Politechnika Łódzka

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

Instytut Informatyki

Maciej Dzwonnik, 157831

System ekspertowy do oceny spółek giełdowych za pomocą analizy technicznej

Praca inżynierska
napisana pod kierunkiem
dr inż. Jana Stolarka

Łódź 2013

Spis treści

Rozdział 1

Wstęp

Niniejsza praca dotyczy inżynierii oprogramowania i systemów ekspertowych. W wyniku pracy powstał system ekspertowy do oceny spółek giełdowych za pomocą analizy technicznej. Temat ten został podjęty ze względu na zainteresowanie analizą rynków finansowych, głównie rynku akcji, a także chęć poszerzenia wiedzy z zakresu budowy systemów ekspertowych. Realizacja systemu została wykonana z wykorzystaniem podejścia funkcyjnego do tworzenia oprogramowania.

1.1 Cele pracy

W pracy stawiam następujące cele:

- Stworzenie systemu ekspertowego oceniającego spółki giełdowe przy pomocy analizy technicznej
- Wykazanie skuteczności powstałego systemu przez analizę porównawczą wyników generowanych przez aplikację i rzeczywistych zachowań cen spółek
- Opis zastosowania programowania funkcyjnego w procesie wytwarzania systemu ekspertowego

1.2 Przegląd literatury oraz uzasadnienie wyboru tematu

Temat analizy technicznej rynków finansowych rozwija się od przeszło 100 lat i wiedza z tej dziedziny cały czas jest rozszerzana. W obecnych czasach bardzo trudno jest jednej osobie, nawet jeśli jest ekspertem, przeprowadzić rzetelnie analizę techniczną wybranego rynku finansowego bądź aktywa na tym rynku w czasie, który pozwalałby na wykorzystanie wniosków z takiej analizy do wykonania pożądanych operacji na rynku. Stąd też potrzeba posiadania narzędzia, które będzie analizować przykładowo spółki giełdowe w czasie znacząco krótszym niż zrobiłby to człowiek,

a przy tym z taką samą bądź lepszą trafnością przewidywań zachowań cen. W mojej pracy podstawowym źródłem wiedzy na temat analizy technicznej jest książka "Analiza techniczna rynków finansowych" [14] autorstwa John'a J. Murphy'ego. Wcześniej wspomnianym narzędziem wykonującym analizę z powodzeniem może być system ekspertowy. Taki system może za użytkownika wykonać obliczenia dla wszystkich potrzebnych wskaźników, a także wykorzystując posiadaną bazę wiedzy i reguł przeanalizować zależności pomiędzy wyliczonymi wartościami. Fundamentalną pozycją z zakresu budowy systemów ekspertowych jest pozycja "Systemy ekspertowe" [13] polskiego autora Jana Mulawki. Swój system ekspertowy do analizy technicznej postanowiłem stworzyć z wykorzystaniem funkcyjnego podejścia do programowania. Wybór taki podyktowany był chęcią poszerzenia swojej wiedzy z zakresu metodologii programowania, a także możliwością porównania wybranego podejścia z podejściem klasycznym. Do pisania aplikacji wykorzystałem jeden z dialektów języka Lisp, język Clojure, który również poddam w swojej pracy podstawowej analizie. Więdzę na temat języka Clojure czerpałem przede wszystkim z książki "Programming Clojure" [12] napisanej przez Stuart'a Halloway'a oraz Aaron'a Bedra.

1.3 Układ pracy

Struktura dalszej części pracy jest następująca: Rozdział 2 zawiera opis teorii ... Rozdział 3 przedstawia nową teorię wprowadzoną przez autora pracy ... Rozdział 4 opisuje technologie i narzędzia wykorzystane w pracy ... Rozdział 5 przedstawia wyniki badań / opis stworzonej aplikacji ... Rozdział 6 podsumowuje uzyskane wyniki oraz płynące z nich wnioski ... W Dodatku A zawarto uwagi dotyczące formatowania pracy z użyciem systemu LaTeX. Dodatek B zawiera płytę CD z aplikacją stworzoną w ramach pracy...

