iLogger->Log( _L8( "End of Downlink\n" ) );
iLogger->Log( _L8( "---->End of GuaranteedBitRate\n" ) );

packetQoS.Close();
packetContext.Close();
iLogger->Log( _L8( "End of QoSProfileInfo\n" ) );
}

```

Jak widać przy obsłudze QoS należy wykorzystać nową klasę, której deklaracja znajduje się w pliku nagłówkowym `etelqos.h`. Z przeprowadzonych testów wynika, iż odczyt informacji o kontekście PDP lub QoS nie stanowi problemu i działa poprawnie. Problemy rodzą się, gdy chcemy ręcznie stworzyć nowy kontekst lub drugi kontekst i samemu je skonfigurować np. ustawić QoS (zadanie stworzenia dwóch kontekstów można zrealizować poprzez utworzenie i wystartowanie dwóch obiektów `RConnection` działających na dwóch różnych APN, jednak wówczas nie mamy wpływu na ustawienia QoS).

API potrzebne do odczytania własności kontekstu PDP pozwala również na utworzenie nowego kontekstu, jednak przeprowadzone testy wskazały, iż API to nie działa poprawnie. Poniżej przedstawiono odpowiedni kod, ale należy pamiętać, iż kod ten zwraca błędy i nie tworzy nowego PDP kontekstu poprawnie.

Tworzenie kontekstu:

```

TInt error;
TRequestStatus reqStatus = KErrNone;
RPacketContext packetContext;

TName contextName;
error = packetContext.OpenNewContext( iPacketServer, contextName );

if ( error != KErrNone )
{
    MessageInt( _L( "OpenNewContext error = %d\n" ), error );
    return;
}

RPacketService::TStatus packetStatus;
TInt status = iPacketServer.GetStatus(packetStatus);
iLogger->LogInt( _L8( "packet staut = %d\n"), packetStatus );

//Set new packet configuration
#define DPCKTTSY_PDP_TYPE1          RPacketContext::EPdpTypeIPv4
#define DPCKTTSY_ACCESS_POINT1     _L8("internet")
#define DPCKTTSY_COMPRESSION1
RPacketContext::KPdpDataCompression

```



```

#define DPCKTTSY_ANONYMOUS_ACCESS1 RPacketContext::ERequired
#define DPCKTTSY_EGPRS_REQUIRED    TRUE
#define DPCKTTSY_PROTOCOL_TYPE1_0 RPacketContext::EProtocolNone
#define DPCKTTSY_NWICONTEXT_FALSE FALSE

RPacketContext::TContextConfigGPRS newConfig;
TPckg<RPacketContext::TContextConfigGPRS> newPacketConfig( newConfig
);

newConfig.iAccessPointName = DPCKTTSY_ACCESS_POINT1;
newConfig.iAnonymousAccessReqd = DPCKTTSY_ANONYMOUS_ACCESS1;
newConfig.iPdpCompression = DPCKTTSY_COMPRESSION1;
newConfig.iPdpType = DPCKTTSY_PDP_TYPE1;
newConfig.iUseEdge = DPCKTTSY_EGPRS_REQUIRED;
newConfig.iProtocolConfigOption.iAuthInfo.iProtocol =

DPCKTTSY_PROTOCOL_TYPE1_0;
newConfig.iNWIconText = DPCKTTSY_NWICONTEXT_FALSE;

packetContext.SetConfig( reqStatus, newPacketConfig );
User::WaitForRequest( reqStatus );

packetContext.Activate( reqStatus );
User::WaitForRequest( reqStatus ); //zwraca błąd -5 Not Supported
iLogger->LogInt( _L8("Activate = %d\n"), reqStatus.Int() );

```

Tworzenie drugiego kontekstu PDP:

```

TInt count;
TInt maxAllowed;
TRequestStatus reqStatus;

iPacketServer.EnumerateContexts( reqStatus, count, maxAllowed );
User::WaitForRequest( reqStatus );

RPacketService::TContextInfo info;
iPacketServer.GetContextInfo(reqStatus, 0, info);

RPacketContext packetContext;
TName newContextName;
Error
= packetContext.OpenNewSecondaryContext(iPacketServer,info.iName,
newContextName );
if ( error != KErrNone )
{
    MessageInt( _L( "OpenNewContext error = %d\n" ), error );
    return;
}

//Set new packet configuration
#define DPCKTTSY_PDP_TYPE1 RPacketContext::EPdpTypeIPv4

```

```

#define DPCKTTSY_ACCESS_POINT1      _L8("ACME INTERNET PROVIDERS LTD")
#define DPCKTTSY_PDP_ADDRESS1       _L8("i-MODE GATEWAY")
#define DPCKTTSY_COMPRESSION1       RPacketCon-
text::KPdpDataCompression
#define DPCKTTSY_ANONYMOUS_ACCESS1  RPacketContext::ENotRequired
#define DPCKTTSY_EGPRS_REQUIRED     TRUE
#define DPCKTTSY_USER_NAME1         _L8("ONYEARA")
#define DPCKTTSY_PASSWORD1          _L8("1234")
#define DPCKTTSY_PROTOCOL_TYPE1_0   RPacketContext::EProtocolNone
#define DPCKTTSY_NWICONTEXT_FALSE   FALSE

RPacketContext::TContextConfigGPRS newConfig;
TPckg<RPacketContext::TContextConfigGPRS> newPacketConfig( newConfig
);

newConfig.iAccessPointName = DPCKTTSY_ACCESS_POINT1;
newConfig.iAnonymousAccessReqd = DPCKTTSY_ANONYMOUS_ACCESS1;
newConfig.iPdpAddress = DPCKTTSY_PDP_ADDRESS1;
newConfig.iPdpCompression = DPCKTTSY_COMPRESSION1;
newConfig.iPdpType = DPCKTTSY_PDP_TYPE1;
newConfig.iUseEdge = DPCKTTSY_EGPRS_REQUIRED;
newConfig.iProtocolConfigOption.iAuthInfo.iUsername
=DPCKTTSY_USER_NAME1;
newConfig.iProtocolConfigOption.iAuthInfo.iPassword =
DPCKTTSY_PASSWORD1;
newConfig.iProtocolConfigOption.iAuthInfo.iProtocol =

DPCKTTSY_PROTOCOL_TYPE1_0;
newConfig.iNWIconText = DPCKTTSY_NWICONTEXT_FALSE;

packetContext.SetConfig( reqStatus, newPacketConfig );
User::WaitForRequest( reqStatus );

```

Wniosek płynący z przeprowadzonych testów jest następujący: nie da się utworzyć ręcznie nowego kontekstu PDP z API Symbiana 8.0, ponieważ API zwraca błędy typu -5. Inaczej sytuacja wygląda w Symbianie 9.1, gdzie po części udało się to zadanie zrealizować dzięki nowej klasie RSubConnection. Dzięki tej klasie możliwe jest ustawienie parametrów QoS tzn. nie tworzymy ręcznie kontekstu PDP, ale korzystamy z klasy RConnection do utworzenia kontekstu a następnie modyfikujemy go przy pomocy klasy RSubConnection. Sytuację tę prezentuje następujący kod:

```

//Tworzenie obiektów klas RConnection i RSubConnection
TInt error = 0;
TRequestStatus reqStatus;

error = iSocketServ.Connect();
iLogger->LogInt( _L8( "iSocketServ.Connect error = %d\n" ), error );
if ( error != KErrNone )
    return;

```

```

error = iConnection.Open( iSocketServ /*,KafInet*/ );
iLogger->LogInt( _L8( "iSocketServ.Open error = %d\n" ), error );
if ( error != KErrNone )
    return;

iConnection.Start( reqStatus );
User::WaitForRequest( reqStatus );
iLogger->LogInt( _L8( "iConnection.Start reqStatus
=%d\n"), reqStatus.Int() );

er-
ror=iSubConnection.Open(iSocketServ,RSubConnection::ECreateNew,iConne
ction);
iLogger->LogInt( _L8( "iSubConnection.Open error = %d\n" ), error );
if ( error != KErrNone )
    return;

iLogger->Log( _L8( "iSubConnection.EventNotification..." ) );
iSubConnection.EventNotification( iNotificationEventBuf, EFalse,
iStatus

);
SetActive();
iLogger->Log( _L8( "----->OK\n" ) );

//Ustawienie parametrów QoS przy pomocy RSubConnection
TInt error = 0;
RSubConParameterBundle subConParameterBundle;
CSubConParameterFamily *subConParameterFamily;
TRAP( error, subConParameterFamily = CSubConParameterFamily::NewL(
                                subConParameterBundle,
1 ) );
iLogger->LogInt( _L8( "CSubConParameterFamily::NewL    error=%d\n"),error
);
if ( error != KErrNone )
    return;

CSubConQosIPLinkR99ParamSet *qosIPLink;
TRAP( error, qosIPLink = CSubConQosIPLinkR99ParamSet::NewL(
                                *subConParameterFamily,
CSubConParameterFamily::ERequested ) );
iLogger->LogInt( _L8( "CSubConQosIPLinkR99ParamSet::NewL  error=%d\n" ),
error);
if ( error != KErrNone )
    return;

iLogger->Log( _L8( "SetMaxBitrateUplink...\n" ) );
qosIPLink->SetMaxBitrateUplink(5120);
iLogger->Log( _L8( "----->OK\n" ) );
iLogger->Log( _L8( "SetMaxBitrateDownlink...\n" ) );
qosIPLink->SetMaxBitrateDownlink(5120);
iLogger->Log( _L8( "----->OK\n" ) );

```

```

iLogger->Log( _L8( "SetMinBitrateUplink...\n" ) );
qosIPLink->SetMinBitrateUplink(5120);
iLogger->Log( _L8( "----->OK\n" ) );

iLogger->Log( _L8( "SetMinBitrateDownlink...\n" ) );
qosIPLink->SetMinBitrateDownlink(5120);
iLogger->Log( _L8( "----->OK\n" ) );

TRAP( error, subConParameterFamily->AddExtensionSetL( *qosIPLink,
CSubConParameterFamily::ERequested ) );
iLogger->LogInt( _L8("subConParameterFamily.AddExtensionSetL
error=%d\n"),error);
if ( error != KErrNone )
    return;

TRAP( error, subConParameterBundle.AddFamilyL( subConParameterFamily
) );
iLogger->LogInt( _L8( "subConParameterBundle.AddFamilyL error = %d\n"
),
error );
if ( error != KErrNone )
    return;

error = iSubConnection.SetParameters( subConParameterBundle );
iLogger->LogInt( _L8( "iSubConnection.SetParameters error = %d\n" ),
error);
if ( error != KErrNone )
    return;

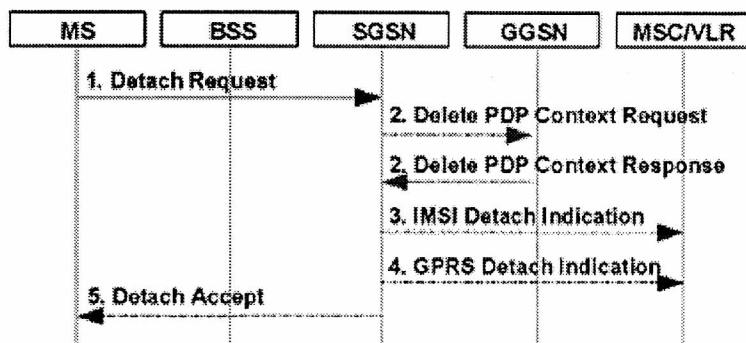
//Odczyt ustawionych parametrów
RSubConParameterBundle subConParameterBundle2;
iSubconnection.GetParameters(subConParameterBundle2);
CSubConParameterFamily *proba = subConParameterBun-
dle2.FindFamily(subConParameterFamily->Id());
CSubConQosIPLinkR99ParamSet*proba2=
(CSubConQosIPLinkR99ParamSet*)proba->FindExtensionSet(
1,CSubConParameterFamily::ERequested );
iLogger->LogInt( _L8("MaxBitrateUplink = %d\n", proba2-
>GetMaxBitrateUplink() );
iLogger->LogInt( _L8("MaxBitrateDownlink=%d\n",proba2-
>GetMaxBitrateDownlink());
iLogger->LogInt( _L8("MinBitrateUplink = %d\n",proba2-
>GetMinBitrateUplink() );
iLogger->LogInt( _L8("MinBitrateDownlink=%d\n",proba2-
>GetMinBitrateDownlink() );

```

W odróżnieniu do API systemu Symbian 8.0 kod stworzony przy pomocy API Symbiana 9.1 wykonuje się bezbłędnie i po ustawieniu parametrów QoS odczytywane są nowo ustawione wartości. Należy jednak zwrócić uwagę, iż przeprowadzone testy

na urządzeniu nie wskazują jednoznacznie na przyspieszenie transferu. Nie wiadomo zatem, czy nowe wartości są respektowane przez system operacyjny.

4.4. PROCES ODŁĄCZANIA OD SIECI GPRS



Rys. 8. Proces odłączania od sieci GPRS [Cisco 2002]

Proces odłączania od sieci GPRS w kodzie Symbiana można uruchomić:

- poprzez wywołanie metody Stop klasy RConnection,
- poprzez wywołanie metody Detach klasy RPacketService w następujący sposób:

