POLITECHNIKA WARSZAWSKA

WYDZIAŁ MATEMATYKI
I NAUK INFORMACYJNYCH

PRACA DYPLOMOWA MAGISTERSKA
INFORMATYKA

# Adaptacyjne zachowania agentów programowych w e-commerce

Autor: Michał Drozdowicz

Promotor: dr Marcin Paprzycki

WARSZAWA STYCZEŃ 2008

…… ……

podpis promotora podpis autora

# Spis treści

# 1 Wstęp

Niniejsza praca traktuje o adaptacyjnych właściwościach agentów programowych działających w wieloagentowym systemie e-commerce. Opisane zostaną podstawowe zagadnienia związane z e-commerce, zwłaszcza pod kątem perspektyw rozwoju tej gałęzi gospodarki oraz automatyzacji pewnych procesów sprzedaży. Zostanie także przedstawiony modelowy wieloagentowy system e-commerce E-CAP. W opisie tym szczególne miejsce zajmie omówienie tych agentów, w przypadku których można mówić o dostosowywaniu się do zmian środowiska oraz analiza tych procesów adaptacyjnych.

W części praktycznej pracy zostaną zaproponowane procesy zbierania i przechowywania wiedzy oraz projekt agenta decyzyjnego działającego w ramach struktury sklepu, który korzystając ze zgromadzonych danych zdolny będzie do modyfikacji swoich działań w odpowiedzi na zmieniające się warunki. W oparciu o tę platformę zostaną zaimplementowane dwa procesy decyzyjne wspierające działania sklepu: prognozowanie sprzedaży produktów w celu dynamicznego uzupełniania stanów magazynowych, oraz modyfikację parametrów sprzedaży w celu zwiększania zysku sklepu.

## *1.1 E-commerce*

### 1.1.1 Definicja

E-commerce (electronic commerce, handel elektroniczny), w swoim najszerszym znaczeniu określa całość procedur i działań wykorzystujących środki i urządzenia elektroniczne (przede wszystkim sieci komputerowe) w celu zawarcia transakcji finansowej, współdzielenia danych biznesowych oraz utrzymywania związków handlowych [8]. Termin ten, którego pierwsze użycie przypisywane jest Kalakocie i Whinstonowi [4] w roku 1996, opisuje więc metody wykorzystywane już od lat 80. pod postacią chociażby EDI (Electronic Data Interchangeable, standard elektronicznej wymiany danych) czy elektronicznego przepływu pieniędzy. Po latach rozwoju technik komputerowego wspomagania biznesu, e-commerce obejmuje swoim zasięgiem bardzo szerokie pole działań handlowych – od zarządzania łańcuchem dostaw i

relacjami z klientem przez realizowanie transakcji bezgotówkowych do zaawansowanych systemów księgowania.

Najpopularniejsze jednak znaczenie pojęcia e-commerce stosowane dziś jest znacznie węższe od przedstawionego powyżej, choć oczywiście często wykorzystuje wiele z wymienionych procesów – chodzi o handel elektroniczny jako sprzedaż towarów i usług przez Internet. To właśnie ten kontekst interesuje nas w niniejszej pracy.

### 1.1.2 Sprzedaż internetowa a tradycyjna

To, czym handel elektroniczny nie różni się na pewno od tradycyjnego, to fakt, że oba opierają się na tych samych fundamentalnych prawach podaży i popytu. W związku z tym, kupiec internetowy, który nie potrafi sprostać wymaganiom klienta jednocześnie maksymalizując swój zysk jest tak samo skazany na porażkę jak kupiec tradycyjny, o czym boleśnie przekonał się rynek podczas upadku „dotcomów" w 2000 roku.

Poza tym podstawowym podobieństwem, te dwa modele biznesowe różnią się pod wieloma względami. To, co chyba najbardziej wyróżnia handel elektroniczny, to jego globalna dostępność – w przeciwieństwie do tradycyjnych sklepów, otwartych zazwyczaj w określonych godzinach i fizycznie osiągalnych dla klientów mieszkających w ich okolicy, sklepy internetowe są „czynne" 24 godziny na dobę i są otwarte dla każdego klienta posiadającego dostęp do Sieci niezależnie od jego położenia geograficznego. Oczywiście należy tu nadmienić, że w większości przypadków wysyłka zakupionego przez internet towaru i tak realizowana jest w normalnych godzinach pracy, i nie zawsze istnieje możliwość dostarczenia zakupu w najdalsze zakątki globu, ale klient może zawsze zapoznać się ze szczegółową ofertą sklepu.

Także liczba różnych sprzedawanych produktów może być w przypadku sklepu internetowego znacznie większa – ograniczeniem jest jedynie powierzchnia magazynowa, nie ma natomiast problemu z prezentacją sprzedawanego asortymentu. W wielu przypadkach mamy nawet do czynienia ze sprzedażą dóbr częściowo wirtualnych – sprowadzanych z odległych hurtowni na życzenie klienta. Taką strategię objął m.in. Empik podpisując w maju 2007 umowę z dużą niemiecką hurtownią Libri, powiększając tym samym swoją ofertę o 2 mln. nowych pozycji ( wcześniej oferował w sumie około 200 tys. tytułów). [24]. Obfitość

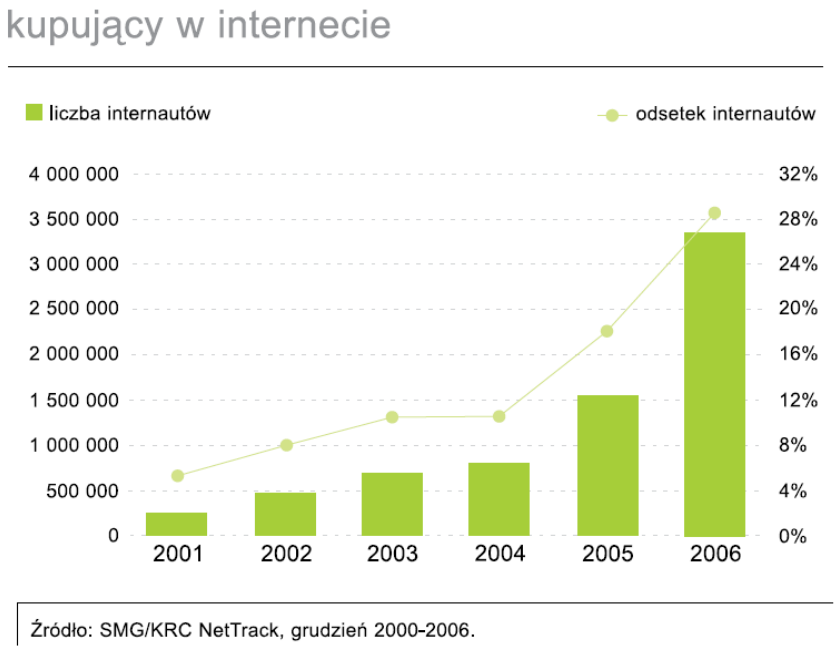
oferty sklepu elektronicznego jak i jego dostępność dla klientów oddalonych geograficznie pociągają za sobą też problemy. W przypadku handlu internetowego system logistyczny musi być znacznie bardziej zaawansowany niż w tradycyjnych sieciach sprzedaży – każda przesyłka musi być odpowiednio zabezpieczona i dostarczona bezpośrednio do klienta, co pociąga za sobą zwiększone koszty dostawy i potrzebę zarządzania stanem wielu dostaw jednocześnie. Kolejnym problemem, z którym borykają się sklepy internetowe jest brak możliwości fizycznego kontaktu klienta z towarem. Z tego też powodu opisy oferowanych produktów są często bardzo rozbudowane a przez internet sprzedawane są najczęściej artykuły, które w każdym sklepie są takie same (książki, muzyka, rtv, perfumy), a więc możliwe do zobaczenia wcześniej chociażby w sklepie tradycyjnym.

Jeszcze innym aspektem sprzedaży internetowej jest znacznie łatwiejsze niż w przypadku sprzedaży tradycyjnej porównywanie cen poszczególnych produktów pomiędzy sklepami – klient może nie wychodząc z domu sprawdzić atrakcyjność oferty dziesiątków sklepów. Zwiększa to oczywiście konkurencyjność i powoduje, że oprócz ceny, której poniżej pewnego progu obniżyć nie można, liczy się także wygoda obsługi, jakość i bezpieczeństwo transakcji.

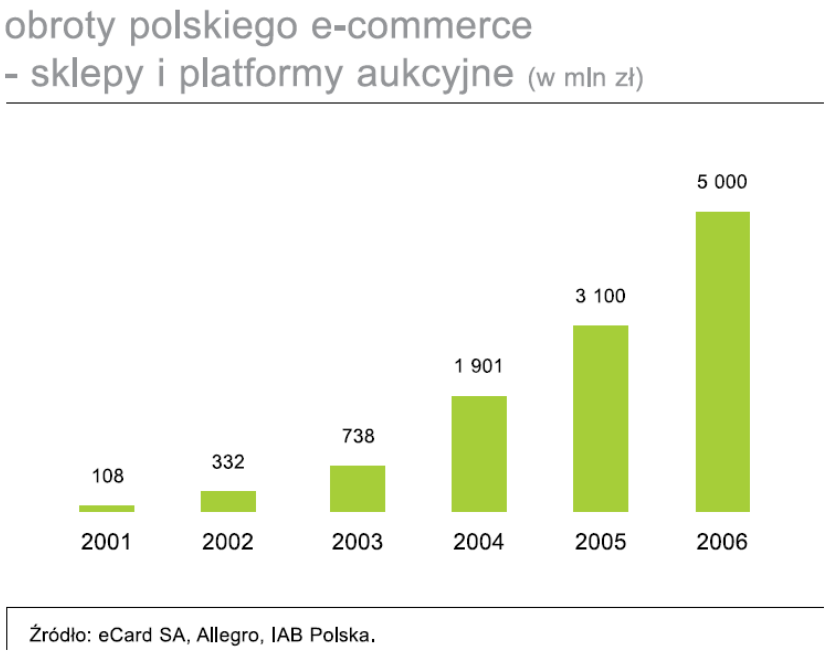
Na koniec pozostawiłem cechę handlu internetowego, która w dalszym ciągu nie jest zbyt mocno wykorzystywana przez istniejące firmy (w szczególności w Polsce), ale niesie ze sobą bardzo atrakcyjne perspektywy na przyszłość – łatwość zbierania informacji na temat zachowania i zwyczajów konsumenckich klientów. Cel, który większe tradycyjne sieci sklepowe próbują realizować przy pomocy programów partnerskich i kart stałego klienta – czyli autentykacja klienta i budowanie jego profilu, w przypadku handlu internetowego jest właściwie efektem ubocznym procesu sprzedaży. Tego typu informacje, po odpowiednim przetworzeniu, mogą stać się wartościowym źródłem wiedzy zarówno o zachowaniu konkretnego klienta, co może być wykorzystane do lepszej personalizacji treści serwisu, jak i zwyczajów całych grup klientów – w celu wyciągania wniosków chociażby o skuteczności akcji promocyjnych lub o zależnościach między produktami (analiza koszyka zakupów). Oprócz tego, co nie mniej istotne, dane o transakcjach pozwalają wnioskować o trendach sprzedaży towarów, dzięki czemu łatwiej jest zarządzać stanem magazynowym. Przetwarzanie tych danych wymaga jednak odpowiedniej infrastruktury analitycznej. Jak pokazują wyniki ankiety [13] większość polskich sprzedawców internetowych nie wykorzystuje w pełni danych o klientach i transakcjach, zarówno w przypadku personalizacji serwisu jak i wyodrębnianiu grup docelowych.

### 1.1.3 Wielkość e-commerce i perspektywy rozwoju

Wg raportu IAB [30] pod koniec 2006 roku 29% polskich internautów robiło zakupy przez internet, a obroty polskiego handlu elektronicznego wyniosły 5 mld zł.




*Rys. 1. Wykres ilości kupujących w Internecie w kolejnych latach*




*Rys. 2. Obroty polskiego e-commerce w kolejnych latach*

W USA natomiast w 2006 roku obrót e-commerce przekroczył 110 mld dolarów, co oznacza ponad dwudziestoprocentowy wzrost w stosunku do roku poprzedniego w porównaniu do 4% wzrostu całkowitych obrotów amerykańskiego handlu [31]. Widać więc, że dynamika rozwoju sektora e-commerce zarówno w Polsce jak i na świecie nie maleje.

### 1.1.4 Platformy aukcyjne

Oprócz sklepów internetowych w krajobraz współczesnego e-commerce bardzo silnie wpisane są platformy aukcyjne. Są to serwisy internetowe pełniące jedynie rolę pośrednika pomiędzy użytkownikami sprzedającymi i kupującymi towary i usługi. Sprzedaż odbywa się najczęściej pod postacią negocjacji cenowej, a najpopularniejszym typem aukcji są aukcje angielskie, w których początkowo niska cena startowa jest podbijana przez kolejnych licytujących. Atrakcyjny dla użytkowników model kupna/sprzedaży oraz często nawet niższe niż w sklepach internetowych ceny spowodowały w ostatnich latach wykładniczy wzrost obrotów najważniejszych platform aukcyjnych. W 2006 roku Allegro – największy tego typu serwis w Polsce zanotował 2,5 mld złotych obrotów, co stanowi połowę obrotów całego polskiego sektora e-commerce [30]. W tym samym czasie globalny gigant aukcyjny – eBay osiągnął prawie 6 mld dolarów obrotów [25].

### 1.1.5 Automatyzacja i usprawnienia

Dosyć istotne w kontekście niniejszej pracy wydaje się być pojawienie się w internecie, także w Polsce, serwisów wspomagających klienta w kupnie interesujących go towarów. Najlepiej jest to widoczne w przypadku serwisów zbierających informacje z wielu sklepów internetowych i umożliwiających porównywanie cen danego produktu. Dzięki temu klient nie musi nawet odwiedzać stron poszczególnych sklepów – porównywarka od razu wyświetla ich listę. W Polsce działa w tej chwili wiele takich serwisów, najważniejsze z nich to Skąpiec, Ceneo i Nokaut [38, 36, 37]. Zwłaszcza ten ostatni jest interesujący, gdyż umożliwia także wyszukiwanie atrakcyjnych ofert przy pomocy telefonu komórkowego z obsługą WAP. Dzięki temu, klient oglądający produkt w tradycyjnym sklepie, może od razu sprawdzić ile mógłby zaoszczędzić dokonując zakupu w sklepie internetowym.

Innym sposobem na ułatwienie klientowi życia, tym razem przy korzystaniu z aukcji internetowych, są serwisy automatyzujące licytację. Serwisy takie jak Bidnapper [35], czy polski Snajper [39] pozwalają na automatyczne złożenie oferty w ostatnich sekundach trwania aukcji, dzięki czemu użytkownik, który nie chce wcześniej podbijać ceny produktu nie musi też do ostatniej chwili obserwować przebiegu negocjacji.

Można więc zauważyć rosnący trend jeśli chodzi o przenoszenie obowiązków człowieka w zakresie sprzedaży w internecie na systemy komputerowe. W ninejszej pracy rozpatrywany będzie temat programów, które w imieniu swoich użytkowników podejmują większość możliwych do automatyzacji działań dotyczących wymiany handlowej w rozproszonym środowisku e-commerce.

## 1.2 Agenci programowi

Pojęcie agenta programowego nie doczekało się jednej powszechnie uznanej za obowiązującą definicji. W wielu publikacjach termin ten definiowany jest w różny sposób, często rozbieżny od innych. Kilka popularnych definicji znaleźć można w pracy [29] oraz wcześniejszej [7]. Autorzy tej ostatniej publikacji podają także własną definicję: *Agent to system stanowiący część otoczenia, w którym jest umieszczony, który jest świadom swojego otoczenia i oddziałuje na to otoczenie reagując na sygnały z niego płynące, dążąc do realizacji swoich celów* (tł. polskie za [27]).

Agent często określany jest także poprzez pryzmat cech jakimi powinien się odznaczać. Do tych, które wymieniane są jako obowiązkowe, aby program nimi się charakteryzujący mógł być uznawany za agenta należą: autonomiczność (zdolność podejmowania samodzielnych decyzji), komunikatywność (umiejętność komunikacji z innymi agentami i użytkownikiem), percepcja i reaktywność (zdolność do postrzegania i reagowania na zmiany środowiska) [22]. Poza wymienionymi cechami, podaje się wiele innych, którymi można opisać agentów, ale ich występowanie zależy już od konkretnego przypadku. Różni agenci mogą więc być opisane niektórymi z następujących cech: umiejętność dopasowywania się (adaptacyjność), umiejętność współdziałania, rozumowanie w oparciu o zgromadzoną wiedzę, umiejętność przemieszczania się, umiejętność przewidywania, inteligencja [29].Wiele przykładów agentów podawanych jako uzupełnienie dyskusji o istocie tego pojęcia, to jednostki

wypełniające dosyć złożone zadania, komunikujące się w większości z użytkownikiem końcowym. Można tu podać takie przykłady jak agent organizujący czas użytkownika lub wyszukujący w sieci informacje, które mogą go zainteresować. Tworzenie agentów programowych może być jednak widziane jako coś więcej – jako metodologia projektowania oprogramowania – zwłaszcza systemów złożonych [29], [10]. Projektowanie komponentów systemu w postaci agentów może być rozważane jako rozszerzenie obiektowej metodologii tworzenia oprogramowania, spełniające fundamentalne założenia, czyli postulaty dekompozycji, abstrakcji i organizacji [1]. Cytując [29] *(a) wyrażenie problemu w terminach niezależnych agentów jest efektywną metodą podziału problemu (abstrakcja), (b) podstawowe abstrakcje związane z podejściem agentowym są naturalnym sposobem modelowania skomplikowanych problemów (dekompozycja), (c) podejście agentowe do modelowania i zarządzania organizacją i oddziaływaniami pomiędzy komponentami systemu jest właściwym sposobem modelowania zależności istniejących w dużych problemach i jest też najlepszym podejściem do ich implementacji (organizacja).*

Siła agentowego podejścia objawia się także w łatwej możliwości rozproszenia tak zbudowanego systemu – ponieważ poszczególne komponenty są autonomiczne i jedynie komunikują się i współpracują ze sobą, to bez problemu mogą być umieszczone na różnych maszynach, co może mieć istotne znaczenie w przypadku systemów ciężkich obliczeniowo i/lub wymagających wysokiej dostępności. Z kolei mobilność agentów, czyli zdolność migrowania z jednej maszyny na inną, wraz ze swoim „programem" i stanem wewnętrznym, pozwala na operowanie w silnie rozproszonych środowiskach, w których połączenia między węzłami sieci nie są zbyt niezawodne. Taka sytuacja ma miejsce w przypadku sieci Internet, gdzie fizyczna odległość między połączonymi komputerami powoduje często spore opóźnienia w komunikacji. W tym przypadku implementacja na przykład programu automatycznie prowadzącego licytację na aukcji internetowej nie jest proste, gdyż nie mamy możliwości zagwarantowania, że nasza oferta dotrze do serwera aukcyjnego przed zakończeniem negocjacji. Rozwiązanie oparte na mobilnych agentach, którzy przed rozpoczęciem negocjacji migrują na serwer aukcyjny, a następnie lokalnie uczestniczą w licytacji jest pozbawione tych problemów. Co więcej, w takim scenariuszu, klient nie musi nawet być podłączony do sieci w trakcie trwania aukcji – rozwiązanie to nadaje się więc do zastosowania także w przypadku urządzeń, których połączenie z siecią jest złej jakości, lub dostępne tylko okresowo, jak np. telefonów komórkowych, czy urządzeń mobilnych.

## 1.3 Java Agent DEvelopment Framework

Od rozpoczęcia badań nad agentami programowymi powstało wiele platform, usprawniających i ułatwiających implementację systemów wieloagentowych. Dziś najpopularniejszą z nich jest JADE (Java Agent DEvelopment Framework, [33], ciągle rozwijany przez TILAB (laboratorium firmy Telecom Italia) w kooperacji m.in. z Motorolą przy wsparciu środowiska Open Source.

JADE jest middleware'em zaimplementowanym w języku Java, wspomagającym tworzenie rozproszonych systemów wieloagentowych W skład pakietu wchodzą: środowisko uruchomieniowe, zwane kontenerem (ang. container); zestaw bibliotek, realizujących podstawowe, powtarzające się operacje oraz narzędzia graficzne do konfiguracji, administracji i monitorowania agentów. JADE realizuje wszystkie cele stawiane platformie agentowej przez standard FIPA (ang. Foundation for Intelligent Physical Agents, [32]): zarządzanie cyklem życia agenta (przy pomocy kontenera); usługi *White pages* i *Yellow pages*, które pozwalają na odkrycie identyfikatora agenta na podstawie jego nazwy lub typów zadań, które potrafi wykonać dla innych jednostek; możliwość przesyłania wiadomości (obsługiwane protokoły, to JAVA-RMI, JICP - własny protokół JADE, HTTP i IIOP); a także wsparcie dla ontologicznie wyrażonej treści przesyłanych wiadomości. JADE umożliwia także migrację agentów do innych kontenerów, nawet jeśli na docelowym systemie nie był obecny kod agenta. Co więcej, pełna zgodność z powszechnie uznanym standardem FIPA sprawia, że agenci działający na platformie JADE mogą współpracować z każdym agentem, spełniającym te same normy. Warto także dodać, że dzięki implementacji w języku Java oraz odseparowaniu kodu użytkownika od operacji niskopoziomowych, udało się uzyskać niezależność platformy od systemu operacyjnego, wersji Javy czy typu protokołu sieciowego. Tworzenie agentów w oparciu o JADE przebiega identycznie niezależnie od tego, czy będą oni działali na platformie J2SE, J2EE czy J2ME.

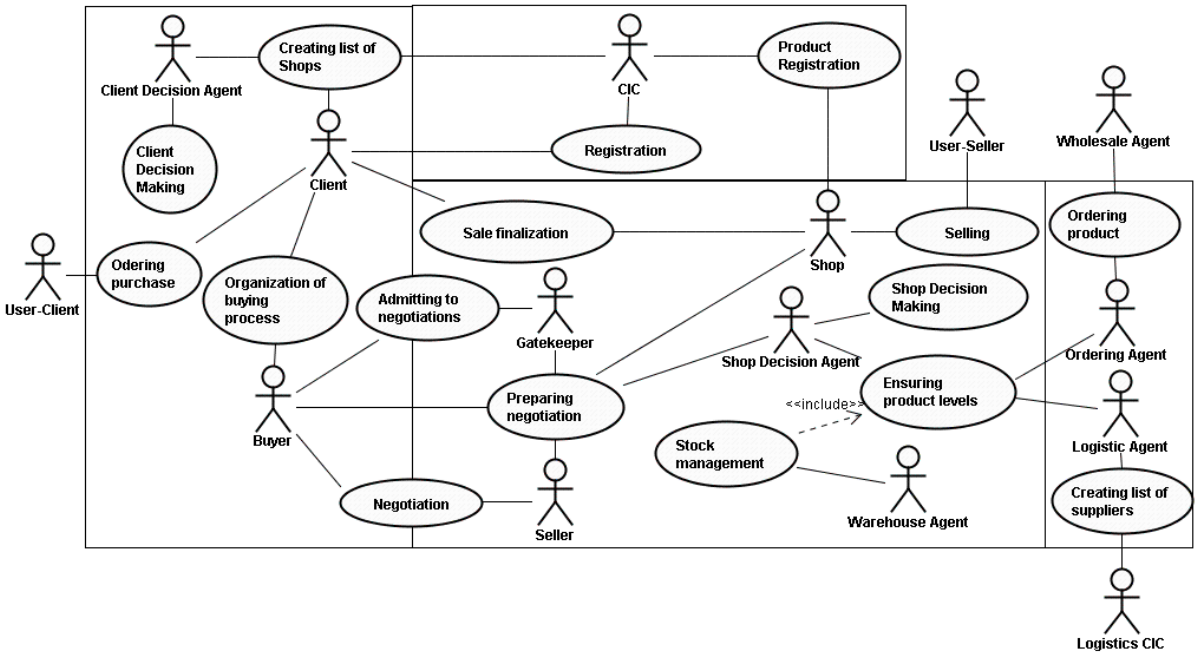
W JADE agent ma postać abstrakcyjnej klasy, po której dziedziczą agenci użytkownika, natomiast swoje cele realizuje dzięki kolejce zachowań (klasy dziedziczące po Behaviour), które są wykonywane „równolegle" – uruchamiane są domyślnie we wspólnym wątku, więc de facto w jednym momencie realizowane jest tylko jedno z nich, nie ma też wywłaszczania działających zachowań. Programista może, natomiast, zatrzymać aktualnie działające zachowanie, powodując uruchomienie innego, czekającego zadania. Komunikacja między

agentami przebiega na zasadzie asynchronicznych wiadomości, z automatycznie obsługiwanym przez kontener kolejkowaniem wiadomości w przypadku, gdy odbiorca nie jest dostępny w momencie wysyłania. To co jest istotne, to fakt, że programista nie musi zajmować się żadnymi szczegółami transmisji wiadomości, tworzy tylko wiadomość, wypełniając odpowiednie właściwości, a następnie zleca agentowi wysłanie jej.

## *1.4 Agentowy system e-commerce*

### 1.4.1 Opis systemu

Systemem, na przykładzie którego zajmę się adaptacyjnymi zachowaniami agentów programowych jest wieloagentowy system e-commerce, E-CAP (E-Commerce Agent Platform), zaprojektowany i rozwijany przez zespół pod kierownictwem dr M.Paprzyckiego i dr M.Ganzha. Poniżej postaram się przybliżyć założenia i architekturę tego systemu w takim stopniu, w jakim jest to konieczne do rozważania tematu mojej pracy. Dokładniejszy opis systemu oraz wnioski dotyczące poszczególnych jego aspektów znaleźć można w takich pracach jak : [11], [17], [16].



*Rys. 3. Diagram agentów i przypadków użycia systemu E-CAP*

System e-commerce, o którym mowa, jest modelem środowiska handlowego, opartym na negocjacjach między agentami. Składa się on z trzech podstawowych elementów: części klienckiej, części sklepowej oraz centrum informacji.

Część kliencka służy za osobistego agenta pomagającemu klientowi dokonać możliwie najbardziej korzystnego zakupu. Część sklepowa pełni funkcje systemu zarządzania wirtualnego sklepu, dbając o obsługę negocjacji, procesów magazynowych i logistycznych, a także starając się maksymalizować zyski sklepu w oparciu o wiedzę zebraną w czasie działania systemu. Trzecia część architektury, centrum informacji (*Client Information Center, CIC*), pełni funkcję katalogu produktów oferowanych przez różne sklepy. Sklep, wprowadzając towar do sprzedaży, rejestruje go w centrum informacji, używanego z kolei przez klientów szukających sklepów, które oferują interesujące ich produkty.

System ten jest w bardzo dużym stopniu autonomiczny – jego użytkownicy pojawiają się na diagramie przypadków użycia jedynie pod postacią *User Seller'a* i *User Client'a*. Założenie jest takie, że użytkownik-klient zleca tylko swojemu agentowi kupno danego produktu, podając maksymalną cenę jaką chce za niego zapłacić – o całą resztę procesu kupna dba agent-klient (*Client Agent, CA*) . Oczywiście użytkownik ma możliwość nadzorowania działań agenta oraz ewentualnej interwencji, natomiast taka interakcja nie jest wymagana dla do działania systemu. Ze strony sklepu, sytuacja wygląda tak, że użytkownik zewnętrzny („właściciel sklepu") podejmuje jedynie najbardziej istotne decyzje, takie jak o wprowadzeniu do sprzedaży nowego towaru lub wycofaniu sprzedawanego, natomiast codzienne działania sklepu, jak na przykład dbanie o poziom produktów w magazynie, wybór odpowiedniej strategii negocjacyjnej czy ceny towarów, podejmowane są autonomicznie przez agenta sklepowego.

### 1.4.2 Scenariusz transakcji

Jak więc wygląda typowy scenariusz transakcji w obrębie zainicjowanego i działającego systemu? Operacja inicjowana jest przez użytkownika-klienta. Przekazuje on agentowi-klientowi (*Client Agent, CA*) opis produktu, który chce kupić. *CA* odpytuje centrum informacyjne o listę sklepów oferujących produkt odpowiadający specyfikacji, a następnie, po

wstępnym przefiltrowaniu otrzymanej listy sklepów na podstawie wiedzy o nich (np. na podstawie zaufania [14, 15]), wysyła do pewnej liczby interesujących kupców instancje agenta kupującego (*Buyer Agent, BA*). *BA* jest lekkim, mobilnym agentem, który migruje na serwer sklepowy aby w imieniu *CA* uczestniczyć w negocjacji. Istnieje także możliwość uczestniczenia w negocjacjach klientów nie dysponujących własnymi agentami *BA* – w takiej sytuacji *CA* zgłasza prośbę o utworzenie w sklepie agenta *BA* mającego go reprezentować. Zanim jednak rozpoczną się negocjacje cenowe, agent klienta jest weryfikowany przez *agenta-bramkarza* (*Gatekeeper Agent*, *GA*) oraz być może musi odczekać pewien czas (potrzebny do zebrania odpowiedniej, dla danego tupu nagocjacji, liczby potencjalnych klientów). Po weryfikacji agentów *BA* przez *GA* następuje także przedstawienie warunków negocjacji cenowych.licytacji. Etap ten zostanie szerzej przedstawiony w sekcji 1.4.3, natomiast ogólnie rzecz biorąc polega on na tym, że agenci *BA* otrzymują od *GA* warunki negocjacji, a następnie proszą swoich *CA* o dostarczenie im strategii działania przy zadanych parametrach negocjacji cenowej. Dopiero gdy *BA* otrzyma stosowne wytyczne, może przystąpić do procesu kupna. Po zakończeniu negocjacji agent *BA* wysyła ich wynik (wygrana lub przegrana, cena, liczba ew. wynegocjowanych sztuk towaru) do agenta-klienta, a *CA* po zebraniu pewnej liczby odpowiedzi (niekoniecznie od wszystkich wysłanych *BA*) podejmuje na podstawie analizy wielu kryteriów decyzję o finalizacji niektórych spośród transakcji rozpoczętych przez *BA*. Jeżeli żadna z otrzymanych ofert nie jest do zaakceptowania, to *CA* może albo poczekać na wynik kolejnych negocjacji, albo odpowiedzieć użytkownikowi, że zakup nie jest możliwy. Agenci *BA*, których oferty zostały wybrane w procesie decyzyjnym otrzymują wiadomość nakazującą im finalizację transakcji, pozostali agenci kupujący zostają natomiast usunięci.