Rozdział 2

Teoria

2.1 Systemy ekspertowe

Według Jana Mulawki:
System ekspertowy jest programem komputerowym, który wykonuje złożone zadania o dużych wymaganiach intelektualnych i robi to tak dobrze jak człowiek, będący ekspertem w tej dziedzinie.[1]
Biorąc pod uwagę powyższe można powiedzieć, że system ekspertowy to aplikacja przeprowadzająca proces wnioskowania na podstawie zbioru specjalistycznej wiedzy. Wnioskowanie takie wymaga oprócz wiedzy, zbioru reguł, według których proces ten będzie przeprowadzany. Można wyróżnić następujące elementy składowe systemu ekspertowego:

- Baza wiedzy - Baza specjalistycznej wiedzy (np. zbiór reguł) z dziedziny, dla której tworzony jest system ekspertowy
- Baza danych stałych - Zbiór danych na temat analizowanego problemu
- Silnik wnioskujący - Mechanizm realizujący proces wnioskowania

Ponadto, program będący systemem ekspertowym może posiadać dodatkowo następujące elementy:

- Interfejs użytkownika - Graficzny bądź tekstowy interfejs dla użytkownika korzystającego z programu
- Edytor wiedzy - Edytor pozwalający rozszerzać bazę wiedzy
- Silnik objaśniający - Mechanizm wyjaśniający proces wnioskowania. Prezentuje "tok myślenia", którym system doszedł do uzyskanych wniosków

Już pod koniec lat siedemdziesiątych XX wieku zauważono, że największy wpływ na efektywność działania systemu ekspertowego ma baza wiedzy. Stwierdzono, że im pełniejsza wiedza, tym szybszy jest proces uzyskiwania wniosków przez

[1] [13], str. 20

program. Stworzenie dobrego systemu ekspertowego opiera się więc w dużej mierze na dostarczeniu dla niego odpowiednio dużej bazy dobrej jakościowo wiedzy. Jest to zadanie niełatwe, ponieważ wymaga pozyskania wiedzy od eksperta (bądź grupy ekspertów), który decyzje podejmuje na podstawie informacji o problemie, ale również w oparciu o swoje doświadczenie. Dodatkowo, zebraną wiedzę należy ustrukturalizować, tak aby była możliwa do przetwarzania przez system.[2]

Systemy ekspertowe możemy podzielić ze względu na kilka kryteriów, jednak ogólnie dzielą się one na:

- Doradcze - prezentują rozwiązania dla użytkownika, który potrafi ocenić ich jakość
- Podejmujące decyzję bez kontroli człowieka - system sam podejmuje decyzję, nie poddaje rozwiązania ocenie użytkownika
- Krytykujące - system ma przedstawiony problem i jego rozwiązanie, a następnie ocenia to rozwiązanie

Możemy również podzielić je na dwie grupy:

- Systemy dedykowane - kompletny system ekspertowy posiadający bazę wiedzy
- Systemy szkieletowe - system z pustą bazą wiedzy

W drugim przypadku proces tworzenia systemu ekspertowego polega na dostarczeniu bazy wiedzy. Innym kryterium podziału jest ze względu na logikę wykorzystywaną podczas prowadzenia procesu wnioskowania:

- Z logiką dwuwartościową (Boole'a)
- Z logiką wielowartościową
- Z logiką rozmytą

Można również dokonać podziału ze względu na rodzaj przetwarzanej informacji:

- Systemy z wiedzą pewną
- Systemy z wiedzą niepewną

Najbardziej szczegółowy podział to podział ze względu na zadania realizowane przez system ekspertowy:

- Interpretacyjne - dedukują opisy sytuacji z obserwacji lub stanu czujników
- Predykcyjne - wnioskują o przyszłości na podstawie danej sytuacji
- Diagnostyczne - określają wady systemu na podstawie obserwacji

[2] [13], str. 21

- Kompletowania - konfigurują obiekty w warunkach ograniczeń
- Planowania - podejmują działania, aby osiągnąć cel
- Monitorowania - porównują obserwacje z ograniczeniami
- Sterowania - kierują zachowaniem systemu
- Poprawiania - podają sposób postępowania w przypadku złego funkcjonowania obiektu
- Naprawy - harmonogramują czynności przy dokonywaniu napraw uszkodzonych obiektów
- Instruowania - systemy doskonalenia zawodowego (np. dla studentów)

Jak już wcześniej wspomniano, najistotniejszą częścią systemu ekspertowego jest baza wiedzy. Systemy ekspertowe których baza wiedzy składa się z faktów i reguł nazywa się systemami regułowymi. Zdecydowana większość systemów ekspertowych to właśnie systemy regułowe. Fakty w bazie wiedzy są to zdania oznajmujące, opisujące jakiś obiekt bądź jego stan. Reguły natomiast są zdaniami warunkowymi. Gdy spełnione są przesłanki występujące w części warunkowej takiego zdania, wtedy do bazy faktów dodawane jest nowe zdanie.