```

TRequestStatus reqStatus;
iPacketServer.Detach (reqStatus);
User::WaitForRequest (reqStatus);
  
```

5. ZAKOŃCZENIE

W pracy przedstawiono aspekty architektury sieci GSM/GPRS oraz procesy zachodzące w tych sieciach w kontekście programowania urządzeń mobilnych. Zaprezentowane zostały przykłady kodu obrazującego jak procesy takie można przeprowadzić wykorzystując urządzenia działające z systemem operacyjnym Symbian OS.

Przedstawiono opisy przebiegu procesów dołączania oraz odłączania urządzenia do/z sieci GPRS, procesów tworzenia i aktywowania kontekstu. Przy omawianiu każdego z tych procesów starano się zaprezentować odpowiedni kod oraz omówić jakie problemy można w kontekście danego procesu napotkać oraz co jest możliwe do realizacji, a co nie.

Z przeprowadzonych testów wynika, iż nie wszystko jest możliwe, np. w przypadku własności sieci takich jak minimalna i maksymalna przepustowość (ang. *min*

and max bitrate) przypisana do danego urządzenia – okazuje się, iż własności te można tylko odczytać, ale nie można już nimi manipulować. Jest to pewne ograniczenie i podczas programowania urządzeń mobilnych i realizacji projektów ograniczenie to, jak i wiele podobnych należy mieć na uwadze. Okazuje się, iż w wielu aspektach programowania sieciowego, w kontekście urządzeń mobilnych, programiści są zdani na to, co oferuje dany operator sieci komórkowej, np. minimalne i maksymalne przepustowości.

Ograniczenia te należy znać i zdawać sobie z nich sprawę, ponieważ może to wpłynąć na przedłużenie czasu trwania projektu, zmianę jego pierwotnych założeń lub zakończenie takiego projektu zupełnym fiaskiem.

LITERATURA

- [Anders 2001] Andersson Ch., *GPRS and 3G Wireless Applications*. Wiley, John & Sons, 2001.
- [Bates 2001] Bates R.J., *GPRS: General Packet Radio Service*. McGraw-Hill Professional, 2001.
- [BraWal 1997] Brasche G., Walke B., *Concepts, Services, and Protocols of the new GSM Phase 2+ General Packet Radio Service*. Aachen University of Technology, 1997.
- [Cisco 2002] *Cisco Mobile Exchange (CMX) Solution Guide*. Cisco Systems, Inc., Rozdział 2, 2002.
- [EbeVoBe 2001] Eberspaecher J., Voegel H., Bettstetter Ch., *GSM Switching, Services and Protocol*, Wiley, 2001.
- [HaRoMe 2002] Halonen T., Romero J., Melero J., *GSM, GPRS and EDGE Performance: Evolution Toward 3G/UMTS*. John Wiley & Sons, 2002.
- [KavBec 2002] Kavanagh A., Beckmeyer J., *GPRS Networks*. Osborne Publishing, 2002.

THE ASPECTS OF THE GSM/GPRS BASED NETWORKS IN THE SYMBIAN PROGRAMMING CONTEXT

In this paper we present and discuss elements of the GSM/GPRS based network architecture and processes occurring within such networks together with code examples. The examples illustrate how such processes should be performed and what problems can be encountered during their execution.

Jacek GRUBER*

Tomasz BRZOZOWSKI*

SERVICE ORIENTED MODELING AND IMPLEMENTING OF BUSINESS PROCESSES ON THE SYBASE PLATFORM

For the last few years Sybase Company [SYB] proved its commitment to improving business modeling by creating integrated infrastructure for modeling enterprise businesses. Thanks to the incorporation of Service Oriented Architecture (SOA), Web services and business process execution languages, Sybase has significantly extended business modeling in almost every direction. This article is intended to provide a short survey of Sybase approach, examining its usage and related key aspects – among them: defining static and dynamic view of business organization, implementing business processes by specifying external services and deploying processes to a target execution platform, which allows orchestration and execution simulation.

1. INTRODUCTION

Historically, the primary goal in Business Process Modeling, as one of the disciplines in Software Development Lifecycle, was to create common communication platform between system developers and people strictly related to the business. Business modeling languages, graphical notations, business UML profiles – they all were genuinely meant to provide infrastructure for expressing business processes (workflows, business objects, rules), in a way, which could make them understandable for IT architects. Along the evolution of business modeling both sides of the contract formulated a few new expectations, which changed a bit the primary purpose of business modeling. Stakeholders and end-users expect to have a tool for orchestration, management and simulation of different business organizations to be able to quickly choose enterprise configuration, which would support company end-goals in most efficient manner. On the other hand, developers demand business

* Instytut Informatyki Stosowanej, Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27, jacek.gruber@pwr.wroc.pl, brzozowski.tomasz@gmail.com

modeling architecture, which could get most out of the latest IT technologies and which could be semi-automatically transformed into executable system architecture by implementing processes using service-oriented approach.

Nowadays there are a lot of different modeling environments, which support enterprise business modeling. One of them was proposed by Sybase and the thing, which seems to make it slightly different from all the others is business driven approach to software development. Implementing this approach forced Sybase to put business modeling in the very foreground and at the time being Sybase is one of the top leaders in business process modeling and implementation. Attempts to fulfill developers' and stakeholders' expectations pushed Sybase to integrate business modeling with SOA. This resulted in creating executable business modeling environment, which remained completely independent from application considerations and which at the same time enables rapid and seamless integration within the enterprise.

This short article is intended to describe key aspects of business modeling approach presented by Sybase, mostly by presenting, how to design analytic and executable business process model using PowerDesigner platform and then how to orchestrate and execute business processes using Unwired Orchestrator. We also decided to present an academic case study to better visualize the whole idea, as well as to highlight couple very specific aspects, especially concerning implementation of business processes using Web services.

2. BUSINESS MODELLING ACCORDING TO SYBASE – QUICK OVERVIEW

Business Process Modeling on Sybase platform is divided into two methodological aspects:

- **Analysis Business Process Modeling** – this aspect is responsible for providing “common language” between people strictly related to the business and people not familiar with it. It provides graphical notation for describing enterprise business organization on a very high level of abstraction.
- **Executable and Orchestration Business Process Modeling** – this aspect covers implementation of previously defined and structured processes by orchestrating activities and executing services according to Service Oriented Architecture. This aspect is also responsible for supervision and simulation of process execution.

Typical strategy for business modeling assumes starting with analysis model, which is perfect for discovering existing processes and data flows within an enterprise organization. Moreover even a non-technical analyst without any IT knowledge is capable of defining company organization by building Analysis Business Process Model (Analysis BPM). Next we can generate the analysis model into a new BPM for

SOA executable language. Choosing SOA for the executable language still gives us very high level of abstraction and still doesn't include any technical considerations. This allows developers and business people to work together on execution business architecture.

Once we have valid Execution BPM (EBPM) we can very easily generate it into another EBPM, but this time choosing less abstract execution language. In practice we can choose between "Sybase Unwired Orchestrator Language" (using Orchestrator as a target execution engine) and "Business Process Execution Language for Web Services" – BPEL4WS (to drive the implementation using application server like BEA@WebLogic). After checking, if newly created model was generated in accordance with the target language, we can move further and start importing WSDL files and implementing processes utilizing imported Web services. But this is not the end, we can step even further by exporting our model to Unwired Orchestrator project, which makes it possible to deploy implemented processes and to refine all the aspects related to processes execution. Figure 1 shows a quick overview of business modeling approach proposed by Sybase.

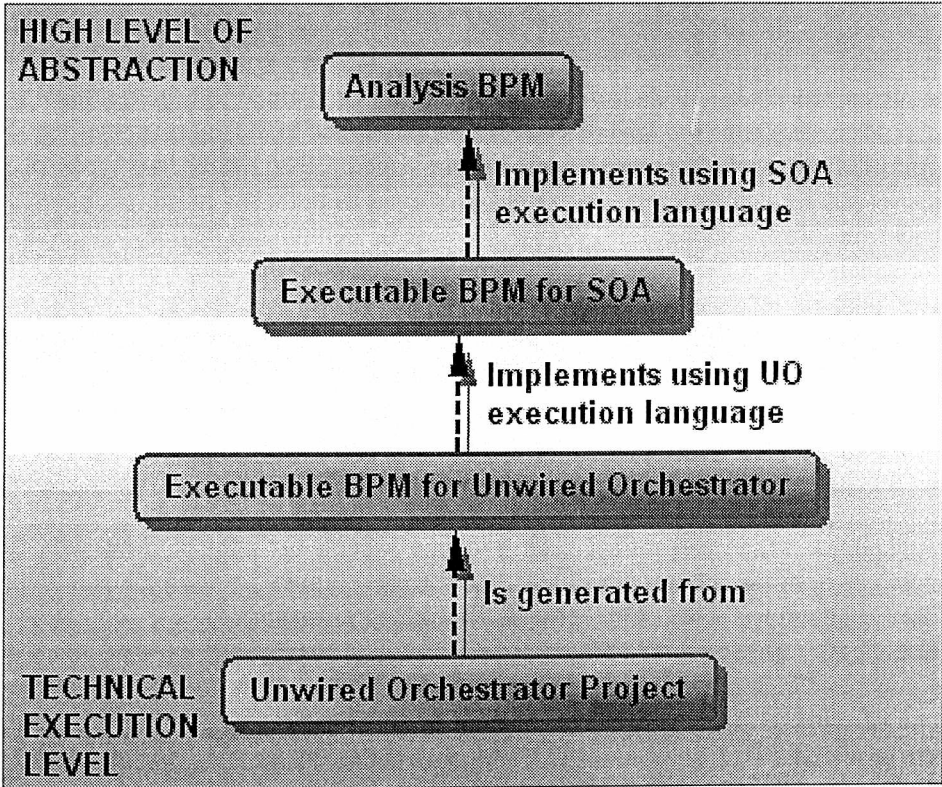


Fig. 1. Relations and dependencies between business modeling artifacts

3. ANALYSIS BPM

Since the very beginning of business modeling IT developers, who are used to technical description, had a huge problem with capturing business requirements as well as capturing the current and future business processes using non-technical description. Analysis BPM is one of the business modeling elements, which appeared when IT people realized that business process modeling is something completely different than application modeling and it requires a lot more abstract approach.

At this stage we should answer three questions:

- What are our business requirements?
- How do existing business processes work?
- How should the new system work?

First step in designing analysis BPM is discovering top-level processes representing top-level business functions. Each top-level business process can be defined as set of activities, which realize common business goal or common business function. In the next steps previously identified top-level processes have to be decomposed (into set of sub-processes) down to the elementary level. Elementary level process is defined as “single activity performed by a single person at single point in time and place, which brings specific business value and leaves company data in a consistent state” [PDUG]. Discovered processes and their decompositions can be defined in a hierarchical business process diagram, which becomes the first artifact of business modeling. This diagram defines static structure of our business model using only two types of elements: processes and decomposition links displayed as branches in the diagram (lines that join the process symbols) – Figure 2 presents hierarchical business process diagram, which defines the scope of our example, which is in this case “order processing system”.

By looking at the Figure 2 we can easily say, which process is composite, and which sub-processes is it made up from. However we can't say anything about execution order for sub-processes and interactions between them. Hierarchic structure is far from satisfactory when it is about expressing dynamic aspect of business process modeling. That's why after finishing hierarchic diagram we must move to analysis of separate processes. For every composite process we need to design business process flow diagram, which can reflect such dynamic aspects as: control flow, data flow, interactions between processes and message formats (Figure 3 presents business process flow diagram for “take order” process from previous example).

Even a very quick glance at the Figure 3 gives us much more information, than it was in case of hierarchic diagram. When defining process flow we have much bigger palette of objects, which allow representing dynamic aspect of any kind of business process. Within these objects we can find many elements typical for standard flow diagrams, among them: organization units (representing company department or single person, who is responsible for executing specific process), starting and ending points,

decision points, branches and synchronizations, and of course process objects. There are also two additional elements: message formats and resources. The latter is quite obvious – as a resource we can understand any data storage or data supplying system (mostly databases but not necessarily). But the most important are message formats, which can be defined as “format of a piece of information exchanged between business processes. The format depends on the nature of the exchanged information and allows objects to find an agreement to communicate” [PDUG]. Why is it so important? The answer is simple – this element is mainly responsible for getting SOA into action. To be able to use Service Oriented Approach we need to define format of messages, which are exchanged between a process and its service provider (object which delivers services that are used for process implementation). This can’t be done properly without specifying messages exchanged between processes first – a process can’t provide its service provider with data, which isn’t delivered to the process from one of the resources or from some other process.

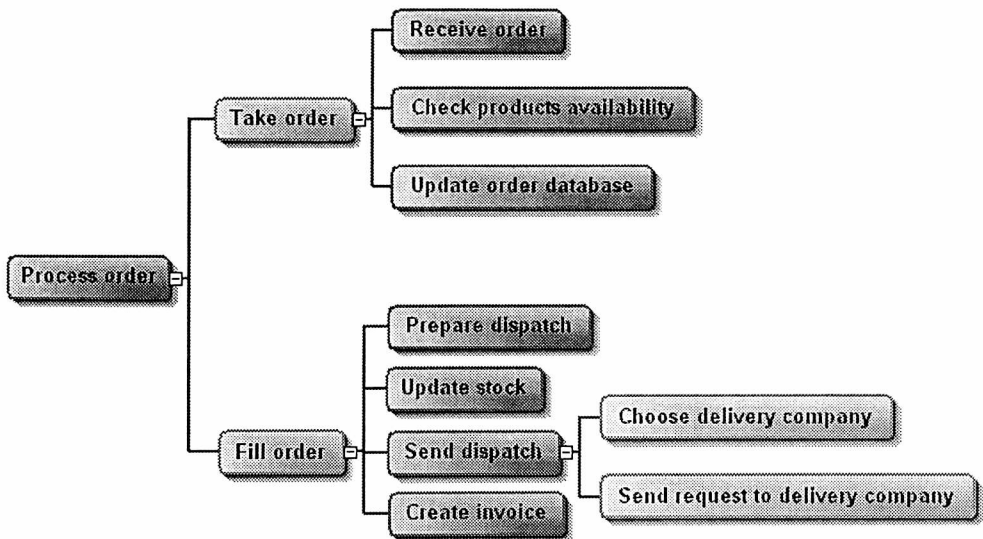


Fig. 2. Hierarchic diagram representing decomposition for a composite process

Message formats can be specified in several ways, e.g. by defining message parts using simple data types or by defining a company form, which is well known inside the organization. However the easiest way of doing this seems to be defining XML documents, such as XML Schema or DTD (Data Type Definition), which are great for describing data structure.

To sum up analysis Business Process Modeling we’d like to emphasize prime goal, which should be realized at this stage – designing company organization by defining decomposition and choreography for top-level business processes. For the reason that

analysis models impose less constraints than execution models, they're perfect for presenting business requirements, the way business processes work and how the new system should work. High level of abstraction and independency from implementation considerations allow very close cooperation with stakeholders and end-users, which is necessary to streamline the communication between business and IT. Analysis business process model belongs to the easiest artifacts to build, however most of the time it appears to be very helpful in optimizing operational costs, integrating data and other information with existing applications, as well as managing internal and cross-enterprises processes.

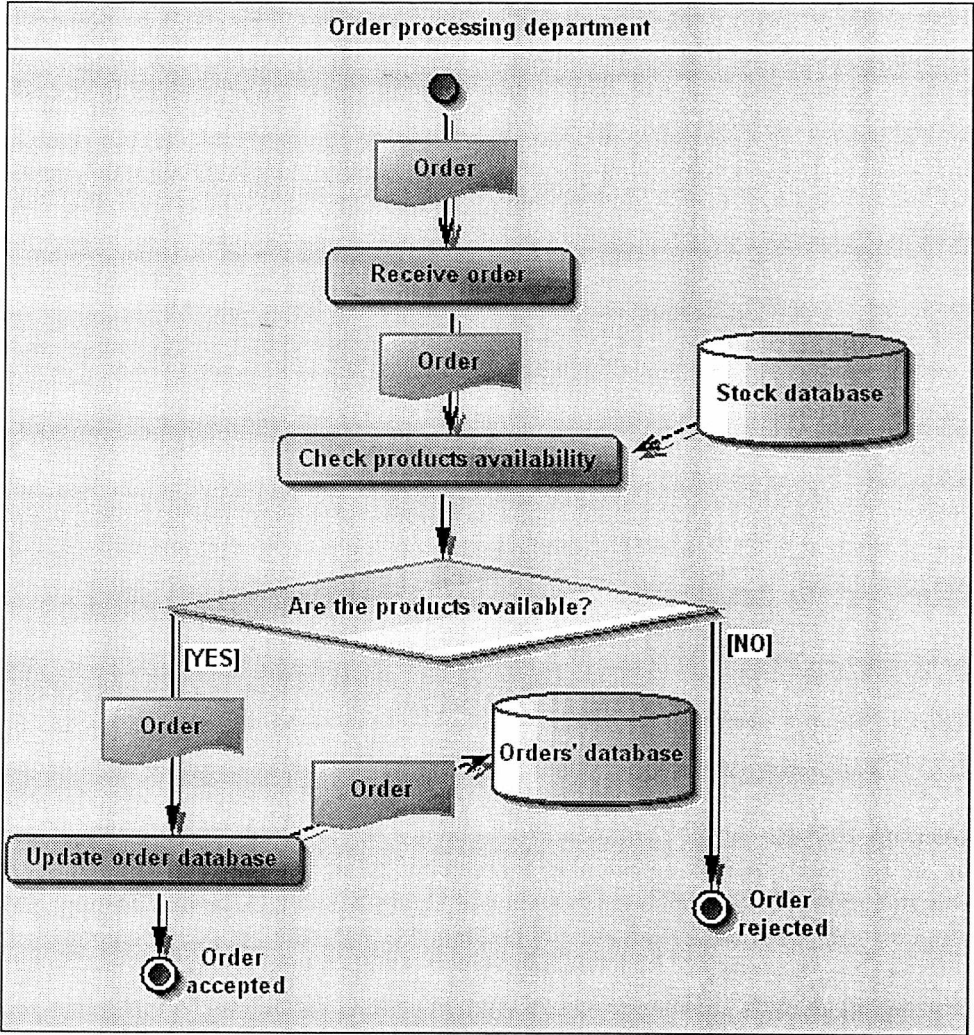


Fig. 3. Business process flow diagram for “take order” process

4. XML MODEL AS SUPPORT FOR BUSINESS MODELING

Previous section for the first time in this paper directed our attention to XML format, which appeared to be useful to specify message formats exchanged between processes. Of course it's not the only element in business modeling, which utilizes XML format. Among the available tools offered by PowerDesigner we can find dedicated XML model, which allows graphically editing and representing the following XML files: XML Schema Definition, Document Type Definition and XML-Data Reduced.

XML model provides two perspectives: tree-hierarchy and XML diagram. We need to remember that there are two things to consider when modeling XML: the data we want to model and how we want to write the XML document, which can be represented in many different ways by different schemas but contain the same structure. Both perspectives provide global and schematic view of all the elements making up our model, which is extremely useful, when we need to quickly check, edit or interpret any XML element.

To show how to model XML elements and then utilize them in business model let's build XML Schema file, which could represent "Order message format" for our "Order Processing System" example. We need to start with creating new XML model, in which we have to define XML element "Supplier Order". All we have left to do is creating XML diagram for our new element (using this diagram we have to build hierarchical structure representing content of our „Supplier order" element). Among the available objects, which can be used for creating XML diagrams, we can find for example: "xml elements", different kinds of groups (sequences, sets, choices...), attributes, references and entities. Figure 4 illustrates possible structure, which defines "Supplier Order" element.

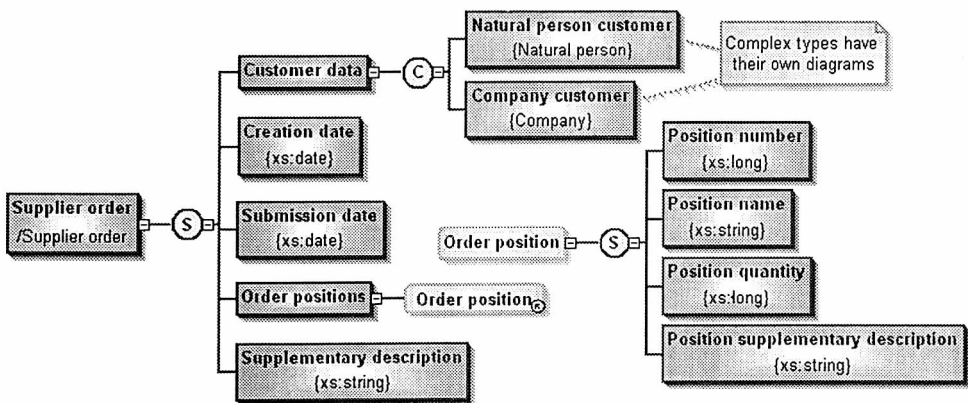


Fig. 4. XML diagram representing hierarchical structure for "Supplier Order" element

As the Figure 4 shows, XML diagram becomes an extremely lucid structure, which makes interpretation and editing much easier because it doesn't require digging through hundreds of XML tags to find interesting element. Of course every valid XML diagram can be automatically transformed into XSD, DTD or XDR file, which can be imported into any BPM and freely utilized. Conversely, we can also reverse engineer a DTD, XSD or XDR file into an XML model.

5. EXECUTABLE BUSINESS PROCESS MODEL (EBPM)

Building analysis business process model, which describes hierarchy and choreography for discovered business processes still makes it impossible to model, how we can use services provided by our business partners, other departments of the company or provided by potentially existing inherited system, which is to be updated. Of course Sybase architects didn't pass over this subject and from version 11.0 PowerDesigner platform includes Executable Business Process Model (EBPM), which becomes the element, where SOA shows the most of its potential.

The most popular approach to building EBPM starts with generating existing analysis model into EBPM for SOA execution language, which remains abstract from any execution platform and at the same time allows implementing elementary processes. This implementation is done by designing choreography for activities, which make up a single elementary process. The activity, which is responsible for getting SOA into action, is of course "external service execution", that can be:

- the call of a stored procedure in a database
- the call of a component in an application server
- the exchange of a message with an external partner

To make use of existing services we need to import appropriate WSDL file. PowerDesigner processes WSDL files in a way, which converts Web services into corresponding service description objects and EBPM objects. However only the abstract description of Web services can be imported: messages, operations and port types. Among the objects, which are created in EBPM are: service providers (objects that gathers a set of service interfaces, for which they represent a namespace), service interfaces (objects that gathers a set of operations, for which they represents a namespace) and operations (objects that describe the implementation of atomic processes).

WSDL elements	EBPM objects
Message	Message format
PortType	Service interface
Operation	Operation
WSDL file	Service provider

Fig. 5. WSDL elements and their corresponding objects in EBPM [PDUG]

When we don't have existing Web services, which could be utilized to implement an activity, instead of importing WSDL files we can define operations manually. We need to start with creating service provider, then define a service interface for it, and within the interface define operations by specifying type of the operation and required message formats. Available operation types are: one-way operation, request-response, solicit response and notification. The differences between them reside in the presence and the sequence of input and output messages. Naturally operations defined manually can't be executed until appropriate components, which implement these operations, are deployed. That's why typically defining operation manually becomes a trigger to create component in Object Oriented Model (Figure 6 presents typical schema for implementing processes in EBPM).

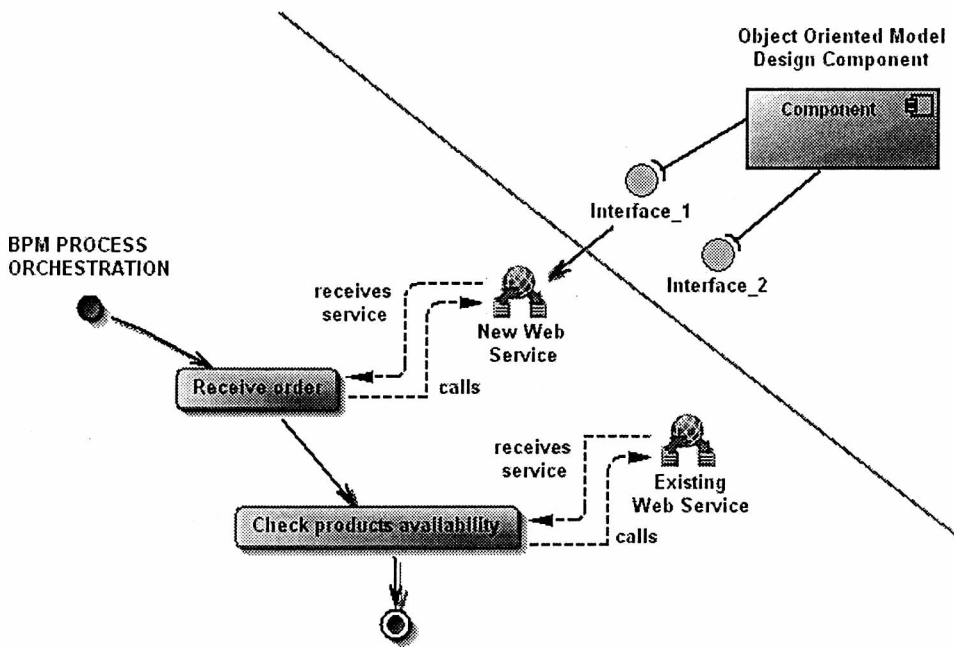


Fig. 6. Typical schema for implementing business process in EBPM [PDUG]

Once again we'd like to use our "Take Order" process to present a possible implementation. Let's suppose that we don't have any existing Web services, which could be used, that's why we have to start by defining a service provider, which will represent our "Order Processing System", and add an interface (just to make it more real let's have two interfaces: one for public customers, and one for enterprise members, which provides some extra logic to receiving order – Figure 7 presents defined structure). Now we can define required operations within the interfaces. For every single operation we must provide input message format and output message format – once