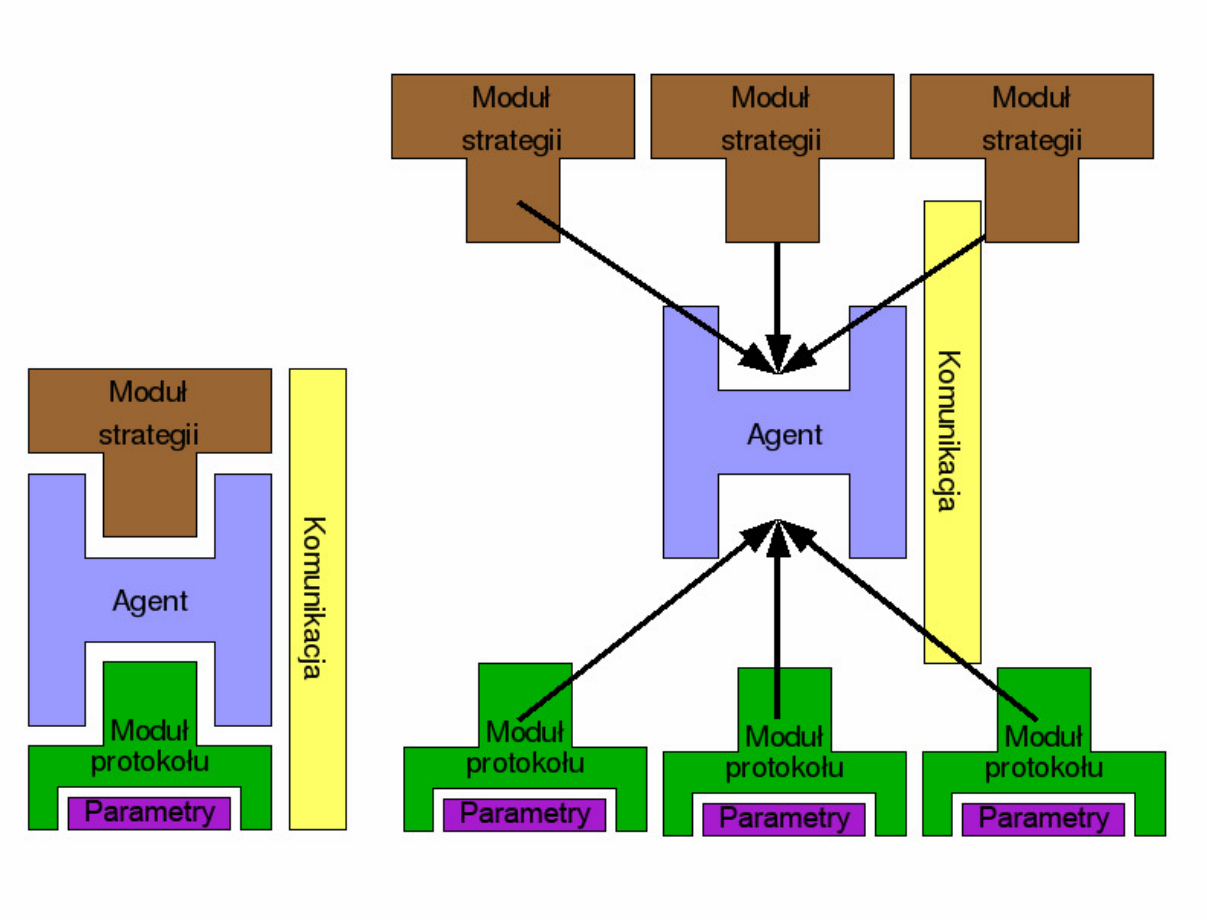
Prowadzenie sprzedaży od strony sklepowej przedstawia się następująco. Agent reprezentujący użytkownika-sprzedawcę, a więc agent sklepowy (*Shop Agent*, *SA*) na początku działania systemu tworzy agentów pomocniczych – *agenta-bramkarza* i agenta zarządzającego magazynem (*Warehouse Agent*, *WA*). Rejestruje także w centrum informacyjnym towary początkowo sprzedawane. Od tego momentu agenci *CA* mogą podejmować próby zakupu w sklepie, komunikując się z *GA*, który tworzy lub odbiera *Buyer Agent*'ów, zainteresowanych udziałem w aukcji, nie dopuszcza do licytacji tych klientów, którym nie należy ufać, a następnie, po zebraniu się wystarczającej dla aktualnie obowiązującego modelu negocjacji liczby uczestników decyduje o rozpoczęciu aukcji przekazując listę uczestników niezależnemu od sklepu agentowi prowadzącemu negocjację

(*Host Agent, HA*). U *HostAgenta* rejestruje się także agent sprzedający (*Seller Agent, SeA*), reprezentujący sklep i gdy wszystkie strony są gotowe do negocjacji, *HA* rozpoczyna aukcję. Warto w tym miejscu dodać, że czas, gdy zarejestrowani już kupujący czekają na zebranie się odpowiedniej ich liczby, zasady negocjacji mogą jeszcze zostać przez sklep zmienione. W takiej sytuacji agenci *BA* otrzymują nowe parametry, które mogą zaakceptować i przystąpić do negocjacji (lub czekać na rozpoczęcie jeśli jest to konieczne) lub odrzucić je i odstąpić od zakupu.

W momencie rozpoczęcia licytacji, *WA* dostaje prośbę o rezerwację wystawionej liczby produktów. Rezerwacja ta jest aktywna do momentu zakończenia licytacji, w przypadku gdy nikt nie wygrał, bądź do momentu finalizacji transakcji – potwierdzenia zakupu, rezygnacji lub wygaśnięcia terminu rezerwacji – przez agenta kupującego. Długość trwania rezerwacji w przypadku pomyślnego zakończenia licytacji ustalana jest w zależności od poziomu zaufania jakim darzony jest klient. W trakcie działania systemu, *Warehouse Agent* dba o zapasy towarów w magazynie, aby produktu nie zabrakło. Więcej informacji o agencie *WA* i związanym z nim podsystemie logistycznym znaleźć można w sekcji 1.4.4.

### 1.4.3 Proces negocjacji

Sam proces prowadzenia negocjacji cenowych jest bardziej złożonym zagadnieniem, a dokładny opis jego przebiegu znaleźć można na przykład w [11], ograniczę się tylko do jego zarysowania. Najważniejszym założeniem dotyczącym procedury negocjacji jest wielość możliwych do zastosowania procedur negocjacji. W różnych sklepach dla różnych produktów mogą być stosowane różne modele (stała cena z wyprzedażą, aukcja angielska, holenderska itd.). Wymaga to od agentów kupujących umiejętności radzenia sobie z wieloma rodzajami negocjacji. Jednocześnie mobilność tychże agentów nie pozwala na każdorazowe przenoszenie ze sobą kompletu zachowań. Dlatego też autorzy systemu zdecydowali się na podział agenta kupującego na 3 części: moduł komunikacyjny, moduł protokołu negocjacji wraz z zestawem parametrów i moduł strategii.




*Rys. 4. Modularna budowa agenta BA*

Moduł komunikacyjny, który umożliwia podstawową interakcję agenta *BA* z agentem prowadzącym sprzedaż jest wspólny dla wszystkich rodzajów negocjacji, więc jest przenoszony jako część szkieletu agenta. Moduły protokołu i strategii są natomiast zależne od ustalonego przez sklep typu negocjacji. Protokół (*NegotiationProtocol*) definiuje ogólny schemat przebiegu negocjacji i jako taki jest wspólny dla wszystkich uczestników – może być więc przekazany agentowi przez *GA*. Razem z protokołem do agentów kupujących wysyłany jest także zestaw parametrów (*NegotiationTemplate*) określających negocjację, do udziału w której są zapisani – w jego skład wchodzą wartości wszelkich zmiennych występujących w protokole negocjacyjnym, a więc na przykład dla aukcji angielskiej: cena wywoławcza (cena, od której zaczynana jest licytacja; ang. *asking/starting price*) i minimalna wartość o jaką można podbić ofertę. Moduł strategii (*BuyerNegotiationStrategy*) zawiera natomiast sposób zachowania się agenta w trakcie negocjacji określonego rodzaju przy danych parametrach. Są to informacje prywatne danego *BA*, mogą więc być dostarczone jedynie przez rodzimego agenta klienckiego.

Podobnie ma się sprawa od strony sklepu – *Seller Agent* odpowiedzialny za prowadzenie negocjacji otrzymuje jej schemat, zestaw parametrów i strategię (*SellerNegotationStrategy*) od *Gatekeeper Agenta*, przy czym schemat jest rzeczą publiczną i jest rozdawany wszystkim uczestnikom, strategia natomiast, jako sposób postępowania agenta sklepowego podczas negocjacji jest prywatna dla agenta sprzedającego. W skład strategii, dla, przykładowo, aukcji holenderskiej, mogą w tym przypadku wchodzić takie informacje jak cena minimalna (cena, poniżej której sklep odmawia sprzedaży; ang. *reserve price*) czy wielkość obniżki ceny w kolejnych krokach negocjacji.

### 1.4.4 Podsystem logistyczny

W poprzednich sekcjach pokrótce przedstawione zostały główne zadania agenta *WA*: rejestrowanie przepływu towarów i dbanie o poziomy magazynowe produktów. Aby jednak skutecznie realizować odnawianie zapasów, zaprojektowany został oddzielny agentowy system logistyczny realizujący zamówienia na towary składane przez agenta *WA*. Dokładny opis tej części systemu znaleźć można w [18, 21], poniżej postaram się jedynie przybliżyć jego ogólny zarys. Sercem tego systemu jest agent logistyczny ( *Logistics Agent, LA*), który korzysta z usług puli agentów zamawiających (*Ordering Agent, OA*), realizujących szczegóły dostaw. Oprócz wymienionych, w podsystemie tym występują jeszcze *Logistics CIC*, czyli odpowiednik *CIC* z głównego systemie oraz dostawcy (*Wholesale Agent, WhA*).

Proces zamawiania rozpoczyna się od dostarczenia agentowi *WA* przez *SDA* prognozy sprzedaży danego produktu. Na podstawie tych danych oraz aktualnego stanu magazynowego danego produktu, *WA* przygotowuje specyfikację zamówienia i wysyła ją do agenta *LA*. *LA* z kolei korzystając z usług *Logistics CIC* wyszukuje hurtowników sprzedających dany towar, a następnie filtruje ich listę biorąc pod uwagę zaufanie oparte na poprzednich interakcjach. Tak spreparowana lista trafia do jednego z Ordering Agentów, który to tworzy zapytanie ofertowe i wysyła je do dostawców z listy. Po otrzymaniu odpowiedzi od *Wholesale Agent'ów*, *OA* wybiera najlepszą ofertę i powiadamia *LA* o sukcesie. O dostarczeniu zamawianego towaru *LA* dowiaduje się od *WA*.

## 1.5 Adaptacyjni agenci w e-commerce

Adaptacyjnych zachowań w rozpatrywanym wieloagentowym systemie e-commerce szukać należy przede wszystkim w częściach klienckiej i sklepowej, gdyż tam właśnie pojawia się wiele sytuacji, w których na podstawie zgromadzonej wiedzy podejmowane są decyzje.

W zaproponowanym rozwiązaniu wszelkie procesy decyzyjne zostały oddelegowane do odpowiednich agentów dedykowanych tym zadaniom. W przypadku systemu klienckiego mamy więc do czynienia z agentem *Client Decision Agent* (*CDA*), który komunikując się z *CA* zbiera informacje o środowisku i na podstawie danych gromadzonych w ten sposób podejmuje odpowiednie działania i decyzje. W przypadku systemu sklepowego zostało zastosowane analogiczne rozwiązanie – agent *Shop Decision Agent* (*SDA*) korzystając z administrowanej przez siebie bazy wiedzy podejmuje wszystkie autonomiczne decyzje w obrębie sklepu.

### 1.5.1 *CDA* – decyzyjny agent klienta

#### 1.5.1.1 Wybór oferty

Podstawowym zadaniem jakie stoi przed *CDA* jest wielokryteriowy wybór oferty spośród nadesłanych przez agentów *BA* wyników licytacji. Powinien on zależeć zarówno od wynegocjowanej ceny jak i poziomu zaufania do sprzedawcy. Ponieważ możliwą odpowiedzią procesu decyzyjnego jest także „poczekaj na lepszą ofertę" należałoby przy kryterium ceny brać pod uwagę nie tylko jej wysokość w porównaniu do innych ofert już otrzymanych, ale także zestawić ją z pojęciem ile dany towar może kosztować. Widać więc, że przy wyborze oferty mamy do czynienia z trzema procesami adaptacyjnymi. Pierwszy z nich to wspomniana już możliwa zmiana ceny jaką uważamy za „rozsądną" w przypadku danego produktu. *CDA* powinno więc zbierać informacje na temat cen produktu w negocjacjach, w których brał udział. Drugi, to zaufanie jakim klient darzy sprzedawcę. Koncepcja zaufania w rozpatrywanym systemie jest na tyle istotnym zagadnieniem, że w całości poświęcone jej są publikacje [14] i [15]. Pisząc w skrócie, zaufanie *CA* do danego sklepu tworzone jest przede wszystkim na podstawie jakości wywiązywania się z transakcji czyli dotrzymania terminu dostawy, dostarczenia zamówionego i dobrego jakościowo towaru.

Mniejszy wpływ mają także odmowy dopuszczenia klienta do negocjacji – nikt nie lubi być odrzucany. Ostatnim punktem, w którym podsystem kliencki powinien przystosowywać się do zmian środowiska są wagi poszczególnych kryteriów wyboru oferty kupna – możemy sobie chociażby wyobrazić, że klient, który wielokrotnie zawiódł się na jakości sprzedaży będzie w przyszłości przykładał większą uwagę do poziomu zaufania sklepu.

#### 1.5.1.2 Wybór strategii negocjacyjnej

Drugim ważnym zadaniem, które stoi przed *CDA* jest przygotowywanie strategii jaką ma kierować się *BA* w trakcie negocjacji. Parametry, które muszą być przy tej okazji ustalone zależą oczywiście ściśle od typu aukcji i algorytmu negocjacji i na tym etapie prac nad systemem nie mogą być dokładnie przedstawione, natomiast powinny one być dostosowane do wiedzy zebranej na temat sklepu, który prowadzi negocjacje cenowe oraz, w wersji zaawansowanej, współuczestników licytacji (jeśli negocjacja prowadzona jest na zasadach dopuszczających taki przypadek – np. różne rodzaje aukcji).

#### 1.5.1.3 Ustalanie wizerunku sprzedawców

Ostatnią ważną sytuacją, w której *CDA* jest proszony o wydanie decyzji jest filtrowanie listy sklepów sprzedających dany produkt otrzymanej od *CIC*. W tym przypadku pod uwagę brany jest ogólny wizerunek sklepu, w skład którego, oprócz opisanego wcześniej zaufania, wchodzi także opinia klienta na temat cen oferowanych zwykle przez sklep – jeśli jest on uważany za wyjątkowo drogi, to *CDA* może zdecydować, że nie warto jest wysyłać tam swoich agentów. Dokładniejszy opis tego procesu opisany został w [14].

### 1.5.2 *SDA* – decyzyjny agent sprzedawcy

Analogicznym do *CDA* elementem systemu, należącym do części sklepowej jest Shop Decision Agent (*SDA*), który stanowi mózg tej strony systemu. Do najważniejszych jego zadań należą podejmowanie decyzji o przyjęciu klienta do negocjacji lub jego odrzuceniu, ustalanie czasu trwania rezerwacji produktu dla klienta, który wygrał licytację, ustalanie

parametrów negocjacji oraz przygotowywanie prognoz sprzedaży na użytek agenta magazynowego. Poniżej postaram się przybliżyć wymienione cele i zadania agenta *SDA*.

#### 1.5.2.1 Zarządzanie zaufaniem do klientów

Podczas gdy według przedstawionego wcześniej scenariusza za dopuszczanie lub odrzucanie klientów starających się o udział w licytacji odpowiadał agent *GA*, to w rzeczywistości, zgodnie z ideą centralizacji procesów decyzyjnych, zgodę na uczestnictwo w negocjacjach danego *BA* wydaje *SDA*. Decyzja ta jest podejmowana w oparciu o poziom zaufania sklepu do klienta [14]. Największy wpływ na zaufanie ma postępowanie klienta po wygranej przez niego negocjacji. Jak już zostało wcześniej nadmienione, klientowi, który złożył wygrywającą ofertę wyznaczany jest termin rezerwacji – czas przez który zakupiony towar trzymany jest dla klienta i nie może być sprzedany nikomu innemu. W takiej sytuacji transakcja może zakończyć się na jeden z trzech sposobów: *CA* może potwierdzić zakup – uznaje się wtedy, że sprzedaż zakończyła się powodzeniem i zaufanie do klienta rośnie; termin rezerwacji może upłynąć zanim *CA* potwierdzi zakup – zaufanie jest wtedy mocno obniżane, gdyż sklep mógł stracić przez to potencjalnie wiele innych możliwości sprzedaży produktu; *CA* może zrezygnować z zakupu przed zakończeniem terminu rezerwacji – wtedy zaufanie jest obniżane, ale o mniejszą wartość niż w poprzednim przypadku.

Drugą sytuacją, w której następuje zmiana zaufania jest okres przednegocjacyjny – chodzi o zachowanie klienta po zgłoszeniu się do *GA*, ale przed rozpoczęciem aukcji. Protokół interakcji agenta *BA* z *GA* zakłada, że *GA* rozpoczyna negocjację i przesyła listę agentów-kupców do *Seller Agenta* dopiero, gdy wszyscy kupujący potwierdzą gotowość – jest to czas, który może być potrzebny do sprowadzenia przez każdego z uczestników odpowiedniej strategii licytacji. Tutaj pojawia się możliwość szkodliwego dla sklepu działania klientów – wrogi agent może chcieć nie dopuścić do negocjacji. Z tego też powodu, po pewnym czasie bezczynności *BA* zostanie usunięty z pamięci a zaufanie do klienta przez niego reprezentowanego zostanie obniżone.

Oprócz dopuszczenia klienta do negocjacji istnieje jeszcze jeden punkt, gdzie *SDA* bierze pod uwagę zaufanie do klienta – moment określania długości terminu rezerwacji po wygraniu przez *BA* licytacji. Tutaj stosowana jest oczywiście zasada, że im niższym zaufaniem cieszy

się dany klient, tym mniej czasu dostaje na potwierdzenie zakupu – minimalizujemy w ten sposób ryzyko utraty innych możliwości sprzedaży w razie niedopełnienia rezerwacji.

#### 1.5.2.2 Ustalanie parametrów sprzedaży

*SDA* odpowiedzialny jest także za ustalanie sposobu w jaki następuje sprzedaż towarów – typ negocjacji dla nich stosowany oraz wszelkie parametry wybranego rodzaju negocjacji, takie jak na przykład liczba wystawianych przedmiotów dla modeli zakładających możliwość sprzedaży więcej niż jednej sztuki w pojedynczej negocjacji , ceny minimalna i wywoławcza w przypadku aukcji angielskiej, czy wielkość kroku obniżania ceny dla aukcji holenderskiej. Odnosząc to dokładnie do scenariusza negocjacji opisanego w punkcie 1.4.3, *SDA* wybiera protokół negocjacji (*NegotiationProtocol*) spośród dostępnych w systemie oraz uzupełnia go wektorem jego parametrów (*NegotiationTemplate*), które to informacje przekazywane są każdemu *BA* biorącemu udział w negocjacji oraz *SeA* reprezentującemu sprzedawcę. Oprócz tego przygotowywana jest także strategia, stosowana następnie przez *SeA* w trakcie licytacji (*SellerNegotiationStrategy*). Właściwości te dla każdego produktu mogą być inne, ale nie są zmieniane jednorazowo dla każdej aukcji. Są one ustalane dla każdego oferowanego produktu raz, a następnie zmieniane i przesyłane do *GA* tylko w sytuacji, gdy *SDA* stwierdzi, że sytuacja zmieniła się na tyle, że ich aktualizacja jest wymagana. Rozwiązanie to pozwala na zmniejszenie ilości komunikacji między *SDA* i *GA*, jednocześnie utrzymując możliwość dostosowywania się sklepu do zmian w globalnej wartości produktu, podaży i popytu, a także maksymalizacji zysków sprzedawcy. Warto nadmienić, że w momencie wprowadzania produktu do sprzedaży, menadżer sklepu ustala początkowe wartości paramentrów. Dopiero gdy ilość zgromadzonych przez system danych jest wystarczająco duża, może być włączona automatyczna ich aktualizacja.

#### 1.5.2.3 Prognozowanie sprzedaży produktów

Ostatnim istotnym procesem, w kontekście którego można mówić o przystosowywaniu się sklepu do zmian środowiska jest zarządzanie systemem logistycznym. Moduł ten, reprezentowany w projekcie jako agent magazynowy (*Warehouse Agent*, *WA*) jest odpowiedzialny za pilnowanie stanów magazynowych poszczególnych produktów oraz

zamawianie towaru od hurtowników w taki sposób, aby zminimalizować możliwość opróżnienia półek w sklepie a jednocześnie zapobiec zaleganiu artykułów w magazynie.

Zadanie uzupełniania magazynu w środowisku, w którym zarówno popyt na produkt jak i zbiór sklepów produkt ten oferujących są zmienne wymaga stale aktualizowanego modelu prognozowania sprzedaży. Funkcjonalność ta, ze względu na silne uzależnienie od ogólnej wiedzy historycznej o transakcjach zawartych w trakcie działania systemu, została jednak przeniesiona z *WA* do *SDA*. Agent decyzyjny co jakiś czas, cyklicznie, przygotowuje prognozę sprzedaży danego produktu w tym okresie czasu. W trakcie działania systemu model, na podstawie którego odbywa się predykcja jest aktualizowany aby odzwierciedlać aktualne trendy sprzedaży. Co więcej, są sytuacje, gdy *SDA* przygotowuje nową prognozę natychmiast po otrzymaniu informacji o pewnym zdarzeniu. Ma to miejsce na przykład wtedy, gdy *SDA* zostaje powiadomiony przez *SA* , że na półkach zabrakło jakiegoś towaru – jest to sygnał, że poprzednia prognoza nie okazała się wystarczająco dokładna i należy przygotować, a następnie dostarczyć agentowi magazynowemu nową.

Także horyzont predykcji, a zarazem także częstotliwość jej aktualizacji może być zmienny w czasie niezależnie dla każdego produktu. Towar, którego sprzedaż nie zmienia się znacząco w czasie, a przynajmniej nie można zaobserwować zjawiska trendu, czy sezonowości jest łatwy do prognozowania i nie wymaga częstego powtarzania obliczeń – dzięki zastosowaniu dłuższego horyzontu czasowego możemy zaoszczędzić na obciążeniu agenta decyzyjnego. Nie jest celem tej pracy analiza sposobu działania agenta *WA* – informacje na ten temat znaleźć można w [18, 21], warto natomiast wspomnieć, że agent magazynowy jest do pewnego stopnia autonomiczny jeśli chodzi o ilość zamawianego towaru i częstotliwość dostaw – nie musi zamawiać jednorazowo pełnej liczby produktów przekazanej jako prognoza, ale może na przykład podzielić tę ilość na kilka mniejszych zamówień składanych w pewnych odstępach czasu. Takie zachowanie służyłoby przede wszystkim ograniczeniu ilości towaru zalegającego w magazynach i opóźnieniu zakupu towarów. Przykładowo, gdy *SDA* obliczy, że w ciągu następnego miesiąca sprzeda się 100 par czerwonych butów, *WA* nie ma obowiązku zakupu od razu 100 par, ale może co tydzień zamawiać 25 par. Tego typu decyzje leżą jednak już w gestii agenta *WA*. *SDA* powinien jednak tak dobierać długość horyzontu czasowego, aby ewentualny podział zamówienia na mniejsze części nie wymagał od *WA* stosowania algorytmów prognozowania sprzedaży – to jest rola agenta decyzyjnego. Z tego też powodu tak istotne jest, aby w obrębie horyzontu czasowego sprzedaż towaru

pozostawała na stałym poziomie. Dolną granicą długości horyzontu czasowego powinien być przewidywany czas realizacji pojedynczego zamówienia przez dostawcę. Nie jest pożądaną sytuacja, kiedy sklep otrzymuje towar później, niż kolejną prognozę.

### 1.5.3 Zbieranie danych o środowisku

Aby skutecznie wypełniać swoje zadania *SDA* musi mieć dostęp do możliwie najpełniejszych informacji na temat działania sklepu. W związku z tym, istotnym elementem całego systemu decyzyjnego sklepu jest zdobywanie danych na temat ważnych zdarzeń oraz przechowywanie ich w łatwy do późniejszej analizy sposób. Ponieważ założenie jest takie, że dostęp do bazy wiedzy sklepu posiada jedynie agent *SDA*, on też musi odpowiadać za zarządzanie tymi informacjami.

Monitorowanie stanu systemu można prowadzić następujące sposoby:

- moduł monitorujący jest stroną aktywną – odpytuje źródła informacji i rejestruje ich odpowiedzi
- źródło informacji jest aktywne – w momencie zmiany stanu środowiska, lub wystąpienia pewnego zdarzenia, źródło informacji wysyła komunikat zawierający komplet informacji do modułu monitorującego
- źródło informacji przesyła jedynie powiadomienie o pojawieniu się nowych danych, natomiast pełne informacje dostępne są po dodatkowym odpytaniu przez moduł monitorujący
- modyfikacja drugiego lub trzeciego sposobu – wiadomości są wysyłane jedynie do tych modułów monitorujących, które uprzednio dokonały subskrybcji odpowiednich informacji

Pierwsze podejście sprawdza się w sytuacjach gdy możemy sobie pozwolić na opóźnioną reakcję na zdarzenia – informacja nie jest odbierana natychmiast po wystąpieniu, ale dopiero gdy odbiorca się o nią upomni. Można je stosować także gdy nie interesuje nas konkretne zdarzenie, ale na przykład sumaryczne dane z pewnego okresu czasu. Zaletą drugiego podejścia jest możliwość prawie natychmiastowej rejestracji informacji o zdarzeniu, dzięki temu, że wiadomość je opisująca jest wysyłana od razu po jego wystąpieniu. Problem w tym podejściu może pojawić się, jeżeli pełne dane są zbyt duże. W takiej sytuacji warto

zastosować trzeci sposób – odbiorca otrzymuje jedynie niewielkie powiadomienie – jeśli w danym momencie nie potrzebuje pełnych danych, nie musi ich pobierać. Ostatnia metoda – modyfikacja drugiej lub trzeciej jest zazwyczaj stosowana wtedy, gdy źródło informacji powinno dostarczać dane do wielu odbiorców lub lista odbiorców może się dynamicznie zmieniać – najprostszym rozwiązaniem tego problemu jest właśnie rejestrowanie się odbiorców u każdego potrzebnego źródła danych.

Jeśli chodzi o zbieranie danych przez *SDA*, to w większości przypadków ważne jest błyskawiczne otrzymywanie informacji o zdarzeniach zaistniałych w systemie. Oprócz tego, wiadomości są stosunkowo niewielkie, nie zachodzi więc potrzeba rozdzielania komunikacji na powiadomienie-dane. Poza tym, informacje otrzymywane przez *SDA* kierowane są praktycznie tylko do niego, nie pobiera też on podobnych danych z innych źródeł – nie jest więc wymagane wprowadzanie mechanizmu subskrybcji. Z wymienionych wyżej powodów najczęściej stosowana jest metoda druga, choć w przypadku cyklicznego pobierania zbiorczych informacji spoza systemu wykorzystane zostało aktywne odpytywanie źródła informacji. Dokładny opis scenariuszy komunikacji *SDA* z innymi agentami przedstawiony zostanie w kolejnych akapitach.

Informacje jakie zbiera *SDA* można podzielić na pochodzące z i spoza systemu

1. Te pierwsze przekazywane są przez agentów *SA* i *WA* i dotyczą konkretnych zdarzeń, które zachodzą w systemie. *SDA* otrzymuje więc komunikat za każdym razem, gdy klient prosi o dopuszczenie do negocjacji – wiadomość ta niesie właściwie tylko informację, że takie zdarzenie miało miejsce i jest potrzebna ze względu na to, że w niektórych przypadkach może to być ostatnia interakcja z danym klientem (np. przy odrzuceniu jego zgłoszenia). Od agenta *SA* komunikaty przychodzą także po każdej zakończonej licytacji oraz po finalizacji transakcji – potwierdzeniu, rezygnacji bądź wygaśnięciu rezerwacji. Komunikaty te zawierają komplet danych odnośnie przebiegu procesu sprzedaży od rozpoczęcia negocjacji, przez sam jej przebieg (wraz z historią ofert składanych przez uczestników) do zakończenia wszystkich transakcji z niej wynikłych. Oprócz tego *SDA* otrzymuje także wiadomości od *WA* zawierające informacje dotyczące dostaw produktów do magazynu.