Systemy ekspertowe są to aplikacje mieszczące się w dziedzinie sztucznej inteligencji. Jest to nauka definiowana na wiele sposobów. Dwie definicje prezentujące odmienne aspekty badań prowadzonych w tej dziedzinie:

- Według Minsky'ego sztuczna inteligencja jest nauką o maszynach realizujących zadania, które wymagają inteligencji wówczas, gdy są wykonywane przez człowieka.[3]
- Według Feigenbauma sztuczna inteligencja stanowi dziedzinę informatyki dotyczącą metod i technik wnioskowania symbolicznego przez komputer oraz symbolicznej reprezentacji wiedzy stosowanej podczas takiego wnioskowania.[3]

2.2 Programowanie funkcyjne

Najbardziej powszechnym podejściem do programowania jest podejście impreratywne. Programowanie imperatywne opiera się na wykonywaniu kolejnych instrukcji zmieniających stan programu. W podejściu tym wykorzystywane są zmienne, które przechowuje informację o aktualnym stanie aplikacji bądź jej fragmentu. Istotą paradygmatu programowania imperatywnego jest możliwość modyfikowania danych przechowywanych w tych zmiennych, a co za tym idzie zmiana stanu aplikacji. W podejściu tym funkcje, pomimo przekazania takich samych parametrów, mogą zwracać różne wyniki. Spowodowane jest to tzw. efektami ubocznymi

[3] [13], str. 17

działania funkcji. Funkcja napisana z podejściem imperatywnym operuje nie tylko na parametrach do niej przekazanych, ale również na stanie aplikacji, czyli różnego rodzaju zmiennych. Zależnie od stanu w jakim podczas danego wywołania funkcji znajduje się program, można uzyskać różny wynik takiego wywołania.

Programowanie funkcyjne opiera się na ewaluacji funkcji, a nie przechowywaniu stanu aplikacji. W podejściu tym każda funkcja zwraca pewną wartość, a działania wykonuje wyłącznie na otrzymanych parametrach. Dzięki temu funkcje są pozbawione efektów ubocznych i mamy pewność, że wywołanie funkcji z takimi samymi parametrami zawsze zwróci taki sam wynik, niezależnie od chwili wywołania. Jako że każda funkcja jest ewaluowana do pewnej wartości, to mogą być one wykorzystywane jako argumenty innych funkcji. Stąd też jedna z głównych zalet podejścia funkcyjnego - modularność. Kod tworzony zgodnie z paradygmatem funkcyjnym jest łatwy do ponownego wykorzystania i dalszej rozbudowy.

Kolejną istotną cechą jest leniwe wartościowanie. W przeciwieństwie do wartościowania zachłannego, które polega na wyliczaniu wartości wszystkich argumentów przed wywołaniem funkcji, nawet jeśli nie są one wykorzystywane, wartościowanie leniwe polega na ewaluacji wartości argumentów dopiero w chwili odwołania się do nich. Dzięki temu czas procesora jest wykorzystywany wyłącznie na obliczenia, które są konieczne w danym wywołaniu. Dzięki temu również można wyliczyć wartość funkcji nawet jeśli nie jest znana wartość któregoś z jej argumentów, pod warunkiem jednak, że nie jest on wykorzystywany w obliczeniach.

Programowanie funkcyjne w dużym stopniu oparte jest na rekurencji, czyli wywoływaniu funkcji przez samą siebie. Gdyby dla każdego takiego wywołania było zajmowane kolejne miejsce na stosie w pamięci operacyjnej systemu, bardzo szybko doszłoby do przepełnienia stosu. Stąd też powszechną techniką stosowaną przy podejściu funkcyjnym jest rekurencja ogonowa. Jest to rodzaj rekurencji, w którym wywołanie rekurencyjne jest ostatnią instrukcją funkcji. Dzięki temu nie ma potrzeby zajmowania nowego miejsca na stosie. Wykorzystywana jest dotychczasowa ramka stosu funkcji, jako że wszystkie operacje zostały już wykonane i zarezerowana pamięć nie jest już dłużej potrzebna. Taka operacja pozwala na optymalizację zarządzania pamięcią, a także ułatwia tworzenie kodu, jako że rekurencja zazwyczaj jest dużo bardziej naturalna niż iteracyjność. Istotne jest również to, że języki funkcyjne automatycznie przeprowadzają optymalizację rekurencji ogonowej, dzięki czemu programista nie musi się tym zajmować.

