

# **Wyższa Szkoła Komunikacji i Zarządzania w Poznaniu**

**Wojciech Jurkowlaniec**

**Aplikacja wspomagająca proces uczenia się  
oraz zapamiętywanie informacji w oparciu  
o technikę Mind Mapping**

Praca dyplomowa inżynierska

Promotor: dr inż. Ewa Idzikowska

Instytut Informatyki

Instytut Informatyki

kierunek: Informatyka

specjalność: Technologie i zastosowania Internetu

Poznań 2011

# Spis treści

<b>1 Wstęp</b>	<b>5</b>
<b>2 Procesy poznawcze towarzyszące nauce</b>	<b>6</b>
2.1 Skupianie uwagi . . . . .	6
2.2 Odbieranie i analizowanie informacji . . . . .	6
2.3 Zapamiętywanie informacji . . . . .	7
2.4 Podsumowanie . . . . .	7
<b>3 Mind mapping</b>	<b>8</b>
3.1 Nauka a ludzki mózg . . . . .	8
3.2 Zasady tworzenia map pamięci . . . . .	10
<b>4 Przegląd aplikacji do tworzenia map pamięci</b>	<b>11</b>
4.1 XMind . . . . .	11
4.2 FreeMind . . . . .	12
4.3 Buzan's iMindMap . . . . .	13
<b>5 Specyfikacja wymagań funkcjonalnych programu</b>	<b>14</b>
5.1 Tworzenie map pamięci . . . . .	14
5.2 Prezentacja map pamięci . . . . .	14
5.3 Serializacja . . . . .	14
5.4 Interakcja z przeglądarką internetową . . . . .	14
<b>6 Specyfikacja wymagań pozafunkcjonalnych programu</b>	<b>15</b>
6.1 Wydajność . . . . .	15
6.2 Użyteczność . . . . .	15
6.3 Przenośność . . . . .	15
6.4 Niezawodność . . . . .	16
<b>7 Wstępna konstrukcja programu</b>	<b>17</b>
<b>8 Wykorzystane technologie i narzędzia</b>	<b>18</b>
8.1 Python . . . . .	18
8.2 Qt . . . . .	18
8.3 PyQt . . . . .	18
8.4 Javascript, XUL i XPCOM . . . . .	18
<b>9 Architektura programu</b>	<b>20</b>
9.1 Zarządzanie mapami pamięci . . . . .	20
9.2 System sygnałów i slotów . . . . .	22
9.3 Interakcja z przeglądarką internetową . . . . .	24
9.4 Konstrukcja kodu źródłowego aplikacji . . . . .	24
<b>10 Interfejs użytkownika</b>	<b>25</b>
10.1 Poruszanie się w widoku mapy pamięci . . . . .	25
10.2 Manipulacja węzłami . . . . .	26
10.3 Dodawanie węzłów z tekstu . . . . .	26

<b>11 Przykład użycia programu</b>	<b>28</b>
11.1 Wybór tekstu . . . . .	28
11.2 Wybór słów-kluczy . . . . .	28
11.3 Dostosowanie mapy pamięci . . . . .	30
11.4 Uczenie się z mapy pamięci . . . . .	31
<b>12 Uruchamianie programu</b>	<b>32</b>
12.1 Instalacja rozszerzenia do przeglądarki Firefox . . . . .	32
12.2 Uruchamiania aplikacji z argumentami . . . . .	33
12.2.1 Tryb tworzenia mapy pamięci z tekstu . . . . .	33
12.2.2 Tryb Wczytania pliku przy uruchomieniu . . . . .	33
<b>13 Podsumowanie pracy</b>	<b>34</b>

## **Spis rysunków**

- 3.1 Przykład sieci semantycznej
- 3.2 Funkcje półkul mózgowych
- 3.3 Przykładowa mapa pamięci
- 4.1 Wygląd interfejsu użytkownika programu XMind
- 4.2 Wygląd interfejsu użytkownika programu FreeMind
- 4.3 Wygląd interfejsu użytkownika programu iMindMap
- 9.1 Schemat UML elementów sceny
- 9.2 Schemat UML interfejsu użytkownika
- 10.1 Wygląd interfejsu użytkownika programu
- 10.2 Wygląd okna edycji węzła
- 10.3 Wygląd okna dodawania węzłów z tekstu
- 11.1 Wygląd okna przeglądarki internetowej
- 11.2 Wygląd interfejsu programu po dodaniu tekstu
- 11.3 Wygląd interfejsu programu po dodaniu paru węzłów
- 11.4 Drukowanie mapy pamięci

# 1 Wstęp

Żyjemy w czasach, gdzie komputery stały się łatwo dostępnym i używanym na co dzień narzędziem pracy czy rozrywki. Wykorzystuje się je do rozrywki, wzajemnej komunikacji, pogłębiania zainteresowań, a także do nauki. Internet stał się nieograniczonym źródłem informacji, w którym niekiedy trudno się odnaleźć. Spowodane jest to ogromną ilością dostępnych danych, przez co ich przyswajanie, czyli nauka staje się problematyczna.

Najpopularniejszą metodą uczenia się jest notowanie mówionego słowa, a następnie powtarzanie przez wielokrotne czytanie. Jednakże nie jest to technika efektywna dla wszystkich. Dlatego alternatywą może być wykorzystanie *Mind Mappingu*, który pozwala na zapamiętanie większej ilości informacji w krótszym czasie. Sprawia, że tworzenie notatek jest mniej uciążliwe, a system powtórek pozwala na zapamiętanie informacji o dowolnej złożoności długofalowo.

Celem niniejszej pracy jest opracowanie aplikacji umożliwiającej wspomaganie procesu uczenia się, przy wykorzystaniu metod efektywnej nauki. Program będzie oparty o tworzenie map pamięci oraz funkcjonalność szybkiego tworzenia ww. map przy pomocy przeglądarki internetowej. Zostanie również przedstawiony interfejsu użytkownika programu oraz objaśnienie działania tej techniki.

Praca została podzielona na dwie części - teoretyczną i praktyczną. Część teoretyczną obejmują rozdziały 2-4, gdzie zostaje objasniona technika Mind Mappingu, omówienie procesów poznawczych towarzyszących nauce oraz analizą istniejących aplikacji wspomagających tworzenie map pamięci.

Część praktyczną obejmują rozdziały 5-12, gdzie zostaje opisana koncepcja programu do tworzenia map pamięci, specyfikacja wymagań funkcjonalnych i niefunkcjonalnych, zostaną opisane wykorzystane technologie i narzędzia a następnie architektura programu i opis interfejsu użytkownika. Przedostatni rozdział traktuje o instalacji programu i różnych trybach pracy.