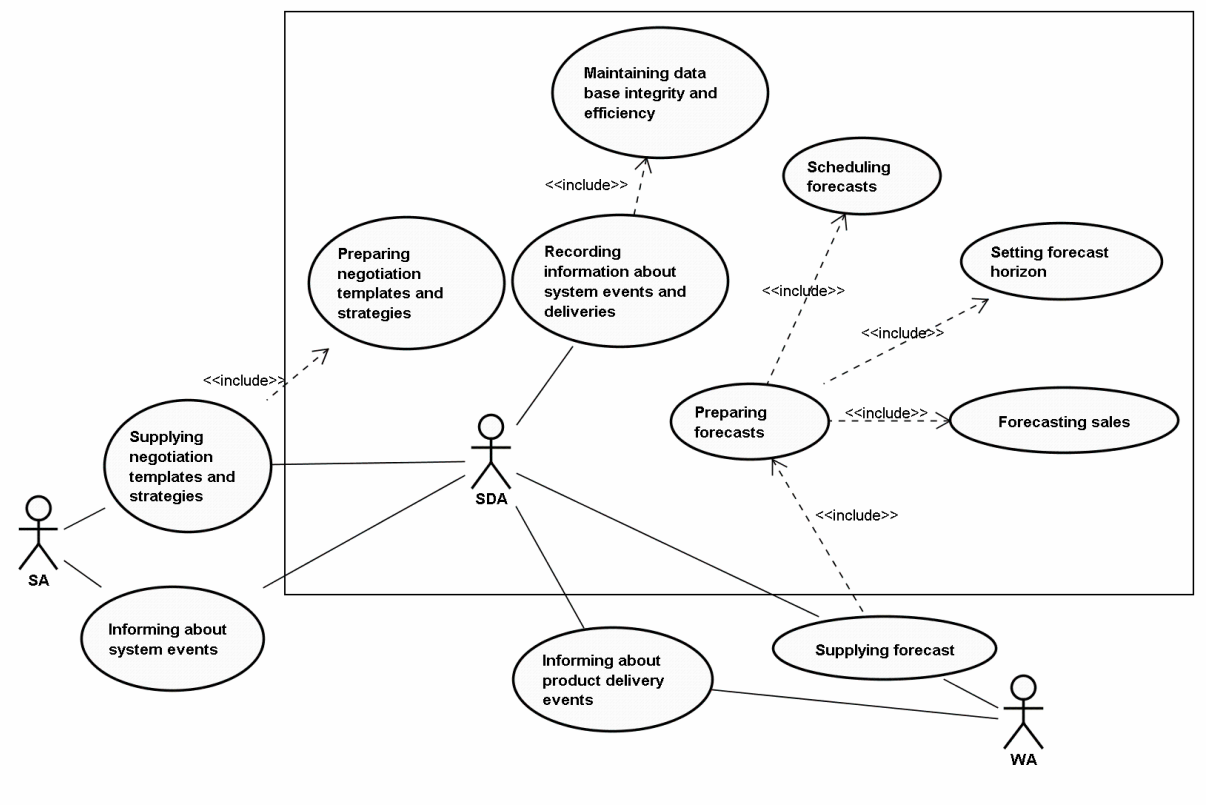
2. Poza wymienionymi danymi gromadzone są także informacje o popycie i podaży na określony produkt. Są one zbierane aktywnie przez *SDA* dzięki cyklicznemu

odpytywaniu centrum informacyjnego o dzienną liczbę zapytań klientów o dany towar oraz listę sklepów go oferujących. Dzięki takim danym sklep może dowiedzieć się chociażby, czy brak klientów jaki obserwuje jest wynikiem swojego słabego wizerunku, czy faktycznej zniżki popytu. Należy w tym momencie nadmienić, że podstawową fukcjonalnością agenta *CIC* jest udzielanie odpowiedzi o listę sklepów oferujących dany towar. Z tej informacji możemy więc dowiedzieć się ilu mamy konkurentów jeśli chodzi o konkretne produkty, a więc o ich podaży, ale nie dowiemy się nic na temat popytu, mierzonego liczbą zapytań o poszczególne artykuły. Oczywiście *CIC* może w łatwy sposób gromadzić takie informacje, udostępnianie ich jednak powinno być uznane za funkcjonalność dodatkową, skierowaną nie do klientów (jak w przypadku listy sklepów), ale do sprzedawców i jako taka mogłoby być serwisem płatnym.

Informacje gromadzone w bazie wiedzy agenta *SDA* są bardzo szczegółowe, a biorąc pod uwagę, że system tego typu powinien zakładać nieprzerwane użytkowanie nawet przez kilka lat, nietrudno jest zauważyć, że w pewnym momencie problemem stać się może zbyt duża ich ilość. Ważne jest więc, że potrzebny jest taki system ich składowania, aby po dłuższym czasie funkcjonowania sklepu nie tracić możliwości wydajnego ich analizowania i ekstrakcji wiedzy w nich zawartej. Sytuację, której chcemy uniknąć, można podsumować zdaniem: bogaci w dane, ubodzy w wiedzę (z ang. „Data rich, information poor") – duża ilość informacji, których jednak nie możemy wykorzystać do polepszenia jakości decyzji wydawanych przez *SDA* z powodu niewystarczającej wydajności przetwarzania danych. Aby temu zapobiec zaproponowany został specjalnie zaprojektowany system bazodanowy oparty o schematy wykorzystywane przy tworzeniu hurtowni danych. Dokładny opis tego rozwiązania przedstawiony zostanie w rozdziale 2.3.

# 2 Projekt systemu

W praktycznej części tej pracy skupiłem się na zaprojektowaniu i implementacji adaptacyjnego agenta na przykładzie *SDA*, w tym jego części odpowiedzialnych za prognozowanie sprzedaży, ustalanie parametrów sprzedaży oraz, wymaganej przez oba wymienione moduły, bazy danych o działaniu systemu.




*Rys. 5. Diagram przypadków użycia agenta SDA wewnątrz systemu sklepowego.*

Zadania jakie realizuje zaimplementowany agent *SDA* przedstawione zostały na diagramie przypadków użycia (Rys. 5) . Jak widać, część *SDA*, jaką miałem zrealizować komunikuje się tylko z dwoma agentami: z *SA*, od którego otrzymuje wszelkie informacje dotyczące działania systemu i któremu dostarcza parametry negocjacji wraz z informacjami o zmianach metod negocjacji cenowych lub strategii, według których sprzedawcy powinni postępować; oraz z *WA*, któremu dostarcza gotowe prognozy sprzedaży i od którego otrzymuje informacje na temat dostaw produktów. Dokładniejszy opis wszystkich scenariuszy komunikacji *SDA* z tymi agentami zostanie przedstawiony w kolejnym podrozdziale. Oprócz wymienionych wyżej, SDA komunikuje się z jeszcze jednym agentem, znajdującym się poza systemem sklepowym CIC. SDA korzysta z jego usług aby zdobyć informacje na temat konkurujących z nim sklepów oraz, jeśli jest taka możliwość, liczby zapytań o produkt.

## 2.1 Scenariusze komunikacji z innymi agentami w systemie

Jak zostało przedstawione w poprzednich rozdziałach, rola *SDA* w systemie jest bardzo rozległa, z czego wynika też mnogość przypadków komunikacji tego agenta z innymi częściami systemu. Dzięki temu, że podsystem sklepowy został zaprojektowany w taki sposób, że *SA* koordynuje funkcjonowanie innych agentów i dla *SDA* działa jako fasada większości systemu sklepowego, okazuje się na szczęście, że *SDA* musi współpracować jedynie z trzema innymi jednostkami – *SA* – w sprawach związanych z większością zadań, *WA*, w przypadku procesów logistycznych oraz *CIC*, przy zbieraniu danych statystycznych spoza systemu. Poniżej postaram się bardziej szczegółowo niż w poprzednich częściach przedstawić scenariusze komunikacji z wymienionymi agentami, wraz z informacjami podczas nich przekazywanymi.

### 2.1.1 Zbieranie danych

#### 2.1.1.1 Zgłoszenie się *BA*

W momencie gdy agent *BA* przybywa do systemu sklepowego, lub *GA* tworzy nowego na prośbę klienta, *GA* wysyła powiadomienie o tym do agenta *SA*, który przekazuje tę informację dalej do agenta *SDA*.

Informacje, które są przekazywane, to:

1. Identyfikator klienta
2. Identyfikator produktu, który klient chciał kupić
3. Czas zgłoszenia
4. Czy BA został utworzony przez GA na prośbę klienta, czy migrował z systemu klienckiego

#### 2.1.1.2 Odrzucenie *BA*

W sytuacji, gdy *GA* zdecyduje o niedopuszczeniu agenta *BA* do negocjacji, lub gdy przekroczony zostanie maksymalny czas (timeout) w jakim *BA* powinien wykonać operację

wymaganą dla postępu procesu sprzedaży (na przykład akceptację warunków negocjacji), agent *BA* zostaje usunięty z systemu sklepowego, a *SDA* jest o tym informowany stosownym komunikatem. Informacja ta jest bardzo istotna pod kątem zarządzania zaufaniem do klientów. [14, 15]

Wiadomość dotycząca odrzucenia agenta *BA* zawiera następujące dane:

1. Identyfikator klienta
2. Identyfikator produktu, który klient chciał kupić
3. Czas zgłoszenia klienta
4. Czas odrzucenia klienta
5. Powód odrzucenia klienta

#### 2.1.1.3 Koniec negocjacji

W momencie zakończenia negocjacji, *GA* przesyła raport z jej przebiegu i wyników do *SA*, który przekazuje wiadomość do *SDA*.

Wiadomość ta zawiera całość danych dotyczących negocjacji, a więc:

1. Identyfikator negocjacji
2. Identyfikator sprzedawanego produktu
3. Czas rozpoczęcia negocjacji
4. Czas zakończenia negocjacji
5. Liczba oferowanych jednostek produktu
6. Rodzaj strategii użytej podczas negocjacji przez SeA, wraz z jej parametrami – w tym m.in. cena minimalna
7. Rodzaj szablonu negocjacji wysłany do uczestników negocjacji, wraz z jego parametrami – w tym m.in. cena wywoławcza
8. Lista uczestników, którzy zostali dopuszczeni do negocjacji i otrzymali szablon licytacji, wraz z dokładnym czasem przekazania im szablonu
9. Lista uczestników, którzy wzięli udział w licytacji – dla każdego z nich:
    a. Czas otrzymania przez *GA* potwierdzenia gotowości do licytacji

b. Lista złożonych przez niego ofert, wraz z czasem złożenia, oferowaną ilością i ceną oraz informacją, czy dana oferta okazała się wygrywająca
c. Jeśli oferta była wygrywająca, to także termin wygaśnięcia rezerwacji produktu oraz identyfikator rozpoczętej transakcji

Na poniższym przykładzie przedstawiona została treść wiadomości opisującej negocjację według protokołu aukcji angielskiej z wieloma sztukami produktu. W aukcji brało udział 4 klientów, spośród których 3 wylicytowało pewną ilość oferowanego produktu. Format wiadomości przedstawionej poniżej, jak również w kolejnych przykładach odpowiada formatowi w jakim informacje te trafiają do logów aplikacji.

```
NegotiationLog: {
      negotiationId: 1205
      productId: Blender ZX-2010-23AT
      negotationStart: Wed Feb 01 20:10:10 CET 2006
      negotiationEnd: Wed Feb 01 20:10:25 CET 2006
      quantityOffered: 50
      strategyId: SimpleMultiItemEnglishAuction
      reservePrice: 15.0
      templateId: MultiItemEnglishAuction
      askingPrice: 10.0
      bids: {
            BidInfo: {
                  clientId: Client3
                  offerTime: Wed Feb 01 20:10:11 CET 2006
                  quantity: 10
                  price: 11.0
                  isWinning: false
                  reservation: null
            }
            BidInfo: {
                  clientId: Client1
                  offerTime: Wed Feb 01 20:10:12 CET 2006
                  quantity: 20
                  price: 14.0
                  isWinning: false
                  reservation: null
            }
            BidInfo: {
                  clientId: Client2
                  offerTime: Wed Feb 01 20:10:13 CET 2006
                  quantity: 20
                  price: 16.0
                  isWinning: false
                  reservation: null
            }
            BidInfo: {
                  clientId: Client3
                  offerTime: Wed Feb 01 20:10:14 CET 2006
                  quantity: 10
                  price: 17.0
```

```
                  isWinning: true
                  reservation: ReservationInfo: {
                        transactionId: 51
                        expiryTime: Wed Feb 01 21:10:14 CET 2006
                  }
            }
            BidInfo: {
                  clientId: Client1
                  offerTime: Wed Feb 01 20:10:16 CET 2006
                  quantity: 20
                  price: 20.0
                  isWinning: true
                  reservation: ReservationInfo: {
                        transactionId: 52
                        expiryTime: Wed Feb 01 20:35:16 CET 2006
                  }
            }
            BidInfo: {
                  clientId: Client4
                  offerTime: Wed Feb 01 20:10:20 CET 2006
                  quantity: 20
                  price: 22.0
                  isWinning: true
                  reservation: ReservationInfo: {
                        transactionId: 54
                        expiryTime: Wed Feb 01 21:40:25 CET 2006
                  }
            }
      }
}
```

#### 2.1.1.4 Zakończenie transakcji

Po zakończeniu negocjacji, dla każdego z klientów, którzy pomyślnie wylicytowali pewną ilość oferowanego towaru, ustalana jest długość terminu rezerwacji. Jest to czas, w którym obowiązują dane warunki transakcji - na przykład w tym czasie można kupić parę spodni za 45 złotych – po tym czasie ta cena nie będzie już obowiązywała. Cały proces może zakończyć sie transakcją – jest to tożsame z pobraniem zapłaty i wydaniem produktu; rezygnacja z rezerwacji – klient się rozmyślił i powiadomił o tym sklep; lub upłynięciem terminu rezerwacji. W każdej z wymienionych sytuacji *SA* przesyła odpowiednią informację do *SDA* celem zapisania tej informacji w bazie wiedzy.

Wiadomość o zakończeniu transakcji zawiera następujące informacje:

1. Identyfikator transakcji

2. Identyfikator klienta
3. Typ zdarzenia – potwierdzenie / rezygnacja / wygaśnięcie rezerwacji
4. Data i czas zdarzenia

Kolejny przykład pokazuje informację o zakończeniu danej transakcji. Typ zdarzenia – *CONFIRMED* – oznacza, że klient zdecydował się zakupić określoną ilość zarezerwowanego wcześniej produktu.

```
TransactionLog: {
      transactionId: 54
      negotiationId: 1205
      clientId: Client4
      time: Wed Feb 01 20:10:20 CET 2006
      finalisationType: CONFIRMED
      quantityReserved: 20
   }
```

#### 2.1.1.5 Dostawa towaru do magazynu

Po zrealizowaniu przez hurtownika zamówienia, agent magazynowy (*WA*) powiadamia o tym zdarzeniu *SDA*, przekazując następujące informacje:

1. Identyfikator otrzymanego produktu
2. Data i czas złożenia zamówienia
3. Data i czas realizacji zamówienia
4. Ilość i cena zamawianego towaru

Przykład, pokazujący informację o dostarczeniu 2000 sztuk towaru o id `Blender ZX-2010-23AT` o określonej wartości. Można także zaobserwować czas zamówienia oraz realizacji dostawy.

```
DeliveryLog: {
      deliveredProductId: Blender ZX-2010-23AT
      orderTime: Sun Apr 02 11:10:40 CEST 2006
      deliveryTime: Wed Apr 05 11:10:40 CEST 2006
      orderedQuantity: 2000
      orderedValue: 20000.0
      wholesalerId: Wholesaler2
}
```

#### 2.1.1.6 Informacje o liczbie zapytań od *CIC*

Jak już było powiedziane w poprzednim rozdziale, *SDA* cyklicznie odpytuje centrum informacyjne o liczbę zapytań dotyczących sprzedawanych przez sklep produktów w ciągu pewnego okresu czasu. Tak jak zostało już wspomniane, ta funkcjonalność *CIC* powinna stanowić dodatkową, płatną usługę skierowaną do sprzedawców. Każde zapytanie dotyczy jednego produktu i zawiera następujące parametry:

1. Identyfikator produktu
2. Data i czas początku okresu, dla którego chcemy otrzymać liczbę zapytań
3. Data i czas końca okresu

Następny przykład pokazuje wiadomość, wysyłaną przez SDA do CIC z prośbą o informację o liczbie zapytań klientów o produkt o id `Blender ZX-2010-23AT` w okresie od 2.04.2006 11:10:40 do 3.04.2006 11:10:40.

```
DemandRequest: {
      productId: Blender ZX-2010-23AT
      startDate: Sun Apr 02 11:10:40 CEST 2006
      endDate: Sun Apr 03 11:10:40 CEST 2006
}
```

Odpowiedź *CIC* zawiera z kolei następujące informacje:

1. Identyfikator produktu
2. Data i czas początku okresu
3. Data i czas końca okresu
4. Liczba zapytań o dany produkt

Poniższy przykład zawiera natomiast odpowiedź CIC, a więc liczbę zapytań o dany product w określonym okresie czasu.

```
DemandData: {
      productId: Blender ZX-2010-23AT
      startDate: Sun Apr 02 11:10:40 CEST 2006
      endDate: Sun Apr 03 11:10:40 CEST 2006
      cicQueryCount: 1520
}
```

#### 2.1.1.7 Informacje o sklepach sprzedających dany towar

Omawiając wcześniej informacje na temat środowiska, które przechowywane są w bazie wiedzy sklepu, wymieniona została liczba sklepów sprzedających dany towar w pewnym momencie w czasie. Dane te niosą istotną wiedzę na temat wielkości konkurencji i podaży jeśli chodzi o dany produkt. *CIC* nie został wyposażony w osobną, dedykowaną sklepom funkcjonalność umożliwiającą dostęp do tych danych, sklep może natomiast wykorzystać istniejący interfejs kliencki do pozyskiwania tych informacji. *SDA*, identycznie jak w normalnym scenariuszu robi to agent kliencki, odpytuje *CIC* o listę sklepów oferujących dany towar, a następnie rejestruje ich liczbę.

Zapytanie do *CIC* zawiera jedynie identyfikator produktu.

Odpowiedź od *CIC* zawiera następujące pola:

1. Identyfikator produktu
2. Lista sklepów oferujących dany produkt

#### 2.1.1.8 Problemy ze zrealizowaniem planu odnawiania stanu magazynowego

Może się zdarzyć sytuacja, w której *WA* nie może spełnić założeń nakładanych na niego przez *SDA* w postaci prognozy, np. gdy hurtownicy nie oferują ilości produktów wystarczającej do utrzymania stanu magazynowego na poziomie określonym w prognozie. *WA* powiadamia wtedy *SDA* o tym problemie, kontynuując próby zamawiania. Jeżeli w dalszym ciągu plan nie może być zrealizowany, *WA* wysyła kolejne zawiadomienie. Po trzeciej nieudanej próbie, *WA* zaprzestaje wykonywanie zleconego mu planu i wysyła stosowną informację do *SDA*.

Powiadomienie zawiera jedynie identyfikator produktu.

#### 2.1.1.9 Problemy z brakiem towaru

Należy liczyć się z tym, że w trakcie działania systemu pojawi się problem z niedostateczną ilością towaru w magazynie. W momencie kiedy liczba jednostek produktu spadnie do zera, *SA* jest zobowiązany do natychmiastowego wyrejestrowania rzeczonego towaru w *CIC* tak,

aby zminimalizować liczbę klientów dowiadujących się o braku towaru dopiero od *GA*. Oprócz wyrejestrowania produktu, *SA* powiadamia o zaistniałej sytuacji *SDA*, który może podjąć odpowiednie działania zaradcze takie jak na przykład przyspieszone zaktualizowanie prognozy sprzedaży danego produktu

Powiadomienie zawiera jedynie identyfikator produktu.

### 2.1.2 Dostarczanie wyników

Jeśli chodzi o dostarczanie wyników procesów decyzyjnych przez agenta *SDA*, postanowiłem ograniczyć się do szczegółowo rozpatrywanych przeze mnie jego aspektów, a więc prognozowania sprzedaży oraz, w mniejszym stopniu, aktualizacji szablonu negocjacyjnego.

#### 2.1.2.1 Dostarczenie prognozy

Po planowym lub wymuszonym nagłą potrzebą przygotowaniu przez *SDA* prognozy, jest ona przesyłana bezpośrednio do agenta *WA*.

Predykcja opisana jest następującymi parametrami:

1. Identyfikator produktu, którego poziom w magazynie ma być pilnowany
2. Ilość jednostek produktu, która według prognozy zostanie sprzedana w czasie następnego okresu predykcji
3. Szacowane możliwe odchylenie prawdziwej sprzedaży od wartości prognozowanej
4. Długość okresu trwania prognozy (w sekundach)

Następny przykład pokazuje opis prognozy wysłanej do WA, według której w ciągu następnej doby (86400 sekund) od 2 kwietnia 2006, godziny 11:10:40 sprzeda się 2000 (± 180) sztuk produktu o id `Blender ZX-2010-23AT`.

```
ForecastDescription: {
      productId: Blender ZX-2010-23AT
      forecastPeriod: 86400
      forecastAmount: 2000.0
      forecastDeviation: 180.0
      preparationTime: Sun Apr 02 11:10:40 CEST 2006
}
```

#### 2.1.2.2 Dostarczenie strategii, protokołu i parametrów negocjacji

W momencie, gdy agent *SDA* zakończy aktualizację parametrów strategii i protokołu negocjacji, przesyła ich zestaw wraz z obiektami opisującymi sam protokół i strategię do agenta *SA*, który przekazuje je do *GA*. Zmiany te oczywiście obowiązują jedynie w przypadku aukcji rozpoczętych po otrzymaniu nowego zestawu parametrow.

Założenie jest takie, że w przyszłości, wraz z implementacją różnych rodzajów negocjacji opis protokołu i strategii, a zatem także ich parametrów, będzie rozwijany. Opis rodzajów negocjacji zaprojektowanych do tej pory można znaleźć w pracach [18, 19]. Dokładna adaptacja parametrów negocjacji do każdej z wymienionych tam metod negocjacji wykracza poza zakres tej pracy, przedstawiony wektor parametrów dotyczy aukcji angielskiej z wieloma sztukami produktu (ang. *multi-item english auction*):

1. Identyfikator produktu
2. Cena wywoławcza przedmiotu – stanowi część parametrów protokołu negocjacji i jest przekazywana wszystkim uczestnikom
3. Cena minimalna (reserve price) przedmiotu – stanowi część parametrów strategii sklepu i jest znana jedynie agentowi prowadzącemu negocjację
4. Ilość towaru wystawiana w pojedynczej aukcji

Na poniższym przykładzie przedstawiony został wektor parametrów negocjacji dla aukcji angielskiej z wieloma przedmiotami, zawierający cenę początkową, cenę minimalną oraz ilość wystawionych sztuk produktu.

```
NegotiationDescription: {
      productId: Blender ZX-2010-23AT
      startingPrice: 11.0
      reservePrice: 15.0
      offeredAmount: 50
}
```

### 2.1.3 Inne (administracyjne)

Powyżej wymienione zostały najważniejsze ze względu na implementowaną funkcjonalność modułu scenariusze interakcji agenta *SDA* z innymi częściami systemu. Oprócz nich można

wymienić jeszcze kilka – nie służących bezpośrednio wykonywaniu zadań agenta, ale raczej pomocniczych względem wymienionych wcześniej. Są to wiadomości administracyjne, otrzymywane od *SA*, wynikające z decyzji i działań menadżera sklepu.

#### 2.1.3.1 Rozpoczęcie sprzedaży produktu

Jeśli użytkownik systemu sklepowego podejmie decyzję o wprowadzeniu do asortymentu nowego towaru, informacja o tym musi być przekazana agentowi *SDA*. Jest to o tyle istotne, że nawet jeśli nie ma potrzeby automatyzacji procesów decyzyjnych w przypadku tego produktu, to *SDA* powinno otrzymać jego opis aby od samego początku móc zbierać szczegółowe informacje o jego sprzedaży, dostawach itp.

W takiej sytuacji *SDA* otrzymuje wiadomość zawierającą ontologiczny opis produktu.

#### 2.1.3.2 Rozpoczęcie prognozowania sprzedaży produktu

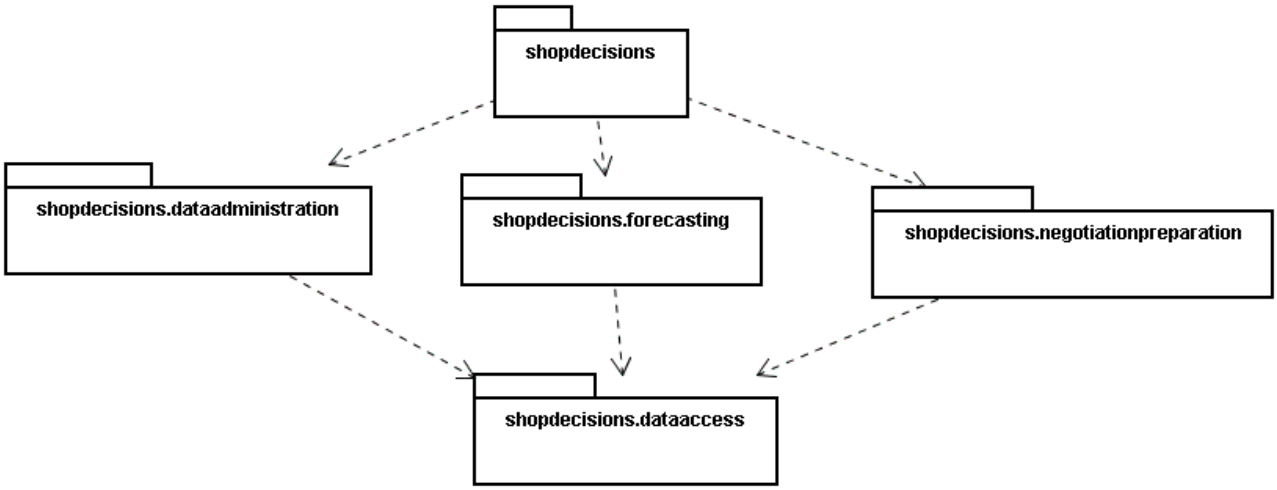
Aby *SDA* przygotowywał prognozy zapotrzebowania na wprowadzony do sprzedaży produkt konieczne jest wysłanie do niego odpowiedniego żądania. W wiadomości tej przekazywany jest jedynie globalny identyfikator produktu. Jeżeli w bazie produktów *SDA* znajduje się towar o danym identyfikatorze, to *SDA* akceptuje żądanie, jeśli nie to zwraca odmowę. W momencie rozpoczęcia prognozowania, *SDA* przygotowuje także natychmiast prognozę danego produktu.

#### 2.1.3.3 Zakończenie prognozowania sprzedaży produktu

Przedmiot można także wyłączyć z automatycznego prognozowania sprzedaży przez wysłanie odpowiedniego żądania z identyfikatorem produktu do agenta *SDA*. Tak samo jak w poprzednim przypadku jeżeli w bazie produktów *SDA* znajduje się towar o danym identyfikatorze, to *SDA* akceptuje żądanie, jeśli nie to zwraca odmowę.

## 2.2 Architektura agenta SDA

Nawet w obecnym kształcie systemu agent *SDA* jest odpowiedzialny za wiele, często krytycznych, procesów decyzyjnych. Biorąc dodatkowo pod uwagę dynamikę rozwoju aplikacji i fakt, że niektóre aspekty jej działania są w dalszym ciągu rozwijane, uszczegóławiane i zmieniane, staje się oczywiste, że architektura *SDA* powinna być jak najbardziej zmodularyzowana i elastyczna. Cel ten został zrealizowany poprzez umieszczenie poszczególnych zadań, a więc prognozowania sprzedaży, przygotowywania parametrów negocjacji i zarządzania danymi, w osobnych, niezależnych od siebie nawzajem modułów.




*Rys. 6. Diagram struktury pakietów agenta SDA*

Rozwiązanie to zakłada, że agent *SDA* pełni funkcję fasady – odbiera jedynie wiadomości, a ich obsługę przekazuje wyspecjalizowanym pakietom. Dołożenie nowej funkcjonalności wymaga więc tylko dodania do zbioru zachowań *SDA* obsługi nowych wiadomości oraz zaimplementowania nowego modułu. Także zmiana jednego z modułów nie pociąga za sobą potrzeby aktualizacji pozostałych części.