Jedną z najważniejszych cech funkcyjnych języków programowania jest posiadanie niemutowalnych struktur danych. Oznacza to, że wartość, bądź zbiór wartości raz przypisany do zmiennej bądź struktury danych nie może zostać zmieniony. Jest to cecha zaskakująca dla osób mających doświadczenie wyłącznie z programowaniem imperatywnym, które opiera się na modyfikowaniu wartości struktur danych i stanu aplikacji. Jednak programowanie funkcyjne opiera się na ewaluacji funkcji, a nie przechowywaniu danych w zmiennych. Dzięki niemutowalności danych unika się również wiele błędów związanych z efektami ubocznymi funkcji, które mogłyby te dane modyfikować.

Języki funkcyjne można podzielić na dwie grupy:

- Języki czysto funkcyjne
- Języki mieszane

Języki czysto funkcyjne są to języki, w których nie występują efekty uboczne, a operacje wejścia/wyjścia odbywają się np. z użyciem monad. W językach tych wartościowanie jest leniwe.

Jezyki mieszanie to takie, które dopuszczają stosowanie zmiennych, operacje wejścia/wyjścia i mieszanie stylu funkcyjnego z imperatywnym (a nawet obiektowym). Funkcje w tych językach mogą mieć również efekty uboczne.

2.3 Analiza techniczna

W tej pracy pojęcie analizy technicznej wykorzystywane jest w odniesieniu do rynku akcji. Analiza techniczna to badanie zachowań rynku, przede wszystkim przy użyciu wykresów, którego celem jest przewidywanie przyszłych trendów cenowych.[4] Zachowanie rynku oznacza w tym wypadku zmiany ceny konkretnego aktywa - akcji spółki giełdowej, a także wolumenu, czyli liczby akcji zmieniających właściciela podczas jednej sesji giełdowej (jednego dnia). Analiza techniczna oparta jest na 3 założeniach:

- Rynek dyskontuje wszystko
- Ceny podlegają trendom
- Historia się powtarza

Pierwszy punkt - rynek dyskontuje wszystko - zakłada, że rynkowa cena akcji w pełni odzwierciedla wszystkie czynniki mogące mieć wpływ na wartość aktywa, takie jak czynniki fundamentalne, polityczne czy psychologiczne. Analityk wykresów opiera się na prawie popytu i podaży. Jeśli popyt przewyższa podaż, to cena powinna rosnąć. Jeśli natomiast podaż przewyższa popyt - cena powinna spadać. Analityka nie interesują przyczyny zmiany popytu i podaży, a jedynie jej skutki.

To, że ceny podlegają trendom jest kolejną przesłanką na jakiej opiera się analiza techniczna. Jej uzupełnieniem jest stwierdzenie, że trend kontynuuje swój bieg w dotychczasowym kierunku tak długo, aż nie pojawią się przesłanki do tego, aby się zmienił.

Ostatnie założenie opiera się mocno na badaniu ludzkiej psychiki. W analizie technicznej zakłada się, że ludzka psychika, która ma przełożenie na trendy cenowe na rynku, raczej się nie zmienia. W związku z tym trendy występujące w przeszłości powinny powtórzyć się również w przyszłości.

Warto w tym miejscu wspomnieć również o drugim rodzaju analizy rynków finansowych - analizie fundamentalnej. Różni się ona tym, że koncentruje się na badaniu gospodarczych uwarunkowań popytu i podaży, które są przyczyną wzrostów, spadków lub stabilizacji cen. Przy podejściu fundamentalnym bada się wszystkie

[4] [14], str. 1

czynniki oddziałujące na cenę danego towaru, w celu określenia jego rzeczywistej wartości. Jeśli rzeczywista wartość danego towaru jest niższa od jego bieżącej ceny rynkowej, znaczy to, że towar ów jest "przewartościowany" i należy go sprzedać. Jeśli cena towaru jest niższa od jego rzeczywistej wartości, jest on "niedowartościowany" i należy go kupić.[5]

Analizę techniczną podzielić możemy na dwa obszary. Pierwszym z nich jest analiza wykresów i odnajdywanie na nich formacji cenowych, sugerujących kontynuację trendu, bądź jego odwrócenie. Drugi to analiza wskaźników analizy technicznej. Stworzono wiele wskaźników pomocnych w rozpoznawaniu określonych stanów w jakich znajduje się rynek, a także generujących sygnały takie jak sygnał kupna/sprzedaży, zmiany bądź kontynuacji trendu.