## 2 Procesy poznawcze towarzyszące nauce

By dobrze zrozumieć pojęcia związane z mapami pamięci oraz efektywnym uczeniem się i zapamiętywaniem potrzebne jest parę słów wstępnu na temat procesów poznawczych. Ten rozdział jest podzielony na cztery podrozdziały które opisują poszczególne fazy związane z uczeniem się.

### 2.1 Skupianie uwagi

Skupianie uwagi jest pierwszym elementem towarzyszącym uczeniu się. Jest bardzo ważny, gdyż bez koncentracji nie można przyswajać informacji. By dobrze zrozumieć ten proces, należy przytoczyć dwie definicje.

*Uwaga* to system odpowiedzialny za selekcję informacji i zapobieganie negatywnym skutkom przeładowania systemu poznawczego przez nadmiar danych [PSYCHO].

*Świadomość* to zdawanie sobie przez podmiot sprawy z treści własnych procesów psychicznych [PSYCHO].

Głównym elementem procesu skupiania uwagi jest *przeszukiwanie pola percepcyjnego*. Mózg ludzki dokonuje tej czynności automatycznie. Przykładowo jak osoba stoi na ulicy w tłumie i szuka swojego znajomego, który ma na sobie czerwoną kurtkę, każde pojawienie się czerwonego koloru w zasięgu pola widzenia sprawia że osoba reaguje ze szczególną uwagą na te wystąpienia. Najłatwiej rozpoznaje się obiekty które posiadają cechy priorytetowe, różnią się od otoczenia kolorem, kształtem czy rozmiarem.

Kolejnym ważnym elementem jest *podzielność uwagi*, które ozacza dzielenie pewnej ograniczonej puli "mocy obliczeniowej" na więcej niż jedno zadanie. Jednak robienie paru rzeczy na raz musi spowodować zakłócenie przynajmniej jednej z nich, dlatego ważne jest by tych czynności nie było zbyt wiele.

Również warto zwrócić uwagę na *przedłużoną koncentrację* (ang. sustained attention). Polega na zdolności utrzymania uwagi selektywnej przez dłuższy czas na tym samym aspekcie otoczenia.

### 2.2 Odbieranie i analizowanie informacji

Na procesy odbierania i analizowania informacji składa się percepcja. Jest to zbiór procesów aktywnej interpretacji danych otoczenia. Dzięki wskazówkom kontekstowym, nastawieniu i wcześniej zdobyta wiedzą możliwe jest rozpoznanie obiektów. Do jej zadań należy dostarczanie informacji z całego ciała, uwzględniając narządy wewnętrzne, zmysły i aktualne położenie kończyn. W wyniku tych procesów pojawiają się spostrzeżenia, zwane *perceptami* [PSYCHO].

Ważną rzeczą przy uczeniu się jest rozpoznawanie wzorców (ang. *pattern recognition*). Jest to bardzo ważna umiejętność, gdyż dzięki niej można na przykład rozpoznawać pismo ręczne. Każdy wzorzec jest podświadomie kategoryzowany, i trwa on zależnie od ilości kategorii który ten proces musi sprawdzić [COGNI]. Warto zauważyć, że im struktura

identyfikowanych informacji jest bardziej ujednolicona, tym szybciej następuje ten proces.

## 2.3 Zapamiętywanie informacji

Pamięć jest zdolnością do przechowywania informacji i późniejszego jej wykorzystania. Składa się na niego zespół procesów poznawczych. Mimo badań nad pamięcią trwającą już od wieków, nie udało się jednoznacznie stwierdzić w jaki sposób informacje są zapisywane. Jednak najbardziej przemawiającym modelem pamięci opartym na kryterium czasu przechowywania informacji jest podział na trzy systemy [COGNI]:

- magazyn sensoryczny (ang. *sensory storage*)
- magazyn krótkotrwały (ang. *short term storage*)
- magazyn długotrwały (ang. *long term storage*)

*Magazyn sensoryczny* jest specyficzny dla powiązanego z nim zmysłu. Podstawową funkcją każdego z nich jest przetrzymanie bodźca przez krótki czas (maksymalnie kilkaset milisekund). Przykładowo gdy zgasi się światło, osoba jeszcze przez chwilę “widzi” obraz pomieszczenia w którym się znajduje. Zbiór wszystkich informacji sensorycznych jest przekazywany do systemu poznawczego.

*Magazyn pamięci krótkotrwałej* działa w inny sposób niż sensoryczny, gdyż zbiera informacje które zostały już przetworzone przez system poznawczy, jak również przywołane z pamięci długotrwałej czy efekt bieżącego przetwarzania informacji. Podstawową funkcją tego rodzaju pamięci jest przechowywanie informacji w czasie niezbędnym do jej przetworzenia zgodnie z celem, jaki aktualnie realizujemy. Czas ten waha się od kilkunastu do kilkudziesięciu sekund.

*Magazyn pamięci długotrwałej* z kolei charakteryzuje się najdłuższym czasem przechowywania informacji, mierzonym w perspektywie lat. Jednak informacja trafiająca do pamięci długotrwałej nie pozostaje tam na zawsze. Przykładem jest nauka języków obcych. W momencie gdy przez długi czas nie używamy danego języka, wiedza na jego temat zanika.

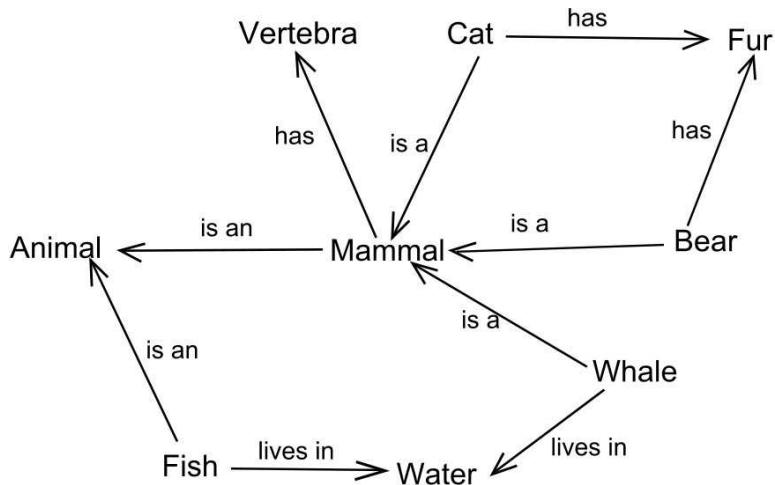
## 2.4 Podsumowanie

Patrząc powyższy rozdział można zwrócić uwagę, że przed zapisaniem informacji “na stałe” potrzebne jest dużo procesów nim trafi ona do magazynu pamięci długotrwałej. Na początku osoba ucząca się musi skupić uwagę na przedmiocie badań, by zacząć go analizować. Ważne w tym momencie jest, by uwaga nie była rozpraszana, przedmiot był zróżnicowany i w zasięgu wzroku nie było elementów przeszkadzających uczeniu się.