Na tym poziomie koncepcyjnym nie jest rozstrzygnięte jeszcze, czy operacje realizowane przez poszczególne moduły wykonywane będą każde przez oddzielnego agenta, czy też w kilku wątkach przez agenta *SDA*. Przydzielenie każdego zadania osobnemu agentowi pozwoliłoby na wykonywanie ich równolegle na różnych maszynach, co mogłoby poprawić wydajność w sytuacjach dużego obciążenia. To rozwiązanie niesie jednak komplikacje związane z asynchronicznym przekazywaniem wiadomości. Na potrzeby tej pracy został

zaimplementowany model z pojedynczym agentem *SDA* wykonującym wszystkie zadania, ale równolegle - w kilku wątkach. Zachowano także pełną modularyzację – cała logika związania z konkretnym zadaniem znajduje się w pojedynczym pakiecie. Ewentualne rozdzielenie funkcjonalności *SDA* na kilku agentów nie wymagałaby więc zmiany tych pakietów, a jedynie osadzenia ich wewnątrz odpowiednich, wyspecjalizowanych agentów i zaimplementowania komunikacji między nimi a *SDA*.

### 2.2.1 Moduł prognozowania sprzedaży

Najważniejszym zadaniem modułu prognozowania sprzedaży jest, oczywiście, tworzenie predykcji potrzebnych agentowi *WA* oraz ustalanie ich horyzontów czasowych . *SDA* musi jednak dbać także o to, aby *WA* otrzymywał odpowiednie prognozy w odpowiednim momencie – planowanie kiedy prognozować jaki produkt także należy do obowiązków tego modułu. Ponieważ zakładamy możliwość sprzedawania przez sklep przedmiotów, dla których przewidywanie popytu nie jest potrzebne lub wskazane (na przykład w sytuacji, gdy sprzedaż produktu jest trudna do prognozowania lub nie mamy dość danych do prognozowania i użytkownik chciałby własnoręcznie sterować poziomem stanu magazynowego), moduł prowadzi także rejestr towarów, dla których prognozy powinny być cyklicznie aktualizowane. Tworzenie predykcji może być operacją kosztowną czasowo i dlatego jest wykonywane asynchronicznie.

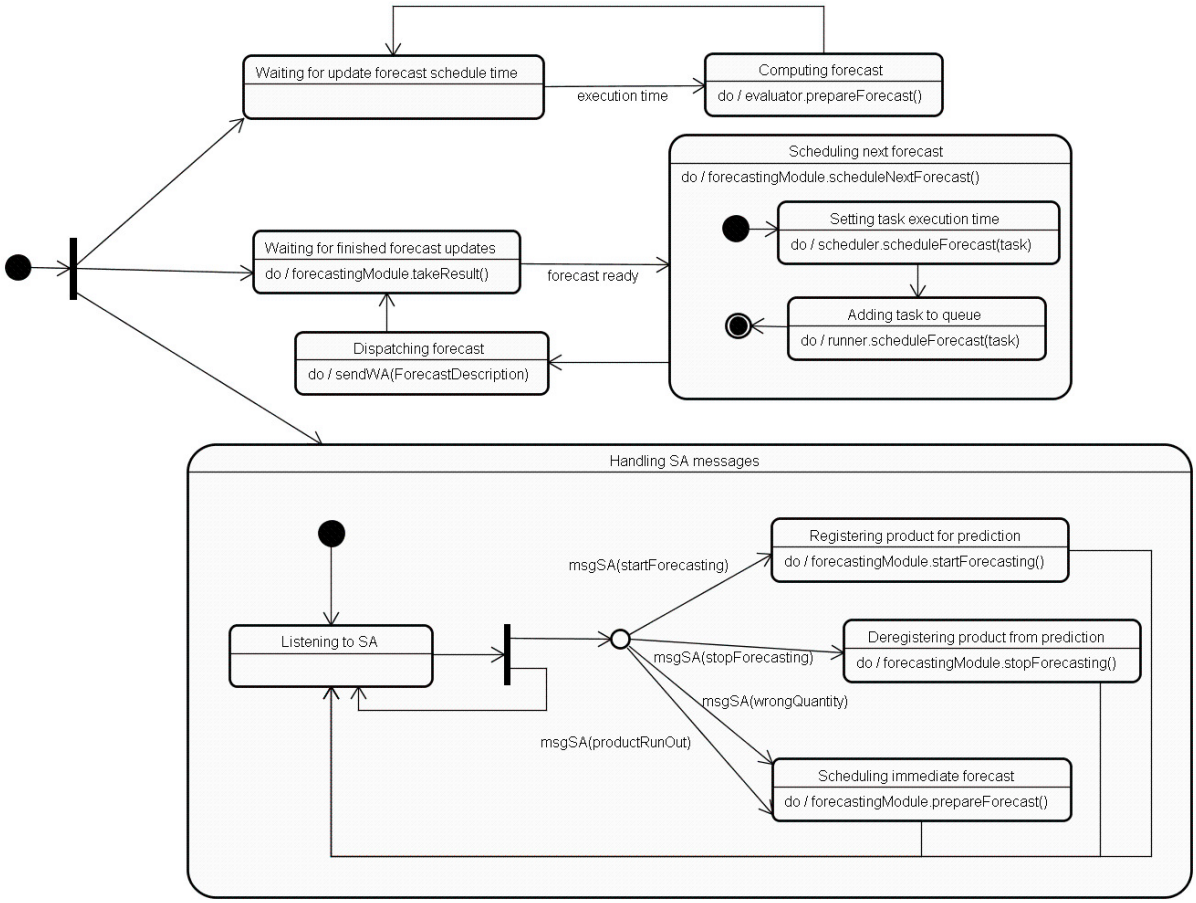
Moduł ten składa się z kilku podstawowych klas, które zapewniają przedstawioną wyżej funkcjonalność:

- **ForecastingModule** – fasada dostępowa do całego modułu. Zorganizowana jest jako singleton. Udostępnia operacje natychmiastowego, asynchronicznego przygotowania prognozy, pobrania gotowej prognozy oraz rejestracji i wycofania produktu z cyklicznego aktualizowania prognoz.
- **ForecastedProductsInventory -** odpowiada za zarządzanie listą produktów, dla których prognozy powinny być aktualizowane cyklicznie. Umożliwia rejestrację i wycofywanie produktów, a także rozstrzyganie, czy dany produkt jest w rejestrze. Przy obecnej implementacji rejestr zarządzanych produktów trzymany jest w pamięci, a więc w sposób nietrwały – jest to rozwiązanie wystarczające do zastosowań próbnych. Zmiana tego założenia wymagałaby jedynie modyfikacji tej klasy w taki

sposób, aby dane te były przechowywane na dysku lub w bazie danych, co nie jest problemem ani trudnym, ani specjalnie pouczającym.

- **ForecastScheduler** – wyznacza terminy w jakich powinny być wykonywane obliczenia prognoz dla konkretnych produktów, a także zarządza listą zaplanowanych zadań. Aktualna implementacja przechowuje tę listę w pamięci, rozwiązanie jest jednak przygotowane do zapewnienia jej trwałości przez zapisanie na dysku lub w bazie danych.
- **ScheduledForecastRunner** – odpowiada za uruchamianie zaplanowanych zadań w odpowiednim momencie i udostępnianie ich wyników. Aktualna implementacja wykorzystuje pulę wątków, z których każdy może być wykorzystany do równoległego wykonania jednego zadania. Jeśli nie są dostępne żadne wolne wątki, to zadanie czeka w kolejce. Wykonane zadania także odkładane są do kolejki, gdzie czekają na odebranie.
- **ForecastEvaluator** – przeprowadza właściwy proces prognozowania sprzedaży danego produktu wraz z ustaleniem horyzontu predykcji. Korzysta z danych historycznych zawartych w bazie wiedzy za pośrednictwem pakietu **shopdecisions.dataaccess.**
- **ForecastTask** – reprezentuje zadanie predykcji wraz z podstawowymi parametrami potrzebnymi aby zaplanować, wykonać w odpowiednim momencie a następnie obliczyć wynik prognozy. Atrybuty obiektów tej klasy są przez kolejne elementy modułu.
- **ForecastDispatcher** – pomocnicze zachowanie, które cyklicznie odpytuje **ForecastingModule** o efekty zakończonych obliczeń i przesyła je do *WA*.

Na poniższym diagramie (Rys. 7) przedstawiony został schemat przejść między różnymi stanami modułu.




*Rys. 7. Diagram stanów modułu przygotowywania prognozy*

### 2.2.2 Moduł przygotowywania negocjacji

Moduł ten jest odpowiedzialny za aktualizowanie szablonów i strategii negocjacyjnych, z których korzysta *GA* przy organizowaniu aukcji. W przeciwieństwie do tworzenia prognoz, nie jest tu potrzebne dokładne planowanie terminu wykonywania obliczeń – podczas gdy *WA* z powodu braku aktualnej predykcji przestaje zamawiać dany towar, to *GA* wykorzystuje ostatnio otrzymane parametry przez cały czas aż do nadejścia nowej ich wersji. Z tego też powodu, wystarczy aby moduł ten cyklicznie aktualizował strategie każdego sprzedawanego produktu raz na jakiś czas. W dalszym ciągu zadania te powinny być wywoływane asynchronicznie, aby nie blokować działania wątku głównego.

Powyższe założenia zdeterminowały architekturę modułu - podobną do znanej z modułu prognozowania sprzedaży, ale z uproszczonym modelem planowania zadań.Na moduł ten składają się następujące klasy:

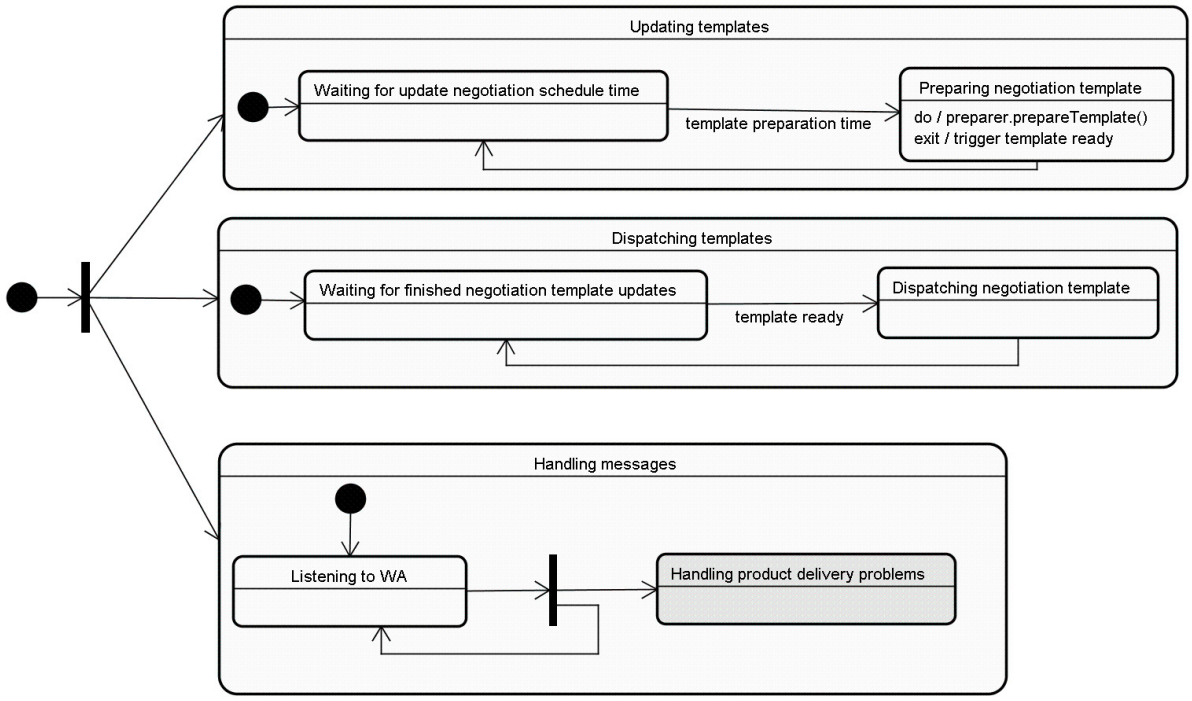
- **NegotiationPreparingModule** - fasada dostępowa do całego modułu. Zorganizowana jest jako singleton. Udostępnia operacje natychmiastowego, asynchronicznego przygotowania opisu szablonu i strategii negocjacji, pobrania gotowego wyniku oraz rejestracji i wycofania produktu ze sprzedaży.
- **NegotiationDescriptionInventory** – zarządza listą sprzedawanych produktów. Umożliwia wprowadzanie sprzedawanego produktu, usuwanie wycofanego a także rozstrzyganie, czy produkt jest na liście. Analogicznie do przypadku **ForecastedProductInventory** rejestr trzymany jest tylko w pamięci.
- **PreparationTaskRunner** – odpowiada za asynchroniczne wykonanie przygotowanego zadania. Podobnie jak w przypadku modułu sporządzania prognoz, zaimplementowany jest na bazie puli wątków z kolejkami przyjmującymi zadania i wyniki obliczeń
- **TemplatePreparer** – ustala parametry negocjacji, w tym szablon, według którego ma odbywać się aukcja oraz strategię jaką stosować będzie agent prowadzący licytację. Klasa ta także zależy od pakietu **shopdecisions.dataaccess**, dzięki któremu pobiera dane historyczne potrzebne do badania.
- **PreparationTask** – zawiera podstawowe parametry wejściowe potrzebne do przygotowania strategii negocjacji, takie jak identyfikator produktu, termin planowanego wejścia w życie strategii oraz poprzedni opis negocjacji. Dane te są przekazywane obiektowi klasy **TemplatePreparer** w momencie uruchomienia obliczeń.

Oprócz wymienionych klas składających się na pakiet, do modułu należą także dwa zachowania, które zapewniają podstawową działalność związaną z aktualizacją strategii:

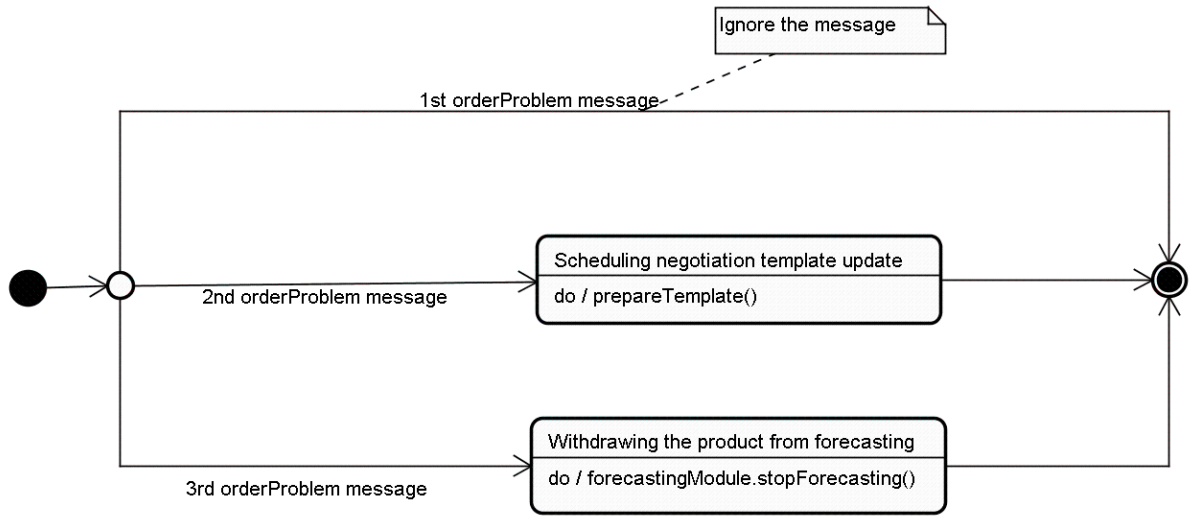
- **NegotiationTemplateUpdater** – zachowanie, które cyklicznie uruchamia operacje przygotowania szablonu negocjacji. Zadania wykonywane są dla kolejnych produktów jeden po drugim, odczekując pewien, ustalony, czas pomiędzy wywołaniami. Po aktualizacji wszystkich produktów rozpoczyna od nowa.

- **NegotiationTemplateDispatcher** – zachowanie, które cyklicznie sprawdza, czy jakieś obliczenia się już zakończyły, a następnie wysyła wyniki tych obliczeń do odpowiedniego agenta – w tym wypadku do *SA*. Ponieważ zachowanie to może blokować wątek, w którym się wykonuje (z powodu blokującego oczekiwania na wyniki obliczeń), uruchamiane jest w osobnym wątku.

Działanie modułu można prześledzić na następujących diagramach stanów (Rys. 8, Rys. 9).




*Rys. 8. Diagram stanów modułu przygotowywania negocjacji*




*Rys. 9. Diagram obsługi problemów z dostawą towaru.*

### 2.2.3 Moduł zarządzania danymi

Oba powyższe moduły do realizacji swoich celów potrzebują wysokiej jakości danych historycznych zorganizowanych w sposób łatwy do wydajnego przetwarzania. Sposób w jaki informacje te są zbierane przez *SDA* został przedstawiony w sekcji 2.1.1, natomiast za przekształcenia tych informacji i ich zapis do bazy danych odpowiedzialny jest ten właśnie moduł.

W kolejnym rozdziale zostanie przedstawiony dokładny projekt bazy wiedzy służącej agentowi *SDA* jako źródło danych potrzebnych do podejmowania decyzji.

## *2.3 Struktura bazy danych*

Nie ulega wątpliwości, że jednym z najważniejszych elementów systemu decyzyjnego jest źródło informacji, na podstawie których wyciągane są wnioski, a więc baza danych. Także w przypadku agenta *SDA* aspekt ten jest kluczowy dla dobrego funkcjonowania całego modułu.

Sposób w jaki zaprojektowana została rzeczona baza danych jest silnie zdeterminowany przez wymagania jakim musiała ona sprostać. Ze względu na bardzo szerokie zastosowanie gromadzonych danych, istotne jest aby tworzyły one jak najpełniejszy obraz funkcjonowania całego systemu – powinny obejmować wszystkie jego elementy w sposób zintegrowany tak,

aby można było łatwo łączyć informacje pochodzące z różnych części aplikacji. Oprócz tego, dane powinny być możliwie szczegółowe, aby nie stracić możliwości analizowania procesów także w skali mikro. To z kolei w połączeniu z założeniem, że chcemy gromadzić dane historyczne przez cały czas działania systemu powoduje, że problemem staje się sama ilość zbieranych informacji. Co więcej, w przypadku nawet bardzo dużych rozmiarów bazy, zależy nam na tym, aby nie utracić możliwości efektywnej analizy jej zawartości, zwłaszcza biorąc pod uwagę, że wiele procesów decyzyjnych wykonywanych jest stosunkowo często.

Cechy te zadecydowały o zaprojektowaniu tej części systemu sklepowego nie jako tradycyjna, transakcyjna baza danych, ale w oparciu o model wielowymiarowy wywodzący się z hurtowni danych.

### 2.3.1 Hurtownie danych

Bazy danych były w użyciu od pierwszych lat elektronicznego przetwarzania informacji, choć na początku nie przypominały oczywiście obecnych systemów i w większości przypadków były dedykowane konkretnym potrzebom firm. Wraz z pojawieniem się relacyjnych baz danych w latach 70. rozpoczął się intensywny rozwój baz danych, a przede wszystkim systemów zarządzania bazami danych (ang. Database management systems, DBMS) ogólnego zastosowania. Technologie takie jak znormalizowany model danych, indeksowanie, języki zapytań i metody optymalizacji przetwarzania bazy danych uczyniły wprowadzanie, modyfikację i przechowywanie informacji wydajnymi i niezawodnymi. Postęp w tej dziedzinie trwał przez całe lata 80. i spowodował, że na początku lat 90. systemy bazodanowe potrafiły już przetwarzać przynajmniej 1000 transakcji na sekundę. Systemy bazodanowe stawały się coraz powszechniejsze i zawierały coraz więcej danych – zwłaszcza w przypadku dużych korporacji. Pojawił się jednak nowy problem – okazało się, że pierwotny cel tworzenia i utrzymywania baz danych w przedsiębiorstwach, a więc analizowanie zgromadzonych w nich informacji i raportowanie w oparciu o nie, stał się bardzo utrudniony. Cytując Ralpha Kimball'a „Przez ostatnie 12 lat musieliśmy odwlekać wydobywanie danych, żeby móc skoncentrować się na ich wstawianiu." [6].

Szybko okazało się, że przetwarzanie transakcyjne i raportowanie w oparciu o ten sam system nie zdaje egzaminu z kilku powodów: odpowiadanie na złożone zapytania zajmowało zasoby, powodując obniżenie wydajności całego systemu; wiele organizacji miało więcej niż jedną

operacyjną bazę danych, więc wnioskowanie na potrzeby całego przedsiębiorstwa było utrudnione; wreszcie istniejące rozwiązania były optymalizowane pod kątem szybkości wstawiania i modyfikacji danych, a nie ich analizy [23].

Aby sprostać tym wymaganiom potrzebne były nowe rozwiązania – stały się nimi rozwijane na poważnie od początku lat 90. hurtownie danych, które przeniosły nacisk z wstawiania i modyfikacji na prostotę i wydajność raportowania. Według definicji W.H.Inmana, jednego z pionierów tej gałęzi informatyki, hurtownia danych to: „..zorientowana tematycznie, zintegrowana, zmienna w czasie i nieulotna kolekcja danych wspierająca proces wspomagania decyzji" ([3], tłumaczenie za [12]). Zorientowanie tematyczne oznacza tu organizację danych wokół kluczowych pod względem przydatności do analizy procesów biznesowych. Integracja hurtowni danych polega na tym, że informacje które do niej trafiają z różnych źródeł powinny zostać zunifikowane pod względem struktury, konwencji nazewniczych, typów danych itp. Ponieważ hurtownia danych magazynuje dane historyczne obejmujące długi okres czasu, wszystkie kluczowe struktury modelu danych obejmują element czasu. Dzięki temu możliwe jest śledzenie w jaki sposób parametry przedsiębiorstwa zmieniały się w trakcie funkcjonowania. Ostatnią cechą, o jakiej jest mowa w definicji jest nieulotność, rozumiana jako niezmienność danych w niej zgromadzonych – w większości przypadków dostęp do hurtowni danych ogranicza się do załadowania do niej pewnej partii danych oraz wydobywania informacji już w niej zgromadzonych [2].

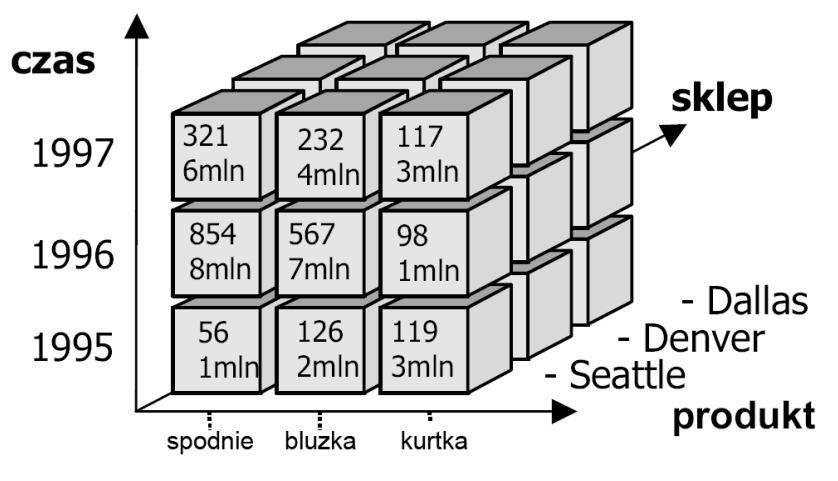
Hurtownie danych różnią się od operacyjnych transakcyjnych systemów bazodanowych pod wieloma względami. Po pierwsze, w większości przypadków systemy te skierowane są do różnych użytkowników. Z operacyjnych baz danych korzystają szeregowi pracownicy i klienci, podczas gdy hurtownie danych są z reguły używane przez analityków i menadżerów. Inny jest także cel przechowywania danych – systemy operacyjne mają za zadanie wspieranie codziennych działań przedsiębiorstwa, opierając się na pojedynczych transakcjach. W przypadku hurtowni danych sposób użycia jest inny – służą one do analizy dużych ilości danych oraz wyszukiwania w nich pewnych trendów, czy prawidłowości. W tym przypadku baza nie operuje na pojedynczych transakcjach, ale na dużych ich ilościach.

W porównaniu z operacyjnymi bazami danych w przypadku hurtowni danych stosowany jest także zupełnie inny sposób modelowania danych. W bazach transakcyjnych w zdecydowanej większości przypadków korzysta się ze zbioru znormalizowanych (w 3. formie normalnej)

tabel tak, aby zapewnić wydajność modyfikacji danych oraz aby wyeliminować potencjalne anomalie ze zmian wynikające. W przypadku hurtowni danych podejście jest inne – modyfikacje praktycznie nie występują, za to zapytania odnoszą się często do wielu rekordów i obejmują dużą ilość obiektów, które przy modelowaniu znormalizowanym przyjęłyby postać encji. W systemach tych dane przechowywane są w postaci wielowymiarowej.

### 2.3.2 Wielowymiarowy model danych

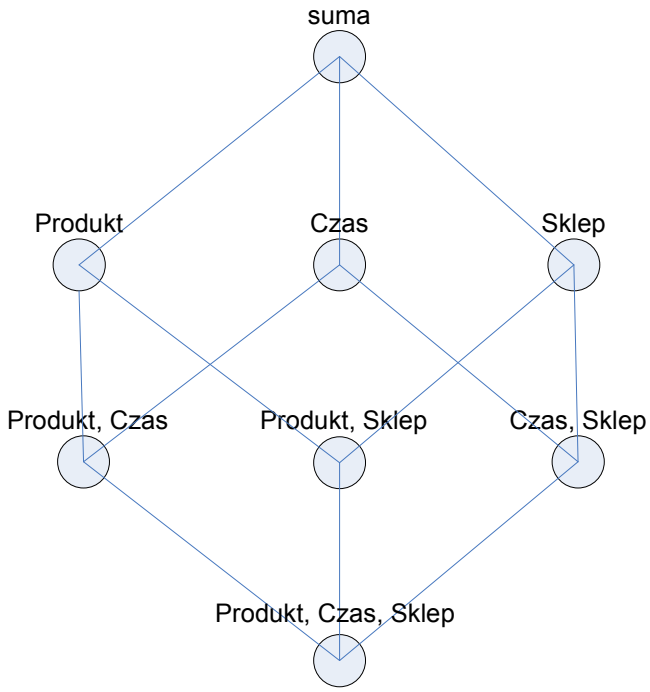
Model wielowymiarowy pozwala na prezentację danych jako *kostka danych*, a więc wielowymiarowego rozszerzenia spotykanych często tablic krzyżowych (ang. cross table, cross tab). Kostka danych zdefiniowana przez wymiary i fakty. Wymiary można rozumieć jako atrybuty klasyfikujące, według których można grupować i agregować dane. Na przykładzie kostki z danymi o sprzedaży sieci sklepów można by utworzyć wymiary określające *Sklep, Towar, Czas*. Wymiary te pozwalają na przechowywanie i analizę danych według dowolnej kombinacji wymienionych atrybutów, np. miesięczna sprzedaż poszczególnych towarów w zależności od sklepu, w którym zostały sprzedane. Fakty natomiast to wartości liczbowe, które znajdują się na przecięciu wymiarów. W naszym przykładzie mogłyby to być *Ilość sprzedanych produktów* lub *Wartość sprzedanych produktów.*




*Rys. 10. Kostka danych z trzema wymiarami*

Agregując (np. sumując) fakty po różnych d-elementowych kombinacjach wymiarów otrzymujemy d-wymiarowe prostopadłościany (ang. cuboid). Kostka danych jest często

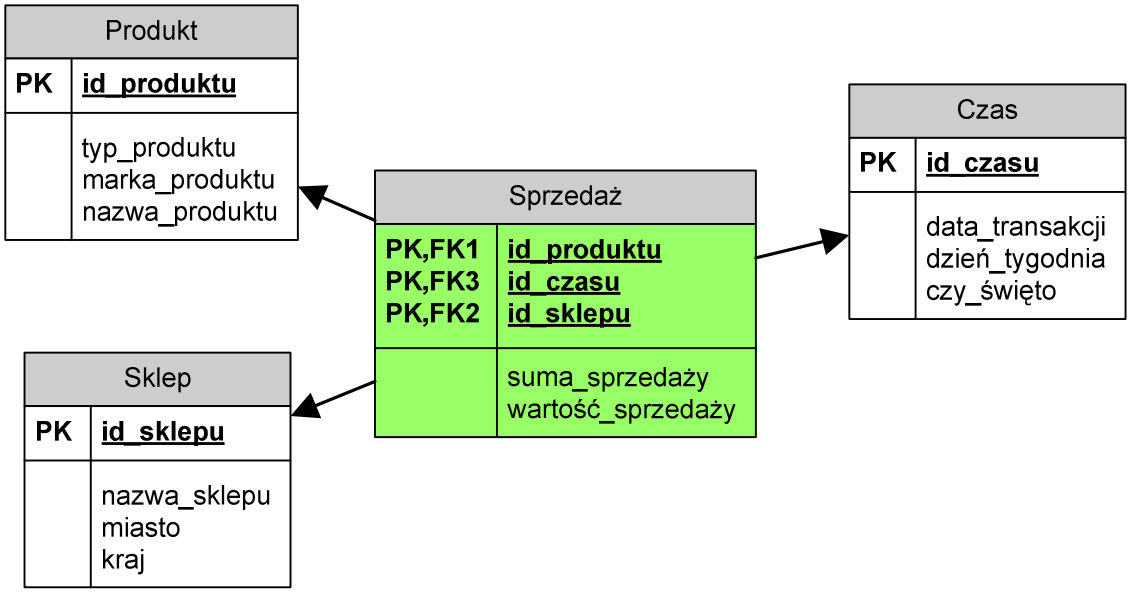
definiowana jako krata powstała z hierarchii wszystkich 0..n wymiarowych agregacji danych, gdzie n jest liczbą wszystkich dostępnych wymiarów.