[5] [14], str. 4

Rozdział 3

Tytuł części z teorią własną

Jeśli w ramach pracy autor rozwinął nową teorię należy zamieścić ją w tym rozdziale.

3.1 Podrozdział...

Rozdział 4

Technologie i narzędzia

4.1 Język programowania

Aplikacja została zrealizowana w języku Clojure [9] w wersji 1.5.1. Jest to język programowania działający na maszynie wirtualnej jawy (ang. JVM - Java Virtual Machine). Jest to język ogólnego przeznaczenia kompilowany do bajtkodu JVM. Clojure jest językiem funkcyjnym, jednym z dialektów języka Lisp. Tak jak Lisp jest językiem listowym (wszystkie elementy języka są listami), realizującym filozofię "kod jako dane".

Pierwszym powodem wyboru tego języka programowania jest to, że języki funkcyjne bardzo dobrze sprawdzają się przy tworzeniu systemów ekspertowych, a także pozwalają na generowanie nowego kodu aplikacji w trakcie jej działania. Samo programowanie funkcyjne jest natomiast dziedziną, która nie jest przybliżana podczas studiów. Stąd też chęć zapoznania się z tym paradygmatem programowania.

Kolejnym powodem użycia języka Clojure jest możliwość wykorzystania w nim elementów języka Java. Dzięki temu w łatwy i szybki sposób mogłem stworzyć graficzny interfejs użytkownika dla aplikacji i więcej czasu poświęcić na te fragmenty aplikacji, które są nieodłącznymi elementami systemu ekspertowego.

Ostatnią przyczyną wybrania Clojure jest to, że programy pisane w tym języku kompilowane są do bajtkodu JVM, a więc można je uruchamiać na każdej maszynie i systemie operacyjnym posiadającym maszynę wirtualną jawy.

4.2 Oprogramowanie

Podczas tworzenia systemu ekspertowego dla tej pracy wykorzystywałem następujące oprogramowanie.

4.2.1 Środowisko programistyczne

Aplikacja była tworzone częściowo na komputerze z systemem operacyjnym Windows 7, a częściowo na komputerze z systemem Windows 8. Dzięki temu, że programy pisane w Clojure uruchamiane są na maszynie wirtualnej jawy, nie występowały

rzadne problemy podczas przenoszenia kodu pomiędzy tymi systemami i dalszym rozwijaniu go i uruchamianiu. Dzięki temu również praca byłaby możliwa chociażby na systemach Unixowych.

Do budowania aplikacji i zarządzania zależnościami wykorzystywałem środowisko Leiningen [10]. Jest to narzędzie do łatwego konfigurowania projektów tworzonych w języku Clojure.

Do budowania i uruchamiania aplikacji wykorzystywane jest środowisko JDK (ang. Java Development Kit) w wersji 7u25.

W trakcie pisania aplikacji korzystałem z edytora Clooj [7]. Jest to proste, zintegrowane środowisko programistyczne dla języka Clojure (jak i tworzone w tym języku). Edytor dostarcza podstawowe funkcjonalności, takie jak kolorowanie składni, dostęp do dokumentacji funkcji języka, czy możliwość łatwego tworzenia nowych plików źródłowych z automatycznym generowaniem nowych przestrzeni nazw.

4.2.2 Wykorzystane biblioteki

Podczas tworzenie aplikacji wykorzystywałem następujące biblioteki:

- Seesaw [6] - biblioteka do tworzenia interfejsów użytkownika w Clojure, jest zbudowana w oparciu o bibliotekę graficzną Swing
- Incanter [8] - biblioteka do obliczeń matematycznych, statystycznych i tworzenia wykresów w Clojure
- Instaparse [5] - biblioteka do tworzenia parserów gramatyk w Clojure
- clj-time [4] - biblioteka do operacji na datach i czasie w Clojure
- clj-http [3] - biblioteka do obsługi zapytań http w Clojure

4.3 Techniki i metodologie programistyczne

Projekt tworzony był z wykorzystaniem paradygmatu programowania funkcyjnego. Nie jest on jednak zrealizowany w czystym podejściu funkcyjnym. Przechowuje w postaci list wyliczone wartości wskaźników analizy technicznej, a także realizuje operację wczytywania danych o spółce giełdowej z pliku.