Śledząc dalsze kroki w celu zapamiętania informacji, po pomyślnym skupieniu się na obiekcie który zawiera informacje, umysł musi: rozpoznać jakiego typu jest to przedmiot, przeanalizowanie jak ważna jest jej zawartość i utrwalic w pamięci długotrwałej. Patrząc na mapę pamięci, wszystkie te zadania mamy ułatwione. Zawiera ona ujednoliconą strukturę jak i bogatą formę przekazu, zmuszającą na aktywowanie obu półkul mózgowych i szybkiego przypomnienia sobie przelanych tam wcześniej informacji.

### 3 Mind mapping

Obrazowe metody utrwalania wiedzy i modelowania systemów były używane od wieków w nauczaniu, wizualnym myśleniu i rozwiązywaniu problemów przez inżynierów, nauczycieli, psychologów i innych. Jeden z pierwszych przykładów takich graficznych zapisów został stworzony przez Porfiriusza, greckiego myśliciela żyjącego w III wieku n.e. Pierwszym krokiem standaryzacji tego rodzaju organizacji myśli były *sieci semantyczne*. Są to sieci reprezentujące semantyczne relacje pomiędzy koncepcjami. Zostały one opracowane przez Richarda H. Richensa w celu zdefiniowania meta-języka do translacji maszynowej na języki naturalne [COGNI]. Przykład takiej sieci semantycznej ilustruje rysunek 3.1

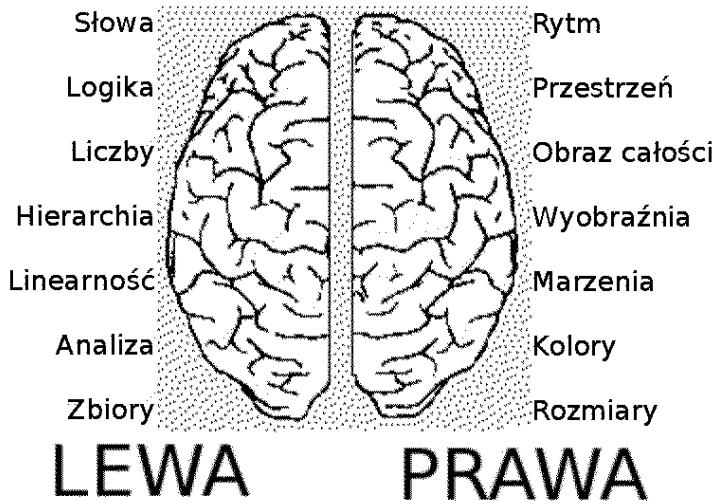


Rys 3.1 Przykład sieci semantycznej

Technika *Mind Mappingu* została opracowana przez dwóch brytyjskich naukowców, Tony'ego i Barry'ego Buzana w latach 60-tych XX wieku. Służy do reprezentowania słów, idei, zadań i innych elementów połączonych i ułożonych dookoła centralnego słowa-klucza bądź idei. Mapy pamięci są wykorzystywane do wizualizacji oraz strukturyzacji. Są przydatne przy rozwiązywaniu problemów, podczas podejmowania decyzji oraz pisania. Poprzez reprezentowanie idei w graficzny, promieniowy i nieliniowy sposób, mapy pamięci są często wykorzystywane podczas "burzy mózgów" jako sposób reprezentacji planów i organizacji zadań. Metoda ta stała się bardzo popularna i jest wykorzystywana przez miliony ludzi na całym świecie [BUZAN2].

#### 3.1 Nauka a ludzki mózg

Mózg ludzki składa się dwóch półkul odpowiedzialnych za różne procesy myślowe. Lewa półkula zajmuje się analizą słowa, logiką, liczbami, hierarchią, linearnością i zbiogrami. Natomiast prawa półkula odpowiada za rytm, świadomość przestrzeni, obraz całości, wyobraźnię, marzenia, postrzeganie kolorów i rozmiarów. W czasie uczenia się z wykorzystaniem notatek linearnych wykorzystuje się głównie lewą półkulę. Rysunek 3.2 przedstawia podział półkul mózgowych na funkcje [COGNI].