*Rys. 11. Hierarchia agregacji trójwymiarowej kostki danych*

Kostka danych jest łatwą do zrozumienia metodą prezentacji danych, pozwala także na efektywne agregowanie oraz zmianę sposobu grupowania dużych ilości danych. W systemach OLAP (Online Analytical Processing) kostki danych są dominującym modelem danych, a w przypadku jednego z ich rodzajów – MOLAP (Multidimensional Online Analytical Processing) stanowią wręcz fizyczne struktury, w których magazynowane są informacje.

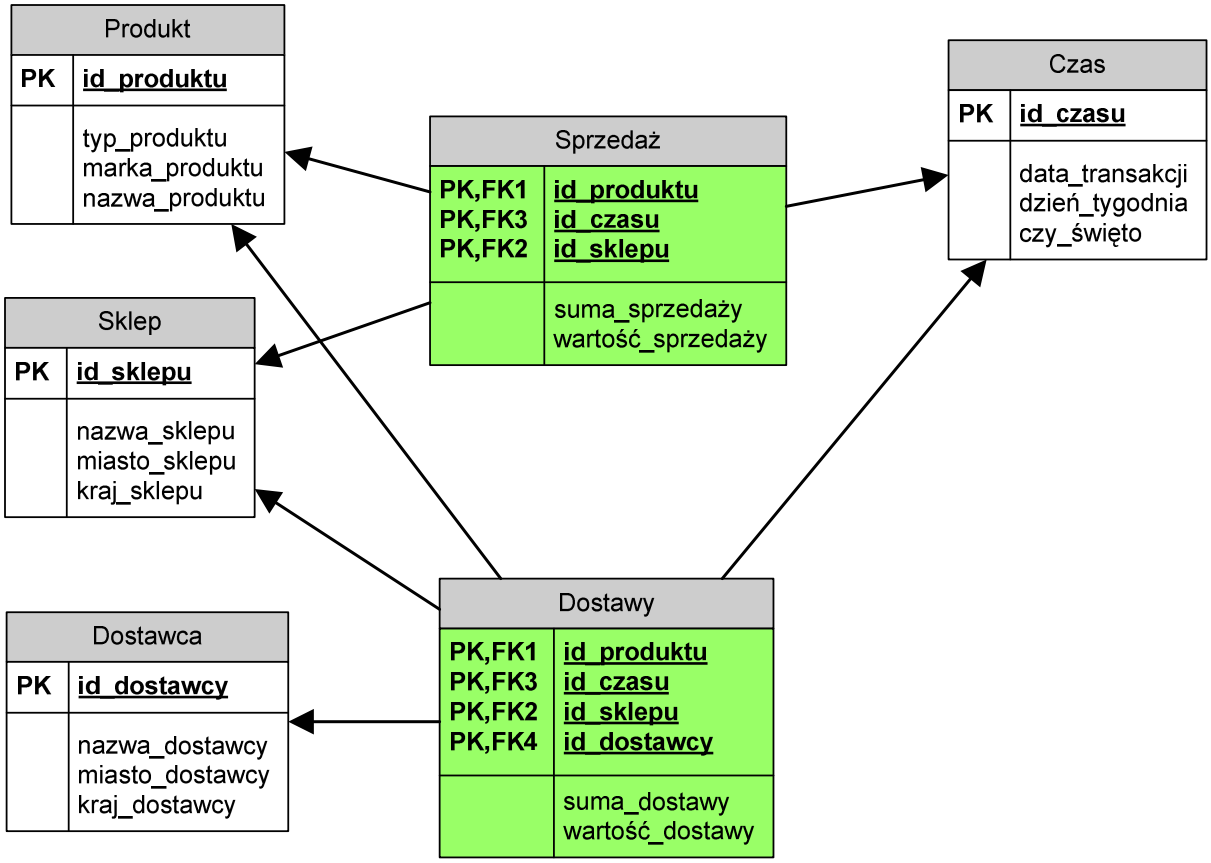
Wielowymiarowy model danych, pomimo wielu zasadniczych różnic w stosunku do znormalizowanych modeli encyjno-relacyjnych (ERM) może być jednak przedstawiany przy pomocy analogicznych diagramów, a implementowany w zwyczajnych relacyjnych bazach danych. Najprostszym, a jednocześnie preferowanym przez wielu, schematem wielowymiarowego modelu danych w relacyjnej bazie jest *schemat gwiazdy*. Schemat ten opiera się na centralnej tabeli faktów otoczonej tabelami wymiarów. Tabela faktów zawiera numeryczne dane zwane miarami (ang. measures) lub faktami (ang. facts) i połączona jest z tabelami wymiarów kluczami obcymi. Tabele wymiarów odpowiadają zdefiniowanym w we wstępnej fazie projektowania wymiarom i zawierają atrybuty określające jednostkę wymiaru, np. *Typ, Marka, Nazwa modelu* w przypadku wymiaru produktu.




*Rys. 12. Przykład schematu gwiazdy (star schema)*

Istotną cechą tego schematu jest fakt, że każdy wymiar opisany jest dokładnie jedną tabelą, a to może prowadzić do redundancji danych, np. jeśli sprzedajemy kilka produktów tego samego typu, to nazwa każdego typu może pojawić się w kilku rekordach. Ogólnie rzecz biorąc w schemacie gwiazdy tabele faktów są znormalizowane (3. forma normalna), natomiast tabele wymiarów są zdenormalizowane (2. forma normalna).

Pewną modyfikacją powyższego schematu jest schemat konstelacji faktów, który zakłada, że w projekcie może być więcej tabel faktów, połączonych ze współdzielonymi wymiarami. Oba te schematy są jest jednak zazwyczaj nazywane po prostu schematem gwiazdy, niezależnie od tego, czy występuje w nim jedna, czy wiele tabel faktów.

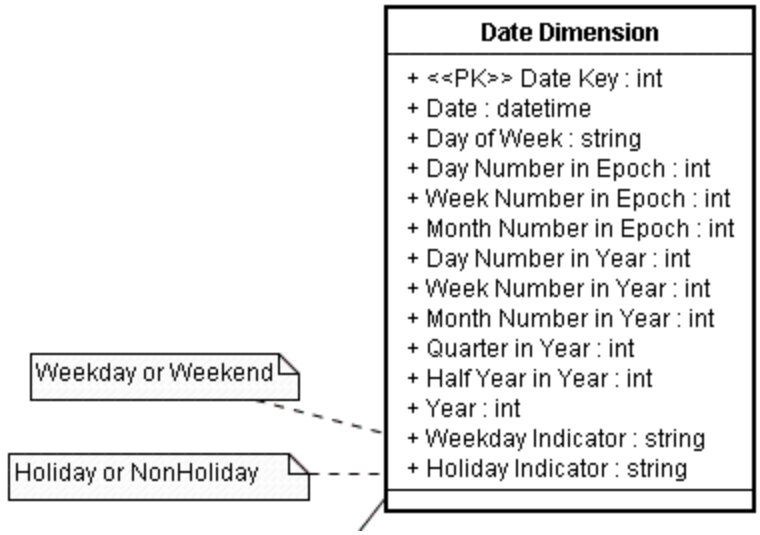



*Rys. 13. Przykład schematu konstelacji faktów*

W mojej pracy baza danych dla agenta *SDA* została zaprojektowana w oparciu o schemat konstelacji faktów / gwiazdy zgodnie z metodami i zaleceniami zawartymi głównie w książce Ralpha Kimballa i Margy Ross *The Data Warehouse Toolkit Second Edition* [5]. W kolejnych sekcjach przedstawione zostały tabele wymiarów i faktów, wraz z opisami ich atrybutów.

### 2.3.3 Tabele wymiarów

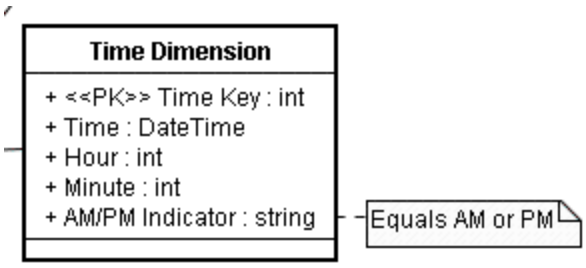
#### 2.3.3.1 Date Dimension




*Rys. 14. Schemat tabeli wymiaru daty*

Tablica wymiaru określającego datę. Każdy rekord oznacza konkretny dzień, wraz z informacjami dodatkowymi go określającymi, takimi jak m.in. numer dnia, tygodnia, miesiąca licząc od pewnego, ustalonego początku epoki czy flagami oznaczającymi czy dzień należy do weekendu (Weekday Indicator) i czy jest dniem świątecznym (Holiday Indicator). Większość pól nie wymaga raczej opisu. Te dodatkowe pola stanowią w gruncie rzeczy logikę biznesową dotyczącą dat - w przypadku chęci podziału roku na inne niż standardowe okresy, do tabeli tej wystarczy dodać odpowiednie pola określające numer nowego okresu i uzupełnić już istniejące wiersze o wartości nowo dodanych pól.

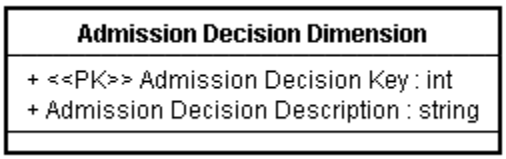
#### 2.3.3.2 Time Dimension




*Rys. 15. Schemat tabeli wymiaru czasu*

Wymiar określający czas z dokładnością do minut. W przypadku potrzeby grupowania rekordów wg niestandardowych okresów, np. 15-minutowych, należy dodać odpowiednie pole do tabeli i uzupełnić już istniejące wiersze o wartości tych pól. Tabela ta używana jest jedynie w przypadku potrzeby grupowania i agregacji danych liczbowych po polach tabeli (suma sprzedaży wg godzin lub przed/po południu) i z tego powodu dokładność do minuty jest wystarczająca. Większa dokładność jest możliwa, ale wymagałaby znacznego zwiększenia liczby rekordów w tabeli. Dla zastosowanej tu dokładności do minuty maksymalna wielkość tablicy to 24*60=1440 wierszy, w przypadku zwiększenia granularności do pojedynczej sekundy ilość ta wzrasta do 86400, co jest w większości przypadków zbyt dużą liczbą jak na tabelę wymiaru. Z tego też powodu, w sytuacji, gdy wymagana byłaby dokładność rejestrowania czasu pewnego zdarzenia z większą dokładnością, należałoby dołożyć pole z pełną datą i czasem (timestamp) do odpowiedniej tabeli faktów. [26].

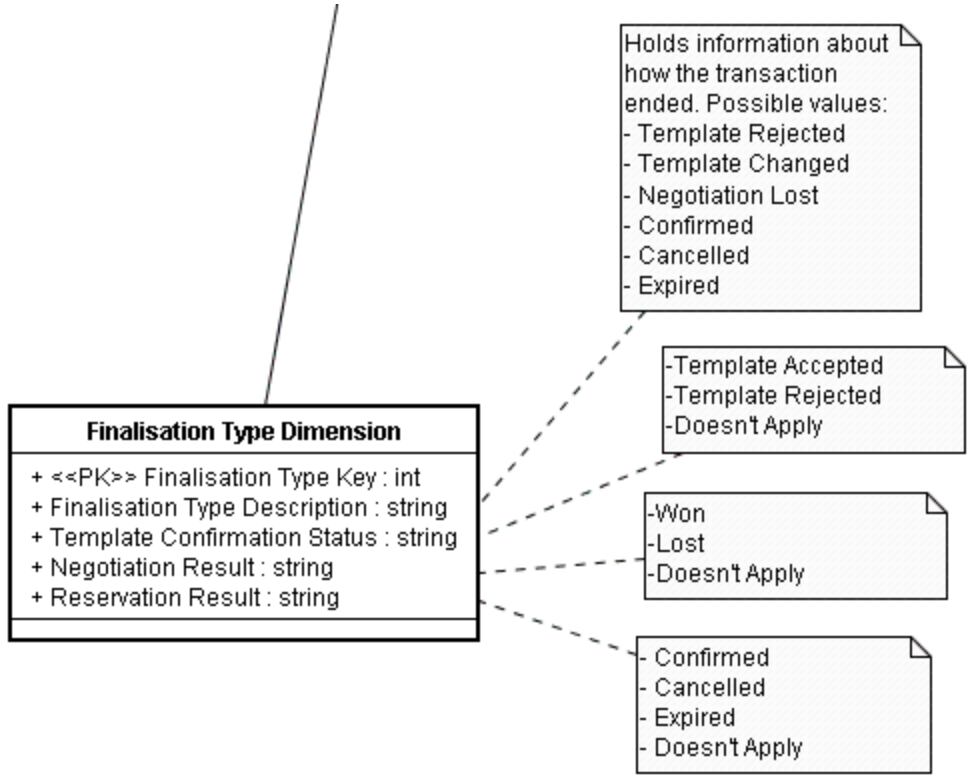
#### 2.3.3.3 Admission Decision Dimension




*Rys. 16. Schemat tabeli wymiaru rodzaju odpowiedzi*

Zawiera informacje o możliwym typie odpowiedzi dla klienta rejestrującego się do negocjacji. Może zawierać wartości takie jak: *Admitted*, oznaczające, że klient został dopuszczony do licytacji lub *Not Trusted*, gdy odmowa była spowodowana brakiem zaufania do klienta.

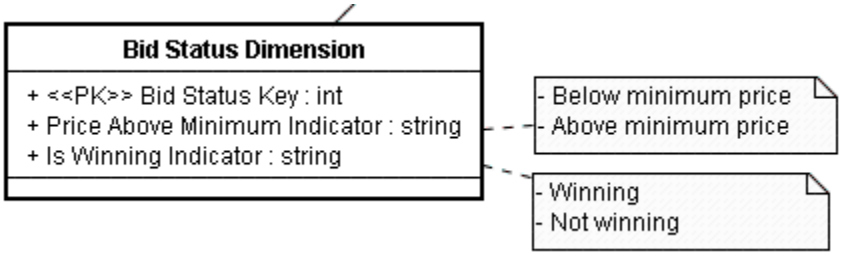
#### 2.3.3.4 Finalisation Type Dimension




*Rys. 17. . Schemat tabeli wymiaru typu zakończenia transakcji*

Zawiera możliwe sposoby, w jaki może zakończyć się interakcja z klientem. Pozwala na grupowanie transakcji na podstawie przystąpienia do negocjacji, wyniku negocjacji, czy sposobu zakończenia rezerwacji (potwierdzenie, rezygnacja, wygaśnięcie).

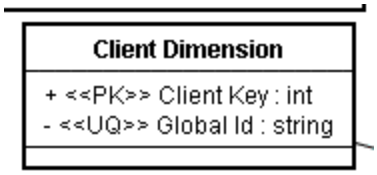
#### 2.3.3.5 Bid Status Dimension




*Rys. 18. Schemat tabeli wymiaru statusu oferty*

Tabela zawiera informacje o ofercie w kontekście całej licytacji: czy oferta była wygrywająca i czy przekroczyła cenę minimalną. Informacje te można by także wyczytać z pozostałych tabel, ale wymagałoby to dosyć kosztownych operacji łączenia z innymi tabelami faktów i mogłoby się okazać zbyt mało efektywne dla bardziej skomplikowanych kwerend wykorzystujących na przykład grupowanie i agregację.

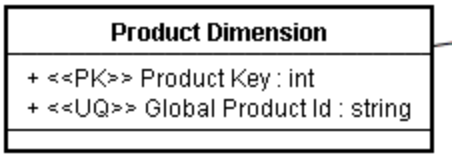
#### 2.3.3.6 Client Dimension




*Rys. 19. Schemat tabeli wymiaru klienta*

Jest to tabela opisująca klienta sklepu. Na chwilę obecną przyjęto założenie, że informacje o klientach trzymane są w osobnej, ontologicznej, bazie danych i nie będą bezpośrednio włączane do bazy wielowymiarowej. Nie jest wykluczone, że w miarę precyzowania się zastosowań bazy danych, tabela ta będzie uzupełniana o często używane w zapytaniach atrybuty dotyczące klienta w taki sposób, aby usprawnić raportowanie w oparciu o nie, natomiast na razie służy ona tylko do łączenia być może długiego i niewygodnego w stosowaniu operacyjnego identyfikatora klienta, ze sztucznym kluczem stosowanym w obrębie hurtowni. Więcej na temat przewagi stosowania kluczy sztucznych nad kluczami naturalnymi w wielowymiarowych bazach danych znaleźć można w [5, str.58].

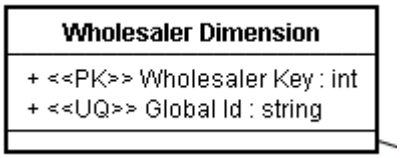
#### 2.3.3.7 Product Dimension




*Rys. 20. Schemat tabeli wymiaru produktu*

Jest to tabela opisująca sprzedawany towar. Podobnie jak w przypadku tabeli klienta, zawiera jedynie globalny, operacyjny identyfikator produktu, zaś pozostałe dane przechowywane są w odrębnej bazie ontologicznej. Uzupełnianie tej tabeli o dodatkowe pola może następować w ramach optymalizacji zapytań i będzie zdeterminowane konkretnymi potrzebami.

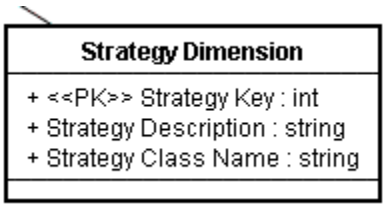
#### 2.3.3.8 Wholesaler Dimension




*Rys. 21. Schemat tabeli wymiaru dostawcy*

Jest to tabela wymiaru opisująca dostarczyciela towaru do magazynu. Zawiera jedynie jego operacyjny identyfikator, gdyż na tę chwilę nie zostały sprecyzowane cechy hurtownika, które mogą być wykorzystane przy wnioskowaniu na ich temat.

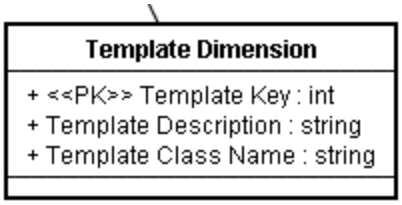
#### 2.3.3.9 Strategy Dimension




*Rys. 22. Schemat tabeli wymiaru strategii*

Wymiar opisujący strategię stosowaną przez sklep w trakcie negocjacji. Zawiera opis strategii oraz pełną nazwę klasy opisującej sposób prowadzenia przez sklep negocjacji. Parametry strategii, które mają często zmieniające się, ciągłe wartości (jak na przykład cena minimalna) umieszczone są w odpowiedniej tabeli faktów opisującej licytacje.

#### 2.3.3.10 Template Dimension

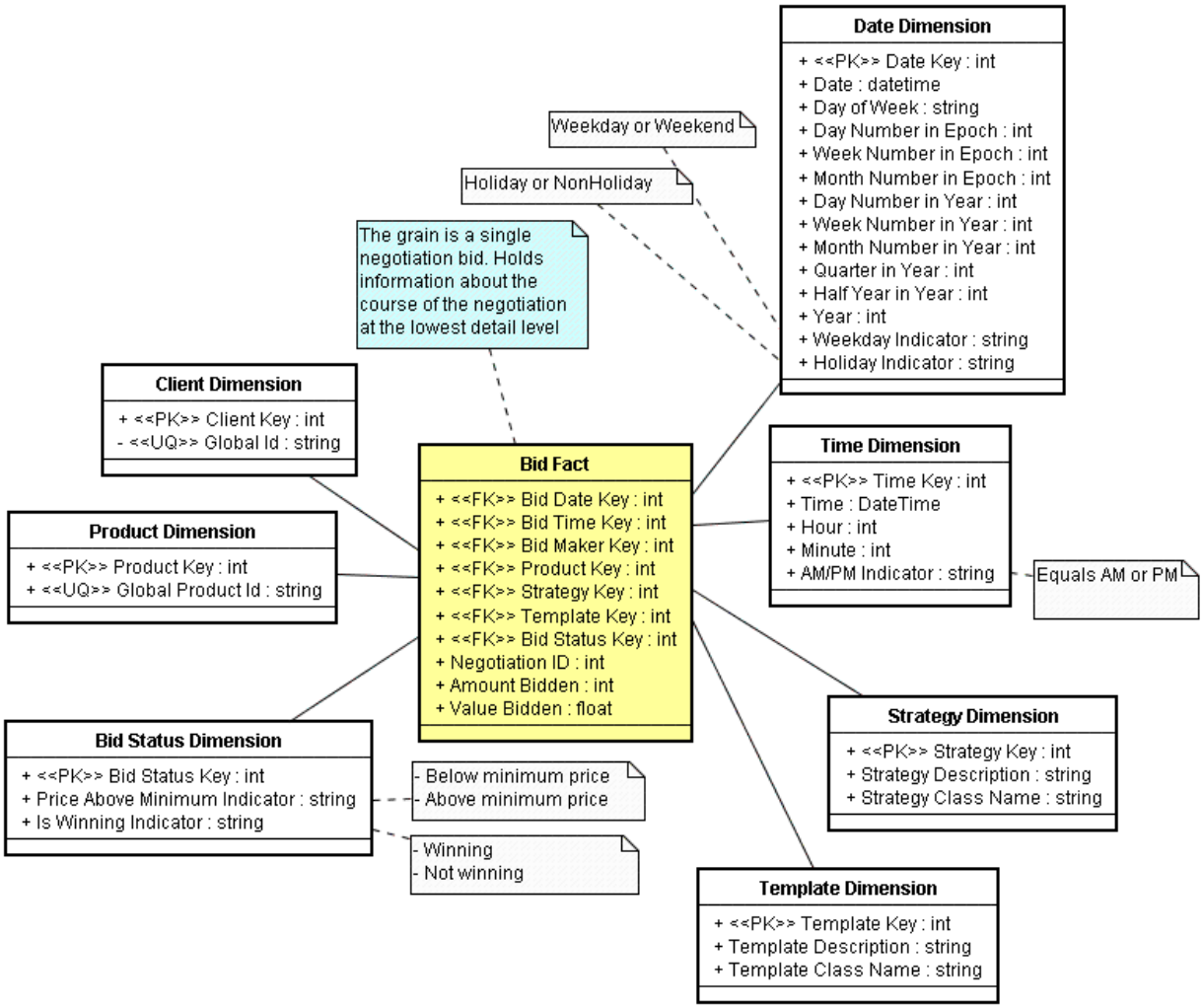



*Rys. 23. Schemat tabeli wymiaru szablonu negocjacji*

Wymiar opisujący szablon aukcji wykorzystywany w licytacjach. W tabeli tej, podobnie jak w przypadku strategii przechowywane są jak na razie jedynie opis i nazwa klasy opisującej sposób w jaki odbywa się negocjacja, a wszelkie parametry o wartościach ciągłych przechowywane są w tabelach faktów.

### 2.3.4 Tabele faktów

#### 2.3.4.1 Bid Fact Table




Najbardziej szczegółowa i największa pod względem liczby wierszy tabela faktów – rekord zawiera informacje o pojedynczej ofercie klienta w trakcie negocjacji. Tabela ta pozwala na śledzenie przebiegu negocjacji w dowolny sposób – w obrębie pojedynczej negocjacji, dla danego klienta, a także wyciągania agregowanych wniosków na temat ofert kupujących takich, jak np. średnia cena oferowana za jednostkę towaru, lub średnia liczba jednostek licytowana. Dane można grupować także po statusie oferty takim jak: poniżej ceny minimalnej, powyżej ceny minimalnej (ale nie wygrywająca), oferta wygrywająca, dzięki czemu możliwe jest zliczanie ofert, ilości licytowanych jednostek i ich wartości dla każdego z

tych etapów oraz łatwe monitorowanie wpływu zastosowanego template'a oraz strategii na przebieg negocjacji.

Wiedza, którą można pozyskać z danych o ofertach może posłużyć przede wszystkim przy badaniu zachowania *BA* w trakcie negocjacji, a więc przy wyznaczaniu poziomu zaufania do klienta oraz przy ustalaniu najbardziej korzystnego szablonu i strategii negocjacji.

#### 2.3.4.2 Negotiation Fact Table

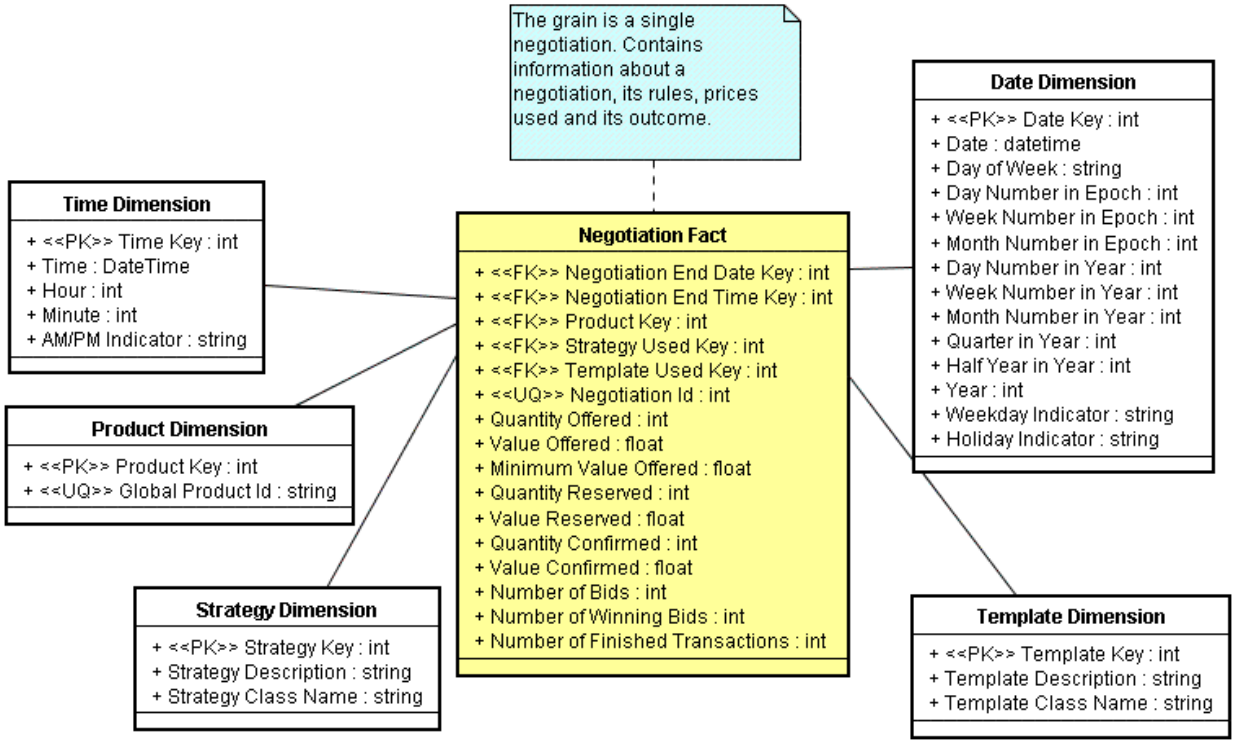
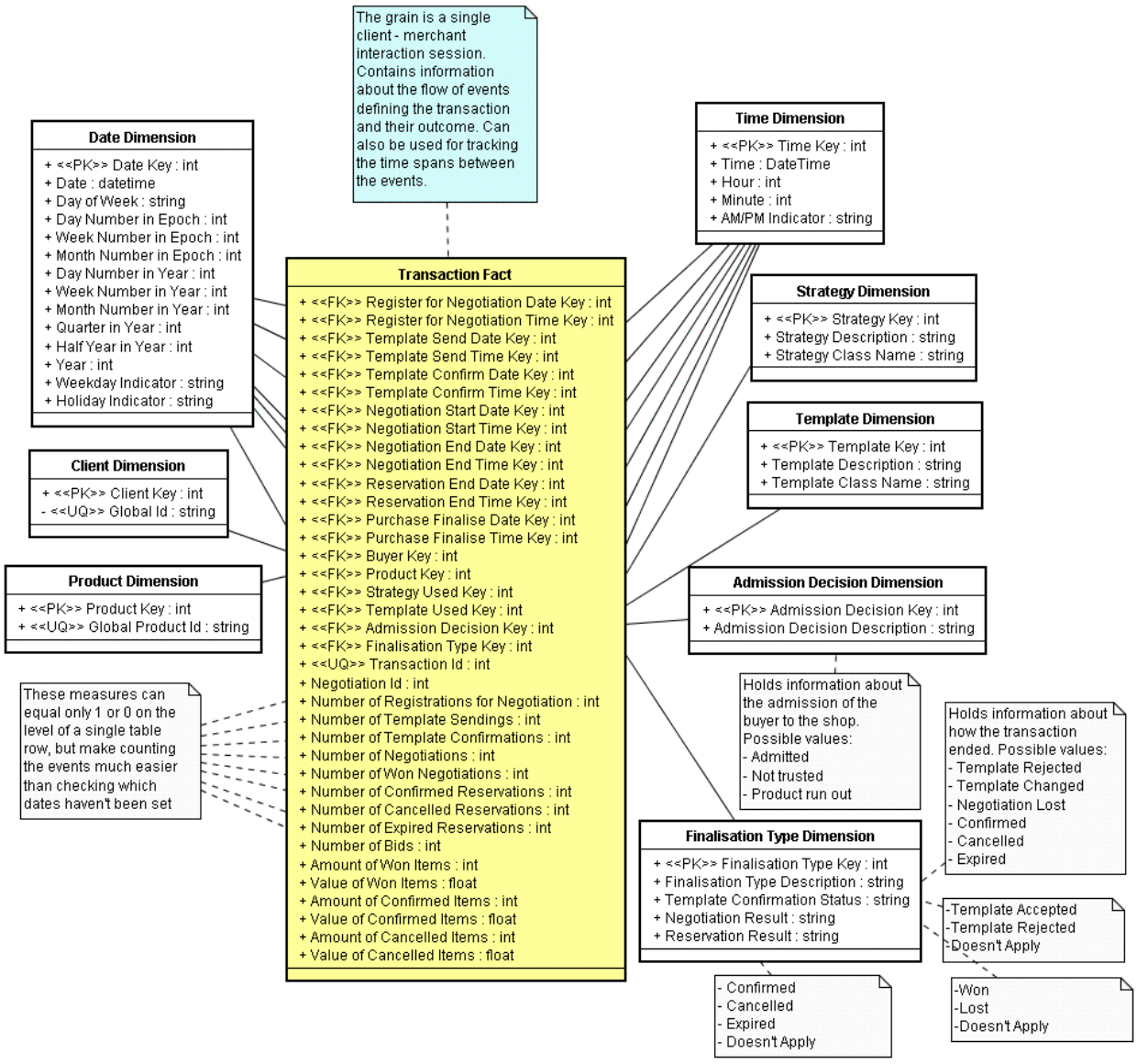



Tabela Negotiation Fact zawiera informacje o przeprowadzonych negocjacjach. Każdy wiersz to pojedyncza negocjacja, bez podziału na klientów. Grupować można po czasie zakończenia negocjacji, produkcie, strategii i szablonie. Najważniejsze miary jakie są w niej przechowywane, to te dotyczące ilości wystawionego towaru i jego wartości przy cenie startowej i minimalnej – dane te występują tylko na tym poziomie szczegółowości i tylko w tej tabeli. Oprócz tego znalazły się tam także zbiorcze dane takie jak ilość i wartość zarezerwowanego towaru, liczba ofert, ofert wygrywających oraz zakończonych transakcji. Są to informacje dostępne także w tabelach Bid Fact oraz Transaction Fact, uznałem jednak, że są na tyle istotne w kontekście wcześniej wymienionych informacji, że warto je umieścić

także tutaj. Korzystając z tej tabeli można przede wszystkim wyciągać informacje o tym, w jaki sposób cena startowa i minimalna wpływa na wynik negocjacji i procesu sprzedaży oraz na ile trafnie określamy liczbę wystawianych w jednej negocjacji przedmiotów w stosunku do faktycznego popytu na nie.