again XML model might come in handy. Of course if we had an existing system instead of creating new service provider we could import appropriate WSDL files (alternatively use UDDI server) and simply pick required services from the list of available ones. Once we have all the necessary services, all we have left to do is assigning services to corresponding elementary processes.

Practically at this level of abstraction this is as far as we can go. To be able to perform more detailed implementation we need to transform our EBPM to another one choosing less abstract language. So what exactly are the advantages of building EBPM for SOA? Every EBPM makes it possible to utilize effort, which was put into analysis level and go further with our design by creating basic implementation architecture. We should think of it as a “bridge” between analysis BPM and one of the execution target platforms.

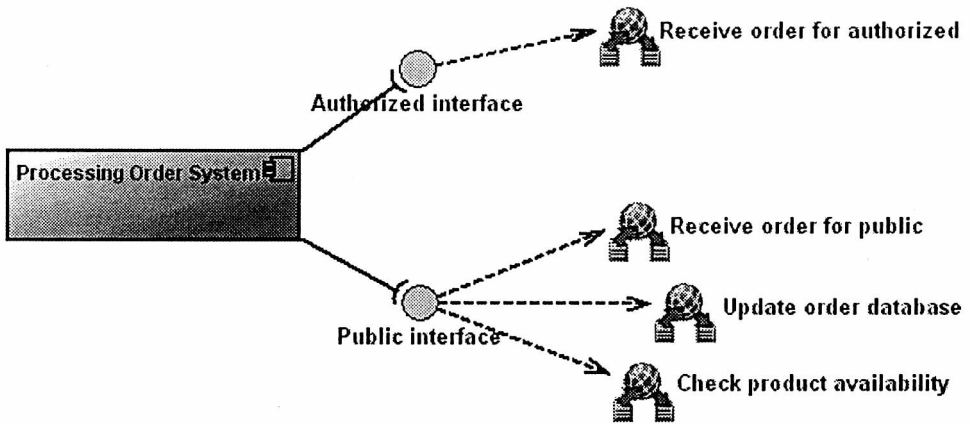


Fig. 7. “Processing order system” as a service provider and its related objects

5. INTEGRATION OF POWERDESIGNER AND UNWIRED ORCHESTRATOR

Now it’s the time, when we need to decide, which executable language we want to use for generating EBPM for target execution engine. We can choose to generate for different business languages like Sybase Integration Orchestrator, Sybase WorkSpace Business Process, or Business Process Execution Language for Web services. This generation enables several levels of abstraction so the analyst can regenerate the executable model several times; it also performs some, but not all, transformations to adapt the source model to the constraints of the selected executable process language. In this article we’ll concentrate on PowerDesigner’s integration with Unwired Orchestrator (Sybase’s composite application engine).

It's really hard to clearly define border between PowerDesigner and Unwired Orchestrator but typically the former is used for designing and the latter for refinement and deploying (Figure 8 presents business modeling phases with its corresponding tools). Choosing target process engine finally gives us full control over process implementation. All the stereotypes used in EBPM for SOA have to be regenerated to assure compatibility with Orchestrator, because at this level we can use only object types supported by chosen execution platform. The three most important concepts (except those presented earlier) at this level are:

- Map – Defines a data transformation using an XSLT file. Used within a business process, it defines a map activity.
- Service interaction – Invocation of one operation defined by a service. It defines which XSD files are used to match the inputs and/or outputs of the operation. It is defined in an interface file (.iface). Used within a business process, it defines a service interaction activity.
- Business process context – Set of variables that defines a state of the process.

Looking at the above list we can see that at this level we can finally put the whole process execution together by defining necessary variables and transformations to share data within complex processes. At this stage we also have a lot bigger tool palette. Among the available tools we can find such new elements as:

- for control flow designing: loops, breaks, delays, splits, rules, joins,
- for error handling: faults, compensates,
- for context defining: maps, assigns.

Once again we'll use our "Take Order" example for illustrating a possible implementation (Figure 9 presents the result). At this level it is pure implementation, we need to choose necessary elements and specify their attributes providing as much details as possible. As soon as our EBPM is valid we are ready to export it to our integration engine using automatic generation mechanism for Unwired Orchestrator code. Next we have to open up the generated project file in Unwired Orchestrator and deploy the business processes. Once deployed they can be automatically tested using a set of previously prepared XML documents containing test data.

To sum up executable business process modeling we'd like to emphasize that although this part of modeling appeared for the first time a short time ago; at the time being it becomes the most important part of the whole system development lifecycle. Thanks to the incorporation of SOA, Web Services, WSDL, etc.; EBPM is expected to revolutionize the use and benefits of software modeling. Looking at companies like Sybase we can observe a methodological paradigm shift. So far we were lost somewhere between data and application modeling, however these days are over – nowadays technology is finally capable to push software development into a higher level and put business in the first place.

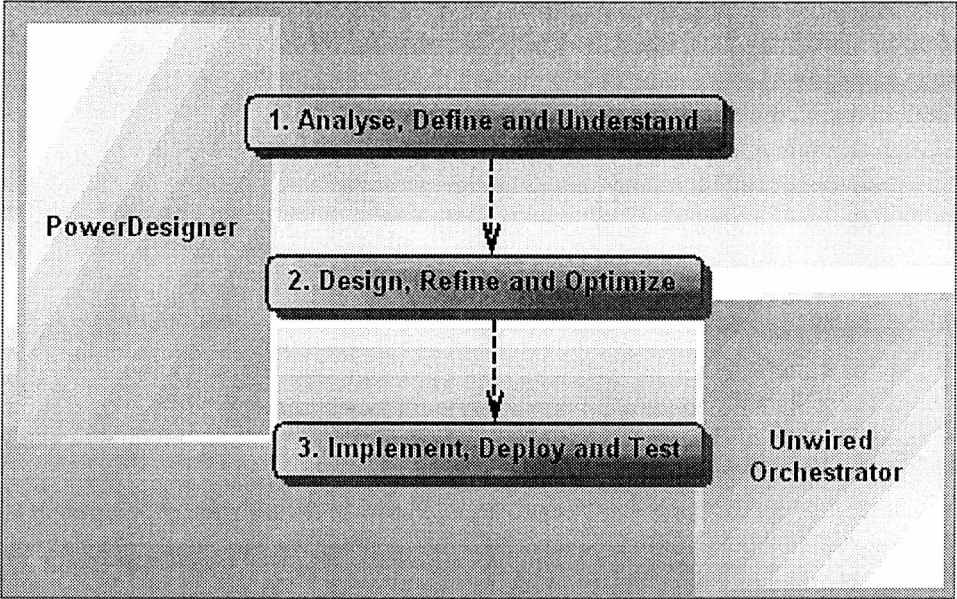


Fig. 8. Business modeling phases and its corresponding tools [SWUG]

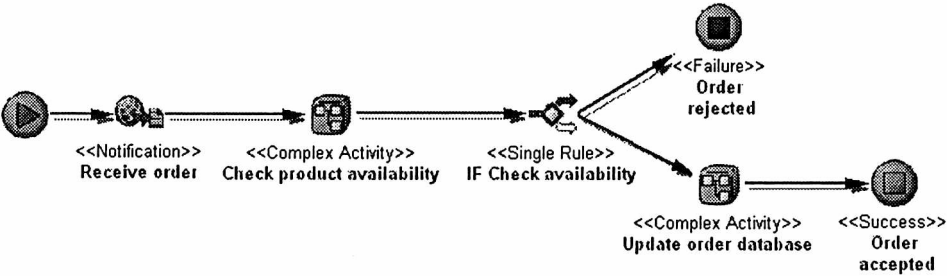


Fig. 9. Possible Orchestrator implementation for “Take Order” process

6. GOING EVEN FURTHER

Every enterprise organization needs to quickly respond to business events as they occur within the context of a business process, that’s why enterprise business modeling must support collecting, analyzing, and displaying business data. So far we didn’t provide any information on, how to monitor business processes during execution. We should remember that this might be crucial for discovering disadvantageous issues in our business architecture such as: bottlenecks, leaks and so on. The easiest way to monitor our business is using Business Activity Monitoring (BAM) included in

BizTracker, which is an integral part of Unwired Orchestrator and provides comprehensive monitoring and auditing functionality. Presenting all the features of BAM and BizTracker would be a great subject for an individual article, that's why in this paper we decided to list some of the key functionalities providing only a very short description. Among BAM's features we can find for example:

- Message and data monitoring – we can monitor process execution having ‘any-time’ access to events, messages, and data used within the process regardless of source.
- Defining metrics and performance indicators – we can easily define any kind of a metric or key performance indicator (KPI), moreover we can define events based on our metrics.
- Visual presentation (customizable dashboard) – for every metric or KPI we can define visual control (graph, chart or table) and put it on a web-based dashboard (accessible through standard Web browser).
- Repairing and reprocessing failed messages – when we discover a failed message we can manually correct it and once again put it into message queue to re-process.
- Probe technology – for sending process data to BizTracker we can use predefined probes or manually define our own ones.
- Alerts and notifications – BizTracker can be also used to generate alerts and notifications for mobile phones, wireless PDA's, pagers, email and so on.

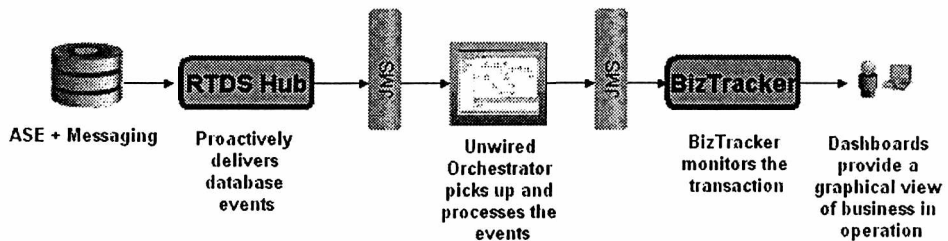


Fig. 10. BizTracker's architecture

We should remember that to be able to fully utilize Orchestrator's monitoring features we need to fulfill a few technical requirements, which arise from BizTracker's architecture. For example, as shown in Figure 10, Business Process Monitoring requires using database management system, which supports messaging and database events technology. We won't have this problem if we use Sybase's native DBMS – Adaptive Server Enterprise, which can get the most out of BizTracker. However decision to use some other DBMS platform may reduce available features.

Let's incorporate Business Activity Monitoring to our example, just to show a possible application and result. We could start by defining a performance indicator to monitor how many messages (in our example "orders") per second can be processed

by “Take Order” process. For our newly created indicator we must also define a chart and put it on a dashboard. More interesting example is defining a new variable within our process, which could indicate message status:

- a message, which is being processed at the moment should have “processing” status,
- a message, which resulted in accepting order should have “accepted” status, otherwise (resulted in rejecting order) should have “rejected” status.

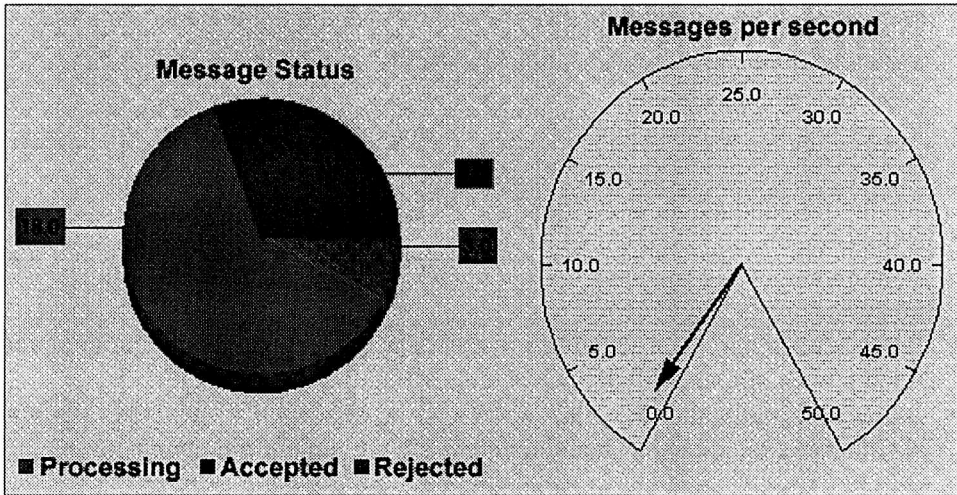


Fig. 11. Possible dashboard view generated by BizTracker

Now we must define a chart presenting the number of messages in each state and same as previously add it to our dashboard. To perform execution we can define set of XML documents representing “orders”, put them into message queue and start the process execution. Figure 11 presents a possible view of our dashboard after couple seconds of execution.

8. SUMMARY

This short article describes practical framework for modeling and deploying business processes. The main goal in creating this framework was to separate business modeling and management from application logic. We need to emphasize that although we didn’t mention a single word about writing code, we were able to build enterprise business architecture and deploy it into executable platform. That’s why we must clearly admit that using Sybase technology allows us to think of an IT system as three separable elements: data management logic, business logic and application logic. Having in memory revolution triggered by separating data management aspect, we

should expect major explosion of new approaches, which put business requirements in the very foreground. We should also expect increase of interest with business modeling tools – in 2005 Sybase carried out a survey, which showed that 37% of developers declared using business modeling tools and another 18% declared investing in such tools within a year. At the time being it is estimated that at the moment business modeling tools are vastly used in 56% of IT developing companies, and by 2009 this number is expected to be just over 70%.

Such rapid development in BPM wouldn't be possible without the appearance of new technologies such as Web services, process description languages, business execution engines, etc. Once all the necessary technologies became grown enough, many different companies tried to utilize them for moving BPM to a higher level. Solution proposed by Sybase seems to push BPM further than ever, moreover it could clear the way for other incoming solutions, which may arise from Sybase's Business Driven Approach.

Right now business modeling is developing faster than ever before, which makes it almost impossible to predict the direction it will follow. The greatest challenge seems to concern an increase in the degree of automation in business process modeling through incorporation of Semantic Web services and AI technologies - we should expect appearance of the first commercial solutions based on Semantic Web services in the immediate future.

BIBLIOGRAPHY

- [SYB] Sybase official portal, <http://www.sybase.com>
- [PDUG] Sybase PowerDesigner User's Guide, v.12.0
- [SWUG] Sybase WorkSpace User's Guide, v.1.0, Business Process Integration chapter.
- [SD 2005] *SOA: Delivering on the promise of Business Driven IT*, www.sybase.com/detail?id=1040323
- [GO 2005] McGoveran D., *Unwired Orchestration*. Alternative Technologies, 2005, www.sybase.com/content/1036885/UnwiredOrchestration_WP.pdf
- [IM 2005] Ivanyuckovich A., Marchese M. *Service Oriented Architectures for Business Process Management Systems*. In IADIS International Conference on Applied Computing, Algarve, Portugal, February 22–25, 2005.
- [GO 2004] Gold N., et al., 2004. Understanding Service-Oriented Software. In IEEE Software, No. 3.

MODELOWANIE I WYTWARZANIE SYSTEMÓW O ARCHITEKTURZE ZORIENTOWANEJ NA USŁUGI

Korporacja Sybase ma od pewnego czasu znaczący udział w rozwoju metodyk i narzędzi do modelowania procesów biznesowych w systemach IT. Zbudowano zintegrowaną infrastrukturę do modelowania procesów biznesowych w skali przedsiębiorstwa lub organizacji. Dzięki wbudowaniu w tę infrastrukturę

architektury zorientowanej na usługi SOA, Web serwisów oraz języków przetwarzania procesów biznesowych, Sybase znacząco rozszerzył modelowanie biznesowe w prawie wszystkich kierunkach i aspektach. Celem niniejszej pracy jest dokonanie zwięzłego przedstawienia podejścia firmy Sybase do zagadnień modelowania biznesowego oraz ocena użyteczności podejścia i takich kluczowych aspektów jak: definiowanie statycznego i dynamicznego spojrzenia na organizację biznesu, implementację procesów poprzez specyfikowanie usług zewnętrznych i rozmieszczania procesów na docelowych platformach i systemach przetwarzających, co w wyniku umożliwia orkiestrację i symulowanie procesu przetwarzania. Przedstawiono studium przypadku do objaśnienia aspektów, na które położono nacisk w natywnych rozwiązaniach proponowanych przez firmę Sybase.

*integracja systemów informatycznych,
Service Oriented Architecture,
Web Oriented Architecture,
Web Oriented Integration Architecture*

Daniel SZEPIELAK*

SEMANTYCZNA INTEGRACJA SYSTEMÓW INFORMATYCZNYCH PRZY UŻYCIU WEB ORIENTED INTEGRATION ARCHITECTURE

Problematyka integracji systemów informatycznych zyskała na znaczeniu na początku lat 90-tych. Od tego czasu pojawiło się wiele podejść i technologii integracyjnych. Ostatnie lata przyniosły dominację rozwiązań bazowanych na *Service Oriented Architecture* (SOA) i *SOAP Web Services*. Niedawny rozwój koncepcji Web 2.0 spowodował, że coraz więcej uwagi poświęca się odmianie SOA zwanej *Web Oriented Architecture* (WOA). WOA upraszcza tradycyjny sposób interakcji pomiędzy rozproszonymi aplikacjami w sieci poprzez wprowadzenie jednolitego interfejsu funkcjonalnego, który ułatwia proces integracji. W niniejszym artykule opisano bazowaną na WOA nowatorską architekturę integracji systemów informatycznych zwaną *Web Oriented Integration Architecture*, która w znaczący sposób rozszerza możliwości WOA jednocześnie zachowując jej najważniejszą cechę – prostotę użycia przez konsumentów usług sieciowych.

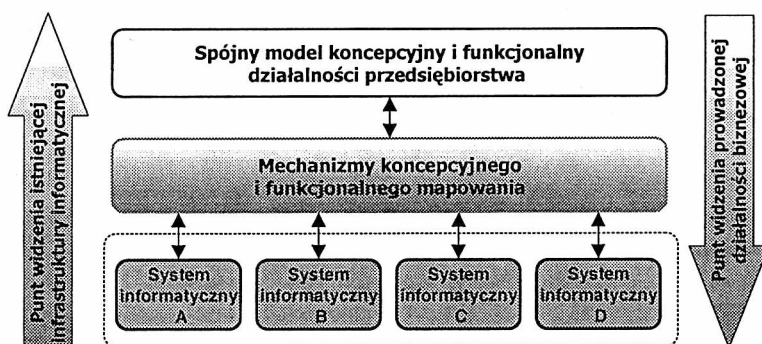
1. WPROWADZENIE

W rezultacie przyspieszonego rozwoju technologii informatycznych w ostatniej dekadzie ubiegłego wieku, znacząca liczba wyspecjalizowanych systemów informatycznych znalazła się w użyciu przedsiębiorstw. Przeciętne przedsiębiorstwo notowane obecnie na liście Fortune 500 posiada 50 lub więcej różnych aplikacji [SmaSad 2002]. Wynikające z tego faktu rozmiary oraz stopień skomplikowania istniejących środowisk oprogramowania powodują, że zwiększony nacisk jest kładziony na ułatwienie dostępu do informacji rozproszonej pomiędzy indywidualnymi aplikacjami, jak również komunikację i koordynację ich działań. Integracja systemów informatycznych przedsiębiorstw jest odpowiedzią na wyżej wymienione wyzwania. Budowa zintegrowanego środowiska oprogramowania w przedsiębiorstwie nie jest zadaniem łatwym i wymaga rozwiązania problemów wynikających z heterogenicznych platform

* Instytut Informatyki, Wydział Automatyki, Elektroniki i Informatyki Politechniki Śląskiej, 44-100 Gliwice, Akademicka 16, dszepielak@o2.pl

sprzętowych i systemowych, różnorodności języków programowania, interfejsów komunikacyjnych czy modeli danych indywidualnych aplikacji. Dodatkowym czynnikiem utrudniającym integrację jest fakt, że systemy informatyczne w przedsiębiorstwach stale ewoluują – są unowocześniane, rozbudowywane bądź zastępowane innymi systemami, w miarę zmieniających się wymagań biznesowych oraz standardów technologicznych. Z powodu tych nieustannych zmian, korporacyjne środowiska oprogramowania wymagają elastycznej i łatwo skalowalnej platformy integracyjnej będącej w stanie sprostać wspomnianym problemom.

Tradycyjne podejścia integracyjne wykorzystujące manualnie definiowane konfiguracje połączeń integracyjnych, nie są w stanie sprostać wyzwaniom stawianym współczesnym rozwiązaniom integracyjnym. Osiągnięcie tego celu wymaga wprowadzenia mechanizmów, które pozwalałyby systemom informatycznym na samodzielne dostosowywanie topologii połączeń i adaptację interfejsów integracyjnych, w celu wyeliminowania bądź minimalizacji wymaganego udziału ludzkiego w procesie integracji. Opisany cel może zostać osiągnięty poprzez proces semantycznej integracji [Noy 2004, UshGru 2004] systemów informatycznych, przy założeniu, że wszystkie uczestniczące w środowisku przedsiębiorstwa systemy informatyczne współdzielą semantyczną interpretację pojęć biznesowych.



Rys. 1. Konfrontacja punktów widzenia na infrastrukturę informatyczną i model prowadzonej działalności biznesowej

Biorąc pod uwagę wspomniane kwestie, można postawić pytanie, dlaczego nie uniknąć problemu integracji poprzez zastosowanie jednej spójnej aplikacji pokrywającej całą działalność przedsiębiorstwa. O ile takie rozwiązanie jest do pewnego stopnia możliwe w określonych przypadkach, różnorodność i odmienność działalności prowadzonej przez przedsiębiorstwa powoduje, że żaden pojedynczy produkt informatyczny nie jest w stanie spełnić oczekiwań każdego z przedsiębiorstw czy nawet ich znaczącej większości [HohWoo 2003, Ruh 2000]. Ponieważ przedsiębiorstwa zdecydowanie preferują kupowanie gotowego oprogramowania niż tworzenie własnego, korporacyjne środowiska oprogramowania są najczęściej komponowane z wielu odrębnych komercyjnych aplikacji wyspecjalizowanych w określonych zadaniach. Powoduje to, że

kompletny i spójny koncepcyjny model, według którego prowadzona jest działalność biznesowa, dzielony jest na fragmenty realizowane przy użyciu odrębnych aplikacji tworzących rzeczywistą infrastrukturę informatyczną przedsiębiorstwa (rysunek 1).

W przedstawionym kontekście integracja oznacza łączenie odrębnych fragmentów w spójną całość poprzez dostarczenie mechanizmów wykonujących koncepcyjne i funkcjonalne mapowanie pomiędzy pokazanymi punktami widzenia.

Nieustanny rozwój podejść i technologii integracyjnych, trwający od lat 90. ubiegłego wieku, przyniósł w ostatnich latach zdecydowaną dominację rozwiązań bazowanych na *Service Oriented Architecture* (SOA) i SOAP Web Services. W wyniku pojawienia koncepcji Web 2.0 i Enterprise 2.0 coraz więcej uwagi poświęca się uproszczonej odmianie SOA zwanej Web Oriented Architecture (WOA), której główną ideą jest umożliwienie budowy systemów o zaletach i charakterystyce sieci World Wide Web. WOA zostało oparte na alternatywnym dla SOAP stylu budowania usług sieciowych (ang. *web services*) zwanym *Representational State Transfer* (REST). Pomimo rosnącej popularności, architektura WOA jest wciąż w początkowej fazie swojego rozwoju i jej wykorzystanie ogranicza się w chwili obecnej do stosunkowo prostych zastosowań.

Celem omawianych badań było opracowanie metodologii i architektury integracji, spełniającej postulaty Web 2.0 i Enterprise 2.0 oraz pozwalającej na semantyczną integrację systemów informatycznych w przedsiębiorstwie. Spełnienie tych specyficznych wymagań było niezbędne przy budowie rozwiązania integracyjnego w Deutsches Elektronen-Synchrotron (DESY) – jednego z największych światowych ośrodków badań nad budową materii oraz fizyką wysokich energii – gdzie przedstawione podejście integracyjne zostało zastosowane i zweryfikowane. Jako międzynarodowy ośrodek naukowy, DESY corocznie gości tysiące naukowców z całego świata, zorganizowanych w grupy badawcze bardzo często tworzące i wykorzystujące samodzielnie budowane proste aplikacje. Tradycyjne podejścia integracyjne wymagają jednakże zbyt dużego nakładu pracy i są zbyt skomplikowane dla tych użytkowników, aby byli oni w stanie z nich skorzystać w celu dołączenia własnych aplikacji do zintegrowanego środowiska oprogramowania. Fakt ten stał się główną motywacją badań nad podejściem integracyjnym pozwalającym budować i korzystać z usług sieciowych nie tylko działom IT, lecz również zwykłym użytkownikom. W niniejszym artykule omawiana jest przydatność WOA i usług sieciowych REST do celów integracji w przedsiębiorstwie o rozbudowanym środowisku oprogramowania oraz stawiane są następujące tezy:

Teza 1 – Proponowana metoda integracji pozwala na odkrywanie usług sieciowych i integrację systemów informatycznych bez konieczności udziału czynnika ludzkiego pod warunkiem, że zaangażowane systemy informatyczne współdzielą semantyczną interpretację pojęć biznesowych; metoda pozwala na przesunięcie większości wysiłków integracyjnych, tradycyjnie skupionych na tworzeniu oprogramowania, w kierunku budowy formalnych modeli biznesowych, co pozwala na redukcję ilości kodu niezbędnego do stworzenia i utrzymania;