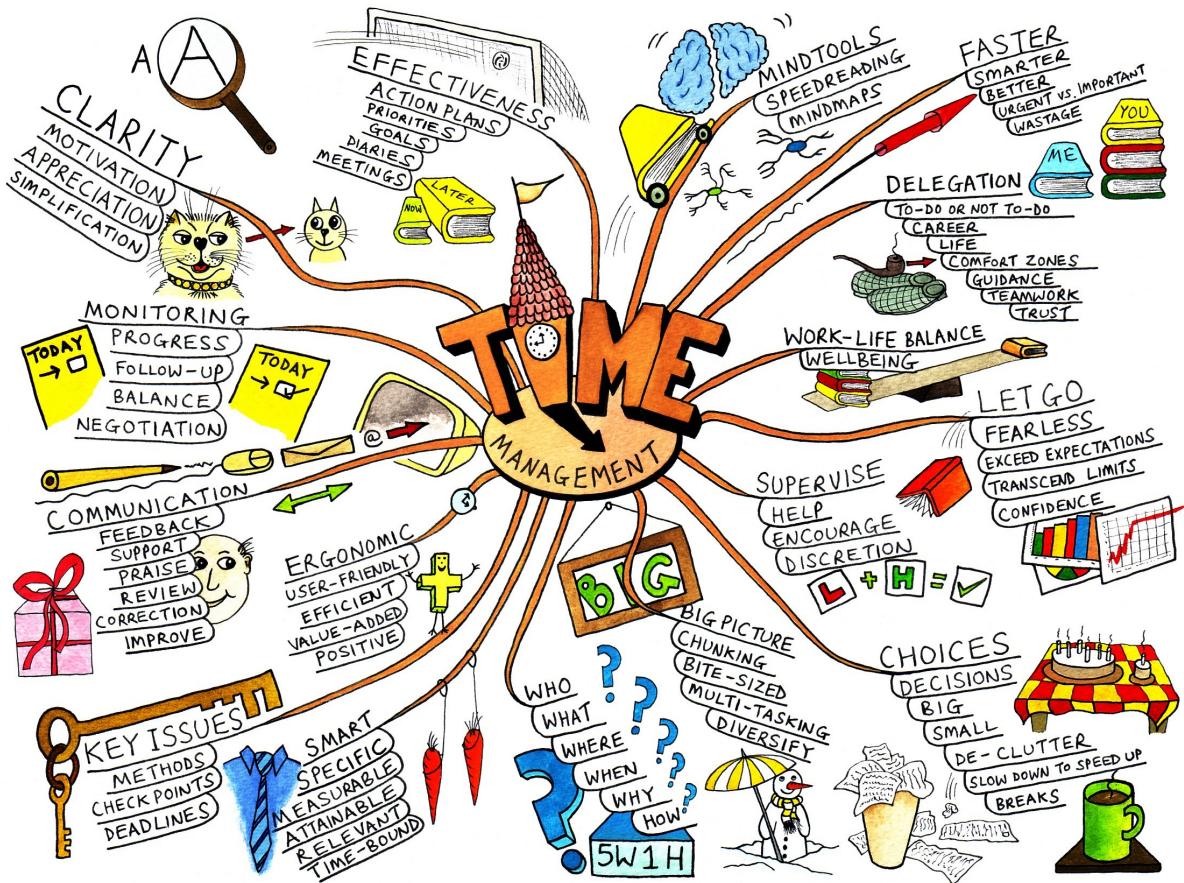


Rys 3.2 Funkcje półkul mózgowych

Najczęściej spotykanymi sposobami notowania są [BUZAN]:

1. Notatki w formie narracyjnej - spisywanie treści komunikatu w formie ciągu zdań
2. Notatki w formie listy myśli w kolejności pojawiania się
3. Notatki w formie hierarchicznej - uporządkowanie wiedzy w strukturę, oznaczoną literami lub liczbami

Ten rodzaj notatek pobudza mózg za pomocą analizy: słów, liczb, kolejności, linii, zbiorów i logiki. Patrząc na rysunek 3.2 można zauważyć, że analizą tych elementów zajmuje się głównie lewa półkula. By się efektywniej uczyć, należy uaktywnić w jak największym stopniu mózg do działania. Elementami które również można użyć podczas tworzenia notatek są: rytm wizualny, schemat, kolor, obraz, wizualizacja, wymiar, zmysł przestrzeni i skojarzenia. Są to rzeczy, które również uczestniczą w procesie myślenia, zapamiętywania i przypominania, dlatego warto wykorzystać je również podczas tworzenia notatek. Jedną z metod pobudzenia wszystkich obszarów mózgu jest technika tworzenia map pamięci. Przykład takiej mapy ilustruje rysunek 3.3



Rys 3.3 Przykładowa mapa pamięci

### 3.2 Zasady tworzenia mapy pamięci

Tony Buzan skonstruował 4 podstawowe zasady potrzebne przy tworzeniu mapy pamięci [BUZAN]:

- Temat mapy symbolizuje centralny rysunek
- Główne zagadnienia w postaci gałęzi wybiegają promieniście z centralnego rysunku
- Gałęzie zawierają kluczowy rysunek lub słowo (wypisane dużymi literami nad odpowiednią linią). Zagadnienia poboczne lub mniej ważne reprezentowane są jako gałęzie podporządkowane gałęziom wyższego rzędu
- Gałęzie tworzą sieć węzłów

Ważne jest również, by wzbogacać je o kolory, rysunki oraz własne skróty myślowe, gdyż dzięki temu pobudzana jest kreatywność i łatwiej sobie przypomnieć wiedzę zgromadzoną na mapie pamięci.

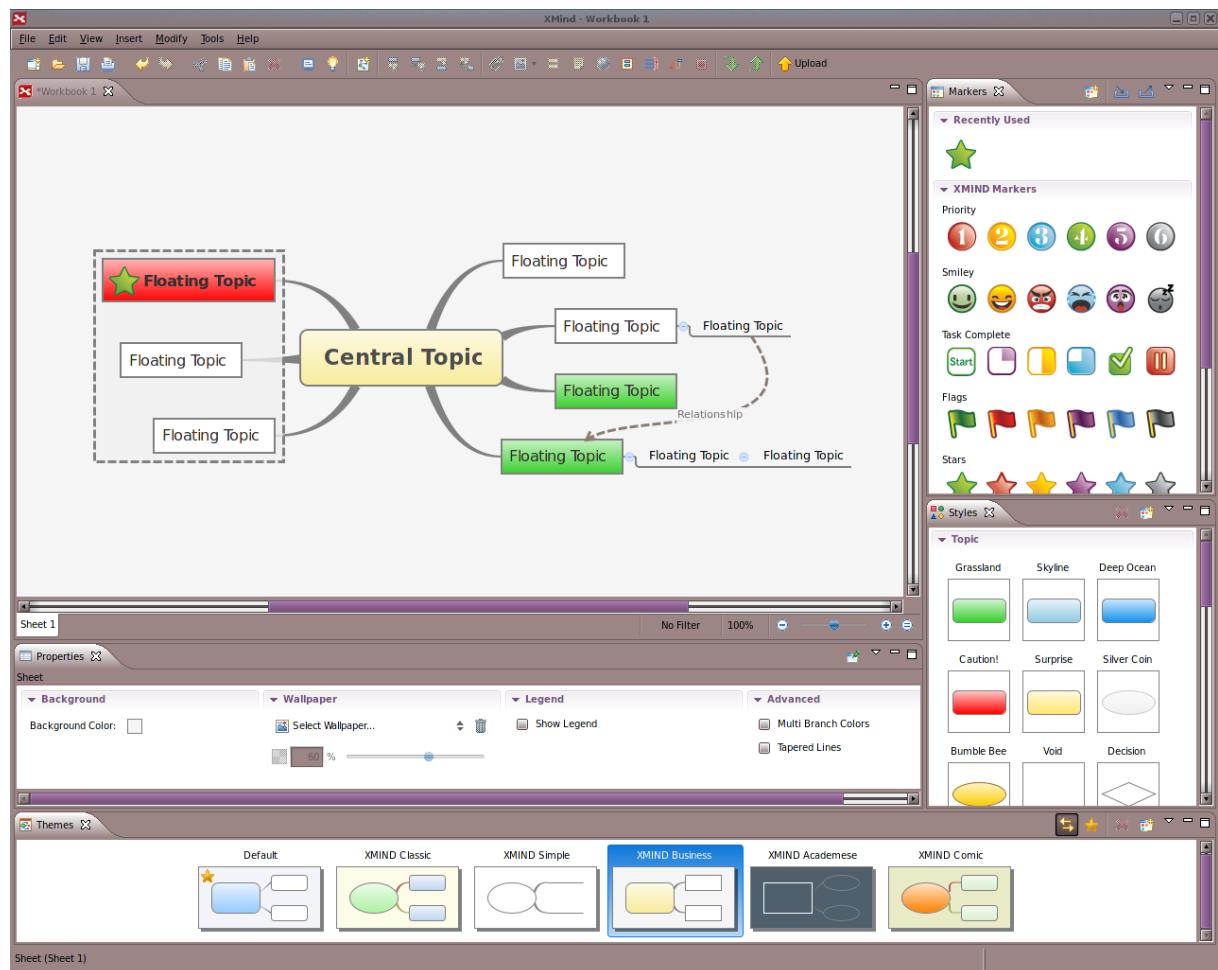
## 4 Przegląd aplikacji do tworzenia map pamięci