### 2.3.4.3 Transaction Fact Table




Największa pod względem liczby kolumn tabela schematu, Transaction Fact Table zawiera informacje dotyczące całego przebiegu sesji komunikacji sklepu z pojedynczym klientem, wyjąwszy jedynie przebieg samej negocjacji, który opisany jest w tabeli Bid Fact Table.

Zbudowana jest wg schematu Accumulating Snapshot [5, str.244] i pokazuje przede wszystkim do jakiego etapu transakcji doszedł w interakcji ze sklepem klient oraz ile trwało przechodzenie przez kolejne stadia transakcji. Długa lista dat służy właśnie do wnioskowania na temat czasu trwania kolejnych etapów, chociażby ile czasu minęło zanim klient potwierdził rezerwację lub po jakim czasie zaakceptował otrzymany protokół negocjacji – wiedza istotna przy ustalaniu poziomu zaufania. Długa lista miar, takich jak Number of Registrations, Number of Negotiations itp, których wartości w obrębie wiersza mogą być równe tylko 1 lub 0 (w trakcie jednej transakcji klient może zarejestrować się i wziąć udział w negocjacji co najwyżej raz), pozwala natomiast w łatwy sposób określać na jakim etapie znajduje się/zakończyła się rozpatrywana interakcja oraz zliczać zdarzenia takie jak wycofanie się klienta po otrzymaniu protokołu w obrębie wielu transakcji.

Interesującą pod względem zapisu w tej tabeli jest sytuacja, gdy protokół negocjacji zostaje zmieniony w momencie, gdy klienci zainteresowani danym produktem są już zarejestrowani i przyjęli poprzednie parametry. W tym przypadku, klient ma możliwość odrzucenia nowego protokołu i odstąpienia od negocjacji lub przyjęcia protokołu i przystąpienia do aukcji. W tabeli transakcji sytuacja taka zostanie odwzorowana przy pomocy dwóch rekordów – jednego opisującego interakcję do momentu dostarczenia nowych parametrów i połączoną z wierszem tabeli Finalisation Type oznaczającym zmianę protokołu (Wartość *Template Changed* w polu Finalisation Type Description); oraz drugiego rekordu, opisującego interakcję z tym samym klientem od momentu zmiany protokołu. Data i czas rejestracji klienta oraz przesłania mu protokołu powinny w przypadku drugiego rekordu mieć wartość zgodną z momentem wysłania nowszego protokołu. W takiej sytuacji dla każdego klienta po zmianie parametrów powinien być wygenerowany nowy identyfikator transakcji, aby nie powodować niejednoznaczności co do protokołu użytego w danej transakcji.

Informacje o liczbie złożonych przez klienta w trakcie negocjacji ofert i wartości zarezerwowanych przedmiotów są jedynie agregowanymi danymi dostępnymi także w tabeli Bid Fact Table i pod tym względem występuje tu redundancja, ale na razie, ze względu na wygodę korzystania z nich w kontekście pozostałych informacji o transakcji pozostawiłem je również w tej tabeli

W przypadku, gdy zaszłaby istotna konieczność wykorzystania informacji o czasie jaki minął między kolejnymi punktami kontrolnymi w procesie transakcji (na przykład ile sklep czekał

na akceptację szablonu negocjacji przez klienta), to do tabeli tej powinny zostać dołączone także pola z dokładnym czasem i datą poszczególnych wydarzeń, nawet pomimo istnienia odniesień do tabel wymiarów daty i czasu. Obliczanie długości tego typu okresów jest po prostu znacznie łatwiejsze i dokładniejsze niż korzystanie jedynie z tabel wymiarów, zwłaszcza, że wymiar czasu ma dokładność pojedynczej minuty (patrz też 2.3.3.2). Zupełna zamiana kluczy obcych do tabel wymiarów daty i czasu na timestampy jest jednak odradzana ze względu na utratę łatwej możliwości grupowania po różnych rodzajach okresów czasu zdefiniowanych w tabelach wymiarów. Więcej na ten temat tego dylematu można znaleźć w [26].

#### 2.3.4.4 Supply Fact Table

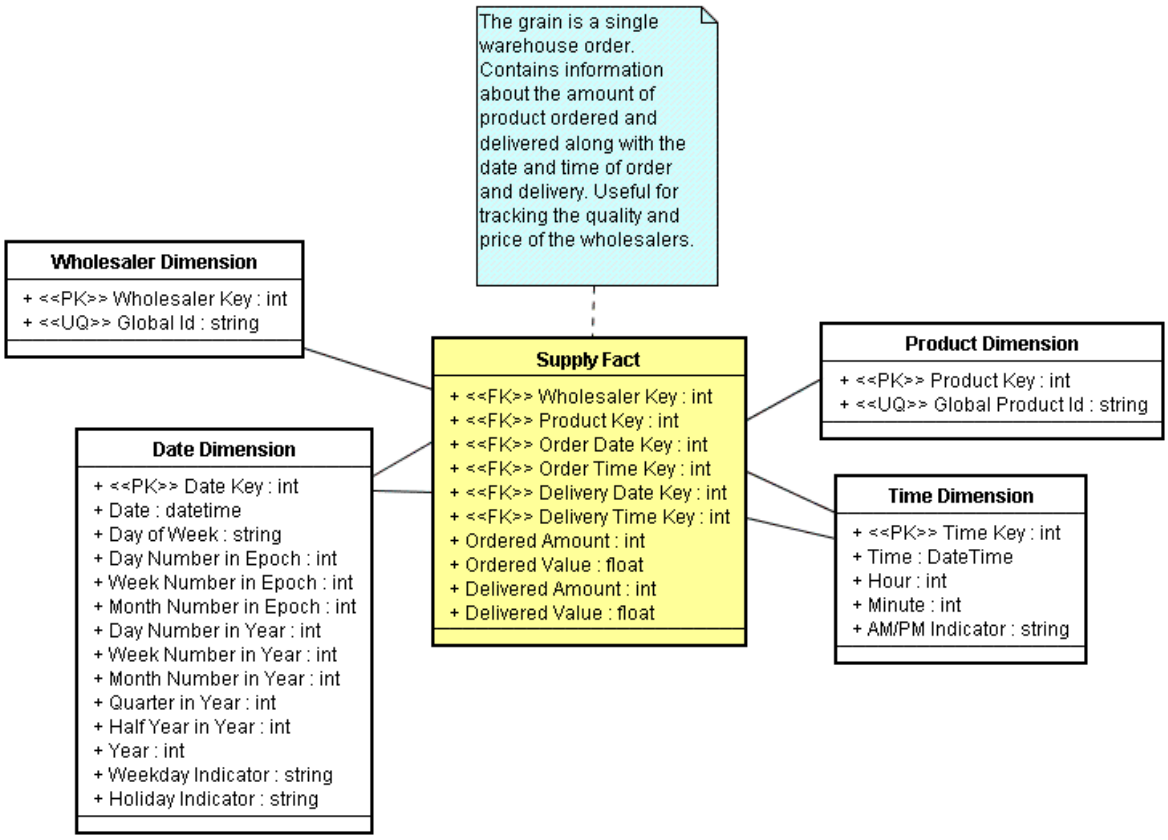



Tabela ta zawiera informacje o zamówieniach produktu złożonych do hurtowników. Czas zamówienia i dostawy pozwalają na śledzenie ile czasu zajęło dostawcy sprowadzenie towaru, natomiast dzięki podwójnym polom na ilość i wartość towaru można zachować informacje, czy kontrakt został dokładnie wypełniony przez hurtownika. Z tabeli tej można

więc pośrednio wnioskować o czynnikach wpływających na wizerunek hurtownika (grupując po polu Wholesaler Key) oraz o czasie potrzebnym na sprowadzenie towaru (grupując po Product Key i licząc różnicę między datą/czasem dostarczenia i złożenia zamówienia).

### 2.3.4.5 Demand Fact Table

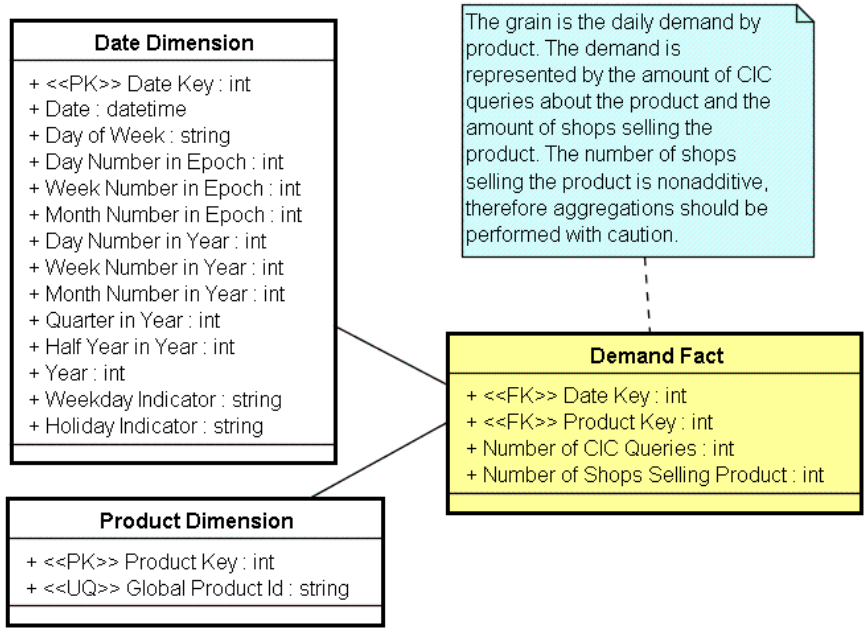



Tabela ta rejestruje dane obrazujące popyt na dany produkt pochodzące z *CIC*, a więc dzienną liczbę zapytań o dany produkt oraz liczbę sklepów go oferujących. Umożliwia agregację danych po czasie i produkcie. Wiedza pochodząca z tego źródła wraz z danymi o liczbie klientów odwiedzających sklep może być istotna przy analizie wizerunku kupca wśród klientów i wpływaniu na ten wizerunek.

Warto zauważyć, że miara określająca ilość sklepów sprzedający dany towar nie jest addytywna przy grupowaniu, należy więc zachować uwagę przy konstruowaniu zapytań dotyczących tej tabeli.

Jeżeli *CIC* nie udostępniałby informacji o ilości zapytań klientów o produkty lub *SDA* nie korzystałby z tych usług, to w polu liczby zapytań znajdowałyby się wartości null.

#### 2.3.4.6 Inventory Snapshot Fact Table

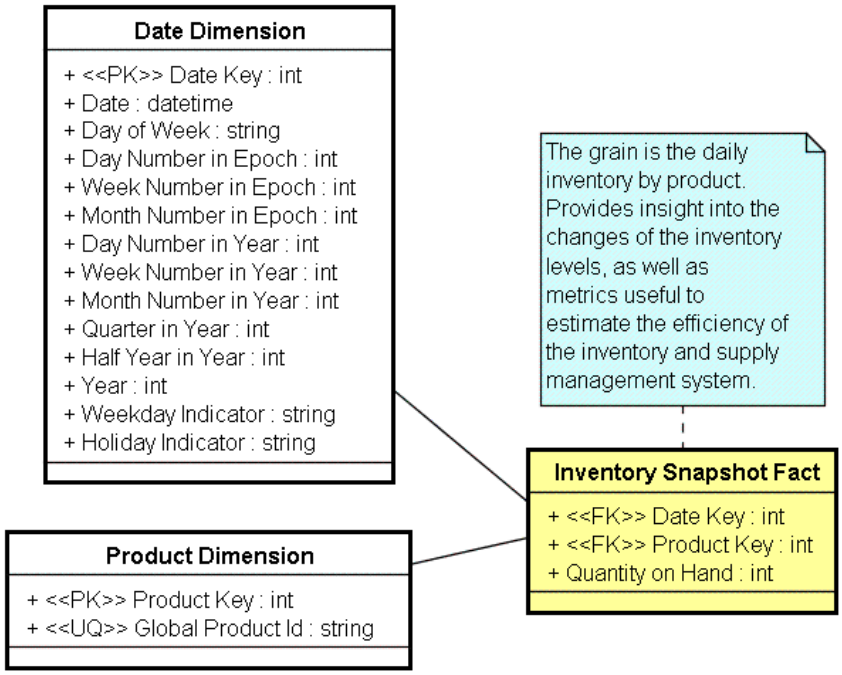



Tabela ta zawiera informacje o dziennym stanie magazynowym w sklepie. Jedyną mierzoną wartością jest ilość aktualnie posiadanego towaru, a grupowanie możliwe jest po dacie i produkcie. Informacje przechowywane w tej tabeli można by pozyskać pośrednio, wykorzystując tabele Sale Fact i Supply Fact – porównując liczbę przedmiotów sprzedanych z liczbą dostarczonych, ale ze względu na wydajność i niezręczność takiego rozwiązania oraz na fakt, że tabela Inventory Snapshot ma bardzo niewielkie rozmiary wydało mi się usprawiedliwione wprowadzić ją do schematu. Przy obecnym projekcie uznałem za wystarczające aby dane te przechowywane były z dokładnością do pojedynczego dnia. Większa rozdzielczość byłaby oczywiście możliwa – aby to osiągnąć trzeba by tylko rozszerzyć tabelę o dodatkowy klucz obcy do tabeli wymiaru czasu.

### 2.3.5 Uwagi implementacyjne

Przedstawiony w powyższych punktach projekt bazy danych opisuje całą działalność części sklepowej omawianego systemu e-commerce. Ze względu jednak na to, że w mojej pracy zajmuję się jedynie wycinkiem tego systemu, nie były mi potrzebne wszystkie dane, które mogłaby zgromadzić zaprojektowana baza. Z tego też powodu zdecydowałem o implementacji jedynie tej części tabel omówionych powyżej, które są istotne dla zadań, jakie

realizuje tworzony przeze mnie fragment agenta *SDA*. Tabele, które zostały zaimplementowane odpowiadają wymiarom: *Date* (2.3.3.1), *Time* (2.3.3.2), *Client* (2.3.3.6), *Product* (2.3.3.7), *Wholesaler* (2.3.3.8), *Strategy* (2.3.3.9) oraz *Template* (2.3.3.10).

## 2.4 Prognozowanie sprzedaży produktu

Pierwszym z adaptacyjnych zadań agenta *SDA*, które zostały zaprojektowane i zaimplementowane w ramach tej pracy jest tworzenie i dostarczanie agentowi zarządzającemu magazynem prognoz sprzedaży produktów. Wstępne założenia i cele tego modułu zostały przedstawione w podrozdziale 1.5, w tej sekcji znajdzie się więc przede wszystkim opis sposobu w jaki tę funkcjonalność udało się zrealizować. W kolejnych akapitach przedstawione są więc zastosowane metody aktualizacji horyzontu predykcji sprzedaży produktów oraz tworzenia samych prognoz.

### 2.4.1 Ustalanie horyzontu predykcji

Jak zostało przedstawione w sekcji 1.5.2.3 horyzont predykcji powinien być ustalony w taki sposób, aby w jego obrębie sprzedaż produktu stanowiła stacjonarny szereg czasowy, a więc jej wartość średnia i odchylenie nie zależały od czasu obserwacji. Dzięki temu, że w ramach pojedynczego okresu predykcji nie występuje trend ani sezonowość, agent *WA* może podzielić wielkość zapotrzebowania na kilka mniejszych dostaw. Proces ustalania długości horyzontu prognozy sprowadza się więc do znalezienia najdłuższego okresu czasu, w obrębie którego sprzedaż danego towaru można uznać za szereg stacjonarny.

W celu sprawdzania, czy szereg czasowy jest stacjonarny wykorzystany została metoda zaczerpnięta z przykładu [34]. W teście tym rozpoczynamy od porównania kolejnych elementów szeregu czasowego ze średnią ich wartością – załóżmy, że wartości mniejsze od średniej oznaczymy liczbą -1, a większe 1. W następnym kroku zliczamy liczbę zmian znaku w tak zbudowanym ciągu liczb -1 i 1. Rozstrzygnięcie, czy dany szereg czasowy jest stacjonarny odbywa się na podstawie P-wartości otrzymanej w poprzednim kroku liczby zmian znaku.

Algorytm rozpoczynamy przyjmując za horyzont najdłuższy czas, na jaki pozwala nam liczba danych o sprzedaży danego produktu. Następnie dzielimy zbiór obserwacji na szeregi czasowe o granularności jednego dnia o długości równej aktualnemu horyzontowi i dla każdego z nich liczymy P-wartość według omówionej metody. Jeśli średnia P-wartość dla wszystkich szeregów jest większa od pewnej, ustalonej z góry wartości, uznajemy, że aktualny horyzont czasowy spełnia wymagania i przerywamy obliczenia. W przeciwnym razie horyzont zostaje skrócony i procedura jest powtarzana. Iteracja trwa dopóki nie zostanie znaleziona długość horyzontu odpowiadająca wymaganiom, lub nie osiągniemy minimalnej długości horyzontu – ustalanej jako parametr algorytmu.

### 2.4.2 Przygotowywanie prognozy

Prognozowanie sprzedaży danego produktu zostało zrealizowane przez analizę szeregu czasowego wielkości sprzedaży z długością okresu równą wcześniej ustalonemu horyzontowi predykcji. Istotną informacją jest, że otrzymany szereg czasowy w przypadku niektórych produktów może nie być stacjonarny. Z tego też powodu zdecydowałem się na zastosowanie takich metod jego analizy, które uwzględniają w modelu trend i sezonowość. Wybór padł na metodę wygładzania wykładniczego.

W kolejnych sekcjach przedstawiona zostanie wybrana przez mnie metoda wraz z jej modyfikacjami, ze szczególnym uwzględnieniem praktycznych jej aspektów. Zostały one opracowana przede wszystkim w oparciu o podręcznik *NIST/SEMATECH e-Handbook of Statistical Methods* [28]

#### 2.4.2.1 Wygładzanie wykładnicze

Metoda wygładzania wykładniczego (ang. exponential smoothing) w najprostszej postaci polega na wyznaczaniu kolejnego elementu szeregu jako ważonej średniej arytmetycznej elementów poprzednich, gdzie waga kolejnych poprzedników maleje wykładniczo w taki sposób, aby najnowsze obserwacje były ważniejsze od wcześniejszych. Jeżeli przez $S_i$ oznaczymy wygładzoną, a przez $y_i$ prawdziwą obserwację w i-tym okresie czasu, to wygładzoną wartość w okresie t możemy wyznaczyć ze wzoru:

$$S_t = \alpha \ y_{t-1} + (1-\alpha)\ S_{t-1} \qquad 0 < \alpha \le 1 \qquad t \ge 3$$

Wartość pierwszego wygładzonego elementu, a więc $S_2$ można ustawić na wartość pierwszej obserwacji – $y_1$ lub na przykład uśrednioną wartość kilku pierwszych obserwacji. W powyższym równaniu parametr α nazywamy współczynnikiem wygładzania. W metodzie tej dokładność dopasowania modelu do prawdziwych obserwacji zależy od doboru współczynnika α oraz pierwszej wygładzonej wartości, czyli $S_2$. Niezależnie natomiast od ich doboru ten podstawowy wariant wygładzania wykładniczego może być stosowany jedynie dla stacjonarnych szeregów czasowych, nie wykazujących cech trendu ani sezonowości.

Modyfikacją powyższego algorytmu, uwzględniającą składnik trendu, jest metoda podwójnego wygładzania wykładniczego. Zmianą jest dodanie do modelu drugiego równania oraz drugiego współczynnika wygładzania. Odpowiadają one za trend i jego zmiany w czasie. Zmodyfikowana metoda opisana jest następującymi równaniami:

$$S_t = \alpha \ y_t + (1-\alpha)(S_{t-1} - b_{t-1}) \qquad 0 \le \alpha \le 1$$
$$b_t = \gamma(S_t - S_{t-1}) + (1-\gamma)b_{t-1} \qquad 0 \le \gamma \le 1$$

Pierwsze z nich opisuje aktualizację wartości wygładzonej, ale w oparciu o wartość trendu z poprzedniego okresu. Drugie aktualizuje trend w znajomy sposób – jako ważoną średnią trendu aktualnego i poprzednich jego wartości, z wagami malejącymi wykładniczo.

Warto zauważyć, że wygładzona wartość w okresie t liczona jest na podstawie prawdziwej obserwacji w tym samym okresie, a więc powyższych wzorów nie można bezpośrednio zastosować w celu obliczenia prognozy. Wartość prognozy, a więc obserwacji, dla której nie mamy jeszcze danych można wyznaczyć ze wzoru:

$$F_{t+1} = S_t + b_t$$

Wraz ze zwiększeniem skuteczności metody w przypadku szeregów czasowych z trendem, przybyło także współczynników, które należy ustalić aby dopasować model do danych: współczynnik wygładzania obserwacji α i trendu γ oraz początkowe wygładzone wartości obserwacji $S_1$ i trendu $b_1$.

Ostatnią z metod wygładzania wykładniczego jest metoda Holta-Wintersa, zwana też metodą potrójnego wygładzania wykładniczego. Powstała z myślą o szeregach, w których występuje

nie tylko trend, ale i sezonowość. Składają się na nią trzy równania aktualizujące poszczególne składniki:

$$S_t = \alpha \frac{y_t}{L_{t-L}} + (1-\alpha)(S_{t-1} + b_{t-1})$$
$$b_t = \gamma (S_t - S_{t-1}) + (1-\gamma) b_{t-1}$$
$$I_t = \beta \frac{y_t}{S_t} + (1-\beta) I_{t-L}$$

Pierwsze z nich odpowiada za wygładzanie obserwacji, druga za wygładzanie trendu, trzecie zaś za wygładzanie sezonowości. Prognozę na okres $S_{t+1}$ wyznaczamy ze wzoru:

$$F_{t+1} = (S_t + b_t) I_{t-L+1}$$

Parametrami modelu są współczynniki wygładzania α, γ i β. Należy także zainicjalizować $S_1$, $b_1$ oraz przynajmniej jeden cykl wartości sezonowości, a więc $I_1$ .. $I_L$. Z tego też powodu metoda ta może być stosowana jedynie wtedy, gdy dysponujemy już danymi z przynajmniej jednego pełnego cyklu, a więc L okresów.

#### 2.4.2.2 Wykorzystane metody

Patrząc na wymagania, jakie zostały postawione zadaniu predykcji, zwłaszcza jeśli chodzi o uwzględnienie sezonowości, wyłania się wybór ostatniej z przedstawionych metod – potrójnego wygładzania wykładniczego. Niestety, przy inicjalizacji modelu wymaga ona danych z conajmniej jednego, a najlepiej większej ilości pełnych cykli sezonowości. Z tego też powodu nie jest możliwe jej stosowanie w przypadku, gdy nie dysponujemy jeszcze informacjami o zbyt wielu minionych okresach. Do rozwiązania tego zagadnienia podszedłem w dwojaki sposób.

Po pierwsze, zastosowałem zmienną długość cyklu sezonowości w zależności od ilości dostępnych danych – a więc, na przykład, jeśli posiadamy dane zaledwie z trzech tygodni, to stosowany jest cykl długości 7 dni (można analizować sezonowość w zależności od dnia tygodnia), natomiast jeśli dostępne dane historyczne obejmują dwa lata lub więcej, to możemy rozpatrywać sezonowość w ciągu całego roku. Dzięki temu można badać pewne

zjawiska sezonowości nawet w przypadku niewielkiej wiedzy historycznej. Rozpatrywane długości cykli, to tydzień, miesiąc lub rok, choć listę tę można rozszerzyć.

Po drugie, w sytuacji, gdy liczba obserwacji jest mniejsza niż dwukrotność długości najkrótszego dopuszczalnego cyklu stosowana jest metoda podwójnego wygładzania, która nie ma dolnego ograniczenia ilości danych potrzebnych do inicjalizacji.

Ostatnią kwestią jaka pozostała do omówienia, jest sposób szacowania możliwego odchylenia prawdziwej sprzedaży od wartości prognozowanej. W tym celu zastosowałem po prostu średnie odchylenie bezwzględne (ang. mean absolute deviation) obliczone na podstawie otrzymanego modelu i wykorzystanych danych historycznych.

Istotnym zagadnieniem we wszystkich wymienionych metodach jest dobór właściwych parametrów wygładzania, gdyż wpływa on dosyć silnie na skuteczność predykcji. W ogólnym przypadku dostosowywanie modelu do posiadanych danych polega na znalezieniu takich wartości tych parametrów, aby zminimalizować błąd średniokwadratowy (ang. mean squared error, MSE) na danych historycznych. W obecnej wersji programu zastosowano bardzo prostą metodę ustalania współczynników wygładzania – sprawdzenie każdej kombinacji wartości parametrów w przedziałach (0 ; 1) w odstępach co 0.1. Wielkość odstępu, można zmieniać przy pomocy pliku konfiguracyjnego.

#### 2.4.2.3 Uwagi implementacyjne

Jeśli chodzi o implementację samych metod podwójnego i potrójnego wygładzania wykładniczego, to zdecydowałem się na wykorzystanie istniejącej biblioteki oferującej wymienione algorytmy. Pakiet, o którym mowa to OpenForecast (http://openforecast .sourcefourge.net) – biblioteka open-source zawierająca podstawowe statystyczne modele predykcyjne, możliwe do wykorzystania dla dowolnych danych. Biblioteka ta dostarcza także szkielet do tworzenia własnych algorytmów. Wykorzystane przeze mnie metody zaimplementowane są w klasach **DoubleExponentialSmoothingModel** i **TripleExponentialSmoothingModel.**

W implementacji zawartej w pakiecie OpenForecast inicjalizacja początkowych wartości wygładzonych parametrów przedstawia się następująco:

- Dla podwójnego wygładzania wykładniczego

    - pierwsza wygładzona wartość ustawiana jest na wartość pierwszej obserwacji: $S_1 = y_1$;

    - początkowa wartość trendu jest inicjalizowana przez różnicę dwóch pierwszych obserwacji: $b_1 = y_2 - y_1$

- Dla potrójnego wygładzania wykładniczego – wymaga danych z dwóch pełnych cykli sezonowości aby zainicjalizować parametry

    - początkowa wartość trendu obliczana jest na podstawie uśrednionych zmian wartości obserwowanych podczas pierwszych dwóch cykli, wg wzoru:

    $$b_1 = \frac{1}{L}\left(\frac{y_{L+1} - y_1}{L} + \frac{y_{L+2} - y_2}{L} + ... + \frac{y_{L+L} - y_L}{L}\right)$$

    - wygładzone wartości dla wszystkich okresów drugiego cyklu wyznaczana jest na podstawie obserwacji z pierwszego cyklu, uśrednionej obserwacji z drugiego cyklu i pierwszej wartości wygładzonego trendu:

    $$S_t = \frac{\sum_{k=L+1}^{2L} y_k}{L} + \left(t + 1 - L - \frac{L+1}{2}\right)b_1$$

    - Początkowe indeksy sezonowe obliczane są na podstawie obserwacji ze wszystkich pełnych cykli, dla których mamy dane, wg wzoru:

    $$I_t = \left(\sum_{k=1}^{M} \frac{y_{t+(k-1)L}}{A_k}\right) \Big/ M ,$$

    gdzie $A_k$ oznacza średnią wartość obserwacji z całego cyklu, a więc

    $$A_k = \frac{\sum_{i=1}^{L} y_i}{L} \qquad k = 1...M ,$$ a M jest liczbą pełnych cykli danych, jakimi dysponujemy.