Teza 2 – Proponowana nowatorska koncepcja Web Oriented Integration Architecture (WOIA), poprzez zastosowanie mechanizmów wirtualizacji zwiększa poziom abstrakcji przy dostępie do zasobów w stosunku do istniejącej koncepcji Web Oriented Architecture, przez co pozwala konsumentom na dostęp do zasobów wszystkich systemów informatycznych w sposób jednolity i niezależny od reprezentacji, lokalizacji oraz interfejsów rzeczywistych zasobów; Web Oriented Integration Architecture rozszerza Web Oriented Architecture w opisany sposób bez wprowadzania dodatkowej złożoności i obowiązków dla konsumentów usług sieciowych;

Teza 3 – Zaprojektowany dla WOIA middleware pozwala na rejestrację, wykonywanie i komponowanie zbioru rozproszonych usług sieciowych, pozwalających na dostęp do rzeczywistych zasobów dostarczanych przez niezależne systemy informatyczne, w tzw. zasoby znaczące dla prowadzonej działalności biznesowej (ang. *business meaningful resources*) na podstawie formalnie zdefiniowanej ontologii oraz danych rejestracyjnych stworzonych przez dostawców usług sieciowych;

Teza 4 – Zaprojektowany framework do budowy usług sieciowych REST dla bazowanej na WOIA integracji systemów informatycznych pozwala na redukcję czasu budowy, kosztów i wysiłków niezbędnych do utrzymania zbudowanego kodu integracyjnego oraz na osiągnięcie wysokiego poziomu wielokrotnego użycia tego samego kodu (ang. *code reusability*) poprzez wprowadzenie wielopoziomowej struktury, definiującej funkcjonalne bloki kodu współdzielone pomiędzy: kompletnym zbiorem usług, podzbiorem usług realizującym taką samą operację na różnych zasobach i podzbiorem usług dostarczanych przez konkretny system informatyczny.

2. PODSTAWY INTEGRACJI

W kolejnych podrozdziałach zostaną omówione rodzaje i modele integracji systemów informatycznych.

2.1. RODZAJE INTEGRACJI

W dziedzinie integracji systemów informatycznych można wyróżnić dwa podstawowe rodzaje integracji: integrację aplikacji oraz integrację informacji, które zostaną omówione w kolejnych podrozdziałach.

Integracja aplikacji [Ruh 2000, SerCra 2002] jest procesem łączenia odrębnych systemów informatycznych w grupę aplikacji współpracujących ze sobą w celu wymiany danych i automatyzacji procesów biznesowych, wymagających uprzednio ludzkiego współudziału. Współpracujące aplikacje mogą koordynować swoje działania poprzez wzajemny dostęp do udostępnianych przez siebie funkcji.

W zależności od miejsca zastosowania, integracja aplikacji jest dzielona na Enterprise Application Integration (EAI) [Linthicum 1999] oraz Business-to-Business inte-

gration (B2B) [Bussler 2003]. EAI dotyczy integrowania aplikacji w obrębie przedsiębiorstwa bądź organizacji w celu rozwiązywania wewnętrznych problemów i zadań, podczas gdy B2B dotyczy integrowania aplikacji odrębnych organizacji celem współdzielenia informacji bądź współpracy nad wspólnymi projektami.

Odrębny rodzaj integracji stanowi integracja informacji, powszechnie określana jako Enterprise Information Integration (EII) [Morgenthal 2005, Halevy 2005]. Wiąże się ona z dostarczaniem mechanizmów pozwalających na wyszukiwanie i gromadzenie powiązanych ze sobą fragmentów informacji pochodzących z wielu różnych aplikacji i ich prezentację przy użyciu pojedynczego interfejsu użytkownika. Pozwala to na dostęp do rozproszonej informacji z tzw. pojedynczego punktu dostępu (ang. *single point of access*) bez konieczności jednoczesnego korzystania z wielu aplikacji i ręcznego łączenia powiązanych fragmentów informacji w celu uzyskania kompletnego i spójnego widoku na dane.

Opisane rodzaje integracji stosowane są w różnych celach i nic nie stoi na przeszkodzie, aby używać ich jednocześnie. W istocie jest to powszechna praktyka stosowana obecnie w wielu przedsiębiorstwach.

2.2. MODELE INTEGRACJI

Jedną z najistotniejszych cech mających dominujący wpływ na jakość rozwiązania integracyjnego jest skalowalność, która zależy przede wszystkim od przyjętego modelu integracji. Z punktu widzenia logiki połączeń pomiędzy aplikacjami w zintegrowanym środowisku oprogramowania można wyróżnić dwa modele integracji [Ruh 2000, SerCra 2002]:

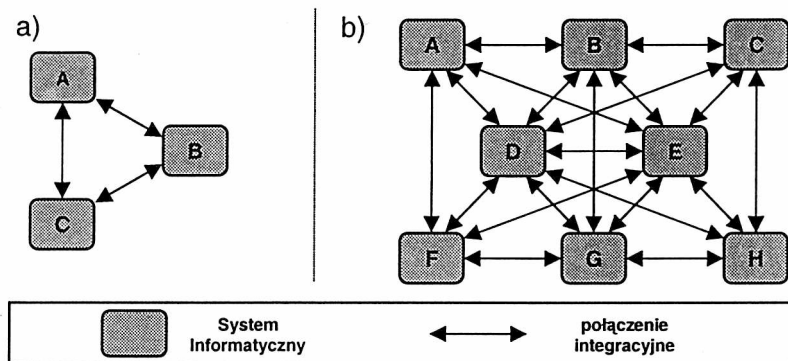
- model typu *point-to-point*,
- model typu *middleware*.

W modelu typu *point-to-point* aplikacje komunikują się ze sobą bezpośrednio przy użyciu dedykowanych połączeń integracyjnych. Każda aplikacja ma odrębne połączenie z pozostałymi aplikacjami, z którymi musi się komunikować. Dodanie nowej aplikacji do środowiska może wymagać stworzenia liczby połączeń integracyjnych równej ilości aplikacji funkcjonujących już w zintegrowanym środowisku. Wraz ze wzrostem liczby aplikacji liczba połączeń drastycznie wzrasta, powodując, że ich skomplikowana sieć staje się niemożliwa do utrzymania (rys. 2).

Dla przykładu, przy trzech aplikacjach potrzeba tylko trzech połączeń integracyjnych (rys. 2a), ale już przy ośmiu aplikacjach, maksymalna liczba połączeń wzrasta do 28 (rys. 2b), zgodnie ze wzorem:

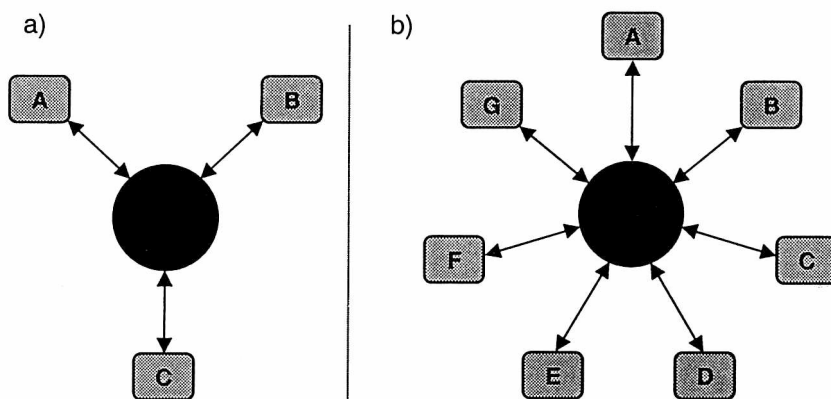
$$C_{\max} = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$$

gdzie $C_{\max}$ jest maksymalną liczbą połączeń integracyjnych, a n jest liczbą aplikacji w środowisku.



Rys. 2. Wzrastająca liczba połączeń integracyjnych w modelu typu point-to-point

W modelu typu *middleware* aplikacje zamiast komunikować się ze sobą bezpośrednio robią to poprzez warstwę pośrednią, która definiuje interfejsy komunikacyjne ogólnego przeznaczenia. Dodanie nowej aplikacji do środowiska wymaga dodania tylko jednego połączenia integracyjnego łączącego middleware z nowo dodaną aplikacją. Rysunek 3 prezentuje wzrost liczby połączeń wraz ze wzrostem liczby aplikacji w środowisku w przypadku integracji typu middleware.



Rys. 3. Wzrastająca liczba połączeń integracyjnych w modelu typu middleware

Maksymalna liczba połączeń w tym przypadku ma charakter liniowy może być wyznaczona przy pomocy wzoru:

$$C_{\max} = n$$

Integracja bazowana na modelu middleware ma niestety również pewne wady. Po pierwsze budowa infrastruktury middleware jest czasochłonna i wymagająca technicznie, a w przypadku niewielkich środowisk koszt jej implementacji bywa

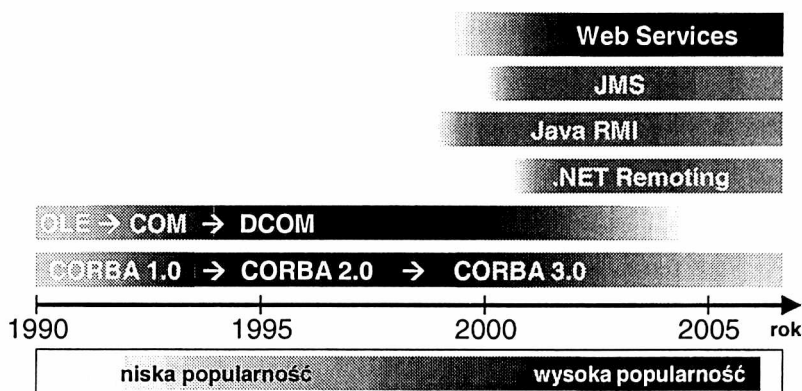
często wyższy niż w przypadku integracji typu point-to-point. Dodatkowo połączenia integracyjne ogólnego przeznaczenia są zazwyczaj mniej wydajne niż dedykowane połączenia w integracji typu point-to-point. Jednakże nawet biorąc pod uwagę wspomniane niedostatki, integracja bazowana na modelu middleware jest zdecydowanie lepszym wyborem. Pozwala ona uniknąć problemów związanych ze skalowalnością środowiska, które są zasadniczą wadą modelu point-to-point, przez co w dłuższej perspektywie większy wstępny koszt inwestycji w przypadku integracji middleware powinien zaowocować w przyszłości niższymi kosztami utrzymania i rozwoju zintegrowanego środowiska oprogramowania w przedsiębiorstwie.

3. TECHNIKI INTEGRACJI SYSTEMÓW INFORMATYCZNYCH

Zanim zostaną omówione podejścia Service Oriented Architecture (SOA) i Web Oriented Architecture (WOA) do integracji systemów, zostanie przedstawiony krótki rys historyczny technologii integracyjnych.

3.1. RYS HISTORYCZNY

Pierwsze poważne próby opracowania technologii integracyjnych miały miejsce na początku lat 90-tych. W 1991 Object Management Group (OMG) opracowała Common Object Request Broker Architecture (CORBA). Jej wczesna wersja posiadała pewne problemy z kompatybilnością sprzętową jak i programową i dopiero wraz z pojawieniem się CORBA 2.0 w roku 1996 nastąpił prawdziwy rozkwit tej technologii. W tym samym czasie własne technologie integracji opracowywała firma Microsoft. W roku 1990 pojawił się Object Linking and Embedding (OLE), którego idea została później rozwinięta wraz z pojawieniem się Component Object Model (COM) w roku 1993 i jego następcy Distributed Component Object Model (DCOM) w roku 1995. W ciągu kolejnych kilku lat DCOM stanowił jedyną istotną konkurencję dla technologii CORBA. Początek nowego wieku zaowocował pojawieniem się kilku nowych technologii z których do ważniejszych można zaliczyć Java Remote Method Invocation (RMI), Java Message Service (JMS) oraz .NET Remoting będący następcą DCOM (rysunek 4). Jednakże dopiero powstanie specyfikacji Simple Object Access Protocol (SOAP) w roku 1999, która wkrótce stała się podstawą technologii Web Services spowodowało wyraźny spadek popularności technologii CORBA i trwającą do chwili obecnej dominację technologii Web Services.



Rys. 4. Ważniejsze technologie integracyjne ostatnich dwóch dekad

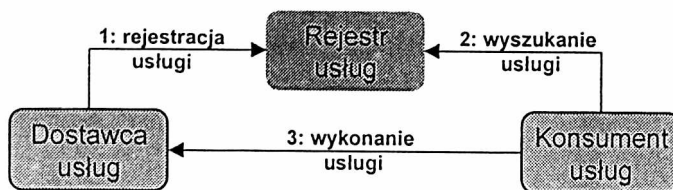
3.2. SERVICE ORIENTED ARCHITECTURE

Dominującym obecnie podejściem do problemu integracji jest wykorzystanie Service Oriented Architecture (SOA) i technologii SOAP Web Services [Curbera 2005, Mayerl 2005]. Główna idea SOA zakłada, że logika biznesowa nie jest realizowana za pomocą monolitycznego programu, lecz przy użyciu rozproszonych usług sieciowych, które z założenia mogą być wykorzystywane przez wielu klientów i komponowane zależnie od ich potrzeb.

Usługa sieciowa definiowana jest przez W3C jako niezależny od platformy i implementacji komponent programowy identyfikowany za pomocą deskryptora URI, którego interfejs może być w formalny sposób zdefiniowany, opisany i wykryty w formie artefaktów XML. Usługa sieciowa umożliwia bezpośrednią interakcję z innymi komponentami programowymi poprzez wymianę wiadomości XML przy wykorzystaniu internetowych protokołów komunikacyjnych [W3C].

SOA definiuje następujące role dla komponentów współdziałających w sieci (rysunek 5):

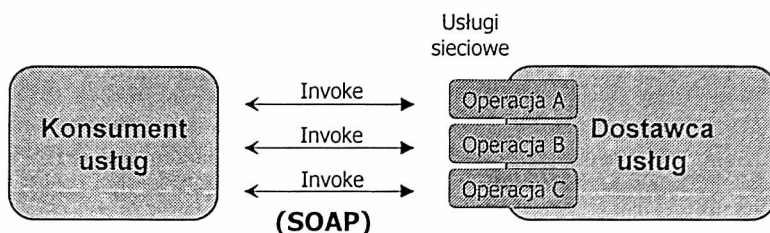
- dostawca usług (ang. *service provider*) jest systemem, który udostępnia swoje dane i funkcjonalność w formie usług sieciowych; jest on odpowiedzialny za opublikowanie interfejsu oferowanych przez siebie usług oraz ich implementację,
- rejestr usług (ang. *service registry*) jest pośrednikiem pomiędzy dostawcami usług i konsumentami usług; pozwala on dostawcom usług na rejestrowanie informacji o oferowanych przez nich usługach, a konsumentom na szukanie i odczytywanie informacji o usługach, których potrzebują,
- konsument usług (ang. *service consumer*) jest systemem lub osobą będącą użytkownikiem usług; może on wyszukiwać potrzebne usługi za pomocą rejestru usług i wysyłać żądania zdalnego wykonania usług do ich dostawców.



Rys. 5. Role komponentów w Service Oriented Architecture

Relacje pomiędzy konsumentami i dostawcami mają charakter wiele-do-wielu co oznacza, że pojedyncza aplikacja może zarówno dostarczać usługi wielu innym aplikacjom jak i być konsumentem usług wielu innych aplikacji.

Na poziomie koncepcyjnym SOA nie jest powiązane z żadną konkretną technologią, jednakże w ciągu ostatnich lat pojęcie to stało się praktycznie równoznaczne z wykorzystaniem technologii SOAP Web Services [Erl 2005]. SOAP jest specyfikacją, która określa sposób kodowania wiadomości XML-owych w procesie komunikacji pomiędzy aplikacjami przy użyciu protokołów internetowych HTTP i SMTP. SOAP używany jest jako mechanizm żądania-odpowiedzi bazowany na modelu klient-serwer i pozwala na zdalne wykonywanie operacji oferowanych przez dostawcę usług (rysunek 6).



Rys. 6. Interakcja między komponentami w SOA przy użyciu usług sieciowych SOAP

Podstawowymi technologiami umożliwiającymi opis i odkrywanie usług sieciowych są Web Services Description Language (WSDL) i Universal Description Discovery and Integration (UDDI). WSDL to język bazowany na XML służący do formalnego opisu sposobu interakcji z usługami sieciowymi. WSDL definiuje interfejs usług sieciowych jako kolekcję punktów docelowych (ang. *network endpoints*) i powiązanych z nimi struktury danych, oraz określa metody kodowania parametrów wywołania i parametrów zwrótnych usługi. Opisy WSDL są składowane w rejestrach UDDI w trakcie rejestracji usługi. Rejestry UDDI wykorzystują trzy poziomy opisy usług:

- *white pages* zawiera podstawowe dane kontaktowe o dostawcy usługi (adres, telefon, identyfikator itp.);

- *yellow pages* zawiera klasyfikację usług według przemysłowych taksonomii i kategoryzacji (np. ISO3166 – klasyfikacja geograficznego położenia dostawcy usługi);
- *green pages* zawiera techniczne informacje o usłudze i sposobach jej wywołania (zazwyczaj zawiera opis WSDL).

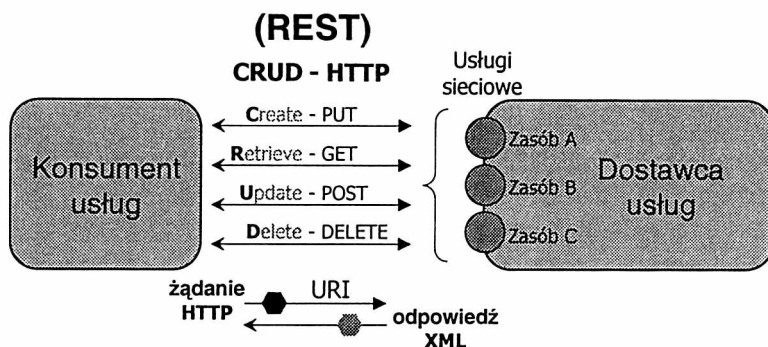
Mimo swej ogromnej popularności, SOA nie udało się jak dotąd osiągnąć jednego ze swoich głównych założeń, którym jest w pełni zautomatyzowane odkrywanie, komponowanie i używanie publicznie dostępnych usług. Większość istniejących podejść do problemu integracji, w tym również tych bazujących na SOA, posiada jedną wspólną charakterystykę: eksponowanie danych i funkcjonalności systemów informatycznych w formie zdalnych funkcji (w SOA reprezentowanych przez usługi sieciowe) specyficznych dla każdego z eksponowanych obiektów. Ponieważ każdy z dostawców może oferować usługi wykonujące dowolne operacje na dowolnych obiektach powoduje to, że formalny opis konkretnej usługi jest zazwyczaj nadmiernie skomplikowany i często niejednoznaczny.

Opisane podejście jest określane powszechnie jako zorientowane na operacje (ang. *operation-centric*) a jego pierwotnym źródłem jest tradycyjny model programowania obiektowego. O ile jednak zasady programowania obiektowego sprawdzają się, gdy obiekty i ich metody są używane ręcznie przez programistę, to w przypadku współczesnych rozwiązań integracyjnych, w których zdalne funkcje (usługi sieciowe) powinny być odszukiwane i używane przez systemy informatyczne w sposób zautomatyzowany, podejście to powoduje spore problemy. Przy pełnej swobodzie interfejsu funkcjonalnego, związanego z dowolnością operacji, poprawna interpretacja opisu funkcjonalności usług dostawców przez aplikacje konsumenckie jest zadaniem wysoce skomplikowanym i jak dotąd nierozwiązanym. Dodatkowo, ponieważ przedsiębiorstwa kupują gotowe aplikacje, często pochodzące od różnorodnych producentów, stosowane przez nie aplikacje dostawców usług i konsumentów usług zbudowane są zazwyczaj zupełnie niezależnie od siebie. Powoduje to, że ich interfejsy rzadko kiedy są kompatybilne, szczególnie, że nawet usługi oferujące tę samą funkcjonalność, ale pochodzące od różnych dostawców, mogą mieć zupełnie różne interfejsy (struktury danych, parametry wywołania i parametry zwrotne). Skutkiem tego, w nie trywialnych rzeczywistych zastosowaniach aplikacje konsumenckie nie są zazwyczaj w stanie poprawnie wykryć oraz przystosować się do interfejsu dostawców usług bez ludzkiej ingerencji [Grigori 2006, Shirky 2002].

3.3. WEB ORIENTED ARCHITECTURE

Problemy związane z tradycyjnymi rozwiązaniami SOA spowodowały, że zaczęto szukać alternatywnych rozwiązań technologicznych. Wraz z niedawnym pojawieniem się koncepcji Web 2.0 [MusORe 2006], która w sposób zdecydowany

promuje używanie prostych i przystępnych rozwiązań technologicznych, powstało pojęcie Web Oriented Architecture (WOA). Główne postulaty Web 2.0 dotyczą między innymi budowy oprogramowania w formie usług sieciowych, zaufania użytkownikom oraz zachęcania ich do aktywnej współpracy, dzielenia się danymi i tworzenia własnego oprogramowania korzystającego z tychże usług, co powinno zostać osiągnięte poprzez stosowanie możliwie najprostszyc modeli budowy oprogramowania (ang. *lightweight programming models*), wymagających od użytkowników minimalnej wiedzy i doświadczenia. Zastosowanie powyższych postulatów w środowisku korporacyjnym zostało niedawno nazwane przez profesora Harvard Business School Andrew McAfee jako „Enterprise 2.0” [McAfee 2006]. Propagatorami wspomnianych idei są między innymi tak poważane organizacje jak O'ReillyMedia czy Gartner Research.



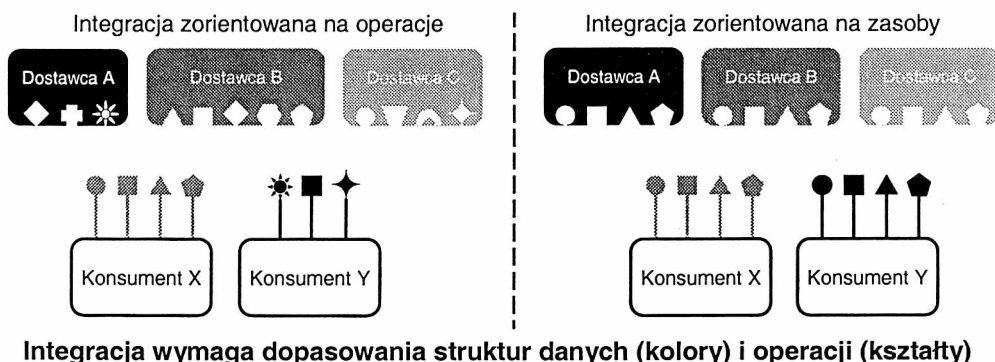
Rys. 7. Interakcja między komponentami w WOA przy użyciu usług sieciowych REST

WOA jest uproszczoną odmianą SOA, mającą za zadanie spełnić postulaty stawiane przez koncepcje Web 2.0 i Enterprise 2.0. WOA bazuje na stylu budowania usług sieciowych zwanym Representational State Transfer (REST) [FieTay 2000], wykorzystującym jedynie podstawowe standardy HTTP, URI i XML. REST jest zastosowaniem podstawowych charakterystyk World Wide Web (WWW) do budowy usług sieciowych. REST reprezentuje podejście integracyjne określane jako zorientowane na zasoby (ang. *resource centric*) i bazuje na pojęciach zasobu i identyfikatora zasobu (URI). Każdy zasób może reprezentować dowolny zbiór danych i funkcji aplikacji, który jest w unikalny sposób adresowalny przy pomocy identyfikatora zasobu. Standardowo zasoby są reprezentowane w formie dokumentów XML, jednakże dopuszczalne są również inne formaty reprezentacji. W metodologii REST aplikacje postrzegane są jako maszyny stanów, gdzie pobranie reprezentacji zasobu powoduje zmianę stanu aplikacji konsumenta, podobnie jak przesłanie reprezentacji zasobu do dostawcy usługi powoduje zmianę stanu aplikacji dostawcy. Kluczową cechą opisywanego podejścia jest to, że zamiast zbioru operacji oferowany jest zbiór zasobów, do których dostęp oferowany jest przy użyciu jednolitego interfejsu HTTP, definiującego stały zbiór operacji możliwych

do wykonania na dowolnym zasobie. Standardowe metody protokołu HTTP są mapowane do tak zwanego zbioru operacji CRUD – Create (POST), Retrieve (GET), Update (PUT) oraz Delete (DELETE) (rysunek 7).

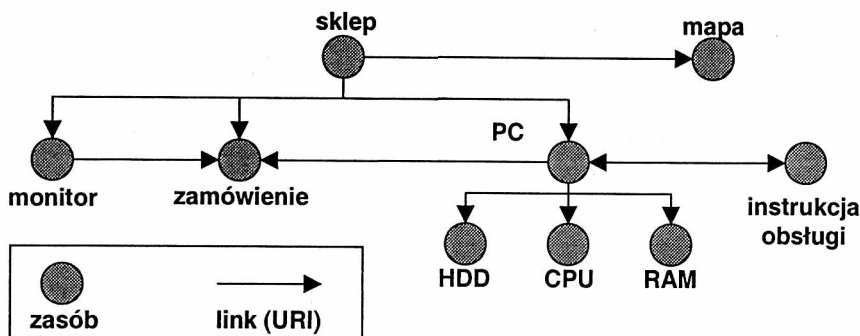
Ponieważ zbiór operacji jest stały, a ich interpretacja znana zarówno dostawcom jak i konsumentom, wyeliminowany zostaje problem poprawnej interpretacji interfejsu funkcjonalnego po stronie konsumenta. Skutkiem ubocznym jest to, że dostawca usług traci swobodę ekspresji funkcjonalnej i może oferować usługi jedynie w ściśle zdefiniowany sposób. Biorąc jednak pod uwagę fakt, że kluczowym problemem integracji jest uzyskanie przez konsumentów i dostawców wzajemnego zrozumienia odnośnie operacji wykonywanej przez usługę jak i obiektu, na którym ta usługa jest wykonywana, wprowadzenie jednolitości interfejsu kosztem swobody ekspresji, wydaje się być uzasadnionym posunięciem, szczególnie jeśli weźmie się pod uwagę fakt że dostawcy i konsumenci są od siebie niezależni i budują swoje aplikacje bez wiedzy o tym, co robi partner (rysunek 8).

Dzięki jednolitości interfejsu funkcjonalnego konsument ma znacznie większą szansę na odnalezienie szukanej usługi niż w przypadku integracji zorientowanej na operacje, wymagającej dostosowania zarówno struktury danych jak i interfejsu funkcjonalnego. W przypadku, gdy dostępne są jedynie usługi częściowo spełniające wymagania konsumenta, przystosowanie się do interfejsu dostawcy wymaga jedynie mapowania struktur danych bez konieczności uzgadniania wspólnej interpretacji operacji [Szepielak 2006].



Rys. 8. Konfrontacja podejść integracyjnych:
zorientowanego na operacje i zorientowanego na zasoby

Kolejną istotną cechą WOA jest to, że reprezentacja zasobu może zawierać linki do innych zasobów będących w relacji z zasobem pierwotnym. W ten sposób pełen zbiór zasobów tworzy sieć powiązań pozwalającą na nawigację bez konieczności znajomości modelu zasobów dostawcy, co ułatwia konsumentom odnajdywanie niezbędnych usług (rysunek 9).



Rys. 9. WOA – przykład zbioru zasobów REST i linków pozwalających konsumentom na nawigację

Prostota podejścia WOA znalazła już liczne grono zwolenników i dobrze sprawdza się w rzeczywistych zastosowaniach. Wiele firm, które początkowo oferowały jedynie usługi sieciowe w formacie SOAP, obecnie preferuje format REST. Najbardziej znanym przykładem jest firma Amazon, która oferuje usługi sieciowe w obu wspomnianych formatach, lecz 95% wszystkich żądań od konsumentów dotyczy formatu REST [MusORe 2006]. REST jest również od niedawna wspierany przez takie platformy programistyczne jak Microsoft .NET, IBM WebSphere czy BEA Systems WebLogic, które rozpoznały potrzebę wsparcia dla tego otrzymującego coraz więcej uwagi standardu.

4. WYZWANIA ZASTOSOWANIA WOA DO CELÓW INTEGRACJI

Pomimo rosnącej popularności architektura WOA jest wciąż w początkowej fazie swojego rozwoju i jej wykorzystanie ogranicza się w chwili obecnej do stosunkowo prostych zastosowań o niewielkim zasięgu, pozbawionych możliwości rejestracji i kompozycji usług sieciowych. Zastosowanie WOA do tak wymagającego zadania, jakim jest integracja systemów informatycznych w środowisku przedsiębiorstwa, stanowi poważne wyzwanie. Na podstawie przeprowadzonej analizy wymagań, zidentyfikowano następujące wyzwania związane z rozszerzeniem architektury WOA dla potrzeb semantycznej integracji:

- **konsumenci muszą mieć możliwość odkrywania URI zasobów;** w chwili obecnej nie istnieją dla WOA mechanizmy analogiczne do istniejących w SOA (np. rejestry UDDI i opisy usług WSDL); niezbędne jest stworzenie takich mechanizmów, przystosowanych do specyficznych cech architektury WOA;

- **konsumenci muszą być odizolowani od zmian URI zasobów;** w prawdziwych zastosowaniach linki URI które identyfikują zasoby często ulegają zmianom (w istocie niestabilność URI jest jednym z głównych zarzutów przeciwko WOA); niezbędne jest wprowadzenie mapowania rzeczywistych linków URI, które identyfikują zasoby dostawców, na stabilne wirtualne linki URI, z których będą korzystać konsumenci; zadanie mapowania powinno spoczywać na zewnętrznym komponencie middleware;
- **odpowiedzialność za linki łączące zbiór zasobów WOA powinna zostać przesunięta z dostawców usług na middleware;** większość dotychczasowych zastosowań WOA ogranicza się do pojedynczego dostawcy usług, który kontroluje pełen zbiór zasobów; przy zastosowaniu WOA do celów integracji pojawia się wielu dostawców, z których każdy posiada wiedzę tylko o własnych zasobach, stanowiących jedynie fragment pełnego zbioru zasobów dostępnych w środowisku oprogramowania przedsiębiorstwa; w rezultacie dostawcy nie są w stanie zidentyfikować wszystkich relacji pomiędzy zasobami i dostarczyć niezbędnych linków pomiędzy nimi; z tego powodu muszą one być zarządzane zewnątrz przez komponent middleware posiadający kompletny widok na zbiór zasobów;
- **konsumenci powinni mieć dostęp do zasobów znaczących dla prowadzonej działalności biznesowej przy użyciu pojedynczego URI;** każdy z dostawców usług w zintegrowanym środowisku oprogramowania dostarcza tak zwanych zasobów rzeczywistych, które często stanowią jedynie fragment kompletnego zasobu istotnego z punktu widzenia konsumentów; jest to sytuacja wynikająca bezpośrednio ze specjalizacji systemów informatycznych w przedsiębiorstwie, w wyniku której informacje o różnych aspektach jednego zasobu są rozproszone pomiędzy wieloma systemami informatycznymi: kompozycja zbioru powiązanych ze sobą zasobów rzeczywistych pochodzących od różnych dostawców tworzy tak zwany zasób znaczący dla prowadzonej działalności biznesowej (ang. *business meaningful resource*) (dla prostoty dalej zwany zasobem biznesowym), którym są zainteresowani konsumenci;
- **prostota używania WOA i usług sieciowych REST z punktu widzenia konsumenta musi być zachowana;** ponieważ jednym z głównych powodów wyboru architektury WOA jest jej prostota użycia, wprowadzenie do niej jakichkolwiek nowych elementów nie może zwiększać stopnia jej skomplikowania z punktu widzenia konsumenta; wszelkie dodatkowe operacje wynikające z rozszerzenia architektury powinny być wykonywane w sposób niewidoczny dla konsumenta, pozwalając mu na tak samo prosty dostęp do zasobów jak w podstawowej architekturze WOA.

5. INTEGRACJA PRZY UŻYCIU WEB ORIENTED INTEGRATION ARCHITECTURE

Odpowiedzią na wymienione w poprzednim rozdziale wyzwania jest wprowadzenie nowatorskiej architektury *Web Oriented Integration Architecture* (WOIA), która znacząco rozszerza możliwości istniejącej koncepcji WOA, umożliwiając zastosowanie jej do celów integracji systemów informatycznych, zachowując jednocześnie jej prostotę i zgodność z postulatami Web 2.0.

5.1. WEB ORIENTED INTEGRATION ARCHITECTURE

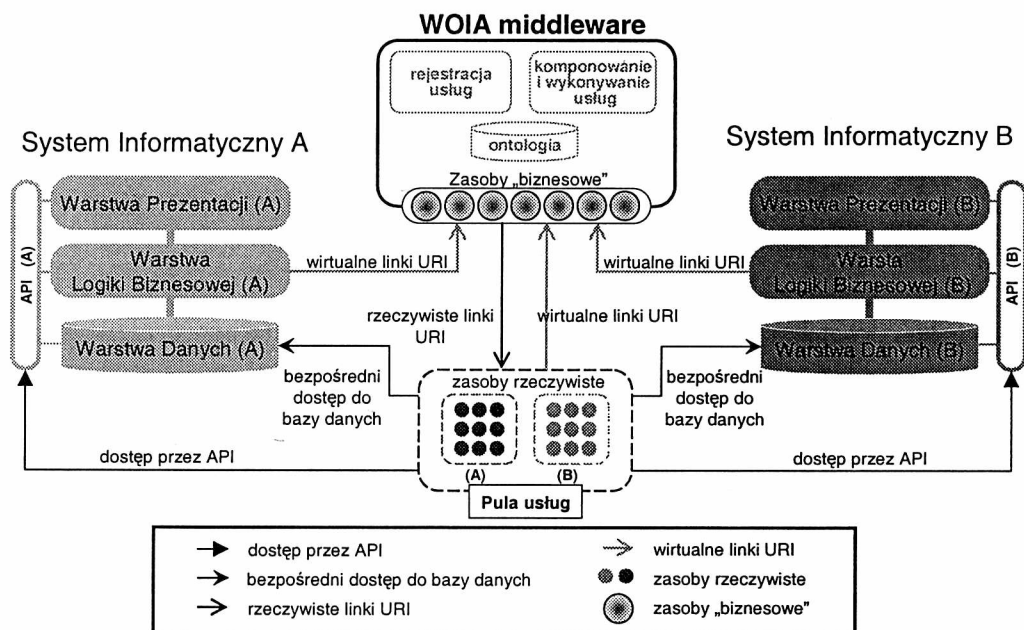
Web Oriented Integration Architecture (WOIA), odseparowuje podlegające częstym zmianom czynniki specyficzne dla konkretnych systemów informatycznych i platform, na których działają, od znacznie bardziej stabilnego ogólnego modelu koncepcyjnego, według którego powinny współdziałać systemy informatyczne w przedsiębiorstwie. Pozwala to indywidualnym systemom informatycznym działać według ich własnego specyficznego punktu widzenia, jednocześnie umożliwiając zewnętrzny dostęp do swoich zasobów zgodnie z ogólnym modelem koncepcyjnym. Rysunek 10 przedstawia architekturę WOIA przedstawioną w odniesieniu do tradycyjnej 3-warstwowej architektury systemów informatycznych.

Bez rozwiązania integracyjnego systemy informatyczne mają dostęp jedynie do własnych danych oraz funkcji logiki biznesowej. Dostęp ten zapewniony jest poprzez API specyficzne dla danego systemu informatycznego, którego użycie w większości przypadków wymaga od programisty szczegółowej wiedzy na temat budowy i zasad działania danego systemu.

Architektura WOIA wprowadza komponent *middleware*, umożliwiający dostęp do zasobów wszystkich systemów informatycznych w zintegrowanym w środowisku oprogramowania przedsiębiorstwa poprzez pojedyncze API – prosty i jednolity interfejs usług sieciowych REST. Dzięki takiemu podejściu programiści aplikacji grających rolę konsumentów mogą w łatwy sposób uzyskać dostęp do dowolnych zasobów w środowisku bez konieczności odwoływania się do niejednokrotnie nadmiernie skomplikowanych API indywidualnych systemów informatycznych.

Każdy z systemów informatycznych musi dostarczyć zbiór usług sieciowych pozwalających na wykonywanie operacji CRUD na wszystkich zasobach, które powinny być dostępne w środowisku. Usługi te pozwalają na dostęp do rzeczywistych zasobów systemów informatycznych i są identyfikowane przez rzeczywiste linki URI. Pełen zbiór usług pochodzący od wszystkich systemów informatycznych w środowisku tworzy Pulę Usług. Komponent *middleware* zawiera ogólny model prowadzonej działalności biznesowej będący zbiorem połączonych ze sobą klas zasobów, zdefiniowany formalnie w postaci ontologii, według której możliwa jest rejestracja i kompozycja

zasobów rzeczywistych do zasobów biznesowych identyfikowanych przez wirtualne linki URI. Wspomniane zasoby mogą być użyte w warstwach logiki biznesowej systemów informatycznych bądź też przez inne usługi z Puli Usług. W obydwu przypadkach żądanie wykonania usługi wysyłane jest do komponentu middleware, a nie bezpośrednio do dostawcy usługi, jak w tradycyjnym modelu interakcji SOA (choć taki dostęp jest również możliwy).



Rys. 10. Web Oriented Integration Architecture

Z technicznego punktu widzenia mechanizm dostępu do zasobów rzeczywistych (żądanie do dostawcy) i zasobów biznesowych (żądanie do middleware) jest jednakowo prosty – konsument musi jedynie wysłać żądanie HTTP pod dany URI (rzeczywisty bądź wirtualny). Żądanie dostępu do zasobu biznesowego niejednokrotnie oznacza jednak konieczność równoczesnego dostępu do wielu zasobów rzeczywistych dostarczanych przez różnych dostawców. Wynikające z tego faktu zadania dekompozycji oryginalnego żądania o zasób biznesowy do zbioru żądań o zasoby rzeczywiste, wykonania tych częściowych żądań oraz zgromadzenie i kompozycję częściowych odpowiedzi realizowane są przez komponent middleware. Korzystając z kompletnego modelu działalności biznesowej (ontologii), middleware ma dodatkowo możliwość wzbogacenia zwracanych odpowiedzi o linki do innych powiązanych zasobów biznesowych.

Opisywana architektura WOIA pozwala konsumentom na dostęp do zasobów w sposób równie prosty jak w przypadku WOA, dając jednocześnie możliwość efek-

tywnego zarządzania, rejestracji, koordynacji i komponowania usług sieciowych poprzez dedykowany middleware.

5.2. ONTOLOGIA

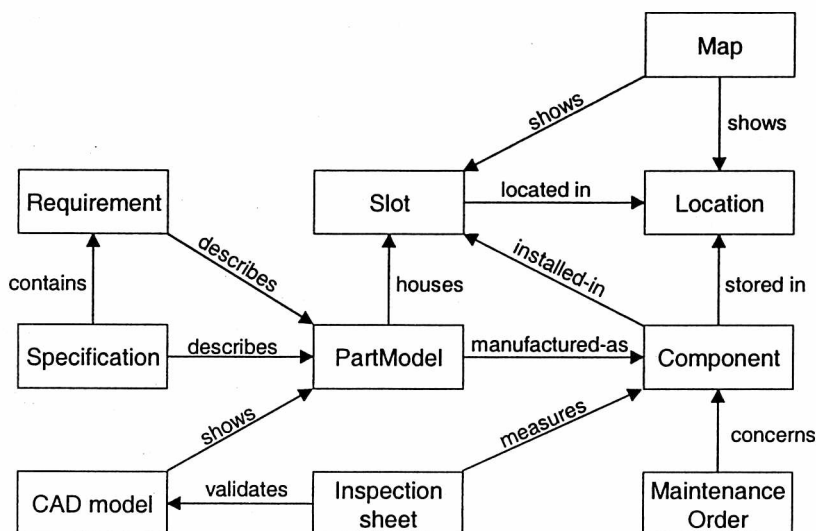
Pierwszym niezbędnym krokiem przy integracji systemów informatycznych przy użyciu opisywanego podejścia, jest stworzenie modelu prowadzonej działalności biznesowej w formie ontologii, definiującej typy zasobów występujących w środowisku oprogramowania przedsiębiorstwa. Należy podkreślić, że ontologia powinna zawierać zasoby zdefiniowane na wysokim poziomie abstrakcji, odpowiadające istotnym obiektom biznesowym, a nie niskopoziomowym klasom specyficznym dla systemów informatycznych.

Zasoby zdefiniowane w ontologii reprezentować mogą zarówno dane jak i procesy. Odzwierciedlenie procesu w formie zasobu wymaga zdefiniowania struktury danych zasobu zawierającej niezbędne parametry wejściowe i wyjściowe procesu oraz przyjęcia następującej interpretacji operacji CRUD: (*Create*) – stworzenie zasobu powoduje uruchomienie procesu, (*Retrieve*) – pobranie zasobu pozwala sprawdzić status procesu, (*Update*) – modyfikacja zasobu pozwala zmienić przebieg procesu oraz (*Delete*) – usunięcie zasobu pozwala na przerwanie wykonania procesu. Zachowanie spójnego interfejsu oznacza, że z punktu widzenia konsumentów nie ma żadnej różnicy pomiędzy operacjami na zasobach reprezentujących dane i zasobach reprezentujących procesy.

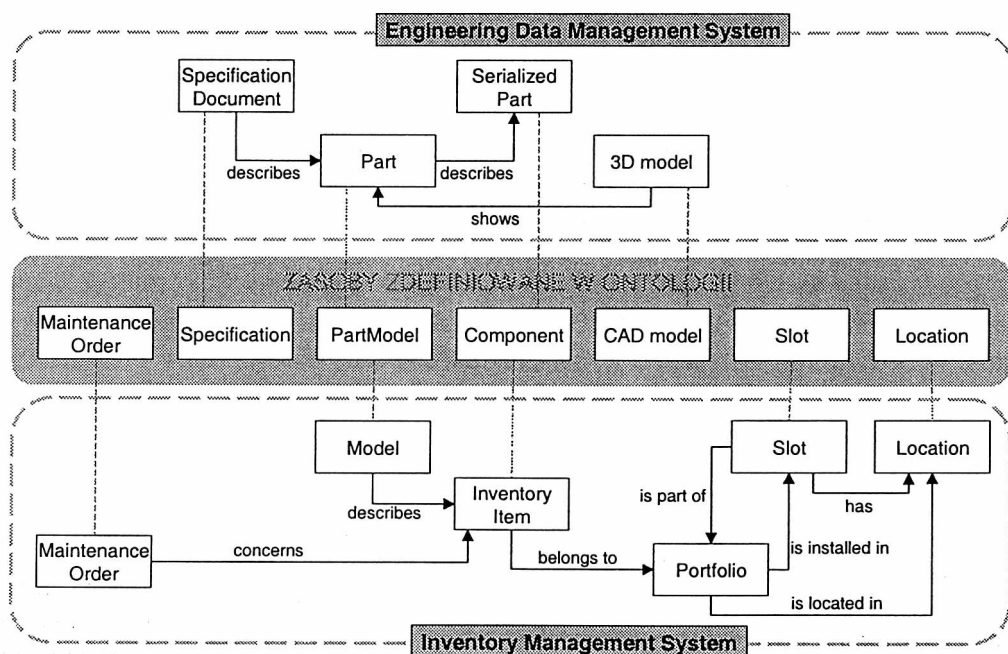
Celem lepszego zobrazowania charakteru i roli ontologii w opisywanym podejściu integracyjnym, przedstawiony zostanie uproszczony widok na fragment ontologii stworzonej na potrzeby zintegrowanego środowiska oprogramowania w Deutsches Elektronen-Synchrotron. DESY zarządza techniczną infrastrukturą zawierającą znaczącą liczbę urządzeń (głównie komponentów akceleratorów cząsteczek), których cykl życia, obejmujący fazy specyfikacji, projektowania, budowy oraz użytkowania, trwa niejednokrotnie 20–30 lat. Każde z urządzeń opisane jest przez specyfikację, zbiór rysunków technicznych, modeli CAD i innych dokumentów, zawierających szeroką gamę informacji, od wstępnych wymagań projektowych po procedury i terminarze przeglądów technicznych. Urządzenia te zainstalowane są na rozległym obszarze laboratoriów i tuneli akceleratorów, bądź przechowywane w magazynach jako części zamienne. Dane i procesy dotyczące wspomnianych obiektów wyrażone są w postaci sieci powiązanych ze sobą zasobów i zdefiniowane w ontologii (rys. 11).

Pojedynczy zasób biznesowy zdefiniowany w ontologii może być rozproszony pomiędzy kilkoma systemami informatycznymi, oferującymi jego częściowe implementacje. Na rysunku 12 przedstawiono przykład opisanej sytuacji na podstawie dwóch systemów informatycznych w DESY: Engineering Data Management System (EDMS), wspomagającego fazy projektowania i produkcji komponentów oraz Inventory Management System (IMS), obsługującego dane związane z rejestracją, użytkow-

waniem oraz utrzymaniem komponentów. W przedstawionym przykładzie zasoby typu *PartModel* oraz *Component* są częściowo realizowane przez obydwa wymienione systemy.



Rys. 11. Uproszczony widok na fragment ontologii w DESY

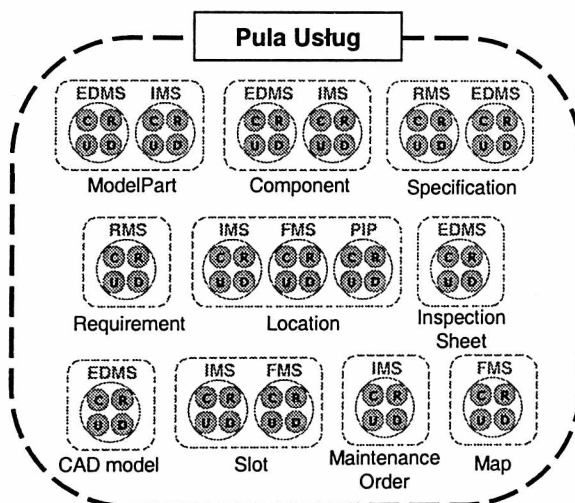


Rys. 12. Przykład: mapowanie ontologii do specyficznych modeli systemów informatycznych

5.3. PULA USŁUG SIECIOWYCH

Aby umożliwić dostęp do zasobów zdefiniowanych w ontologii, każdy z systemów informatycznych w środowisku powinien dostarczyć zbiór usług sieciowych pozwalających na wykonywanie operacji CRUD na tych zasobach. Pełen zbiór usług pochodzący od wszystkich dostawców w środowisku tworzy Pulę Usług. Pula Usług odpowiadająca przedstawionemu wcześniej fragmentowi ontologii przedstawiona jest na rysunku 13 – skróty FMS, RMS, PIP odpowiadają nazwom systemów informatycznych Facility Management System (FMS), Requirements Management System (RMS) i Personal Information Pool (PIP).

Należy zaznaczyć, że indywidualne systemy dostarczają jedynie usług pozwalających operować na zasobach rzeczywistych. Dostęp do zasobów biznesowych realizowany jest poprzez kompozycję usług dostarczanych przez indywidualne systemy. Przykładowo, zasób biznesowy typu *Component* realizowany jest jako kompozycja usług dostarczanych przez EDMS (fragment odpowiadający klasie *SerializedPart*) i IMS (fragment odpowiadający klasie *InventoryItem*).

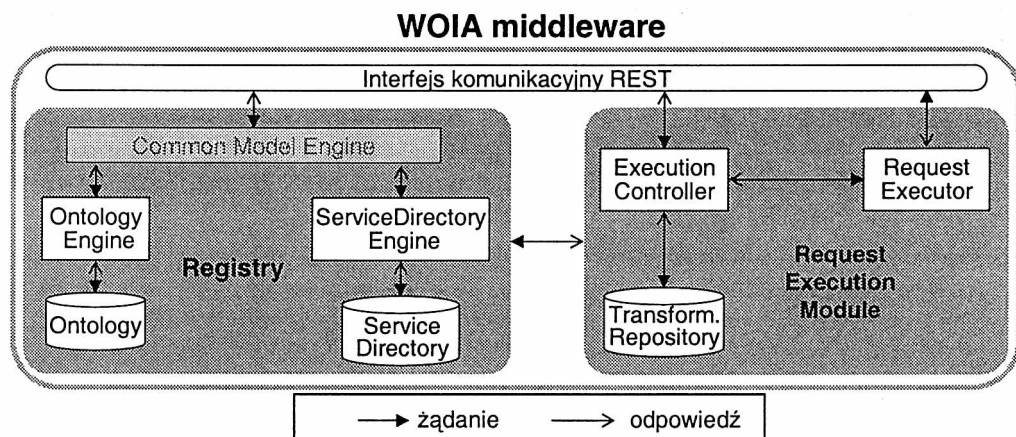


Rys. 13. Pula usług odpowiadająca przykładowej ontologii

5.4. WOIA MIDDLEWARE

Prezentowany komponent middleware jest kluczowym elementem rozwiązania, umożliwiającym semantyczną integrację systemów informatycznych bazowaną na architekturze WOIA. Składa się on z dwóch głównych modułów funkcjonalnych: rejestru usług (*Registry*) oraz modułu wykonawczego (*RequestExecutionModule*),

współdziałających ze sobą w celu umożliwienia rejestracji, kompozycji oraz wykonywania usług sieciowych. Ogólny schemat WOIA middleware przedstawiony jest na rysunku 14.



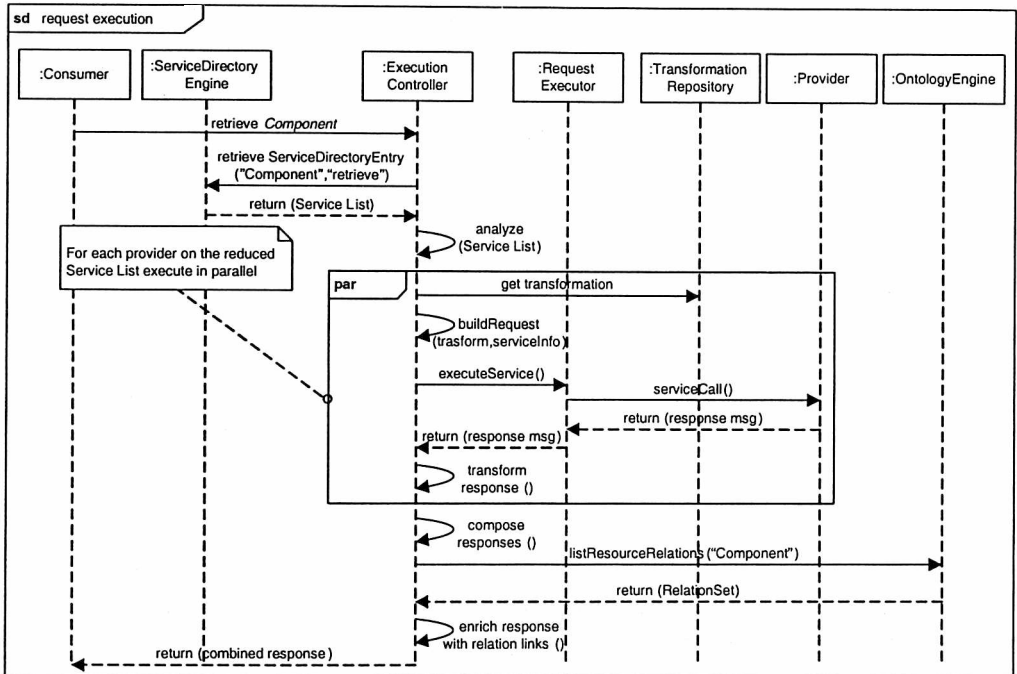
Rys. 14. Architektura WOIA middleware

Podobnie jak w innych rozwiązaniach bazowanych na usługach sieciowych, w proponowanym podejściu kluczową rolę odgrywa rejestr usług, pozwalający rejestrować i odkrywać usługi sieciowe. Wykorzystuje on koncepcyjny model ontologii do stworzenia katalogu usług (*ServiceDirectory*), rozszerzającego informacje zawarte w ontologii o dane dotyczące dostawców usług obsługujących zdefiniowane zasoby. Głównymi elementami rejestru usług, które pozwalają na dostęp do ontologii i katalogu usług są: *OntologyEngine* i *ServiceDirectoryEngine*. Dostęp do nich jest synchronizowany przy pomocy dodatkowego modułu *Common Model Engine*, który zapewnia zgodność modeli ontologii i katalogu usług poprzez zapewnienie ich równoległej modyfikacji. Dodatkowo pozwala on łączyć i wybierać sterowniki dla różnych formatów ontologii i katalogu usług. Działanie *OntologyEngine* polega na obsłudze ładowania modelu ontologii, generowaniu na jego podstawie szkieletu katalogu usług (wypełnianego danymi o dostawcach w miarę ich rejestracji), obsłudze żądań dotyczących przeglądania i modyfikacji modelu ontologii.

ServiceDirectoryEngine obsługuje rejestrowanie i wyrejestrowywanie usług przez dostawców oraz odpowiada na zapytania dotyczące usług dostępnych dla wyspecyfikowanego typu zasobów.

Konsumenci usług mogą w razie konieczności pracować w tradycyjnym trybie SOA, tzn. wysyłać zapytania o listę dostawców usług do rejestru i na jej podstawie wysyłać żądania bezpośrednio do dostawców usług. Ponieważ jednak dostawcy oferują usługi pozwalające operować jedynie na zasobach rzeczywistych, a konsumenci są zazwyczaj zainteresowani zasobami biznesowymi, taki tryb interakcji jest dla kon-

sumentów niewygodny i nieefektywny. W proponowanej architekturze konsumenci w ogóle nie muszą odpytywać rejestru o listę dostawców, a jedynie wysłać żądanie wykonania usługi do modułu wykonywania żądań (*RequestExecutionModule*) pod wirtualny link URI identyfikujący zasób biznesowy. Na podstawie informacji o dostępnych dla danej usługi dostawcach pobranej z katalogu usług, *ExecutionController* przeprowadza analizę przyjętego żądania oraz dekomponuje je do zbioru żądań częściowych dotyczących zasobów rzeczywistych (rysunek 15).

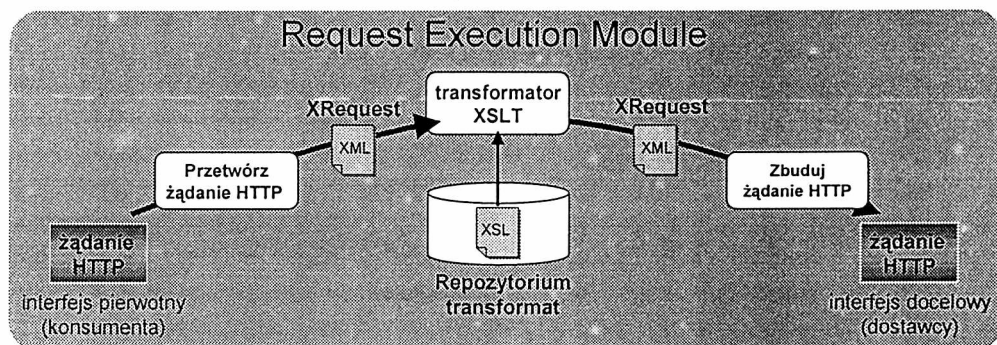


Rys. 15. Wykonanie żądania konsumenta przez RequestExecutionModule

Opcjonalnie może nastąpić również transformacja wybranych żądań częściowych do specyficznych cech interfejsu dostawcy na podstawie skryptów transformacyjnych (zgłaszanych w trakcie rejestracji usługi przez dostawcę). Transformacja ta wykonywana jest za pomocą translatora XSLT, co jest możliwe dzięki przekształceniu oryginalnego żądania HTTP na odpowiadający mu dokument w stworzonym na potrzeby WOIA dialekcie XML nazwanym XRequest. Po wykonaniu transformacji dokument XRequest jest przekształcany w nowe żądanie HTTP zgodne z interfejsem dostawcy (rysunek 16).

Tak przygotowane żądania częściowe przekazywane są do wykonywanych równolegle wątków procesu *RequestExecutor*, które wysyłają je do dostawców. Częściowe odpowiedzi zbierane są przez *ExecutionController*, a następnie komponowane do ostatecznej odpowiedzi, która na podstawie ontologii jest dodatkowo

rozszerzana o informację dotyczącą powiązanych zasobów i ostatecznie odsyłana do konsumenta.



Rys. 16. Proces transformacji żądania konsumenta

Nawiązując do wcześniejszego przykładu, konsument może wysłać do *RequestExecutionModule* żądanie dotyczące zasobu biznesowego typu *Component*, które zostanie zdekomponowane na dwa żądania częściowe odnoszące się do pary zasobów rzeczywistych tworzącej zasób biznesowy *Component* i dostarczanych systemy EDMS i IMS. Uzyskana odpowiedź będzie zawierać reprezentację (lub kolekcję) instancji zasobu typu *Component*, wraz z bezpośrednimi linkami do powiązanych z nimi instancji zasobów typu *Slot*, *PartModel*, *Location* oraz *InspectionSheet*.

Oprócz omówionej do tej pory podstawowej funkcjonalności, WOIA middleware posiada kilka dodatkowych cech, pozwalających na jego sprawne i efektywne wykorzystanie. Do najważniejszych z nich należy zaliczyć:

- mechanizmy monitorujące status usług zarejestrowanych przez dostawców i na bieżąco aktualizujące katalog usług,
- mechanizmy rozgłaszania adresu sieciowego, pozwalające konsumentom oraz dostawcom wykrywać w sposób automatyczny obecność i położenie komponentu middleware,
- mechanizmy zapewniające równomierne rozłożenie obciążenia (ang. *load balancing*) pomiędzy serwerami, na których działają usługi dostawców.

5.5. WEB ORIENTED ARCHITECTURE

Typowy model bezpieczeństwa stosowany dla usług sieciowych REST daje możliwość zastosowania dowolnego standardu bezpieczeństwa oraz narzędzi zarządzania ruchem w sieci stworzonych dla protokołu HTTP [Prescod 2002]. Wynika to z pełnej zgodności REST ze standardem HTTP. Biorąc pod uwagę fakt, że każdy zasób posia-

da swój własny unikalny URI, polityka bezpieczeństwa może np. zostać zrealizowana przy użyciu mechanizmów ACL (*Access Control List*).

W przedstawionym rozwiązaniu zastosowano dodatkowo protokół SSL do szyfrowania transmisji przesyłanych reprezentacji zasobów i weryfikacji tożsamości użytkowników, w połączeniu natywnymi mechanizmami autoryzacji systemów informatycznych udostępniających usługi sieciowe.

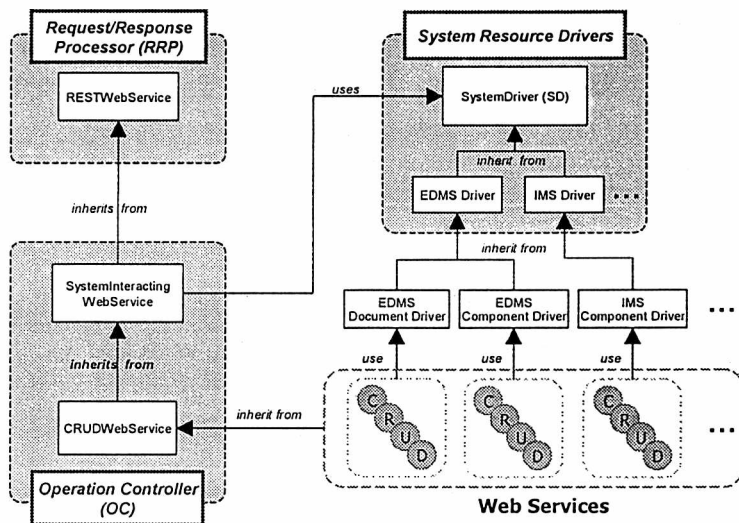
6. FRAMEWORK DO BUDOWY USŁUG SIECIOWYCH REST DLA WOIA

Budowanie rozwiązania integracyjnego w oparciu o architekturę WOIA wymaga stworzenia usług sieciowych REST dla wszystkich zasobów zdefiniowanych w ontologii. W środowisku przedsiębiorstwa, które składa się ze znaczącej ilości systemów informatycznych, proces ten może zostać zdecydowanie przyspieszony poprzez użycie frameworka służącego do budowy usług sieciowych. Proponowany framework wprowadza reguły strukturyzacji kodu pozwalające zidentyfikować, uogólnić i wyodrębnić bloki kodu współdzielone pomiędzy: wszystkimi usługami, podzbiorem usług realizujących taką samą operację na różnych zasobach oraz podzbiorem usług dostarczanych przez konkretny system informatyczny. Umożliwia to osiągnięcie wysokiego poziomu ponownego użycia kodu (ang. *code reusability*), co znacząco ułatwia jego utrzymanie i modyfikację. Funkcjonalne bloki kodu odpowiadające wymienionym zbiorom reprezentowane są we frameworku odpowiednio jako moduły: *Request-Response Processor*, *Operation Controller*, *System Resource Drivers* (rysunek 17).

Niezależnie od wykonywanej funkcji, każda z usług sieciowych musi realizować pewną grupę zadań, do których zaliczyć można między innymi parsowanie żądań konsumentów, parsowanie plików konfiguracyjnych, obsługę wyjątków, formatowanie odpowiedzi itd. Ta grupa zadań realizowana jest przez moduł *Request-Response Processor* implementowany przez klasę *RESTWebService*.

Zadania wykonywane przez moduł *Operation Controller* są specyficzne dla danej operacji ze zbioru CRUD, ale mogą być uogólnione dla wszystkich typów zasobów. Przykładowo, ogólny schemat operacji Update realizowany jest według następujących kroków:

- otwarcie transakcji,
- wyszukanie i pobranie zasobu,
- modyfikacja zasobu (jeśli został odnaleziony),
- zatwierdzenie transakcji w przypadku powodzenia,
- lub przywrócenie stanu sprzed otwarcia transakcji w przypadku wystąpienia błędu.



Rys. 17. Struktura frameworka

Moduł *OperationController* jest implementowany przez klasy *SystemInteractingWebService* i *CRUDWebService*, definiujące szkielety operacji CRUD i ogólny sposób interakcji z zasobami systemów informatycznych.

Moduł *System Resource Drivers* implementowany jest w postaci hierarchii klas sterowników grupujących kod odpowiedzialny za wykonywanie podstawowych operacji na specyficznym systemie informatycznym. Do operacji tych można zaliczyć między innymi: otwieranie i zamykanie sesji połączenia, logowanie i wylogowywanie użytkownika oraz otwieranie i zamykanie transakcji, jak również tworzenie, pobieranie, modyfikowanie i kasowanie zasobów. Dodatkowym celem sterowników jest wewnętrzna enkapsulacja specyficznych API systemów informatycznych w celu zapewnienia jednolitości interfejsów.

W praktyce, budowa usług sieciowych przy użyciu opisanego frameworka sprowadza się do dostarczenia zbioru sterowników dla zasobów rzeczywistych oferowanych przez indywidualne systemy informatyczne.

7. WERYFIKACJA ROZWIĄZANIA