Na rynku komputerowym istnieje parę programów do tworzenia map pamięci. Najczęściej są one prostym edytorem graficznym grafiki wektorowej który pozwala na łączenie ze sobą informacji w sposób zgodny z zasadami tworzenia map pamięci. Podstawowe funkcje obejmują:

- tworzenie węzłów według struktury drzewa
- kolorowanie węzłów
- dodawanie ikonek do węzłów
- drukowanie i import do pliku graficznego

### 4.1 XMind

Xmind jest programem Open Source stworzonym za pomocą języka Java oraz Eclipse Rich Client Platform i Eclipse Graphical Editing Framework. Użycie tych bibliotek pozwoliło na stworzenie interfejsu który jest przystępny użytkownikom. Aplikacja jest rozwijana od 2007 roku, a od końca 2008 roku jest dostępna na licencji LGPL. Przykładowy zrzut ekranu został przedstawiony na rys 4.1.



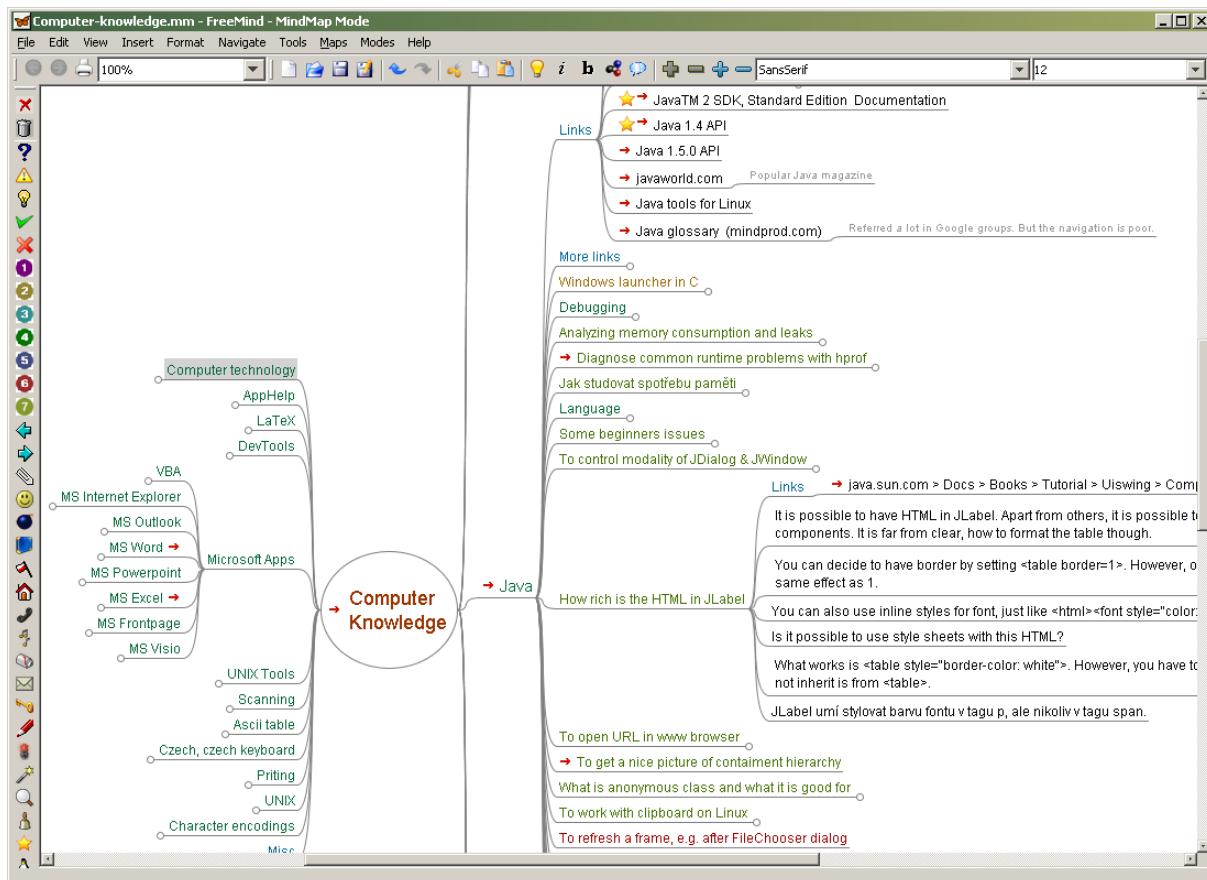
Rys 4.1 Wygląd interfejsu użytkownika programu XMind

Jego funkcjonalność pozwala na:

- tworzenie map o różnej strukturze (diagramy drzewa, Ishkawy , wykresy organizacyjne czy arkusze kalkulacyjne)
- tworzenie węzłów o dowolnym poziomie zagnieżdżenia
- użycie zdefiniowanych stylów kolorowania węzłów mapy pamięci
- dodawanie notatek w dowolnym miejscu struktury mapy
- zagnieżdżanie map wewnętrz innych (w ramach jednego dokumentu)
- publikowanie map pamięci w na stronach internetowych

## 4.2 FreeMind

FreeMind jest aplikacją typu open source wydaną na licencji GPL. Została stworzona za pomocą języka programowania Java i zestawu narzędzi Swing do tworzenia interfejsu użytkownika. Jest rozwijana od roku 2000. Przykładowy zrzut ekranu został przedstawiony na rys 4.2.