## 2.5 Przygotowywanie parametrów negocjacji

Drugą z zaimplementowanych w ramach tej pracy funkcjonalności agenta *SDA*, które umożliwiają dostosowanie się do zmian w środowisku jest przygotowywanie warunków, na których sklep oferuje produkty. Wynik tego procesu, a więc trójka – strategia-protokół-parametry, został przedstawiony w sekcjach 1.4 i 1.5, natomiast architektura modułu, który wykonuje to zadanie w sekcji 2.2.2. Poniżej opiszę natomiast metody zastosowane przy realizacji tego zadania. Ze względu na przyjęty, uproszczony schemat sprzedaży towaru, przygotowanie negocjacji można sprowadzić do wyznaczania dwóch wartości: ceny produktu – będącej jednocześnie ceną wywoławczą i ceną minimalną; oraz maksymalnej ilości towaru oferowanej podczas pojedynczego procesu sprzedaży. Warto przypomnieć, że zestaw parametrów negocjacji jest zależny od metody prowadzenia negocjacji i może zawierać różne zmienne – przykładowo klasyczna aukcja angielska zakłada sprzedaż pojedynczej sztuki towaru, za to umożliwia zmianę minimalnej wartości o jaką licytujący musi podnieść cenę, aby jego oferta została zaakceptowana.

### 2.5.1 Ustalanie ceny produktu

Podstawowym celem dynamicznego modyfikowania cen produktów w trakcie działania systemu jest takie reagowanie na zmiany rynkowej wartości towaru (kształtowanej przez podaż, popyt jak i subiektywną ocenę klientów) aby maksymalizować zysk sklepu. Bardzo ciekawe zestawienie kilku metod dynamicznego wyceniania produktów wraz z bardzo bogatymi wnioskami z niego wynikającymi znaleźć można w [9] Opierając się na tym artykule postanowiłem zastosować metodę Derivative Following (podążanie za efektem). W porównaniu do innych, opisanych tam algorytmów, metoda ta nie wymaga ciągłej i kompletnej informacji na temat cen produktu u konkurencyjnych sprzedawców a jednocześnie jest wystarczająco skutecznaa (zwłaszcza w przypadku homogenicznego środowiska, w którym wszyscy agenci posługują się tą samą strategią).

Metoda DF opiera się na podejmowaniu decyzji co do przyszłej ceny na podstawie zmian zysku jakie zaszły w wyniku poprzedniej zmiany. Początkowo cena wybierana jest losowo, po zastosowaniu ceny przez pewien okres czasu, i otrzymaniu pewnego zysku, następuje zwiększenie lub zmniejszenie (kierunek wybierany losowo) ceny o pewną wartość $\delta$. Po

upływie kolejnego okresu, porównywany jest zysk z dwóch poprzednich okresów. Jeżeli zysk się zwiększył w wyniku zmiany ceny, to kolejną cenę modyfikujemy w tym samym kierunku. Jeśli zysk się zmniejszył, to odwracamy kierunek zmian ceny. W każdym okresie wartość modyfikatora maleje według wzoru:

$$\delta_n = \frac{\delta_0 (n_0 + 1.0)}{n_0 + currentPeriod}$$ , gdzie $n_0 = currentPeriod / 10$.

### 2.5.2 Ustalanie ilości oferowanego produktu

W przypadku rodzajów negocjacji, w których istnieje możliwość zaoferowania podczas pojedynczej negocjacji przez sprzedającego większej niż jedna sztuka ilości towaru, odpowiednie ustalenie tej ilości ma o tyle duże znaczenie dla sprawnego funkcjonowania systemu sklepowego, że podczas licytacji sprzedawane produkty są zarezerwowane i niedostępne w innych aukcjach. Z tego też powodu wystawianie zbyt dużej liczby przedmiotów na raz nie jest korzystne, gdyż może prowadzić do zablokowania możliwości ich kupna przez innych klientów. Z drugiej jednak strony zbyt mała ilość oferowanego towaru może spowodować, że sklep straci klientów, chcących kupić więcej niż wystawiono. Najkorzystniejsze dla sklepu wydaje się więc oferować tyle, ile prawdopodobnie może wynieść maksymalna oferta pojedynczego klienta.

W celu ustalania ilości oferowanego produktu posłużyłem się więc metodą średniej ruchomej (ang. moving average), a więc wartość ta ustalana jest jako średnia arytmetyczna maksymalnych wielkości ofert klientów z *N* ostatnich okresów wg. wzoru:

$$M_t = \frac{\sum_{i=1}^{N} X_{t-i}}{N},$$

gdzie $X_i$ oznacza maksymalną ilość towaru, jaka pojawiła się w pojedynczej ofercie klienta podczas i-tej licytacji.

Do otrzymanej uśrednionej wartości dodawany jest jeszcze pewien margines, który z jednej strony pomaga w przypadku większych wahań zapotrzebowania, a z drugiej pozwala na reakcję na ewentualny wzrost zapotrzebowania. Margines liczony jest jako ustalony procent

od uśrednionej wartości. Zarówno wielkość marginesu, jak i ilość okresów, na podstawie których liczona jest średnia przekazywane są do programu jako parametry.

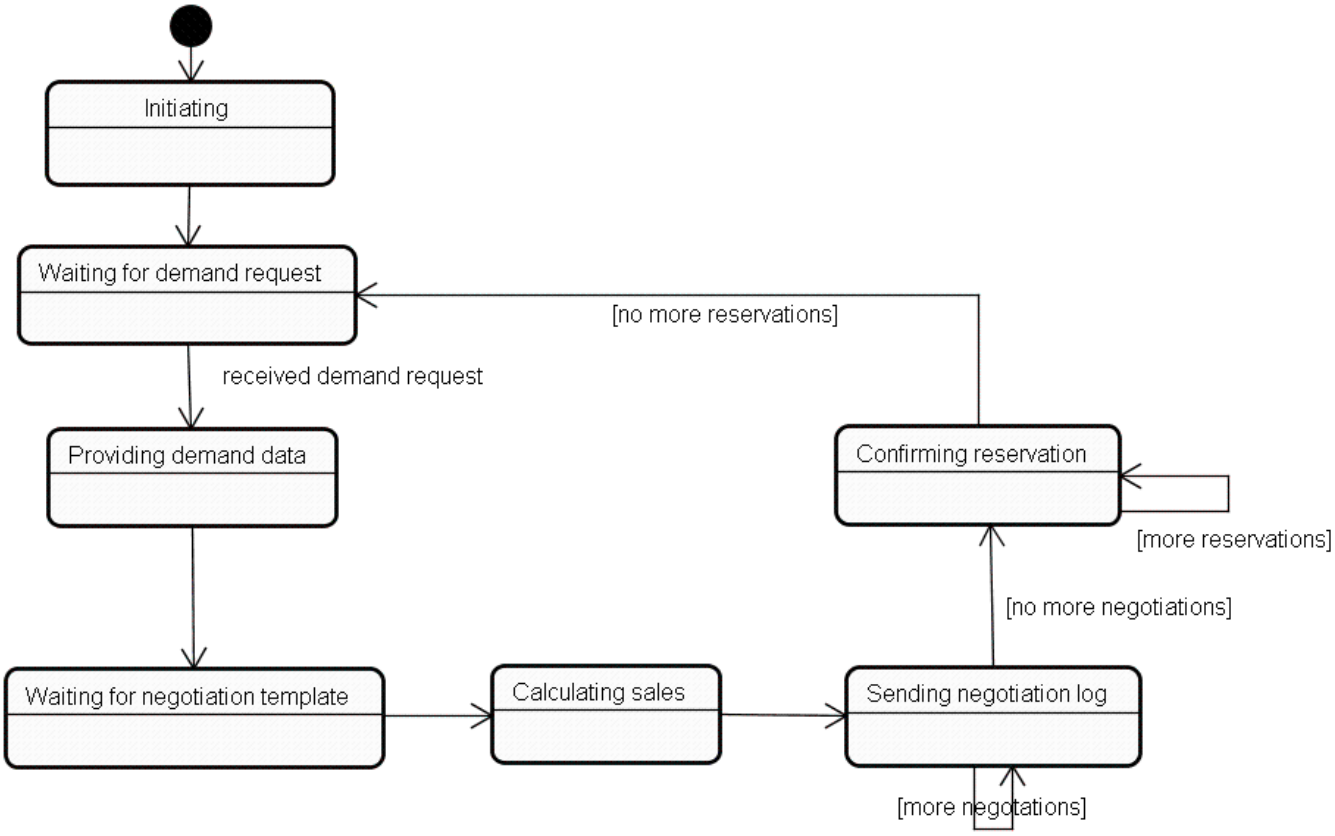
# 3 Testy

## 3.1 Metodologia testów

W momencie pisania tej pracy nie były jeszcze gotowe pozostałe części agentowego systemu ecommerce. Z tego też powodu w celu przeprowadzenia testów funkcjonowania agenta *SDA*, a także sprawdzenia skuteczności jego adaptacji do zmian w środowisku przygotowałem system testowy, który komunikując się z *SDA* symuluje działanie agentów *SA* i *WA*. Powstało więc dwóch agentów: *ShopAgentStub* (*SAS*) i *WarehouseAgentStub* (*WAS*), udający przed *SDA* odpowiednio agentów ShopAgent i WarehouseAgent. W poniższych sekcjach postaram się przybliżyć sposób w jaki *SAS* i *WAS* imitują działania *SA* i *WA* oraz tych części systemu, które leżą poza nimi. Jeśli chodzi o *SDA*, to jedyną znaczącą zmianą w porównaniu do założeń wstępnych jest wprowadzenie klasy **TimeConverter**, dzięki której możliwa jest kompresja czasu symulacji, czyli na przykład ustalenie, że jeden dzień symulacji odpowiada minucie czasu rzeczywistego. Reszta funkcjonalności *SDA* nie wymagała dostosowywania do warunków testowych.

### 3.1.1 ShopAgentStub

Celem agenta *SAS* jest imitowanie działań *SA* w interakcji z agentem *SDA*, a więc wymiana komunikatów opisana w sekcji 2.1. Aby uprościć opis i działanie agenta ShopAgentStub, jego zachowanie zostało zaprojektowane i zaimplementowane w sposób sekwencyjny, ograniczony do tych działań, które były istotne przy opracowywaniu wyników dotyczących rozpatrywanych w tej pracy procesów adaptacyjnych. Algorytm według którego działa agent *SAS* został przedstawiony na poniższym diagramie. Dokładniejszy opis jego najistotniejszych elementów zostanie przedstawiony w kolejnych podsekcjach.




*Rys. 24. Diagram stanów agenta ShopAgentStub*

#### 3.1.1.1 Inicjalizacja symulacji

Działanie agenta *SAS* rozpoczynane jest zawsze wysyłanymi do *SDA* wiadomością informującą o wprowadzeniu do sprzedaży produktu o danym identyfikatorze oraz prośbą o rozpoczęcie prognozowania jego sprzedaży.

#### 3.1.1.2 Dostarczenie danych o popycie

Na początku każdego właściwego cyklu działania agenta, czeka on na prośbę *SDA* o informacje dotyczące zainteresowania klientów produktem. W odpowiedzi na tę prośbę, *SAS* wysyła liczbę wszystkich zapytań dotyczących danego produktu otrzymanych przez *CIC*, liczbę sklepów sprzedających dany produkt oraz liczby klientów zainteresowanych kupnem produktu, którzy zgłosili się do sklepu. Dwie pierwsze informacje pokrywają się z danymi dostarczanymi oryginalnie przez *CIC* ( patrz 2.1.1.6 i 2.1.1.7 ), natomiast ostatnia w gruncie

rzeczy zawiera zagregowane dane z założenia dostarczane do *SDA* w postaci serii wiadomości dotyczących pojawienia się *BA* w sklepie. Wszystkie dane wysyłane przez *SAS* pochodzą z pliku wejściowego i dotyczą dnia poprzedzającego aktualny. Aktualna data, którą posługuje się *SAS* w trakcie cyklu dostarczana jest mu przez *SDA* w wiadomości zapytania o popyt.

#### 3.1.1.3 Tworzenie listy negocjacji

Po wysłaniu danych o zainteresowaniu klientów produktem, *SAS* oczekuje na aktualizację parametrów negocjacji. Gdy je dostanie, rozpoczyna symulację negocjacji z klientami. W tej sekcji postaram się przybliżyć sposób w jaki generowane są efekty licytacji, przesyłane w kolejnych krokach do *SDA*.

Symulacja procesu negocjacji opiera się na uproszczonym modelu sprzedaży. Nie zakłada on de facto negocjacji ceny – uznajemy, że każdy klient posługuje się maksymalną ceną, jaką jest skłonny zapłacić za dany produkt. Jeśli cena zaproponowana przez sklep jest niższa lub równa cenie maksymalnej klienta, to dokonuje on zakupu, w przeciwnym przypadku nie akceptuje on warunków negocjacji i do niej nie przystępuje. Zakładamy także, że w każdej negocjacji bierze udział pojedynczy klient. Aby wyznaczyć ilość negocjacji zakończonych sukcesem zakładam, że cena maksymalna każdego klienta jest wartością zmiennej losowej według normalnego rozkładu prawdopodobieństwa z wartością średnią pobieraną z pliku wejściowego i zmieniającą się każdego dnia (w każdym cyklu) oraz wariancją jako pewien procent wartości średniej - stałą dla całego doświadczenia i podawaną jako parametr programu. Przy takim założeniu liczba klientów, którzy dokonali kupna liczona jest za pomocą dystrybuanty używanego rozkładu wg wzoru:

$$s = v * \Phi_{p_c, f*p_c}(p_s),$$

gdzie s oznacza liczbę klientów, którzy dokonali kupna; $p_c$ – średnią cenę maksymalną klienta; f – względną wariancję ceny maksymalnej klienta; $p_s$ – cenę sklepową. Liczba sztuk towaru kupowana w pojedynczej negocjacji wyznaczana jest z kolei według rozkładu normalnego o średniej wartości pobieranej w każdym cyklu z pliku i wariancji ustalanej procentowo (w zależności od wartości średniej) – względna wielkość wariancji występuje jako parametr symulacji. Wielkość ta jest ograniczana przez liczbę produktów wystawianych

w pojedynczej licytacji – jest to element szablonu negocjacji, dostarczanego przez *SDA*. Po wyznaczeniu liczby klientów dokonujących kupna oraz ilości towaru kupowanego przez każdego klienta, można więc utworzyć odpowiadającą jej wielkością listę negocjacji zakończonych powodzeniem, każda z których zawiera jedną ofertę kupna pewnej ilości towaru po cenie początkowej sklepu.

#### 3.1.1.4 Wysyłanie danych o negocjacjach

Po stworzeniu listy negocjacji, następuje ich sekwencyjne wysyłanie. Zgodnie z interfejsem agenta *SDA*, każda wiadomość zawiera informacje o jednej zakończonej negocjacji.

#### 3.1.1.5 Finalizacja transakcji

Po wysłaniu danych o wszystkich negocjacjach, *SAS* rozpoczyna wysyłanie informacji o zakończeniu transakcji. W rozpatrywanych przeze mnie przypadkach testowych, założeniem jest, że wszystkie rezerwacje są potwierdzane, a więc etap ten sprowadza się do wysłania odpowiedniej ilości wiadomości potwierdzających zakup wynegocjowanych produktów.

Po zakończeniu finalizacji, *SAS* przechodzi do stanu oczekiwania na prośbę o dostarczenie danych o zainteresowaniu produktem, a więc kończy jednodniowy cykl symulacji.

#### 3.1.1.6 Parametry i dane wejściowe

Parametry agenta *ShopAgentStub* można podzielić na te, które podlegają zmianom w kolejnych cyklach działania i te, które są stałe przez cały czas trwania symulacji. Do pierwszych należą:

- liczba zapytań o produkt w *CIC*,
- liczba sklepów oferujących produkt,
- liczba klientów chcących kupić produkt w sklepie,
- średnia liczba sztuk towaru, jaką chce kupić pojedynczy klient

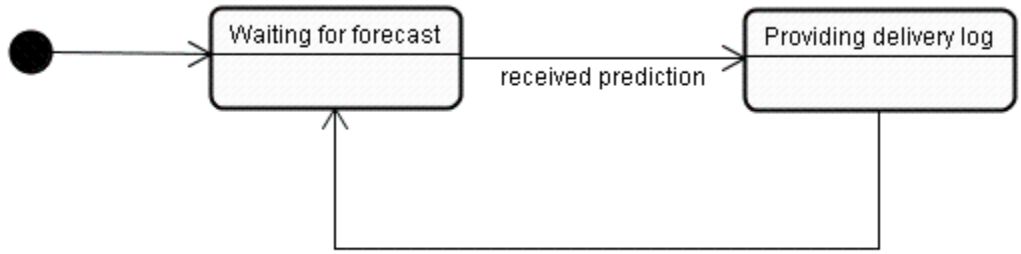
- średnia cena maksymalna, po której klient chce kupić produkt

Wszystkie te wartości przekazywane są agentowi w dwóch plikach tekstowych: demand.txt oraz price.txt. W pliku **demand.txt** podawane są pierwsze 4 wartości, natomiast średnia cena maksymalna powinna znajdować się w pliku **price.txt**. Oba pliki zapisane są w formacie „tab separated values", a więc w pojedynczej linii znajdują się dane dotyczące jednego dnia, z kolejnymi kolumnami oddzielonymi znakiem tabulacji.

Parametrów, które nie zmieniają się w trakcie symulacji jest znacznie więcej i są one przekazywane agentowi poprzez plik właściwości shopagentstub.properties. Jest to plik w standardowym dla Javy formacie właściwości.

### 3.1.2 WarehouseAgentStub

Agent *WAS* jest drugim agentem symulującym funkcjonowanie niezaimplementowanej jeszcze części systemu – w ramach testowanej przeze mnie funkcjonalności udaje w interakcji z *SDA* agenta WarehouseAgent. W porównaniu do *SAS* jego funkcjonalność jest jednak znacznie prostsza i sprowadza się w gruncie rzeczy do odbierania od *SDA* wiadomości zawierających prognozę sprzedaży oraz odpowiadania informacjami o dostawie prognozowanej ilości towaru. To proste zachowanie agenta przedstawione jest na poniższym diagramie.




*Rys. 25. Diagram stanów agenta WarehouseAgentStub*

#### 3.1.2.1 Parametry i dane wejściowe

Parametry agenta *WAS* takie, jak opóźnienie czasu zamówienia względem czasu otrzymania predykcji oraz czasu dostawy względem czasu zamówienia przekazywane są w pliku wastub.properties. Identycznie jak w przypadku agenta *SAS* jest to standardowy plik właściwości Javy.

## *3.2 Scenariusze testowe*

Testy systemu zostały przeprowadzone na podstawie scenariuszy, każdy z których pozwala na obserwację adaptowania się agenta *SDA* do pewnego rodzaju zmian w środowisku. Bardziej szczegółowa analiza zaimplementowanych algorytmów pod względem ich skuteczności dla różnego typu danych – w tym pochodzących z prawdziwych obserwacji; a także ich modyfikacja i usprawnianie w celu poprawy ich wyników wykraczają poza zakres tej pracy – przedstawione poniżej eksparymenty (czyli widze, ze sa LITEROWKI!!! Trzeba strawdzic dokladnei calos) miały raczej na celu pokazanie, że ilość i rodzaj danych zbieranych przez agenta *SDA*, a także sposoby w jaki zgromadzona przez niego wiedza jest pożytkowana przez pozostałych agentów sklepowych jest wystarczająca, aby system zarządzania sklepem wykazywał adaptacyjność.

### 3.2.1 Rosnąca liczba klientów

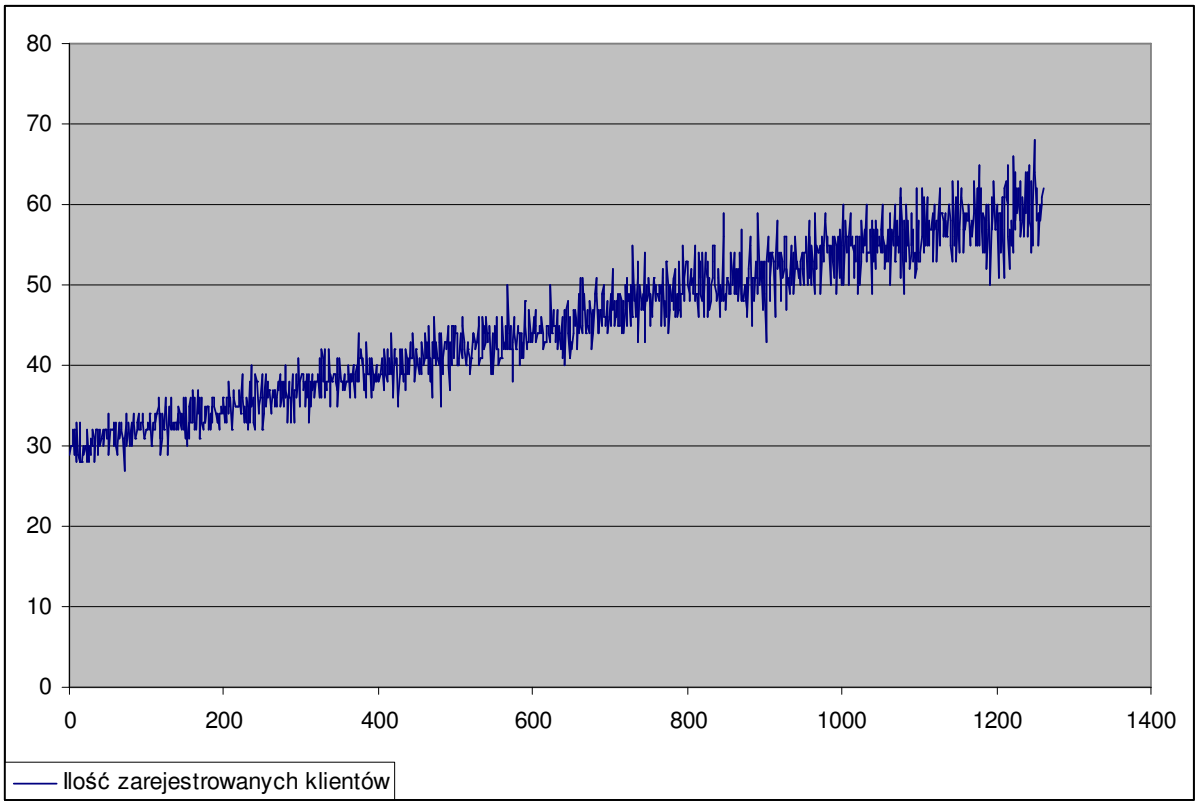
W pierwszym scenariuszu testowym, który sprawdzić ma zdolność przystosowania się systemu do rosnącego zapotrzebowania na produkt przyjęto następujące założenia:

- Czas trwania symulacji obejmuje 1200 dni.
- Rozpatrywany sklep jest jedynym sprzedającym dany produkt.
- Klient dokonujący zakupu, rezerwuje, o ile wystawiono do sprzedaży wystarczającą ilość produktu, 10 jego jednostek.
- Średnia cena maksymalna, po której klient chce kupić produkt (wartość $p_c$ z sekcji 3.1.1.3) została wygenerowana na podstawie rozkładu normalnego o średniej 5 i

wariancji 0.5. Względna wariancja ceny maksymalnej (oznaczona we wzorze jako $f$ ) ustalona została na 0.2.

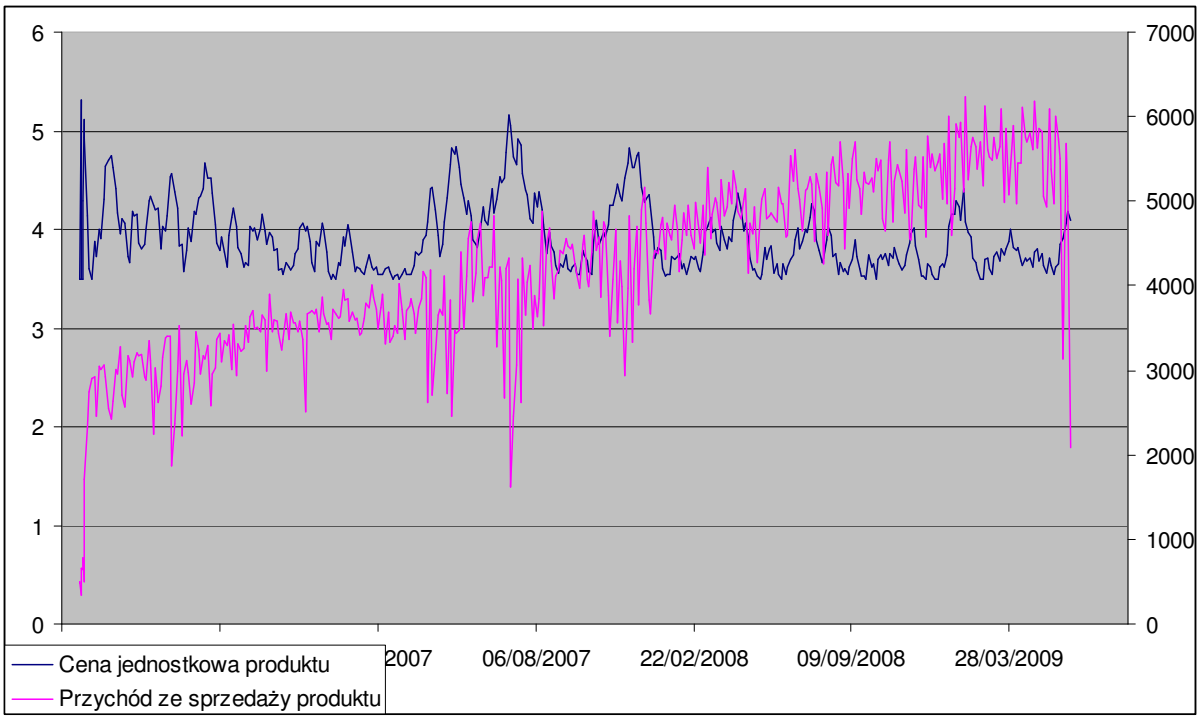
- Liczba zapytań do *CIC* dotyczących danego produktu, równa jest liczbie klientów zarejestrowanych w sklepie i została wygenerowana na podstawie rozkładu normalnego o średniej rosnącej liniowo od 30 na początku symulacji do 60 na końcu oraz wariancji równej 0.05 wartości średniej dla danego okresu.

Na poniższym wykresie (Rys. 26) przedstawione zostały założone przez ten scenariusz zmiany liczby klientów potencjalnie chcących kupić dany produkt.

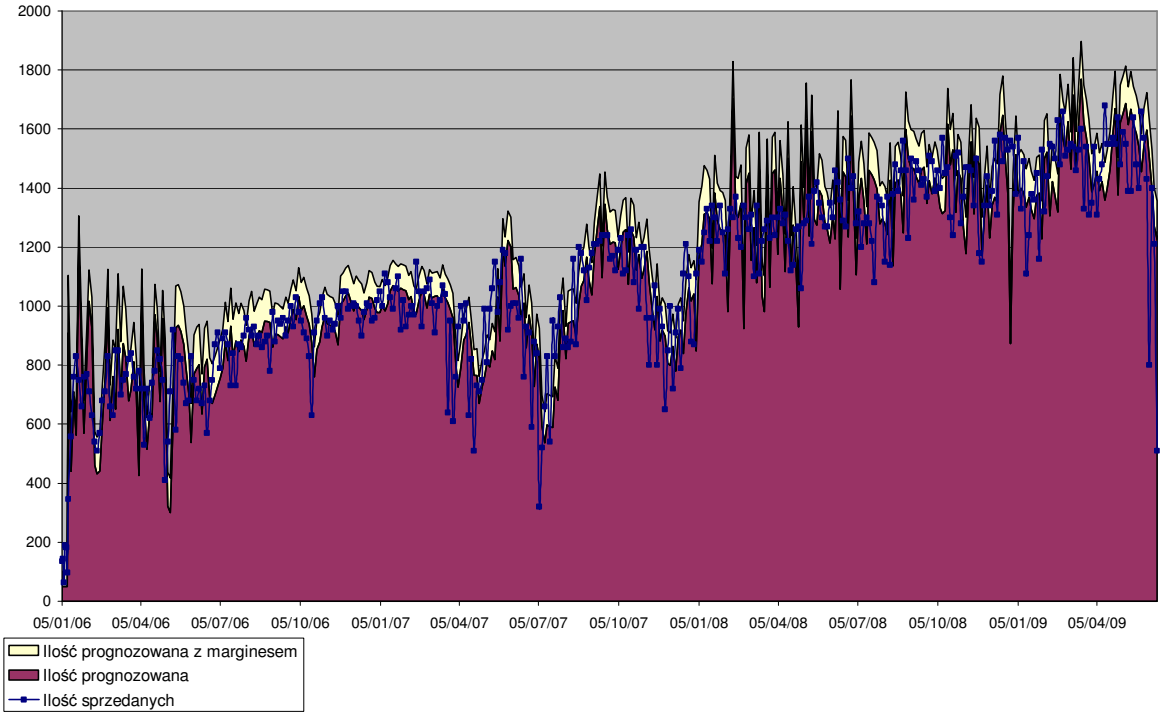



*Rys. 26. Wykres ilości zarejestrowanych klientów w funkcji czasu*

Następne wykresy przedstawiają natomiast wyniki przeprowadzonego testu. Warto nadmienić, że zostały one wyznaczone nie w funkcji dnia symulacji, ale sumowane w obrębie horyzontu predykcji, który w wyniku funkcjonowania systemu ustalony został na 3 dni.