Rozdział 5

Wyniki badań eksperymentalnych

5.1 Opis stworzonej aplikacji

5.1.1 Podstawowy opis

Aplikacja powstająca podczas pisania tej pracy to system ekspertowy, którego zadaniem jest ocena spółek giełdowych i podejmowanie decyzji, czy w danej chwili konkretną spółkę warto kupić bądź sprzedać. Użytkownik aplikacji może pobrać notowania spółek giełdowych notowanych na Giełdzie Papierów Wartościowych w Warszawie z serwisu bossa.pl. Może również korzystać z dowolnego innego źródła notowań, pod warunkiem, że dane będą archiwum ZIP zawierającym pliki w formacie MST o nazwach odpowiadających nazwom spółek. Po tym na liście dostępne są spółki, których dane znajdowały się w archiwum. Aplikacja po wyborze konkretnej spółki tworzy wykres jej ceny i wolumenu w cały okresie w jakim była notowana. Użytkownik ma możliwość wyboru z listy ile sesji chce wyświetlić na wykresie, a także czy ma być tworzony wykres wartości wolumenu. Po wczytaniu danych o konkretnej spółce przycisk Analizuj pozwala na przeprowadzenie procesu wnioskowania dla danej spółki i określenie czy w danej chwili warto sprzedawać bądź kupować dane akcje.

5.1.2 Struktura projektu

Projekt podzielony jest na 3 przestrzenie nazw:

- bachelor.core
- bachelor.wsk
- bachelor.zip

Przestrzeń nazw	Funkcjonalności
bachelor.core	Zawiera wszystkie funkcje parsera reguł, funkcje silnika wnioskującego, wczytywanie listy spółek, których dane znajdują się w katalogu z notowaniami. Tutaj również zdefiniowany jest interfejs użytkownika. Dodatkowo jest tu funkcja uruchamiająca wypakowywanie archiwum ZIP z notowaniami
bachelor.wsk	Tutaj zdefiniowana jest funkcja wczytująca notowania z pliku i tworząca z nich odpowiednią strukturę umieszczaną na liście notowań. W tej przestrzeni nazw zdefiniowane są również wszystkie wskaźniki wykorzystywane przez system wnioskujący, a także funkcje pomocnicze do wyliczania wskaźników i operacji na liście notowań. Jest tu również umieszczona funkcja tworząca listy wszystkich zdefiniowanych wskaźników, a także funkcje odwołujące się do tych list, które wykorzystywane są w regułach.
bachelor.zip	Zdefiniowana jest tu funkcja do pobierania archiwum ze wskazanego adresu internetowego i zapisu go na dysku. Znajdują się tu także wszystkie funkcje pomocnicze do rozpakowywania archiwum

5.1.3 Gramatyka reguł

Jedną z funkcjonalności systemu jest parser reguł, które są wczytywane z pliku tekstowego, a następnie na ich podstawie generowany jest kod nowych funkcji, które przekazywane są do mechanizmu wnioskującego. Aby można było stworzyć taki parser najpierw trzeba było zaprojektować gramatykę, zgodnie z którą można zapisywać reguły w pliku tekstowym.

Rozdział 6

Podsumowanie i wnioski

Podsumowanie jest, obok Wstępu, najważniejszym rozdziałem pracy. Należy tutaj jeszcze raz podsumować wykonane prace. Szczególny nacisk należy położyć na wkład własny autora i uzyskane oryginalne rezultaty. Należy odwołać się do celów pracy z rozdziału 1.1 – można je powtórzyć – i jasno wskazać, że zostały one zrealizowane (należy powołać się na wyniki z rozdziału 5). Wyniki należy podsumować zwięźle i precyzyjnie, np. *uzyskano przyspieszenie algorytmu o X%..., skrócono czas o ...* itd. **Należy wskazać perspektywy dalszych badań.**

Bibliografia

[1] KBibTeX. http://home.gna.org/kbibtex/.

[2] Kile - an Integrated LaTeX Environment. http://kile.sourceforge.net/.

[3] Repozytorium biblioteki clj-http. https://github.com/dakrone/clj-http.

[4] Repozytorium biblioteki clj-time. https://github.com/clj-time/clj-time.