Opisane podejście integracyjne zostało zastosowane i zweryfikowane przy budowie zintegrowanego środowiska oprogramowania w DESY. Weryfikacja została przeprowadzona częściowo na działającym obecnie w DESY środowisku produkcyjnym (tam gdzie to było możliwe), a częściowo w środowisku testowym stworzonym na potrzeby eksperymentów.

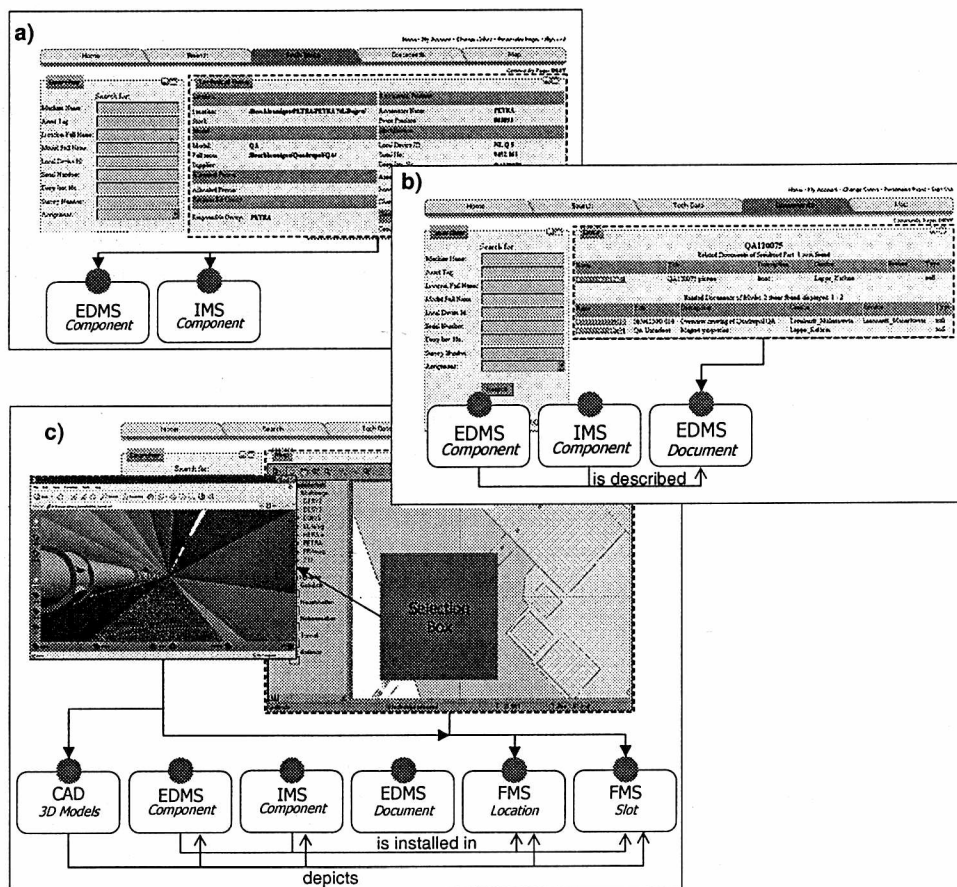
W środowisku testowym przygotowano konfigurację infrastruktury (WOIA middleware, dostawców i konsumentów) oraz dane testowe, pokrywające wszystkie przewidziane przez projekt sytuacje. Przeprowadzone eksperymenty w pełni potwierdziły zgodność działania wszystkich opisanych wcześniej mechanizmów komponentu middleware oraz usług sieciowych z ich zaprojektowanym zachowaniem. Wykonane pomiary wydajności dowodzą, że średni narzut czasowy infrastruktury integracyjnej wynosi w przybliżeniu 0.1 s, co stanowi około ~15% całkowitego czasu realizacji żądania konsumentów (pozostałe ~85% to czas realizacji żądania przez systemy informatyczne).

Doświadczenia zdobyte podczas wykorzystania opisanego frameworka przy integracji środowiska oprogramowania w DESY potwierdzają jego skuteczność w redukcji czasu budowy, kosztów i wysiłków niezbędnych do utrzymania budowanego kodu integracyjnego. Stopień ponownego użycia kodu osiągnięty dla zbioru usług sieciowych w DESY wyniósł 93%, przy czym wynik ten nie uwzględnia zewnętrznych bibliotek użytych do budowy frameworka ani faktu, że usługi zbudowane na jego bazie są używane przez wielu konsumentów (w takim wypadku wynik byłby znacząco wyższy).

Opisane rozwiązanie zostało z powodzeniem zastosowane w wielu scenariuszach integracyjnych w DESY. Jednym z najbardziej efektywnych przykładów zastosowania jest prototypowy Portal Integracyjny, zbudowany jako pojedynczy punkt dostępu do istotnych informacji dotyczących komponentów akceleratorów w różnych fazach cyklu życiowego. Portal pozwala użytkownikom na wyszukiwanie niezbędnych informacji na podstawie tradycyjnych zapytań specyfikujących szukane wartości metadanych, lub w oparciu o wizualną nawigację przy użyciu map elektronicznych lub wirtualnej rzeczywistości. Działanie portalu zilustrowano na rysunku 18.

Użytkownik portalu poszukujący informacji o wybranym zasobie typu *Component*, ma możliwość wyświetlenia pełnej informacji o tym zasobie na zakładce „*Technical Data*”, komponowanej na podstawie informacji pochodzących z systemów EDMS i IMS (rysunek 18a). Wyszukanie powiązanych z komponentem dokumentów umożliwia użytkownikowi zakładka „*Documents*” (rysunek 18b), zaś zakładka „*Map*” prezentuje mapę pokazującą położenie komponentu na obszarze kampusu DESY. Istnieje również możliwość zaznaczenia fragmentu mapy i wyświetlenia go pod postacią modelu wirtualnej rzeczywistości (rysunek 18c) budowanego „w locie” z zasobów typu *3D_Model* oraz koordynat przestrzennych przechowywanych w zasobach typu *Slot* oraz *Location*. Należy zaznaczyć, że wyświetlanie informacji w trakcie przełączania się między zakładkami odbywa się na bazie wirtualnych linków URI (wskazujących na zasoby biznesowe będące w relacji z zasobem aktualnie używanym) dostarczanych przez WOIA middleware. Przykładowo, przełączenie się z zakładki „*Technical data*” na zakładkę „*Map*” spowoduje wyświetlenie mapy wycelowanej na slocie zawierającym wyszukany wcześniej komponent. Podobnie kliknięcie na reprezentację komponentu na mapie

bądź w wirtualnej rzeczywistości pozwoli wyświetlić dotyczącą go listę dokumentów bądź jego dane techniczne.



Rys. 18. Przykład zastosowania – Portal Integracyjny w DESY

8. PODSUMOWANIE I KIERUNKI PRZYSZŁYCH BADAŃ

W ostatnich latach wystąpiło wyraźnie dostrzegalne dążenie podejść integracyjnych w kierunku prostych i przystępnych rozwiązań, bazowanych na rozproszonym przetwarzaniu przy użyciu usług sieciowych. Proponowana nowatorska architektura Web Oriented Integration Architecture, w znaczącym stopniu rozszerza możliwości zyskującej w ostatnim czasie na popularności architektury WOA.

Doświadczenia zdobyte w trakcie wdrażania i eksploatacji opisanego rozwiązania w DESY, potwierdzają postawione tezy oraz zgodność stworzonego rozwiązania z pierwotnymi założeniami projektu. Pomimo, że najważniejsze cele projektu zostały już osiągnięte wciąż istnieją dalsze możliwości rozwoju przedstawionej architektury. W szczególności badania nad mechanizmami pozwalających na pełną obsługę przez WOIA middleware transakcji rozproszonych oraz nad wprowadzeniem w zintegrowanym środowisku mechanizmów Single Sign-On, mogłyby zapewnić użyteczne rozszerzenie dla zaawansowanych mechanizmów dostępu do zasobów systemów informatycznych zastosowanych w WOIA middleware. Kolejnym możliwym kierunkiem rozwoju są badania nad zastosowaniem mechanizmów enkrypcji dokumentów XML jako sposób zapewnienia jeszcze wyższego poziomu bezpieczeństwa w krytycznych zastosowaniach opisanego rozwiązania.

LITERATURA

- [Bussler 2003] Bussler C., *B2B Integration*. Springer, 2003.
- [Curbera 2005] Curbera F., Weerawarana S., Leymann F., Storey T., Ferguson D.F., *Web Services Platform Architecture : SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall PTR, 2005.
- [Erl 2005] Erl T., *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, 2005.
- [FieTay 2000] Fielding R., Taylor R.N., *Principled design of the modern Web architecture*. ACM Press New York, NY, USA, 2000.
- [Grigori 2006] Grigori D., Corrales J.C., Bouzeghoub M., *Behavioral matchmaking for service retrieval*. Proceedings of the ICWS'06, p. 145–152. IEEE Computer Society Press, Los Alamitos, CA, 2006.
- [Halevy 2005] Halevy A.Y., Ashish N., Bitton D., Carey M., Draper D., Pollock J., Rosenthal A., Sikka V., *Enterprise information integration: successes, challenges and controversies*. Proceedings of the 2005 ACM SIGMOD international conference on management of data. ACM Press, 2005.
- [HohWoo 2003] Hohpe G., Woolf B., *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional 2003.
- [Linthicum 1999] Linthicum D.S., *Enterprise Application Integration*. Addison-Wesley, November 1999.
- [Mayerl 2005] Mayerl C., Vogel T., Abeck S., *SOA-Based Integration of IT Service Management Applications*. IEEE International Conference on Web Services (ICWS'05).
- [McAfee 2006] McAfee A., *Enterprise 2.0: The Dawn of Emergent Collaboration*. MIT Sloan Management Review 47, No. 3, 2006, pp. 21–28.
- [Morgenthal 2005] Morgenthal J.P., *Enterprise Information Integration: A Pragmatic Approach*. Lulu Press, 2005.
- [MusORe 2006] Musser J., O'Reilly T., *Why Web 2.0 Matters and How You Can Make the Most of It*. O'Reilly Radar, 2006.

- [Noy 2004] Noy N.F., *Semantic integration: a survey of ontology-based approaches*. ACM SIGMOD Record Volume 33, pp, 65–70, ACM Press New York, NY, USA, 2004.
- [Prescod 2002] Prescod P., *Second Generation Web Services*. O'Reilly Web Services <http://webservices.xml.com/pub/a/ws/2002/02/06/rest.html>
- [Ruh 2000] Ruh W.A., Maginnis F.X., Brown W.J., *Enterprise Application Integration: A Wiley Tech Brief*. Wiley, 2000.
- [SerCra 2002] Serain D., Craig I., *Middleware and Enterprise Application Integration*. Springer, 2002.
- [Shirky 2002] Shirky C., *Planning for Web Services: Obstacles and Opportunities*. O'Reilly Media, Sebastopol, CA, 2002.
- [SmaSad 2002] Smatani G., Sadhwani D., *Enterprise Application Integration (EAI) and Web Services*. Expert Press, 2002.
- [Szepielak 2006] Szepielak D., *REST-Based Service Oriented Architecture for Dynamically Integrated Information Systems*, Proceeding of the IBM PhD Students Symposium at Fourth ACM International Conference on Service Oriented Computing ICSOC'06. Chicago, USA, pp. 7–12.
- [UshGru 2004] Uschold M., Gruninger M., *Special section on semantic integration: Ontologies and semantics for seamless connectivity*. ACM SIGMOD Record Volume 33. pp 58–64 ACM Press, New York, NY, USA, 2004.
- [W3C] <http://www.w3.org/2002/ws/arch/2/06/wd-wsa-gloss-20020605.html>

WEB ORIENTED INTEGRATION ARCHITECTURE FOR SEMANTIC INTEGRATION OF INFORMATION SYSTEMS

Integration of information systems became an important issue in the early 90's. Since that time various approaches and technologies have been devised for addressing the integration problem. In recent years a dominant position among integration solutions has been reached by the Service Oriented Architecture (SOA) and SOAP Web Services. With the latest sprout of the Web 2.0 concept a subtype of SOA called Web Oriented Architecture (WOA) has gained increasing attention. WOA simplifies the traditional model of interaction for networked applications by introducing a uniform functional interface which facilitates the integration process. This paper describes a novel integration architecture called Web Oriented Integration Architecture that offers a much more functionality than original WOA and at the same time does not introduce any additional complexity for Web service consumers.

Zygmunt MAZUR*

Teresa MENDYK-KRAJEWSKA*

THE BATTLE FOR INFORMATION SYSTEM SECURITY

This paper concerns the problem of IT system protection and the authors emphasise the complexity of the protection problem and resulting from it difficulties in achieving high security level. It covers causes of threats that arise from connecting the system to a network as well as problems related to the protection of an IT system against undesired operations.

1. INTRODUCTION

More and more frequently companies are becoming dependent on the global network, which makes them a perfect target for criminal activity within the network. And whereas in the case of IT systems of large institutions, there is implemented a customised security policy, considerable amounts of money can be spent on such protection, and the system security is monitored by a trained personnel, an ordinary user has to face all the problems that arise individually.

Ensuring security of the system is a difficult and complex task.

2. THREATS POSED BY THE INTERNET AND THE INTRANET

Everything that a user seeks on the Internet may create problems: transfer of files, using e-mail and instant messengers or accessing webpages. Together with a file downloaded from the network, a harmful software may be introduced to the system, an e-mail attachment may contain a virus, instant messengers create many possibilities of transferring a malicious code, and webpages often include a spyware. P2P networks are very dangerous as they are likely to infect the system with a spyware. For instance,

* Instytut Informatyki Stosowanej, Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27, {zygmunt.mazur, teresa.mendyk-krajewska}@pwr.wroc.pl

the system is threatened with Java applets and ActiveX controls that constitute a source of harmful operations.

Network threats are becoming more and more dangerous. The viruses that are installed on computers at present can send their copies to addresses included in the user's address book, many of them are activated automatically during each start of the operating system. They may not only damage files or the main board of a computer, obtain lists of active processes, installed applications, the structure of disc catalogues and information on the operating system version or the installed network card, but also download files from the network, from webpages specially prepared for that purpose. A virus can make of the infected system a tool for distributing spam or carrying out a DoS attack, disable protection programs it finds in the system and lock Internet services of companies producing this type of software.

A spyware monitors user's actions and intercepts confidential data whereas Trojans can connect remotely and transfer collected data to an address indicated and carry out various commissioned tasks. Such programs can duplicate themselves on a disc and take a random name in this way effectively concealing their presence in the system.

Viruses are more and more insidious. However, not Internet viruses or worms pose the main threat but Trojans that act on their own and insidiously enter the system, and they additionally decrease the level of computer security maintained so far by changing, for example, parameters of a network firewall or removing installed system overlays.

It is a great arduousness for the user to fight spam (advertising mail), even when anti-spam filters are in place both on the mail server and on the workstation. Experts raise the alarm, as spammers have started to employ illegal methods in their fight against protection. For example, an Israeli company Blue Security decided to give up its further activity after their successes in the area, because of unceasing attacks on their servers and the threat of system infection.

Mass media inform about ever more frequent successful hackings of various IT systems (also those strategic ones), and the phenomenon of the network threat has been called "the Internet criminalisation". According to leading network security experts, the scale of Internet crime has increased to a horrifying degree. Networks are controlled by organised gangs, more than two thirds of computers are used for criminal purposes, and the age of hackers is ever earlier. A report published at the beginning of 2006 by FBI shows that losses incurred by American companies and institutions as a result of various computer attacks in 2005 exceeded USD 67 billion. The losses were estimated by FBI based on information gathered from more than two thousand private and public organisations in four states of the USA. The attacks had become a daily occurrence, only one in ten companies had been able to resist them. The main threat had been posed by viruses and spyware and one in five institutions or companies had experienced computer sabotage. Half of the registered attacks had been triggered in the USA and China (36 countries altogether; it is hard to establish this fact due to applied masking). It is much more difficult to assess losses suffered by private users due to computer crimes.

All respected institutions and companies have their own websites (usually protected by a password) containing a lot of important information about their business and institutional activity. It is not surprising that web pages are break-in targets for capturing data that can be used in different ways by the competition or during negotiations between the organizations. An example of such unlawful actions has been noted by the press which informed that the doctors from "Porozumienie Zielonogórskie" (the association of the primary care providers in the Zielona Góra region) accused the National Health Fund (NFZ) of an officially confirmed illegal gain of information useful in the financial negotiations between NFZ and the doctors. An appointed expert established that attacks on the protected website of Porozumienie were carried out from one of the divisions of NFZ for over a year, which ended with an investigation initiated by the public prosecutor's office.

Specialists unanimously say that the observed network crime will continue growing.

An IT system is also really threatened by its co-users. Some sources even point out that threats posed by intranets are greater and reach two thirds of all registered events that undermine system security. Usually the systems have poorer protection from the inside, however, realising of internal threats, companies producing protection tools have introduced a new type of security solutions against such attacks, i.e. they proposed a kind of intelligent gates which segment an intranet. Such a division into zones and the control of the flow between them are to prevent unauthorised access by co-users of a given local network. Any detected unauthorised action results in blocking the traffic and putting the suspicious computer in quarantine.

Actions that are contradictory to the scope of responsibilities and harmful for companies are most frequently undertaken by unsatisfied (low-paid, underestimated) employees and they pose the major threat coming from the intranet. They have relevant knowledge about an IT system, network configuration, conducted activity and are strongly motivated to perform destructive actions. System security may also be violated by employees accidentally, due to negligence, disorderliness or the lack of competence. Even the best designed security system will not help if people responsible for protecting it fail.

During recent years, much attention has been paid to sociotechnical attacks. The easiest way to gain confidential information, which allows for evading a complex security system, consists in the application of psychological techniques. For example, a telephone conversation skilfully conducted with a chosen person allows making them reveal interesting information or perform given actions that enable an intruder to enter a system. Such contact may also be made in person or via the electronic mail, or instant messengers. Institutions' employees are not much aware of it and the only means of protection against such actions is adequate training of employees.

Recently there has appeared a relatively easy way to enter a network from the inside. In order to do so it is enough to have a notebook fitted with a radio card that allows for cordless connection to a local network. Also local networks in Wi-Fi technology, due to their specificity, poor security and ready-made tools to crack

them, display high vulnerability to threats from the outside as well as from co-users. The protection of cordless networks constitutes a serious problem and is not discussed in this paper.

3. SYSTEM DEFENSE AGAINST THREATS

Problems with protecting an IT system arise mainly from the following: high complexity of operating systems and applications due to which they are not free from defects (gaps) and which make configuration difficult, imperfection of the very system security mechanisms (their defects and insufficient protection against attacks of the new type), installation of system overlays always with some delay, great variety of attacks and rapid development of hacking techniques, availability of automatic hacking tools that allow even an inexperienced person to conduct an attack, lack of sufficient knowledge within the scope of safe system administration, any negligence on the part of the user within the system protection, risk related to the installation of a software via the Internet, risk related to the use of free versions of a security software. Modern security packages (e.g. Norton Security 2006 or Panda Platinum 2006 – Internet Security) contain almost all elements that ensure high level of security. i.e. an antivirus monitor and scanner, an e-mail scanner, a firewall, a register monitor, an antidiabler, an anti-spam program, a webpage filter, an anti-phishing module, a network management module, an automatic update module, a system state analysis module that allows for quick diagnostics if an infection is suspected, as well as heuristic mechanisms that identify unknown threats. Despite that fact, users frequently use popular free versions whose functionality is significantly limited compared to commercial solutions. The tools cannot be updated (e.g. update of virus bases and firewall rules) after a given time of usage, hence when they are used, maximum system security cannot be expected. Moreover, buyers of the commercial software are provided with free technical service in case of any problems with the program they use. Comprehensive protection tools will still be unavailable for an average Internet user due to their high price. It is also important to have protection software that operates in real time, and not only at request, however, it often slows down performance of the system and many users consciously resign from this function of the tool.

The majority of traditional security mechanisms operates basing on a signature base and their efficiency is determined by the frequency of updating relevant bases. A better solution is to use a mechanism that controls generated network traffic paying special attention to compatibility with given requirements. IDS (Intrusion Detection Systems) for hacking detection and IPS (Intrusion Protection Systems) can operate this way, and they also are to protect against such threat. However, there arises a problem, i.e. what should be treated as a standard behaviour for a given IT system. Both systems are also to notify about a violation of security rules.

In order to protect a system, unnecessary services should be deactivated competently in compliance with its functionality, and at the same time the interdependence between services should be carefully analysed.