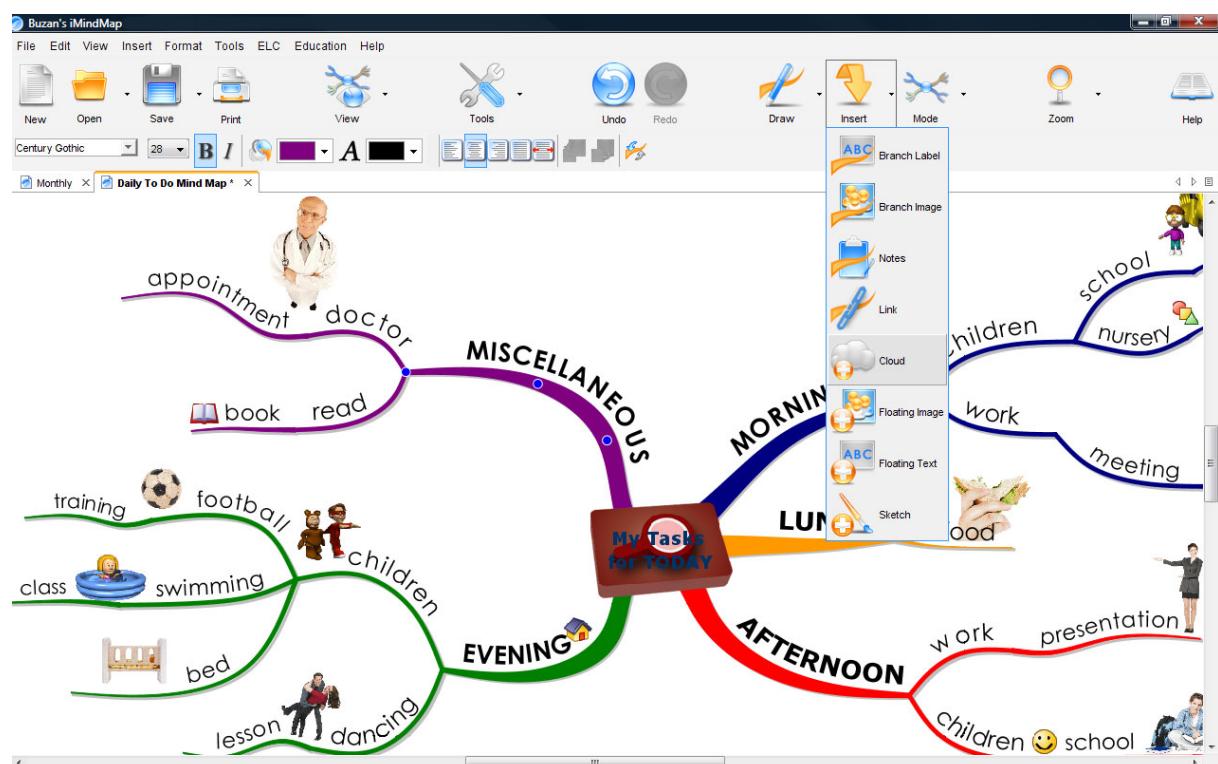
Rys 4.2 Wygląd interfejsu użytkownika programu FreeMind

Jej funkcjonalność pozwala na:

- zapisywanie dokumentu w wielu formatach, uwzględniając XML, HTML, OpenDocument czy PDF
- prezentacja na stronach internetowych w technologii Flash i Java
- wyszukiwanie wewnątrz mapy pamięci
- tworzenie hiperłączy w danych węzłach
- rozszerzenia do wielu systemów CMS (ang Content Management System)

### 4.3 Buzan's iMindMap

Buzan's iMindMap jest komercyjnym oprogramowaniem stworzonym przez firmę ThinkBuzan Ltd. Jej funkcjonalność wykracza poza zwykłe tworzenie mapy pamięci i umożliwia ponadto zarządzanie projektami i planowanie. Wyróżnia go wygodny interfejs użytkownika i bogate graficznie mapy pamięci. Rysunek 4.3 prezentuje wygląd aplikacji.



Rys 4.3 Wygląd interfejsu użytkownika programu iMindMap

## 5 Specyfikacja wymagań funkcjonalnych programu

W celu przedstawienia wymagań funkcjonalnych została stworzona lista wszystkich głównych funkcji aplikacji i każda z nich opisana. Są to wymagania ogólne, definiujące szkic aplikacji.

### 5.1 Tworzenie map pamięci

Tworzenie map pamięci powinno umożliwiać:

- Tworzenie węzłów
- Łączenie węzłów w sposób dowolny ze sobą
- Zmianę koloru i kształtu węzłów
- Automatyczne rozmieszczanie pozycji węzłów uwzględniając połączenia pomiędzy nimi

### 5.2 Prezentacja map pamięci

Prezentacja mapy pamięci ma być płynne i animowane. Powinna być również możliwość drukowania mapy pamięci oraz zapisywanie jej w formie pliku graficznego. Powinna być możliwość przesuwania widoku mapy za pomocą myszki oraz zmianę powiększenia prezentacji.

### 5.3 Serializacja

Zapisywanie i odczytywanie danej mapy pamięci powinno być przeźroczyste dla użytkownika. Powinno dokładnie odzwierciedlać stan danej mapy pamięci.

### 5.4 Interakcja z przeglądarką internetową

W ramach szybkiego tworzenia map pamięci, przeglądarka internetowa powinna umożliwić kopiowanie w wygodny sposób dowolnego tekstu ze strony internetowej do przeglądarki i szybkie tworzenie elementów bazowanych na tym tekście.

## 6 Specyfikacja wymagań pozafunkcjonalnych programu

W tym rozdziale zostaną omówione wymagania pozafunkcjonalne, spełniane przez stworzoną aplikację, w oparciu o standard ISO/IEC 9126-1:2001 [ISO]. Standard ten jest używany do oceny jakości oprogramowania. Odniesiono się zatem do niego, opisując stosowne założenia spełniające przez aplikację.