[5] Repozytorium biblioteki instaparse. https://github.com/Engelberg/instaparse.

[6] Repozytorium biblioteki seesaw. https://github.com/daveray/seesaw.

[7] Repozytorium projektu clooj. https://github.com/arthuredelstein/clooj.

[8] Strona domowa biblioteki incanter. http://incanter.org/.

[9] Strona domowa języka clojure. http://clojure.org/.

[10] Strona domowa projektu leiningen. http://leiningen.org/.

[11] TeX Live. http://www.tug.org/texlive/.

[12] S. Halloway. *Programming Clojure*. 2012.

[13] J. Mulawka. *Systemy ekspertowe*. 1997.

[14] J. Murphy. *Analiza techniczna rynków finansowych*. 2008.

[15] A. Niewiadomski. Szablon pracy dyplomowej. http://ics.p.lodz.pl/~aniewiadomski/mgr/szablon-konspekt.pdf, 2010.

[16] T. Oetiker. The not so short introduction to LaTeX2e. http://tobi.oetiker.ch/lshort/lshort.pdf, 2011.

Dodatek A

Edycja i formatowanie pracy

Pracę należy przygotować korzystając z systemu składu LaTeX (czyt. *latech*). Bardzo dobre wprowadzenie do LaTeX stanowi "The Not So Short Introduction to LaTeX 2_ε" [16].

A.1 Kwestie techniczne

Aby móc składać dokumenty z użyciem LaTeX należy go oczywiście najpierw zainstalować. Dostępnych jest wiele dystrybucji LaTeX. Osobiście korzystam z TeX Live [11], dostępnego zarówno pod Windowsa jak i Linuksa. Dobrze jest również zaopatrzyć się w środowisko do LaTeX i BibTeX (o tym w sekcji A.3). Użytkownikom Linuksa polecam do tego programy Kile [2] oraz KBibTeX [1].

A.2 Formatowanie

Należy zachować formatowanie zgodne z niniejszym szablonem. Preferowany jest druk dwustronny. W związku z tym proszę zwrócić szczególną uwagę na położenie szerszego marginesu. Powinien się on oczywiście znajdować od strony bindowania.

Wszystkie ustawienia marginesów, stylu nagłówków, wykorzystanych pakietów etc. znajdują się w pliku `praca_dyplomowa.sty`.

Należy unikać wiszących spójników (zwanych też sierotami). W tym celu należy stosować po spójnikach twardą spację. W LaTeX uzyskuje się ten efekt poprzez umieszczenie pomiędzy spójnikiem a następującym po nim słowem znaku tyldy. W celu ułatwienia tego procesu do szablonu dołączono plik `korekta.sh`. Jest to prosty skrypt basha – użytkownicy Windowsa muszą go dostosować do swojego systemu – który wywołuje polecenia perlowe wstawiające twardą spację po spójnikach we wszystkich plikach z rozszerzeniem *.tex w bieżącym katalogu. Przed uruchomieniem skryptu dobrze jest wykonać kopię zapasową.

Przy składaniu pracy przydatny może okazać się tryb draft, który sprawi że zaznaczane będą miejsca w których tekst nie został prawidłowo złamany. Jeśli LaTeX nie potrafi prawidłowo złamać jakiegoś słowa można w preambule dokumentu użyć polecenia `hyphenation`.

A.3 Bibliografia

Wszystkie źródła informacji, rysunków, danych itd., które zostały wykorzystane przy tworzeniu pracy muszą zostać podane w bibliografii. Ponadto, wszystkie źródła podane w bibliografii muszą być cytowane w tekście. Za każdym razem, kiedy w pracy pojawia się treść na podstawie jakiegoś tekstu źródłowego czyjegoś autorstwa, oznaczamy takie miejsce przypisem[1]. **Należy pamiętać, że korzystanie ze źródeł bez ich podania w bibliografii może być podstawą do oskarżenia o plagiat.**