In the case of network firewalls, it is important to maintain a balance between security requirements and system functionality (or network functionality). On the one hand, too complicated principles of their operation reduce the speed of system operations, whereas on the other hand, each activated service may contribute to lowering security level. For instance, a harmful software installed via e-mail will allow for the initiation of remote connection from a protected area (of a computer or a network) in a way that will not alert a firewall. The firewall of Windows operating system does not control the outgoing traffic, which is a weak point of such tools. A user informed by a personal network firewall about a potential threat may “manually” stop the outgoing flow of data or lock activated unknown programs, however, making the right decision within this scope requires extensive experience and considerable knowledge concerning the activity of given applications, if for example, from the level of one application another one is activated. A user must be not only notified about threats or an attack carried out, but they should also be instructed how to deal with problems if a protection software does not fulfil this task automatically. There are not many tools that make such function available, as the identification of a situation triggered by the threat is not an easy task.

Many problems are still caused by the used network protocols, such as Telnet (*Telecommunication Network*) or POP3 (*Post Office Protocol v.3* – an Internet protocol of an application that allows for downloading e-mail messages from a server to a local computer), which do not have encrypting mechanisms embedded and send data in the explicit form. Another threat is a possibility to falsify IP addresses by changing the content of information in the address box of a package, which allows hackers to deceive filters of the network firewall.

Other threats result from the fact that numerous applications and services use HTTP, which operates on port 80 intended for accessing webpages, hence such traffic will not be blocked by firewalls.

4. SUMMARY

Applied tools and all actions performed in order to protect a system present numerous problems and in the face of the newest threats, such actions turn out insufficient. Maintaining high level of IT system protection requires not only relevant security but also considerable knowledge, commitment and regularity in undertaking activities to ensure security of a system by both the administrator and all system's users.

The achieving and maintaining high security level of an IT system is a complicated task. Defects in a system software and an application (that result from their complexity and too short testing period), the attractiveness of network resources for criminal ac-

tivities, imperfection of protection tools, the development of hacking techniques, limited users' knowledge about safe administration of a system – these are just a few of the causes of problems related to the effective IT system protection.

The protection of a system is a never-ending process in which creators of security systems struggle with hackers who provide tools to crack the systems. The trend of the network threat phenomenon will keep increasing and the problem with efficient IT system protection will keep deepening as harmful software is now created by professional hackers who benefit from it financially.

Experts agree that the situation requires urgent, resolute actions and cooperation between international security organisations, telecommunications companies, hardware and software producers as well as ordinary users. One of the basis tasks is the development of mechanisms that will allow for identifying network criminals, at the same time guaranteeing (as much as it is possible) anonymity of other network users.

BIBLIOGRAPHY

- [Mendyk 2005] Mendyk-Krajewska T., *Identyfikacja zagrożeń bezpieczeństwa komputerów w sieci Internet*. W: Nowe technologie w komputerowych systemach zarządzania. Internet w społeczeństwie informacyjnym. WKŁ, Warszawa 2005.
- [ScheMen 2004] Scheliga M., Mendrala D., *Bezpieczeństwo Twojego komputera*. Helion, Gliwice 2004.
- [SchGrCar 2002] Schetina E., Green K., Carlson J., *Bezpieczeństwo w sieci*. Helion, Gliwice 2002.
- [MazMen 2006] Mazur Z., Mendyk-Krajewska T., *Analiza stanu plików dla wzrostu bezpieczeństwa systemów bazodanowych*. W: Bazy danych. Struktury, algorytmy, metody. Praca zb. pod red. S. Koziełskiego. Wybrane technologie i zastosowania. WKŁ, Warszawa 2006, s. 325–334.
- [MenTre 2006] Mendyk-Krajewska T., Trelka D., *System kontroli i analizy aktywności plików*. W: Systemy informatyczne z ograniczeniami czasowymi. Praca zbiorowa pod red. A. Kwietnia i P. Gaja. WKŁ, Warszawa 2006, s. 471–482.

WALKA O BEZPIECZEŃSTWO SYSTEMU INFORMATYCZNEGO

Praca dotyczy zagadnienia bezpieczeństwa systemów informatycznych, przy czym autorzy kładą nacisk na złożoność problemu ochrony oraz na wynikające z tego trudności w osiągnięciu wysokiego poziomu bezpieczeństwa. W pracy opisano przyczyny zagrożeń wynikające z połączenia systemu z siecią oraz poruszono wybrane problemy związane z zabezpieczaniem systemu informatycznego przed niepożądanym działaniem.

Teresa MENDYK-KRAJEWSKA*
Zygmunt MAZUR*

BEZPIECZEŃSTWO ROZPROSZONYCH SYSTEMÓW BAZODANOWYCH W SIECIACH BEZPRZEWODOWYCH

Praca dotyczy zagadnień bezpieczeństwa rozproszonych systemów bazodanowych w sieciach bezprzewodowych. W dobie Internetu oraz technologii bezprzewodowych, zapewnienie optymalnego bezpieczeństwa systemom bazodanowym staje się zadaniem coraz trudniejszym i bardziej złożonym. Utrata informacji (poufnych lub na pozór nieistotnych) może spowodować wymierne straty finansowe lub utratę prestiżu oraz zaufania. Firmy i instytucje starają się zagwarantować jak najlepszą ochronę danych, umożliwiając jednocześnie obsługę dodatkowych użytkowników. Równocześnie muszą radzić sobie z zarządzaniem coraz większą liczbą zdarzeń związanych z bezpieczeństwem oraz muszą mieć pełną kontrolę nad rodzajem i zasadami przepływu informacji w sieciowych systemach informatycznych.

1. WSTĘP

Realizacje sieci komputerowych w technologii bezprzewodowej, przede wszystkim z uwagi na krótki czas wdrażania, wygodę instalacji i użytkowania oraz elastyczność, cieszą się dużym zainteresowaniem, jednak dynamiczny ich rozwój hamowany jest trudnościami w zapewnieniu pożądanego poziomu bezpieczeństwa.

Bezprzewodowa technologia transmisji danych wykorzystywana w systemach teleinformatycznych niesie dodatkowe zagrożenia, nieznane w rozwiązaniach konwencjonalnych stosujących połączenia kablowe. Wynika to ze specyfiki łączy bezprzewodowych. Problemem jest ograniczenie zasięgu sygnału radiowego do wyznaczonego obszaru oraz łatwość ingerencji w strumień danych przesyłanych w otwartej przestrzeni. Stwarza to nowe zagrożenia dla bezpieczeństwa systemów bazodanowych, których serwery zawsze stanowiły interesujący cel ataków hakerskich. Wiele jest sposobów przeprowadzenia ataku na sieć bezprzewodową. Pierwszą przeszkodę do poko-

* Instytut Informatyki Stosowanej, Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27, {teresa.mendyk-krajewska, zygmunt.mazur}@pwr.wroc.pl

niania, może stanowić ukryta nazwa sieci lub stosowane filtry adresów czy protokołów. Bardziej zaawansowane działania mogą polegać na wykradaniu kluczy kryptograficznych oraz łamaniu zabezpieczeń stosowanych standardów kryptograficznych, do czego służą wyspecjalizowane narzędzia.

Niestety, użytkownicy sieci bezprzewodowych nie przywiązują wagi do ochrony danych, co jest wynikiem braku wiedzy na temat realnych zagrożeń oraz nieznajomości słabości powszechnie stosowanych standardów bezpieczeństwa.

2. ROZWÓJ TECHNOLOGII TRANSMISJI BEZPRZEWODOWEJ

Używanie połączeń bezprzewodowych w sieciach komputerowych to wygoda użytkowania i komfort dostępu do zasobów sieciowych. W bieżącym roku mija dokładnie 10 lat od wprowadzenia pierwszego standardu dla realizacji lokalnych sieci komputerowych bez użycia kabli. Definiuje on trzy warstwy fizyczne sieci. Dwie z nich (różniące się techniką modulacji i rozpraszania widma) wykorzystują fale radiowe o częstotliwości 2,4 GHz, trzecia warstwa, która dotyczy promieniowania podczerwonego, nie znalazła szerszego zastosowania w praktyce.

W ostatnich latach zainteresowanie sieciami realizowanymi w technologii bezprzewodowej szybko rośnie. Obserwuje się swego rodzaju modę na tego rodzaju dostępność Internetu i jest to w pełni uzasadnione zaletami bezprzewodowych rozwiązań. Najczęściej wykorzystywane w Polsce zakresy częstotliwości nie podlegają bowiem koncesjonowaniu, nie ma też potrzeby układania kabli i wykonywania przepustów przez ściany. Do tego dochodzą takie zalety jak: niskie koszty wdrażania i eksploatacji sieci, łatwość rozbudowy i rekonfiguracji, a także łatwość użytkowania, bo stosowanie technologii bezprzewodowej nie wymaga od użytkownika dodatkowej wiedzy.

Standardy sieci bezprzewodowych, opracowane przez Instytut Inżynierów Elektryków i Elektroników IEEE dotyczące warstwy fizycznej modelu odniesienia OSI/ISO¹, zostały zebrane w tabeli 1. Wszystkie one odnoszą się do transmisji pakietowej.

Pierwszym standardem sieci bezprzewodowych był IEEE 802.11 oparty o protokół sieci kablowych CSMA/CA (ang. *Carrier Sense Multiple Access with Collision Avoidance*)², umożliwiający transmisję z prędkością 1–2 Mbit/s. Zanim został on rozpowszechniony, w 1999 roku pojawił się kolejny standard, IEEE 802.11b, dotyczący pracy w tej samej częstotliwości (2,4–2,5 GHz), pozwalający na przesył z maksymal-

¹ *Open System Interconnection* – 7-warstwowy model odniesienia zdefiniowany przez międzynarodową organizację normalizacyjną ISO dla sformalizowania modułowej budowy oprogramowania systemowego sieci.

² Protokół wielodostępu do łącza ze śledzeniem stanu nośnika i unikaniem kolizji.

ną prędkością 11 Mbit/s, i on wprowadza nazwę Wi-Fi (ang. *Wireless Fidelity*)³ dla sieci bezprzewodowych.

Tabela 1. Standardy sieci bezprzewodowych

Oznaczenie standardu IEEE	Maksymalna prędkość transmisji [Mbit/s]	Zakres częstotliwości [GHz]	Rok	Uwagi
802.11 (802.1y)	1, 2	2,4–2,5	1997	Protokół CSMA/CA sieci kablowych
802.11a	54	5	1999	Częstotliwość głównie wykorzystywana przez radary i połączenia satelitarne; OFDM – zwielokrotnienie częstotliwości i polaryzacja
802.11b	11 w praktyce: 5,5	2,4–2,5	1999	Wi-Fi; DSSS – przesył całego strumienia bez polaryzacji fal nośnych: sieci WLAN
802.11g	54 w praktyce: 22	2,4–2,5	2003	OFDM – częstotliwości pakietów wybierane pseudolosowo; sieci WLAN
802.11n	540 w praktyce: 100	2,4 5	2008 ?	MIMO – zwielokrotnienie przestrzenne
802.16	75 w praktyce: 2	10–66 2–16	2007 ?	WiMax; OFDM; sieci WMAN

Najmniej popularnym standardem do realizacji sieci lokalnych jest IEEE 802.11a (również opracowany w 1999 roku), którego częstotliwość (5 GHz) wymaga w Polsce koncesjonowania z powodu większego poboru mocy i problemów z zasięgiem. Standard ten wprowadza technikę modulacji OFDM (ang. *Orthogonal Frequency-Division Multiplexing*) stosowaną dotąd w telewizji cyfrowej. Maksymalna prędkość transmisji w tym standardzie wynosi 54 Mbit/s. Tę samą wartość można teoretycznie osiągać stosując standard 802.11g dla nie koncesjonowanej częstotliwości 2,4–2,5 GHz, który pojawił się w 2003 roku, jednak, niestety, maksymalne szybkości przesyłu osiągane są jedynie w optymalnych warunkach pracy, przy zastosowaniu odpowiednich urządzeń, dobrych antenach, bardzo dobrym zestrojeniu całej sieci oraz niedużych odległościach. Zwykle w praktyce osiągana prędkość transmisji nie przekracza połowy maksymalnej wartości dla danego standardu. Standardy 802.11b i 802.11g to najpopularniejsze w Polsce standardy sieci WLAN.

³ Termin Wi-Fi został stworzony przez stowarzyszenie Wi-Fi Alliance założone w 1999 roku w celu certyfikowania urządzeń w bezprzewodowych sieciach lokalnych.

Ze specyfiki technologii bezprzewodowej wynika szereg ograniczeń. Uzyskanie określonego zasięgu wymaga odpowiedniego dobrania anten, gdyż w przypadku anten dookólnych fala zostaje spłaszczona, co powoduje, że sygnał nad i pod anteną zanika. Ponadto anteny dookólne są wrażliwe na wszelkiego rodzaju zakłócenia, dlatego też w przypadku dużych odległości lepszym rozwiązaniem jest stosowanie anten kierunkowych, wysyłających skupiony sygnał w określone miejsce. Większy zasięg osiągany może być między innymi dzięki specjalnemu trybowi pracy punktu dostępowego, który pełni wówczas rolę wzmacniaka. Przy dużej liczbie segmentów wydajność sieci zostaje obniżona, a niska przepustowość łączy może powodować blokadę dostępu innym użytkownikom. Problemem jest też znaczący spadek prędkości transmisji wraz ze wzrostem odległości między terminalami. Ponadto dla minimalizacji strat sygnału długość kabla łączącego antenę z punktem dostępowym ogranicza się do 10 m, gdyż powyżej tej odległości moc emitowanego sygnału znacząco spada.

Znaczne zainteresowanie sieciami bezprzewodowymi, przy wszystkich ograniczeniach wynikających z tej technologii, wymusiło dalsze intensywne prace nad technikami umożliwiającymi poprawę parametrów transmisji, przede wszystkim prędkości przesyłu oraz zasięgu sygnału. W 2003 roku rozpoczęto oficjalne prace (pierwsze działania już w 2001 roku) nad technologią WiMax (ang. *World Interoperability for Microwave Access*) przeznaczoną dla bezprzewodowych rozwiązań sieci miejskich WMAN, a w styczniu następnego roku nad technologią MIMO (ang. *Multiple Input Multiple Output*). W technologii MIMO stosowane jest zwielokrotnienie przestrzenne polegające na podziale strumienia danych na kilka strumieni (w ramach jednego połączenia) poruszających się w różnych kierunkach. Sygnały pochodzące z różnych stron docierają innymi drogami, zatem w różnym czasie, wobec czego istnieje możliwość ich identyfikacji poprzez pomiar różnicy czasu odbioru fal. Inna metoda pozwala identyfikować sygnały z różnych kierunków na podstawie polaryzacji. MIMO wymaga zatem instalacji kart radiowych wyposażonych w odpowiednią liczbę anten, a także implementacji specjalnych algorytmów umożliwiających wydzielanie (po stronie nadawczej) i łączenie (po stronie odbiorczej) wyróżnianych strumieni danych. Dzięki takiemu rozwiązaniu można osiągać większy zasięg oraz prędkość transmisji (przy dwóch antenach jedynie jeden z parametrów ulega poprawie). Kolejną zaletą wprowadzaną przez MIMO, jest poprawa jakości odbioru w wyniku możliwości przełączania między antenami, w celu uzyskania optymalnego poziomu sygnału (bo fale nakładają się i wzajemnie wygaszają). Ze względu na osiąganą szybkość transmisji, technologia ta pociąga za sobą potrzebę zmiany (lub przynajmniej modyfikacji) stosowanych mechanizmów szyfrowania.

WiMax to rodzina standardów 802.16 dotyczących bezprzewodowego dostępu szerokopasmowego. Osiągane parametry to: 50 kilometrowy zasięg stacji bazowej (niestety w praktyce nadal nie więcej niż 10 km) oraz maksymalna prędkość transmisji 75 Mbit/s. Stacje bazowe i terminale pracujące w wyższym paśmie częstotliwości (10–66 GHz) muszą być w zasięgu swej widoczności, dla niższego pasma (2–16 GHz)

– bezpośrednio pole widzenia nie jest wymagane. Technologię WiMax cechuje większa stabilność pracy (dzięki separacji anten i podziałowi na podkanały), bazowanie na częstotliwościach licencjonowanych (którymi można zarządzać, co ma wpływ na bezpieczeństwo sieci), jednak wadą rozwiązania jest brak zgodności ze standardem 802.11 (dopuszcza się jedynie współpracę na poziomie stacji bazowych). Mobilnej dostępności Internetu dotyczy standard 802.16e. Rozwiązanie polega na tworzeniu stref dostępowych (ang. *hot zones*), stanowiących rozszerzenie zasięgu punktów dostępowych (hot spotów), a dopuszczalna prędkość poruszania się użytkownika mobilnego wynosi 60 km/godz. Mobilna wersja WiMax nosi nazwę WiBro i nie jest jeszcze dostępna w większości krajów. Technologię tę cechuje wysoki koszt wdrożenia sieci, wysoki koszt utrzymania (co wynika z dostępności dużego pasma częstotliwości) oraz znacząca cena terminali abonenckich. Wszystko to wskazuje, że technologia WiMax przeznaczona jest dla firm, a nie dla klientów indywidualnych. Kolejnym standardem rodziny 802.16 jest 802.16m (zgodny z 802.16e), w którym maksymalna prędkość transmisji wynosi 1 Gbit/s, co może już stanowić konkurencję dla realizacji kablowych. Wprowadzenie tego standardu przewiduje się na rok 2009.

Oprócz wymienionych standardów opracowano wiele innych, wśród których warto wymienić standard 802.11e dotyczący przesyłania głosu i obrazu wideo, zatwierdzony w 2005 roku, oparty na technologii WMM (ang. *Wi-Fi Multimedia*). Współpracuje on ze standardami 802.11a, 802.11b i 802.11g gwarantując wyższy priorytet danym multimedialnym.

3. STANDARDY BEZPIECZEŃSTWA SIECI BEZPRZEWODOWYCH I ICH SŁABOŚCI

Odrębną grupę standardów opracowano dla zapewnienia bezpieczeństwa transmisji w technologii bezprzewodowej. Niestety, proponowane mechanizmy ochrony sieci Wi-Fi nie są pozbawione wad. Na ich temat pojawiło się wiele artykułów, a nawet całych książek, których autorzy wykazują słabości udostępnianych rozwiązań.

Pierwszym standardem był WEP (ang. *Wired Equivalent Privacy*) – opracowany przez komitet, w składzie którego nie było żadnego kryptografa. Do szyfrowania danych jest tu wykorzystywany szyfr strumieniowy RC4 (o zmiennej długości klucza), którego wymaganie stosowania niepowtarzalnego ciągu klucza nie jest spełnione. Kolejne pakiety są zatem szyfrowane tym samym kluczem, który jest w dodatku zbyt krótki (40 lub 104 bity). Ponadto, 24-bitowy wektor inicjujący jest albo wartością stałą, albo zmienianą cyklicznie, co świadczy o jego słabości. Z powodu braku zdefiniowanego sposobu zarządzania kluczami trudniejsza staje się jego aktualizacja. Jeden wspólny klucz (użytkowników i punktu dostępowego) jest używany do uwierzytelniania i szyfrowania danych, zatem każdy może odczytać wiadomość kierowaną do innego odbiorcy. Stosowana jest zasada ufania klientom, jednak oczywistym jest fakt, że

wraz z upływem czasu i wzrostem liczby użytkowników, poziom zaufania wobec tajności klucza maleje, a w przypadku złamania klucza podsłuchiwana jest cała sieć (a nie jedynie pojedynczy użytkownik). Na domiar złego szyfrowanie domyślnie jest wyłączone. Autentyczność danych w WEP potwierdzana jest przy pomocy łatwo podającego się kryptoanalizie mechanizmu CRC32, zaś autoryzacja użytkowników stosowana jest opcjonalnie. Nie wprowadza się też mechanizmu kontroli integralności ramek transmisyjnych, co umożliwia włączanie do strumienia danych nielegalnych pakietów.

Wynikiem opisanych wad było szybkie (w 2003 roku) wprowadzenie nowego standardu – WPA (ang. *Wi-Fi Protected Access*), kompatybilnego z WEP. Nadal stosuje się w nim algorytm szyfrowania RC4, jednak do samego procesu wprowadzono szereg zmian. Zastosowano dłuższy klucz (128 bitów) oraz lepiej rozwiązano problem operowania, tu 48-bitowym, wektorem inicjującym. Ponadto klucz, który jest inny dla każdego użytkownika (generowany na podstawie adresu MAC (ang. *Media Access Control*) komputera i numeru seryjnego pakietu), jest zmieniany co 10 kB przesyłanych danych. Do zarządzania kluczami standard WPA wprowadza protokół TKIP (ang. *Temporal Key Integrity Protocol*), w którym klucz właściwy generowany jest na podstawie klucza bazowego. Klucz bazowy może być albo współdzielony przez wszystkich użytkowników, które to rozwiązanie wystarcza w przypadku sieci domowych, albo jest tworzony dla danej sesji przez serwer RADIUS (ang. *Remote Authentication Dial-In User Service*)⁴, w przypadku dużych firm. Do zapewnienia integralności danych dostępny jest mocniejszy mechanizm MIC (ang. *Message Integrity Code*), zaś dla uwierzytelniania stron, które jest tu obowiązkowe – protokół EAP (ang. *Extensible Authentication Protocol*). Uwierzytelnianie może dotyczyć tylko klienta, albo obu komunikujących się stron, czyli również serwera.

Okazało się, że dopuszczenie uwierzytelniania jednostronnego stanowi poważną lukę w tym systemie zabezpieczeń. Sam protokół serwera RADIUS też nie jest pozbawiony wad, skutkiem czego istnieje możliwość ominięcia procesu uwierzytelniania. Udowodniono również, że istnieje możliwość podjęcia ataku na protokół TKIP, jednak nie ma pewności, czy są dostępne gotowe narzędzia do jego przeprowadzenia.