### 6.1 Wydajność

1. Wykorzystanie zasobów (ang. *resource utilisation*) - rodzaj i liczba wykorzystanych zasobów, np. pamięć operacyjna
  - Aplikacja nie powinna wykorzystywać więcej niż 30 MB pamięci operacyjnej.
2. Wydajność (ang. *efficiency compliance*) - szybkość odpowiedzi aplikacji na akcje użytkownika
  - Aplikacja powinna natychmiastowo reagować na interakcję użytkownika

### 6.2 Użyteczność

1. Łatwość zrozumienia (ang. *understability*) - łatwość zrozumienia czym jest dany produkt i jak może być użyty.
  - Aplikacja powinna posiadać pełną dokumentację funkcjonalności.
2. Łatwość operowania (ang. *operability*) - zdefiniowana jako łatwość użytkowania i sterowania oprogramowaniem.
  - Aplikacja powinna posiadać interfejs użytkownika.
3. Atrakcyjność (ang. *attractiveness*) - wygląd produktu
  - Komunikacja z aplikacją powinna odbywać się za pomocą interfejsu graficznego przystępnego dla użytkownika.

### 6.3 Przenośność

1. Łatwość adaptacji (ang. *adaptability*) - pracochłonność potrzebna do dostosowania aplikacji by działała na innych systemach operacyjnych
  - Aplikacja zostanie napisany w języku Python, w celu zapewnienia przenośności pomiędzy systemami operacyjnymi
  - Aplikacja powinna umożliwiać uruchomienie na platformach systemowych Windows oraz Linux
2. Łatwość insatalacji (ang. *installability*) - nakład pracy potrzebny do zainstalowania i konfiguracji aplikacji

- Aplikacja dostarcza dokumentację zainstalowania bibliotek potrzebnych do instalacji na danej platformie
- Aplikacja jest zaprojektowana w sposób przenośny (ang. *portable*), nie wymaga zapisania jakichkolwiek informacji w rejestrze czy ustawieniach systemu operacyjnego

## 6.4 Niezawodność

1. Odporność na błędy (ang. *failure recoverability*) - zdolność do działania nawet po wystąpieniu błędu w aplikacji
  - Błąd w czasie uruchomienia nie może powodować awarii całej aplikacji
  - Aplikacja nie powinna narażać systemu operacyjnego na niestabilność działania

## 7 Wstępna konstrukcja programu

Program powinien być skonstruowany w taki sposób, by oddzielić od siebie poszczególne elementy. Zapewni to łatwość rozszerzania aplikacji i naprawy błędów. Przewidziano następujące elementy aplikacji:

- Interfejs użytkownika
- Prezentacja map pamięci
- Manipulacja każdym elementem
- Interakcja z użytkownikiem
- Serializacja
- Interakcja z przeglądarką internetową

Interfejs dodawania mapy pamięci powinien działać jak najwydajniej, z uwagi na możliwość uruchomienia go na różnych platformach o różnej mocy obliczeniowej. Większość produktów programistycznych jest dostępnych w formie binarnej, i mimo dużej możliwości dostosowywania ich działania, czasami istnieje potrzeba głębszej modyfikacji, co nie zawsze jest możliwe. Dzięki społeczności OpenSource wiele rozwiązań posiada swoje kompilacje na różne platformy sprzętowe i systemowe, co w tym przypadku ma duże znaczenie. Przejawem tego podejścia jest użycie następujących technologii:

- Python - język programowania z otwartym kodem źródłowym
- QT - otwarty zbiór bibliotek programistycznych
- PyQT - nakładka biblioteki QT na język Python
- XUL, Javascript i XPCOM - zbór technologii wykorzystywany do tworzenia rozszerzeń do produktów Mozilli

## 8 Wykorzystane technologie i narzędzia

Głównym założeniem aplikacji jest możliwość uruchomienia jej na wielu platformach, bez potrzeby dostosowywania jej do każdej z nich. Ma być obsługiwana za pomocą klawiatury, myszki i ekranu dotykowego. Program ma działać szybko na komputerach o słabej mocy obliczeniowej, nie zamując dużo miejsca w pamięci operacyjnej.

### 8.1 Python

Python jest interpretowanym, interaktywnym, zorientowanym obiektowo językiem programowania. Jego składnia jest bardzo charakterystyczna ze względu na umieszczanie wcięć w kodzie zamiast nawiasów. Jedną z charakterystycznych cech tego języka programowania jest tworzenie funkcji anonimowych jak również to, że wszystko jest obiektem. Istnieją setki gotowych modułów do Pythona która umożliwiają na użycie bibliotek napisanych w innych językach programowania [PYQT].

### 8.2 Qt

QT jest zbiorem bibliotek stworzonych przez firmę Trolltech, następnie przejętą przez firmę Nokia. Obejmuje kilkaset klas odpowiedzialnych za projektowanie interfejsu użytkownika, programowanie wielowątkowe, współdzielenie pamięci, komunikację z bazami danych, grafikę OpenGL, dostęp do multimediów, obsługę XML, komunikację międzyprocesową, sieciową, grafikę wektorową i innych. Programy które używają tych bibliotek można uruchomić na następujących platformach: Linux, Windows, Mac OS X, Embedded Linux, Windows CE, Symbian i Maemo. Są też dostępne porty na inne platformy, jak iPhone, Android, webOS, openSolaris czy OS/2. Bibliotek Qt można używać za pomocą języka C++, jednak dzięki zastosowaniu MOC (ang. *Meta Object Compiler*) jest możliwość pisania swoich aplikacji w innych językach programowania, takich jak Java, C#, Lisp, Pascal, Python czy Ruby [PYQT].

### 8.3 PyQt

PyQt jest nakładką stworzoną by używać bibliotek QT wraz z językiem programowania Python. Została stworzona poprzez narzędzie SIP, które automatycznie generuje moduły Pythona z plików nagłówkowych bibliotek C++. To narzędzie zostało stworzone na potrzeby stworzenia tej nakładki, przez firmę Riverbank Computing [PYQT].

### 8.4 Javascript, XUL i XPCOM

Javascript jest interpretowanym, obiektowym językiem programowania wykorzystywanym głównie na stronach internetowych. Stosuje się go głównie do zapewnienia interaktywności stron internetowych po stronie klienta

XUL (ang. *XML-based User-interface Language*) jest językiem XML przeznaczonym do definiowania interfejsów użytkownika. Jest używany głównie przez produkty fundacji Mozilli typu Firefox, Flock czy Thunderbird.

XPCOM (ang. *Cross Platform Component Object Model*) jest przenośną platformą obiektów COM, podobną do Microsoft COM. Posiada dowiązań do wielu języków programowania. Interfejsy w XPCOM są definiowane jako dialekty IDL (ang. *Interface Definition Language*) zwane XPIIDL. XPCOM dostarcza zestaw componentów i klas pozwalających na zarządzanie plikami i pamięcią, wątki, struktury danych i inne.

Zestaw tych trzech technologii pozwala na tworzenie rozszerzeń do aplikacji Firefox i Thunderbird.