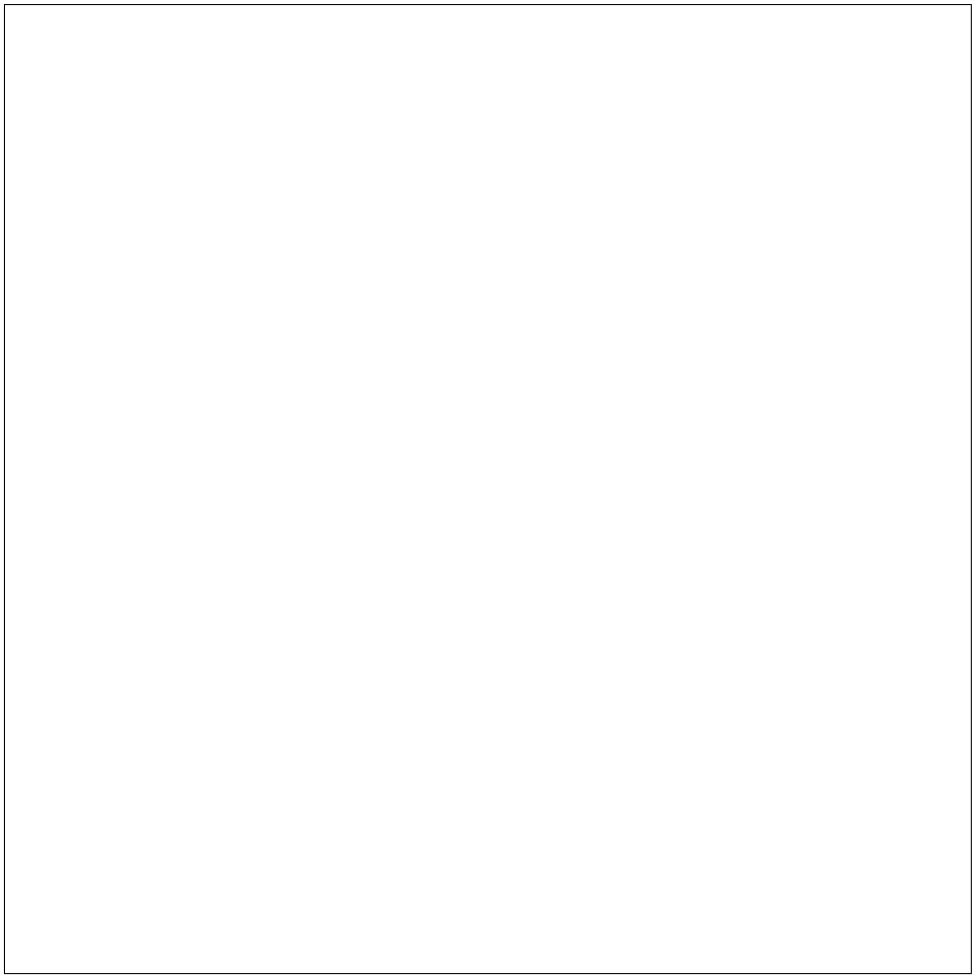
Należy zwrócić szczególną uwagę na jakość cytowanych źródeł internetowych. Najlepszym rozwiązaniem jest ograniczenie się, na ile to możliwe, do oficjalnych stron projektów. Ponadto, odnośniki do źródeł elektronicznych muszą zawierać pełną ścieżkę do zasobu.

Bibliografię należy przygotować korzystając z mechanizmów dostarczanych przez LaTeX (patrz rozdział 4.2 w [16]). Zalecam korzystanie w tym celu z BibTeX. BibTeX sam wygeneruje bibliografię na podstawie przygotowanej prostej bazy danych i zagwarantuje że w spisie literatury pojawią się tylko te pozycje, które są faktycznie cytowane w pracy. Pozwoli to zaoszczędzić naprawdę sporo czasu.

[1] [15], str. 9

Dodatek B

Płyta CD



Do pracy należy dołączyć podpisaną płytę CD w papierowej kopercie. Poniżej należy zamieścić opis zawartości katalogów.

Zawartość katalogów na płycie:

dat : pliki z danymi wykorzystane w trakcie badań

db : Zrzut bazy danych potrzebnej do działania aplikacji

dist : dystrybucyjna wersja aplikacji przeznaczona do uruchamiania

doc : elektroniczna wersja pracy magisterskiej oraz dwie prezentacje wygłoszone podczas seminarium dyplomowego

ext : ten katalog powinien zawierać ewentualne aplikacje dodatkowe potrzebne do uruchomienia stworzonej aplikacji, np. środowisko Java, PostgreSQL itp.

src : kod źródłowy aplikacji (projekt środowiska Eclipse / Netbeans / Qt Creator / ...)

Oczywiście nie wszystkie powyższe katalogi są wymagane, np. dat, db albo ext mogą być niepotrzebne.