Mocniejsze zabezpieczenia oferuje kolejno opracowany (w 2004 roku) standard WPA2 (IEEE 802.11i), który stosuje nowszy sposób szyfrowania AES (ang. *Advanced Encryption Standard*). Jest to szyfr blokowy przyjęty w 1997 roku jako następca DES'a, dotąd powszechnie stosowanego standardu szyfrowania. Klucz główny jest uzgadniany między klientem a serwerem RADIUS i wysyłany przez serwer do punktu dostępowego. Następnie klient i punkt dostępowy uzgadniają klucz, z którego tworzy się kilka podkluczy wykorzystywanych do podpisywania komunikatów, szyfrowania danych i do szyfrowania klucza grupowego. Podobnie jak w WPA, do sterowania kluczami służy protokół TKIP, a obowiązkowe tu uwierzytelnianie realizowane jest w oparciu o protokół EAP.

⁴ Serwer weryfikujący tożsamość strony i wydający decyzję o jej akceptacji bądź odrzuceniu.

Niestety, WPA2 wymaga stosowania drogich urządzeń ze zintegrowanym procesorem i nie jest kompatybilne z dotychczasowym wyposażeniem sieci Wi-Fi. Użytkownicy starych rozwiązań, na skutek problemów z wymianą wbudowanego oprogramowania firmware lub kłopotów z aktualizacją sterowników, nie mogą używać algorytmu AES, który w nowych urządzeniach realizowany jest sprzętowo. Ogromna liczba operacji podczas procesów szyfrowania i deszyfracji stwarza problem mocy obliczeniowej przy próbach programowej realizacji AES w starszych urządzeniach. Z tej samej przyczyny stosowanie tak mocnych zabezpieczeń powoduje znaczne obniżenie wydajności sieci.

Nową propozycję uzupełniającą dotychczasowe rozwiązania stanowi technologia SSM (ang. *Secure Services Module*). Stwarza ona możliwość bieżącego skanowania przesyłanych pakietów przez punkty dostępne i uaktywnianie sygnału zakłócającego w razie wykrycia strumienia danych nieuprawnionego użytkownika. Ponadto wprowadza wysyłanie sygnału użytecznego w jednym wybranym kierunku z jednoczesnym wysyłaniem szumu zakłócającego w innych kierunkach.

Niektóre rutery (np. Cisco z LEAP) są wyposażone we własny protokół zabezpieczeń dla ochrony lokalnej sieci bezprzewodowej.

4. OCHRONA SYSTEMÓW BAZ DANYCH

W celu ochrony baz danych stosuje się przede wszystkim kontrolę dostępu, filtry bezpieczeństwa oraz szyfrowanie. Te powszechnie stosowane wzajemnie uzupełniające się rozwiązania nie są pozbawione wad.

Bezpieczeństwo systemów bazodanowych, oprócz kontroli dostępu do danych, wymaga zapewnienia ich operacyjnej i semantycznej spójności oraz zabezpieczenia przed niepożądanym wnioskowaniem (w przypadku statystycznych baz danych).

Dla kontroli dostępu i sprawdzania spójności danych można stosować mechanizmy zwane filtrami bezpieczeństwa pracujące zgodnie z przyjętą polityką. Odtworzenie danych na skutek ich bezprawnej modyfikacji jest możliwe dzięki systematycznemu tworzeniu kopii zapasowych. Niestety, przy dużej liczbie nieprawidłowości takie postępowanie prowadzi do obniżenia wydajności systemu, a w skrajnych przypadkach nawet czasowej blokady bazy danych. Dla zwiększenia sprawności działania filtrów bezpieczeństwa wymaga się rozwiązań integrujących filtry z bazą danych oraz innymi mechanizmami zabezpieczeń. Tak implementowany system ochrony powinien podlegać systemowi zarządzania bazą danych.

Ważnym elementem systemu bezpieczeństwa zwiększającym poziom ochrony danych – niezbędny w przypadku danych niejawnych (poufnych, tajnych) jest szyfrowanie. Niestety, proces ten może mieć wpływ na czas dostępu do danych – szczególnie w przypadku potrzeby przetwarzania całych rekordów dla szukania informacji według

fragmentów danych lub w celu szukania zakresów. Stąd możliwość wyboru jednego ze sposobów szyfrowania w bazach danych (rekordów, pól, fragmentów danych lub całych plików) dla optymalizacji procesu szyfrowania zależnie od potrzeb. Ważne jest, by szyfrowanie danych nie wprowadzało nadmiaru (bo zwiększa to rozmiar bazy), jednak nie zawsze takie rozwiązania są możliwe.

Pliki zawierające dane bazy mogą trafić do niepowołanych osób nie tylko w przypadku włamania do systemu, ale także w innych sytuacjach, np. w wyniku niewłaściwego przechowywania kopii zapasowych lub przekazania dysku z danymi wskutek jego awarii do serwisu. Z tych przyczyn, dla bezpieczeństwa baz danych, wskazane jest szyfrowanie.

Wiele systemów baz danych stosuje różne mechanizmy dla zabezpieczania przesyłanych danych pomiędzy serwerem a użytkownikiem. Na przykład w bazach danych Oracle 9i odpowiednie zabezpieczenia wprowadza pakiet Oracle Advanced Security, który umożliwia zarówno zachowanie poufności jak i integralności danych transmitowanych. Ochrona danych przed ich ujawnieniem jest realizowana z wykorzystaniem między innymi takich szyfrów jak RSA⁵ i TDES⁶, natomiast przed aktywną ingerencją zabezpieczają wybrane funkcje skrótu – SHA-1 (ang. *Secure Hash Algorithm*)⁷ i MD5 (ang. *Message Digest Algorithm*)⁸. Algorytmy SHA i MD5 pozwalają też zabezpieczać przed przechwytywaniem pakietów i powtarzaniem transakcji. Dodatkowo w pakiecie udostępnia się mechanizmy uwierzytelniające dla kontroli komunikujących się stron (klienta i serwera).

Do bezpiecznego uwierzytelniania systemy zarządzania bazami danych stosują też inne standardy jak na przykład protokół Kerberos (dostępny np. w Oracle czy My-SQL). Znajduje on zastosowanie w dużych systemach z wieloma serwerami, gdy istnieje potrzeba korzystania z rozproszonych zasobów. Umożliwia weryfikację tożsamości komunikujących się stron (klienta i serwera), a opcjonalnie zapewnia również integralność i poufność przesyłanych danych. Kerberos posiada jednak pewne wady, a jego skuteczność będzie zależała od długości, złożoności i okresu ważności hasła [MazKra 2006]. W powszechnej opinii system Kerberos jest przeznaczony do pracy w środowisku liczącym niezbyt wielu użytkowników i nie sprawdza się w systemach wielodostępnych, co jest spowodowane przeciążeniem serwera uprawnien. Wiele problemów wiąże się też z koniecznością przechowywania kluczy i biletów oraz faktu, że uwierzytelnianie opiera się na synchronizacji zegarów w sieci [Schn 2002]. W środowisku wielodostępnym system Kerberos narażony jest na niebezpieczeństwa związane z obecnością innych użytkowników w systemie.

⁵ Opublikowany w 1977 roku standard szyfrowania asymetrycznego dwukluczowego.

⁶ Triple DES – dane przekształcane z wykorzystaniem standardu szyfrowania DES trzykrotnie przy pomocy dwóch kluczy, co zwiększa długość klucza szyfrującego do 128 bitów.

⁷ Algorytm skrótu o 160-bitowym ciągu wyjściowym; nowe funkcje: SHA-256, SHA-512.

⁸ Funkcja mieszająca dająca w wyniku 128-bitowy skrót.

Moc stosowanych zabezpieczeń kryptograficznych między innymi zależy od poufności wykorzystywanych kluczy. Do bezpiecznej ich dystrybucji na przykład pakiet Oracle Advanced Security wykorzystuje algorytm Diffiego–Hellmana, w którym niepowtarzalne klucze generowane są dla każdej nowej sesji.

Ustanawianie szyfrowanych połączeń (stosowanych na przykład w różnego rodzaju instytucjach finansowych) to skomplikowane działania podejmowane dla zapewnienia bezpieczeństwa realizowanej usługi, łączące ze sobą mechanizmy kryptograficzne i odpowiednie procedury organizacyjne. Bazy danych nie zawsze posiadają wbudowane mechanizmy zapewniające ochronę przesyłanych danych. W środowisku bazodanowym do bezpiecznej transmisji danych można wykorzystywać dostępne protokoły, takie jak: SSH (ang. *Secure Shell*), SSL (ang. *Secure Socket Layer*)⁹, którego następcą jest TLS (ang. *Transport Layer Security*) czy protokół IPsec. Protokół SSH¹⁰ pozwala szyfrować dane przesyłane pomiędzy bazą danych a klientem, natomiast SSL opracowany przez NCC (ang. *Netscape Communications Corporation*), służy do kompleksowego zabezpieczania połączeń zapewniając zarówno szyfrowanie i integralność danych, jak i uwierzytelnianie stron. Na przykład w środowisku bazodanowym Oracle można go stosować do uwierzytelniania klienta lub serwera przy dostępie do jednego lub wielu serwerów oraz dowolnego serwera dla pewności klienta, że łączy się on z żądanym serwerem.

IPsec stanowi szkielet standardów, które zostały wykorzystane do stworzenia bezpiecznego zestawu protokołów uruchamianych z wykorzystaniem protokołu IP. IPsec składa się z trzech protokołów służących odpowiednio uwierzytelnianiu, szyfrowaniu danych i wybranych pól nagłóweków pakietów (dowolnym algorytmem) oraz do negocjacji metody i parametrów szyfrowania, a także autoryzacji. Protokoły te mogą być zaimplementowane w dowolnym urządzeniu wykorzystującym do komunikacji protokoły IP. IPsec może być również stosowany w połączeniu z protokołami tunelowania działającymi w warstwie drugiej modelu OSI.

5. ZAGROŻENIA BEZPIECZEŃSTWA SIECI BEZPRZEWODOWYCH

Sieci bezprzewodowe są znacznie bardziej narażone na zagrożenia niż sieci kablowe. Dostęp do nich jest znacznie łatwiejszy (sygnał przenoszony w otwartej przestrzeni), w związku z czym łatwiej jest sieć podsłuchiwać (jeśli strumień danych nie jest szyfrowany), łatwiej jest zakłócić pracę takiej sieci, natomiast prawdopodobieństwo wykrycia niepożądanego ingerencji jest minimalne.

⁹ Pierwsza wersja pojawiła się w 1994 roku, w 1996 opublikowano TLS.

¹⁰ SSH-1 stworzony przez helskiego naukowca w 1995 roku; poprawiona wersja SSH-2 w 1996 roku. Open SSH – otwarta implementacja protokołu, w 1999 roku.

Wśród nielegalnych działań można wymienić przede wszystkim:

- podszywanie się pod uprawnionego użytkownika (np. poprzez przejęcie jego adresu MAC),
- instalowanie punktu dostępowego bez zezwolenia dla przechwytywania ruchu celem podsłuchu lub ingerencji w strumień danych,
- bezprawne odszyfrowywanie danych,
- zakłócanie pracy urządzeń sieciowych,
- ataki „man-in-the-middle”,
- zablokowanie sieci (w skrajnym przypadku).

Szczególny problem stanowi łatwość dostępności sieci w warstwie fizycznej i łącza danych (w I i II warstwie modelu OSI/ISO). Atak w warstwie pierwszej polega na generowaniu szumu zakłócającego pracę sieci, w warstwie drugiej może przebiegać w postaci przesyłania niepoprawnych ramek, które mogą skutecznie paraliżować pracę urządzenia odbiorczego. Specyfika technologii bezprzewodowej jest ważną, ale nie jedyną przyczyną zagrożeń. Sieć bezprzewodowa narażona jest na zagrożenia również z powodu błędów instalacji i konfiguracji sieci. Ich lista jest długa, wystarczy wymienić podstawowe błędy, do których należą:

- używanie zbyt prostej nazwy sieci SSID (ang. *Service Set Identifier*),
- nieukrywanie identyfikatora sieci,
- pozostawienie punktów dostępowych bez haseł (stwarza to np. możliwość modyfikacji listy ACL (ang. *Access Control List*),
- podłączanie punktów dostępu do sieci lokalnej bez użycia zapory sieciowej,
- brak kontroli adresów MAC,
- brak kontroli obszaru łączności,
- niestosowanie najmocniejszych dostępnych zabezpieczeń (z powodu braku kompatybilności albo spowolnienia pracy sieci).

Każdy producent ustawia takie same opcje domyślne dla wszystkich egzemplarzy swojego sprzętu i przy pierwszym połączeniu z siecią należy to zmienić. Z badań wynika, że ok. 10% administratorów nie zmienia domyślnego identyfikatora sieci, a nazwa sieci powinna być traktowana jak hasło dostępu spełniając wszystkie jego wymogi. W momencie konfiguracji rutera trzeba podać nową nazwę i hasło użytkownika. Okazuje się, że jedynie 10% sieci nie rozgłasza swego identyfikatora. Nieukrywanie nazwy jest błędem, choć można ją oczywiście wykryć (dla wielu użytkowników już to stanowi barierę). Wiele ruterów oferuje możliwość nie odpowiadania na sygnał ping¹¹, zatem dla bezpieczeństwa warto z tego skorzystać. Duży problem stanowi odpowiedni dobór mocy sygnału. Stosowanie sygnału o zbyt dużej mocy zwiększa ryzyko podłączenia się osób nieupoważnionych. Sygnał radiowy trudno jest ograniczyć do wyznaczonego obszaru, a przy tym atakujący sieć zwykle dysponują mocnymi antenami umożliwiającymi wykrycie nawet najsłabszego sygnału. W niektórych rozwiązaniach

¹¹ *Packed Internet Groper* – program diagnostyczny do kontroli dostępności urządzeń w sieci.

wystarczy ruter umieścić w pomieszczeniu centralnie, by sygnał nie wychodził poza jego granice.

Inne przyczyny zagrożeń to:

- słabości standardów bezpieczeństwa,
- brak kompatybilności z powodu różnorodności stosowanego sprzętu i standardów (co wymusza rezygnację ze stosowania dostępnych mocnych zabezpieczeń),
- dostępność specjalistycznych narzędzi do wyszukiwania niezabezpieczonych sieci (zbierających szczegółową informację na ich temat, z podaniem lokalizacji punktów dostępowych dzięki możliwości współpracy z GPS (ang. *Global Positioning System*)¹²),
- dostępność narzędzi do przełamывania stosowanych zabezpieczeń kryptograficznych (często wykorzystujących ich słabości, co przyspiesza skuteczność ich działania).

Istotnym czynnikiem mającym wpływ na bezpieczeństwo jest także tzw. czynnik ludzki. Wiąże się z tym wszelkie zaniedbania administratorów i obsługi systemu wynikające z braku wiedzy z zakresu bezpieczeństwa (na temat realnych zagrożeń i możliwości ochrony) oraz z nieprzestrzegania zasad bezpieczeństwa ustalonych stosowną polityką.

6. ZABEZPIECZANIE SIECI W PRAKTYCE

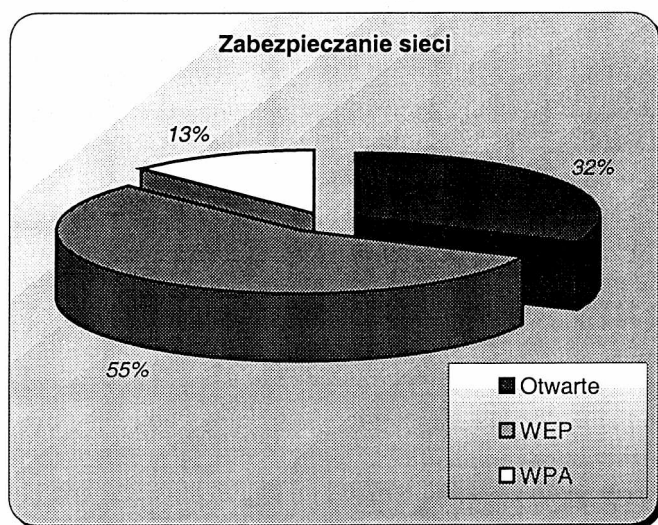
Firma netSecurity na zlecenie RSA Security przeprowadziła w 2005 roku badania, z których wynikało, że 1/3 korporacyjnych sieci w wybranych metropoliach (w Nowym Jorku, San Francisco, Londynie i Frakfurcie) była podatna na ataki z zewnątrz. Ponownie podejmowane badania w tym zakresie jedynie potwierdzały słabość zabezpieczeń sieci bezprzewodowych.

Podobny eksperyment został przeprowadzony we Wrocławiu w 2007 roku. W tym celu wykorzystano laptop wyposażony w program airodump-ng, który między innymi pozwala stwierdzić, czy dana sieć jest otwarta, czy nie, oraz określić rodzaj zabezpieczeń w przypadku rejestrowanego ruchu. Wyniki uzyskane w efekcie podjętych badań przedstawiono na rysunku 1.

We Wrocławiu, podobnie jak w innych miastach świata, około 1/3 sieci bezprzewodowych to sieci otwarte – nie stosujące żadnych zabezpieczeń, zaś jedynie 13% sieci stosuje mocne dostępne zabezpieczenia w postaci WPA. Większość użytkowników wybiera słabe, ale nie spowalniające pracy sieci zabezpieczenia, których dodatkową zaletą jest prostota użytkowania. Informacje dostarczane przez program airodump-ng to między innymi: identyfikator sieci (jeśli nie jest ukryty), numer kanału transmisyjnego, prędkość przesyłu oraz informacja o stosowanych

¹² System nawigacji satelitarnej obejmujący zasięgiem całą kulę ziemską.

zabezpieczeniach (czy sieć jest otwarta czy nie). Do określenia, który z systemów bezpieczeństwa jest używany, WEP czy WPA, potrzebny jest ruch na łączu transmisyjnym. W przypadku braku transmisji w kanale, uzyskane wyniki można przybliżyć przyjmując w przypadku braku strumienia danych, jednakowe proporcje pomiędzy stosowanymi systemami bezpieczeństwa, jak to odnotowano w sytuacjach, gdy ruch na łączu dało się zarejestrować. Potwierdza się, iż około 10% sieci bezprzewodowych nie ma zmienionego identyfikatora i posługuje się domyślną nazwą producenta (np. default, wireless, WLAN, linksys, neostrada). Z kolei niektóre z wprowadzonych identyfikatorów wskazują na konkretną firmę (np. SSK-Gazoprojekt, WBZ), co może zachęcać intruzów do podjęcia ataku na zasoby danej sieci.



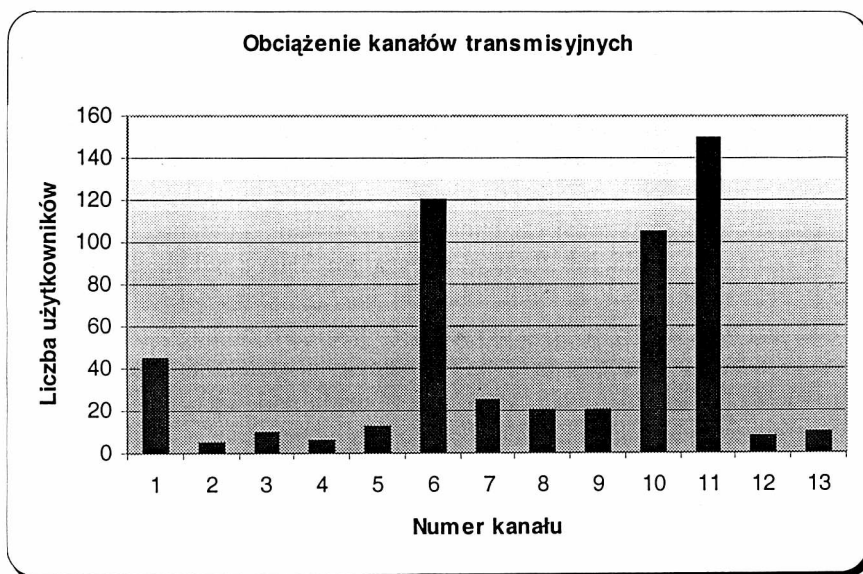
Rys. 2. Zabezpieczanie sieci bezprzewodowych

Zebrane przez program airodump-ng informacje można również wykorzystać do formułowania wniosków na temat praktycznie wykorzystywanych kanałów transmisyjnych. Dostępne w ramach standardu pasmo jest bowiem podzielone na niezależne podkanały, dzięki zastosowaniu częstotliwościowego zwielokrotnienia. Oznacza to, że wiele fal radiowych o różnych długościach tworzy wiele łącz radiowych mogących jednocześnie ze sobą współistnieć. Każdy odbiornik dostraja się do określonej częstotliwości, jednak sąsiadujące kanały mają tendencję do przesłuchów. Średnie częstotliwości kanałów radiowych zgodnych z najczęściej wykorzystywanymi standardami 802.11b i 802.11g do realizacji lokalnych sieci bezprzewodowych zestawiono w tabeli 2.

Tabela 2. Średnie częstotliwości kanałów radiowych zgodnych ze standardami 802.11b i 802.11g

Numer kanału	1	2	3	4	5	6	7	8	9	10	11	12	13
Średnia częstotliwość [MHz]	2412	2417	2422	2427	2432	2437	2442	2447	2452	2457	2462	2467	2472

Jak wskazują wyniki przeprowadzonego badania, powszechnie stosowane są kanały domyślnie ustawiane przez producentów, czyli kanały 1, 6 i 11 oraz kanał 10 wykorzystywany przez TP S.A. dla realizacji usługi Neostrada (rysunek 2). Kanały nie są zatem obciążone równomiernie, co prowadzi do pogorszenia parametrów transmisji i pośrednio ma wpływ na bezpieczeństwo sieci.



Rys. 3. Wykorzystanie dostępnych kanałów w transmisji bezprzewodowej

Program airodump-ng służy do przechwytywania pakietów, natomiast do ich przetwarzania w celu złamania klucza WEP można wykorzystać program aircrack-ng. Narzędzi realizujących podobne zadania jest udostępnianych więcej, dla różnych systemów operacyjnych. W Internecie publikowane są artykuły, których autorzy udowadniają, iż podjęte działania mogą przynieść pożądany efekt w postaci ujawnienia klucza już w ciągu kilkunastu czy kilkudziesięciu minut, zależnie od obciążenia sieci, gdyż czas osiągnięcia sukcesu jest odwrotnie proporcjonalny do liczby przechwyconych pakietów. W wyniku tak dokonanego ataku znajomość klucza oraz nazwy sieci

umożliwia nieuprawnionemu użytkownikowi dostępność danej sieci bezprzewodowej, mimo stosowanych zabezpieczeń.

Dla zwiększenia skuteczności łamania kryptograficznych zabezpieczeń sieci bezprzewodowych można generować ruch w sieci celowo. Wystarczy do tego wykorzystać dowolne gotowe narzędzie, na przykład program Void11 pozwalający wprowadzać trzy rodzaje ramek: uwierzytelniającą, zrywającą uwierzytelnienie oraz kojarzącą. Atak z użyciem celowo generowanych ramek może również prowadzić do zablokowania punktu dostępowego innym użytkownikom (tzw. atak DoS – *Denial of Services*).

7. PODSUMOWANIE

Sieci bezprzewodowe nie są bezpieczne – najczęściej chroni je WEP (mimo wszelkich słabości, ze względu na prostotę i łatwość konfiguracji) lub w ogóle nie są zabezpieczane.

Korzystanie z baz danych w rozproszonych systemach wykorzystujących technologię bezprzewodową staje się jeszcze większym ryzykiem dla ich bezpieczeństwa niż w przypadku sieci konwencjonalnych. Dlatego też już na etapie wdrażania i konfiguracji sieci bezprzewodowych należy poświęcić szczególną uwagę konfiguracji dostępnych mechanizmów bezpieczeństwa. Niestety, stosowane zabezpieczenia kryptograficzne, takie jak szyfr TDES lub funkcje mieszające SHA i MD5, często stosowane w pakietach ochronnych baz danych, obecnie nie gwarantują wysokiego poziomu ochrony, jaki jest wymagany w środowiskach o szczególnym zagrożeniu. Także protokoły wykorzystywane dla zapewnienia bezpiecznych połączeń pomiędzy klientem a serwerem nie są pozbawione wad, które umożliwiają podejmowanie różnych rodzajów ataków, nierzadko z wykorzystaniem gotowych narzędzi.

Z powodu narastającego zagrożenia sieciowego, szczególnie w środowisku sieci bezprzewodowych, pojawia się potrzeba wprowadzenia nowych mocniejszych mechanizmów kryptograficznych. Jest to pożądane także z uwagi na wzrost prędkości transmisji w nowo wprowadzanych technologiach, takich jak MIMO i WiMaX mających na celu poprawę parametrów transmisji sieci bezprzewodowych.

LITERATURA

- [BaMiKriSri 2005] Balinsky A., Miller D., Kriszna A., Sri S., *Bezpieczeństwo sieci bezprzewodowych. Ochrona sieci 802.11 – porady eksperta*. MIKOM, Warszawa 2005.
- [EdAr 2003] Edney J., Arbaugh W., *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*. Addison Wesley, 2003.
- [EnFle 2005] Engst A., Fleishman G., *Sieci bezprzewodowe. Praktyczny przewodnik*. 2005.

- [MazKra 2006] Mazur Z., Mendyk-Krajewska T., *Analiza stanu plików dla wzrostu bezpieczeństwa systemów bazodanowych*. W: Bazy danych. Struktury, algorytmy, metody. Praca zbiorowa pod red. S. Kozielskiego [i in. T. 2]. Wybrane technologie i zastosowania. WKŁ, Warszawa 2006.
- [PotFle 2004] Potter B., Fleck B., *802.11. Bezpieczeństwo*. Helion, Gliwice 2004.
- [Schn 2002] Schneier B., *Kryptografia dla praktyków. Protokoły, algorytmy i programy źródłowe w języku C*. WNT, Warszawa 2002.
- [VlaGavMik 2005] Vladimirov A., Gavrilenko K. V., Mikhailovsky A. A., *Wi-Foo. Sekrety bezprzewodowych sieci komputerowych*. Helion, Gliwice 2005.
- [Wiki] www.pl.wikipedia.org

SECURITY OF DISTRIBUTED DATABASE SYSTEMS IN WIRELESS NETWORKS

This paper concerns the problem security of distributed database systems in wireless networks.

**Wydawnictwa Politechniki Wrocławskiej
są do nabycia w księgarni
„Tech”
plac Grunwaldzki 13, 50-377 Wrocław
budynek D-1 PWr., tel. (0-71) 320-29-35
Prowadzimy sprzedaż wysyłkową**

ISBN 978-83-7493-376-6