*Rys. 27. Wykres zmian ceny jednostkowej oraz przychodu ze sprzedaży produktu*



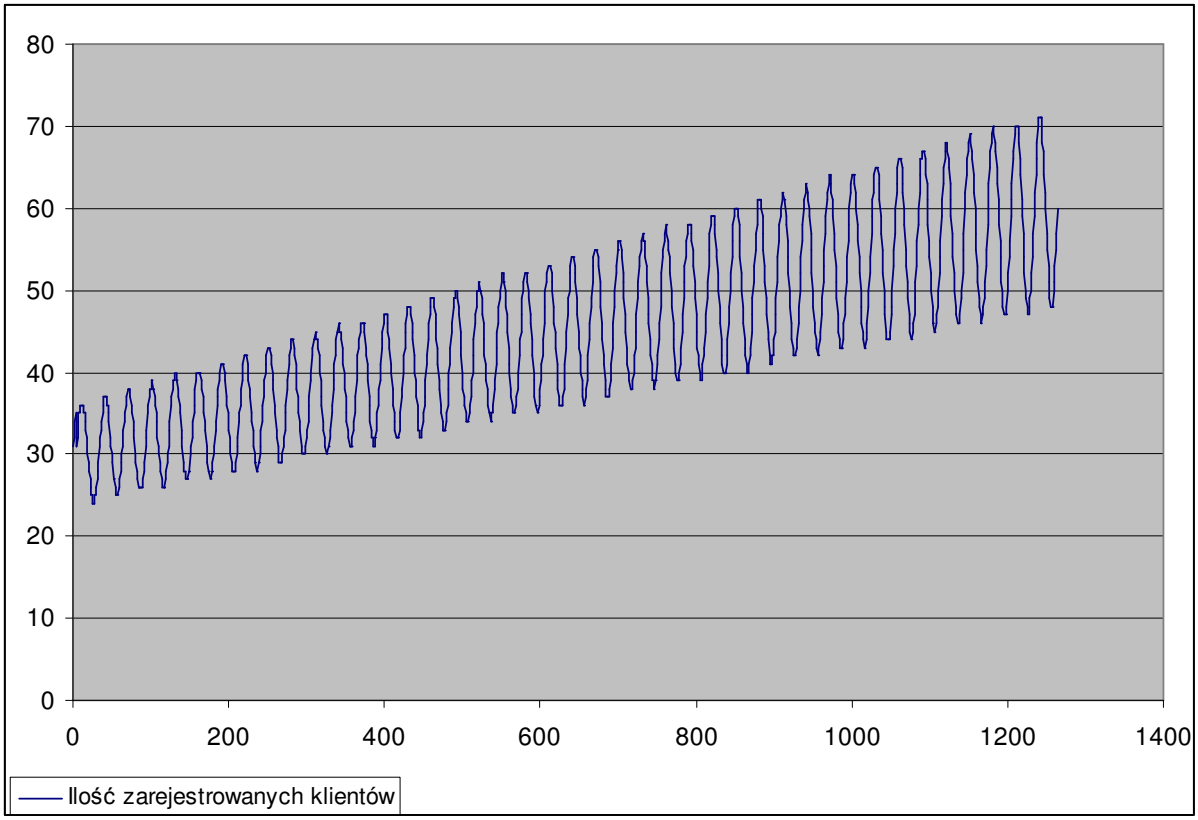

*Rys. 28. Wykres prognozowanego zapotrzebowania na produkt kontra faktyczna sprzedaż w tych samych okresach czasu*

W scenariuszu tym, próbie poddany został przede wszystkim moduł prognozowania sprzedaży. Jego zadanie było utrudnione nie tylko przez rosnący trend wejściowych danych i ich wstępne zaburzenie, ale przede wszystkim przez dodatkowe odchylenia spowodowane działaniem modułu ustalania ceny. Pomimo tego, w przypadku 78% okresów udało się uniknąć braku towarów, przy średniej względnej ilości nadmiarowego towaru równej 20%. W przypadkach, gdzie moduł niedoszacował sprzedaży, średnia względnej wielkości niedoszacowania wyniosła 10% , a jej mediana niecałe 8%.

### 3.2.2 Rosnąca liczba klientów z sezonowością

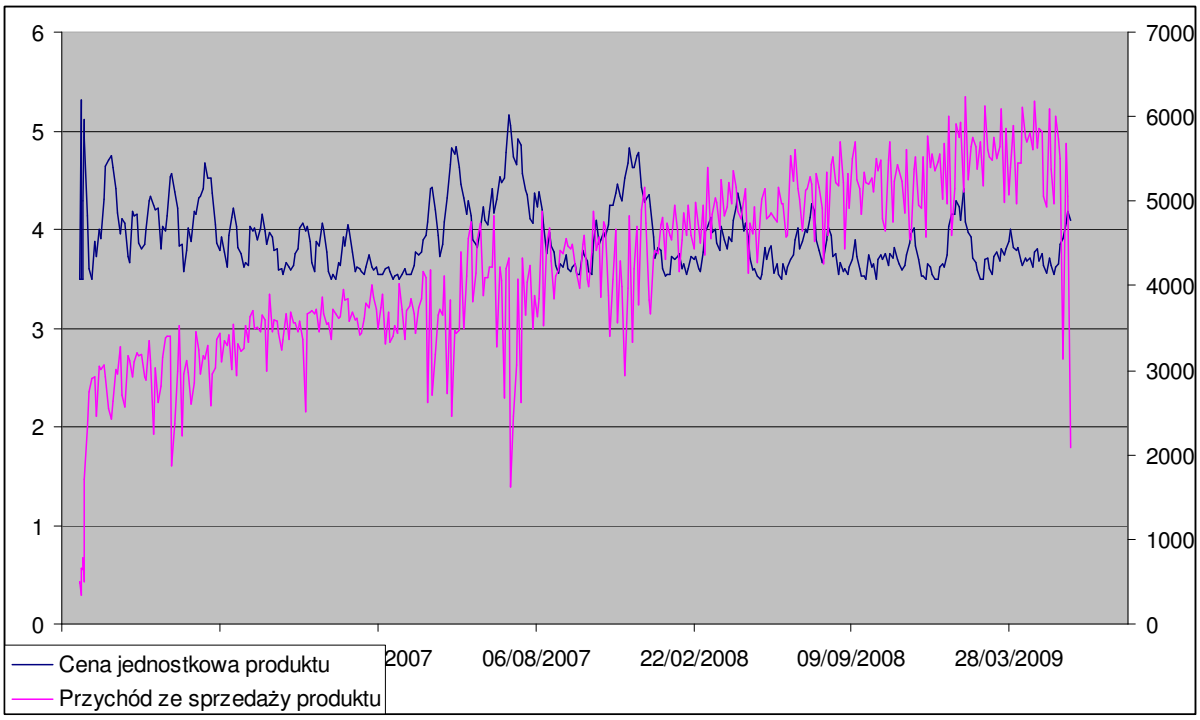
Drugi scenariusz oprócz rosnącego trendu wprowadza do wejściowych danych także efekt sezonowości multiplikatywnej (wielkość wahań sezonowych jest proporcjonalna do wielkości sprzedaży). Trend został ustalony tak samo jak w poprzednim przypadku, a więc od 30 do 60 sztuk w trakcie czasu trwania symulacji. Sezonowość została wprowadzona przez przemnożenie wartości średniej przez funkcję sinusoidalną o okresie 30 dni i amplitudzie równej 0.2 aktualnej wartości (a więc indeksy sezonowe przyjmowały wartości <0.8 , 1.2>). Pozostałe parametry symulacji pozostały takie same jak w pierwszym scenariuszu.

Na poniższym wykresie (Rys. 26) przedstawione zostały założone przez ten scenariusz zmiany liczby klientów potencjalnie chcących kupić dany produkt.

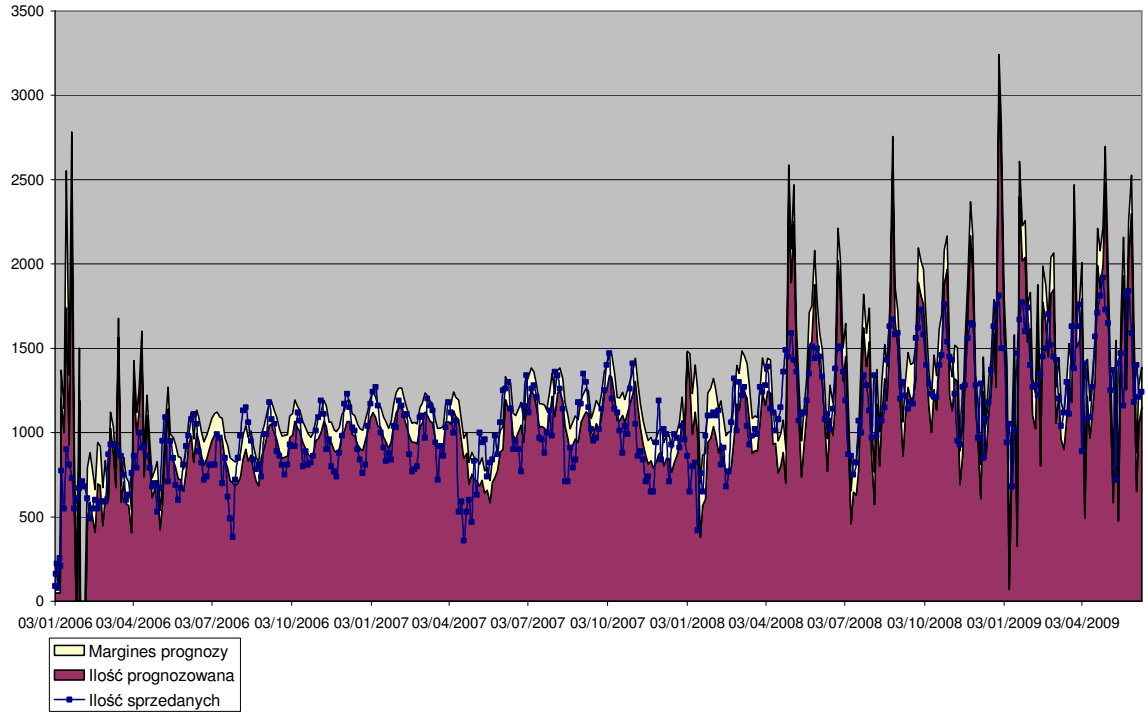



*Rys. 29. Wykres ilości zarejestrowanych klientów w funkcji czasu*

Poniżej zaprezentowane są natomiast wyniki przeprowadzonego testu. Analogicznie do poprzedniego przykładu są one sumowane po 3-dniowych okresach.




*Rys. 30. Wykres zmian ceny jednostkowej oraz przychodu ze sprzedaży produktu*



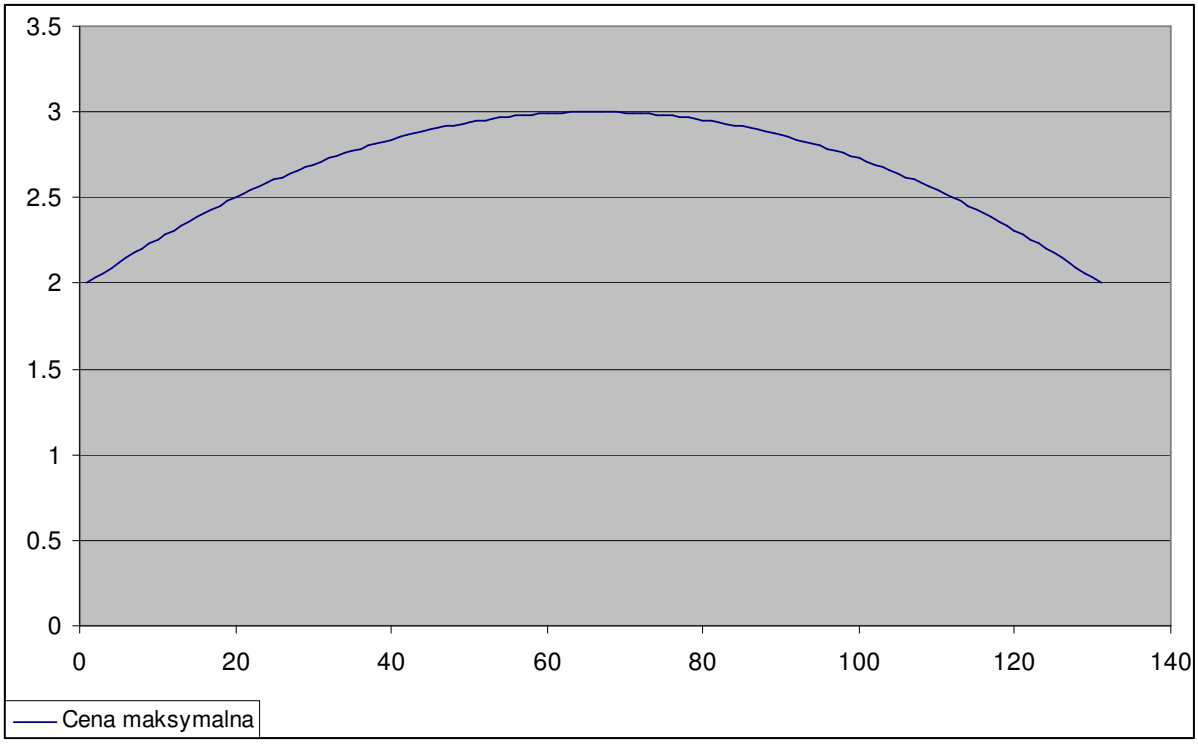

*Rys. 31. Wykres prognozowanego zapotrzebowania na produkt kontra faktyczna sprzedaż w tych samych okresach czasu*

W tym przypadku osiągnięte wyniki były nieco gorsze od tych z pierwszego scenariusza. Co prawda liczba niedoszacowanych okresów była porównywalna – 22%, ale przeciętna wielkość przeszacowania wzrosła do 29 % , a niedoszacowania 21%. To, na co warto zwrócić uwagę, to fakt, że model prognostyczny bardzo skutecznie dopasował się do okresu wahań sezonowych, błędy były przede wszystkim skutkiem niedokładnego oszacowania ich amplitudy.

### 3.2.3 Zmienna wycena produktu przez klientów

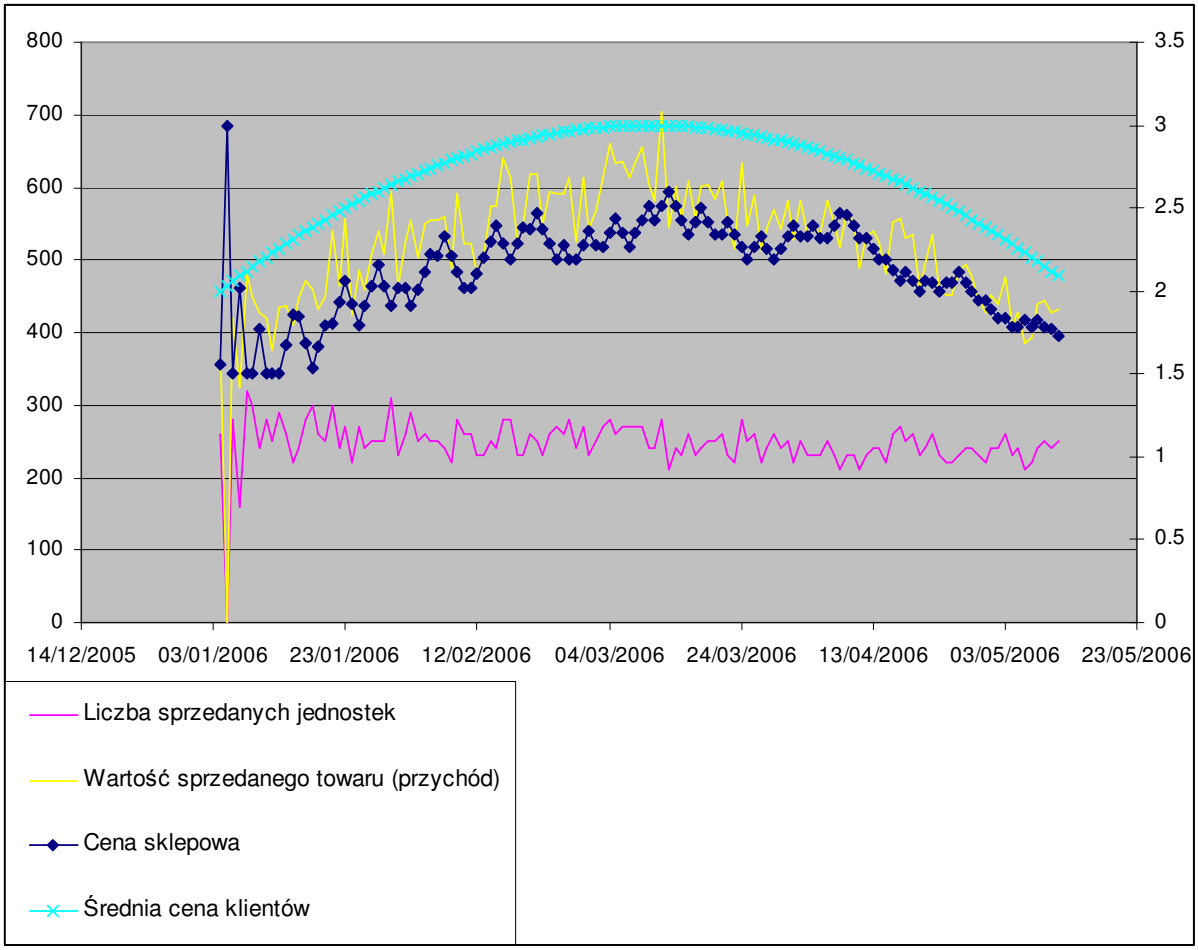
Kolejny ze scenariuszy testowych pokazuje w jaki sposób agent *SDA* przystosowuje się do zmian w postrzeganiu wartości produktu przez klientów. Został on przeprowadzony zgodnie z następującymi warunkami:

- Czas trwania symulacji obejmuje okres 130 dni.
- Rozpatrywany sklep jest jedynym sprzedającym dany produkt.
- Klient dokonujący zakupu, rezerwuje, o ile wystawiono do sprzedaży wystarczającą ilość produktu, 10 jego jednostek.
- Liczba zapytań do *CIC* dotyczących danego produktu, równa jest liczbie klientów zarejestrowanych w sklepie i została wygenerowana na podstawie rozkładu normalnego o średniej równej 30 oraz wariancji równej 1.5 dla całego czasu trwania symulacji.
- Średnia cena maksymalna, po której klient chce kupić produkt (wartość $p_c$ z sekcji 3.1.1.3) została wygenerowana na podstawie wartości funkcji kwadratowej o minimalnej wartości 2 na początku i końcu okresu oraz maksimum równym 3. Zmiany ceny klienta zostały przedstawione na rys.. Względna wariancja ceny maksymalnej (oznaczona we wzorze jako $f$ ) ustalona została na 0.2.




*Rys. 32. Wykres wejściowych danych o maksymalnej cenie, jaką klient może zapłacić za towar*

Na kolejnym wykresie przedstawiony został wynik testu, a więc cena produktu, jaką *SDA* ustalał w kolejnych okresach symulacji oraz przychodu osiąganego przez sklep.




*Rys. 33. Zmiany ceny sklepowej w odpowiedzi na zmiany wyceny produktu przez klientów*

Jak widać na powyższym rysunku, wraz ze wzrostem ceny, jaką klienci byli skłonni zapłacić za dany produkt, agent *SDA* poprawnie zwiększał także sklepową cenę produktu. Zareagował odpowiednio także w drugiej części symulacji, gdy cena produktu zaczęła spadać.

## 3.3 Ogólne wnioski

Testy przeprowadzone na zimplementowanym podsystemie pokazały, że zaprojektowany model zbierania informacji, ustalania parametrów funkcjonowania sklepu oraz reagowania na zmiany w środowisku zewnętrznym jest wystarczający aby agenta *SDA* można było uznać za wykazującego adaptacyjność. Także metody zastosowane do konkretnych zadań agenta, a więc prognozowania sprzedaży, czy dynamicznej modyfikacji ceny przyniosły w prostych przypadkach oczekiwane efekty.

# 4 Podsumowanie

Niniejsza praca przedstawiła zagadnienia związane z adaptacyjnymi agentami programowymi wspierającymi e-commerce w kontekście modelowego systemu wieloagentowego E-CAP. Omówione zostały te elementy systemu, które wykazują cechy przystosowywania się do zmian, ze szczególnym uwzględnieniem agenta podejmującego decyzje w ramach części sklepowej. Zaprojektowane zostały procesy gromadzenia wiedzy oraz składowania jej, pozwalające na wydajne działanie modułów automatyzujących niektóre procesy decyzyjne sklepu. W oparciu o tę platformę zostały zaimplementowane dwa takie procesy – przewidywanie wielkości sprzedaży produktów oraz ustalania parametrów negocjacji. Przeprowadzone testy wykazały, że przyjęty model pozwala systemowi sklepowymi na adaptację do zmiennych warunków zewnętrznych, takich jak zmiana wartości produktów, a także zróżnicowane zapotrzebowanie na towar. Dzięki modularnej, obiektowej architekturze agenta decyzyjnego możliwe jest wykorzystanie stworzonej platformy do testowania bardziej złożonych algorytmów i metod realizujących zadania stojące przed agentem, a także jego rozwijania o kolejne grupy funkcjonalności.

# Bibliografia


1. G. Booth, Object—oriented Analysis and Design with Applications, Addison Wesley, 1994.

2. J. Han, M. Kamber, Data Mining: Concepts and Techniques, 2nd ed., Morgan Kaufmann Publishers, Marzec 2006, rozdz. 3. str.106

3. W. H. Inmon. Building the Data Warehouse. *QED Technical Publishing Group*, Wellesley, Massachusetts, 1992

4. R. Kalakota, A.B Whinston, *Frontiers of electronic commerce*. Addison Wesley Publishing Co., 1996.

5. R. Kimball, M. Ross, The Data Warehouse Toolkit Second Edition: The Complete Guide to Dimensional Modelling, John Wiley and Sons, Inc., 2002

6. R. Kimball, The Database Market Splits*, DBMS,* Vol. 8, No. 10, Miller Freeman, Inc., Wrzesień 1995.

7. S. Franklin and A. Graesser, Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents, *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag, 1996.

8. V. Zwass, Electronic Commerce: Structures and Issues. *International Journal of Electronic Commerce*, 1, 1 (Fall), 1996, 3-23.

9. J. Kephart, J. Hanson, A. Greenwald, Dynamic Pricing by Software Agents. *Computer Networks*. vol. 32 (6), 2000, 731-752.

10. N.R. Jennings, An agent-based approach for building complex software systems, *Communications of the ACM*, 44(4), 2001, 35-41.

11. M. Ganzha, M. Paprzycki, A. Pîrvănescu, C. Bădică, A. Abraham, JADE-based Multi-agent E-commerce Environment: Initial Implementation, *Analele Universităţii din Timişoara, Seria Matematică-Informatică*, Vol. XLII, 2004, 79-100.

12. M.Morzy, Aktywne hurtownie danych. *Najnowsze rozwiązania i praktyczne zastosowania: hurtownie danych i business intelligence*, Centrum Promocji Informatyki, Warszawa, 23 marca 2004

13. P. Drygas, M. Kowalkiewicz, M. Paprzycki, Marketing Activities of Polish Internet Retailers. W: N. Guimarães, P. Isaías (eds.) *Proceedings IADIS AC'05, International Conference on Applied Computing*, Algarve, Portugal, IADIS Press, Lisbon, 2005, 479-486.

14. C. Bădică, M. Ganzha, M. Gawinecki, P. Kobzdej, M. Paprzycki, Towards Trust Management in an Agent-based E-commerce System - Initial Considerations. W: *A. Zgrzywa (ed.), Proceedings of the MISSI 2006 Conference*, Wroclaw University of Technology Press, Wrocław, Poland, 2006, 225-236.

15. M. Ganzha, M. Gawinecki, P. Kobzdej, M. Paprzycki, C. Bădică, Functionalizing trust in a model agent based e-commerce system. W: *M. Bohanec et. al. (eds.), Proceedings of the 2006 Information Society Multiconference*, Josef Stefan Institute Press, 2006, 22-26.

16. M. Gawinecki, M. Ganzha, P. Kobzdej, M. Paprzycki, C. Bădică, M. Scafes, G. G. Popa, Managing Information and Time Flow in an Agent-based E-commerce System. W: *D. Petcu et. al. (eds.), Proceedings of the Fifth International Symposiom on Parallel and Distributed Computing*, IEEE CS Press, Los Alamitos, *CA*, 2006, 352-359.

17. C. Bădică, M. Ganzha, M. Paprzycki, Developing a Model Agent-based E-commerce System, W: *Jie Lu et. al. (eds.) E-Service Intelligence - Methodologies, Technologies and Applications*, Spriner, Berlin, 2007, 555-578.

18. M. Paprzycki, M. Ganzha, Adapting Price Negotiations to an E-commerce System Scenario. W*: K. Saeed et.al. (eds.), Proceedings of the CISIM Conference*, IEEE CS Press, Los Alamitos, CA, 380-386, 2007.

19. M. Gawinecki, P. Kobzdej, M. Ganzha, M. Paprzycki, Introducing interaction-based auctions into a model agent-based e-commerce system — preliminary considerations.

W: *R. do Nascimento et.al. (eds.) Proceedings of the EATIS Conference*, ACM Digital Library, ACM Press, New York, NY, 2007.

20. M. Ganzha, M. Gawinecki, P. Kobzdej M. Paprzycki, T. Serzysko, Implementing Commodity Flow in an Agent-Based Model E-commerce System. W: *Proceedings of the PPAM Conference*, LNCS, Springer, ukaże się.

21. T. Serzysko, M. Ganzha, M. Gawinecki, P. Kobzdej M. Paprzycki, C. Bădică Introducing Commodity Flow to an Agent-Based Model E-commerce System. W: Proceedings of the 2007 IAT Conference, IEEE CS Press, ukaże się

22. Agent (programowanie). *Wikipedia : wolna encyklopedia*, 2007-04-27 07:14Z. Dostępny w Internecie: http://pl.wikipedia.org/w/index.php?title=Agent_%28programowanie%29&oldid=7577588

23. Data warehouse. *Wikipedia : wolna encyklopedia* , 2007-04-27 07:14Z. Dostępny w Internecie: http://en.wikipedia.org/w/index.php?title=Data_warehouse&oldid=175925195

24. Z. Domaszewicz, 2 mln zagranicznych tytułów w ofercie Empik.com, *Gazeta Gospodarka*, http://gospodarka.gazeta.pl/gospodarka/1,68367,3336729.html

25. EBay. *Wikipedia : wolna encyklopedia*, 2007-04-27 07:14Z. Dostępny w Internecie: http://en.wikipedia.org/w/index.php?title=EBay&oldid=177450303

26. R. Kimball, Latest Thinking On Time Dimension Tables, *Kimball Design Tip*, Number 51, 1 lutego 2004. Dostępny w Internecie: http://kimballgroup.com/html/designtipsPDF/KimballDT51LatestThinking.pdf

27. E. Lach, Laboratorium Grafiki Komputerowej: Komputerowa animacja trójwymiarowych postaci. Dostępny w Internecie: http://grafika.iinf.polsl.gliwice.pl/doc/08-ANI.pdf

28. NIST/SEMATECH e-Handbook of Statistical Methods. Dostępny w Internecie: http://www.itl.nist.gov/div898/handbook/,

29. M. Paprzycki, Agenci programowi jako metodologia tworzenia oprogramowania, *E-Informatyka*. Dostępny w Internecie: http://www.e-informatyka.pl/article/show/422#d0e28

30. Raport strategiczny IAB Polska. Internet 2006 - Polska, Europa i Świat. Dostępny w Internecie: http://dc1.sabela.pl/raport_IAB_2006.pdf

31. U.S. Census Bureau, Quarterly Retail E-Commerce Sales. Dostępny w Internecie: http://www.census.gov/mrts/www/ecomm.html

32. Foundation for Intelligent Physical Agents [on-line]. http://www.fipa.org

33. Java Agent DEvelopment Framework [on-line], http://jade.tilab.com

34. Stationary Testing Process [on-line], http://home.ubalt.edu/ntsbarsh/Business-stat/otherapplets/Stationary.htm

35. Bidnapper [on-line], www.bidnapper.com

36. Ceneo [on-line], www.ceneo.pl

37. Nokaut [on-line], www.nokaut.pl

38. Skąpiec [on-line], www.skapiec.pl

39. Snajper [on-line], www.snajper.net

# Spis rysunków

Warszawa, dnia 6.01.2008

## *Oświadczenie*

Oświadczam, że pracę magisterską pod tytułem; „Adaptacyjne zachowania agentów programowych w e-commerce.” , której promotorem jest dr M. Paprzycki wykonałem samodzielnie, co poświadczam własnoręcznym podpisem.

.....