## 9 Architektura programu

Aplikacja została napisana w języku programowania Python w oparciu o biblioteki QT.

### 9.1 Zarządzanie mapami pamięci

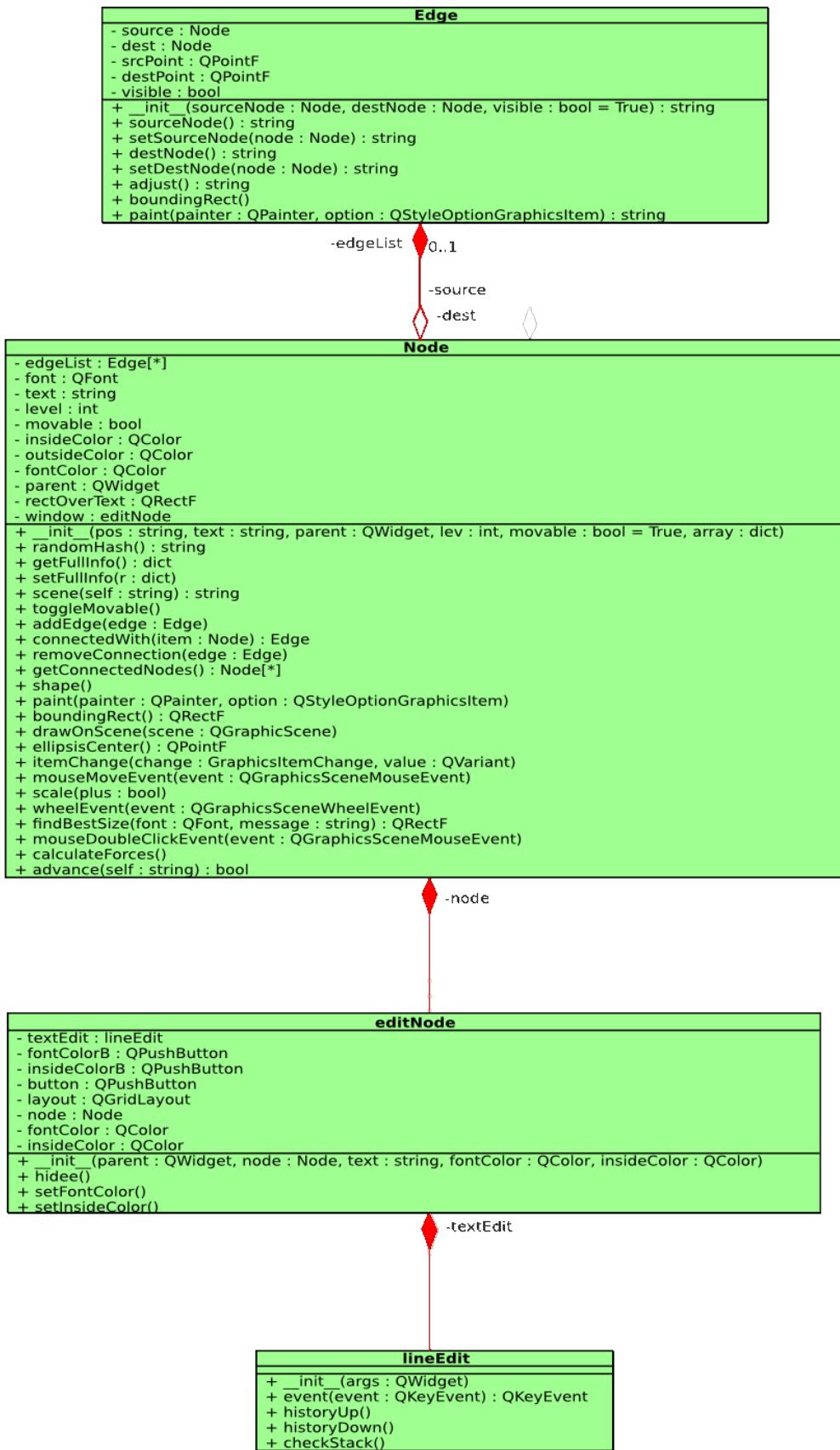
Głównym systemem zarządzającym obiektami na mapie pamięci oraz ich wyświetlaniem jest *Graphics View Framework* który jest zawarty w bibliotekach QT. Zawiera on 3 podstawowe elementy:

- QGraphicsScene - scena w której przechowywane są obiekty wraz z ich położeniem
- QGraphicsView - widok danej sceny
- QGraphicsItem - obiekt sceny

Scena jest kontenerem na obiekty, zapewniając szybki dostęp do nich. Jej zaletą jest zarządzanie stanem obiektów, detekcją kolizji i propagacją zdarzeń. QGraphicsScene używa algorytmu *Binary Space Positioning* do efektywnego zarządzania widzialnością danych obiektów. System propagacji zdarzeń wysyła zdarzenia do obiektów do których się dane zdarzenie odnosi. Jeżeli scena otrzyma zdarzenie kliknięcia myszą w danym miejscu, przekazuje to zdarzenie do obiektu który znajduje się na danej pozycji.

Widok dostarcza kontrolkę która wizualizuje zawartość sceny. Możliwe jest połączenie wielu widoków do jednej sceny by zapewnić wiele rzutni na dany zbiór obiektów. Widok otrzymuje bezpośrednio zdarzenia klawiatury i myszki i tłumaczy je na zdarzenia sceny przetwarzając koordynaty widoku na koordynaty sceny.

Obiekt sceny jest bazową klasą dla każdego graficznego elementu sceny. Dostępne są również predefiniowane klasy dziedziczące po QGraphicsItem, definiując style rysowania elipsy, tekstu czy prostokąta [PYQT].



Rys 9.1 Schemat UML elementów sceny

W aplikacji zostały wykorzystane wszystkie trzy klasy do stworzenia mechanizmu map pamięci. Z uwagi na zadowalającą funkcjonalność klasy QGraphicsScene nie było potrzeby jej rozszerzać, w przypadku pozostałych klas zaistniała potrzeba przeciążenia dużej części metod.

Klasa GraphicsView dziedziczy po klasie QGraphicsView i opisuje zarządzanie interakcją z użytkownikiem. Zaimplementowane są tam wszystkie funkcje związane ze zdarzeniami myszy oraz klawiatury, które odpowiadają za zaznaczanie, łączenie i rozłączanie obiektów, przesuwanie i zmienianie powiększenia widoku mapy oraz animacje obiektów. Schemat UML prezentuje rysunek 9.1.

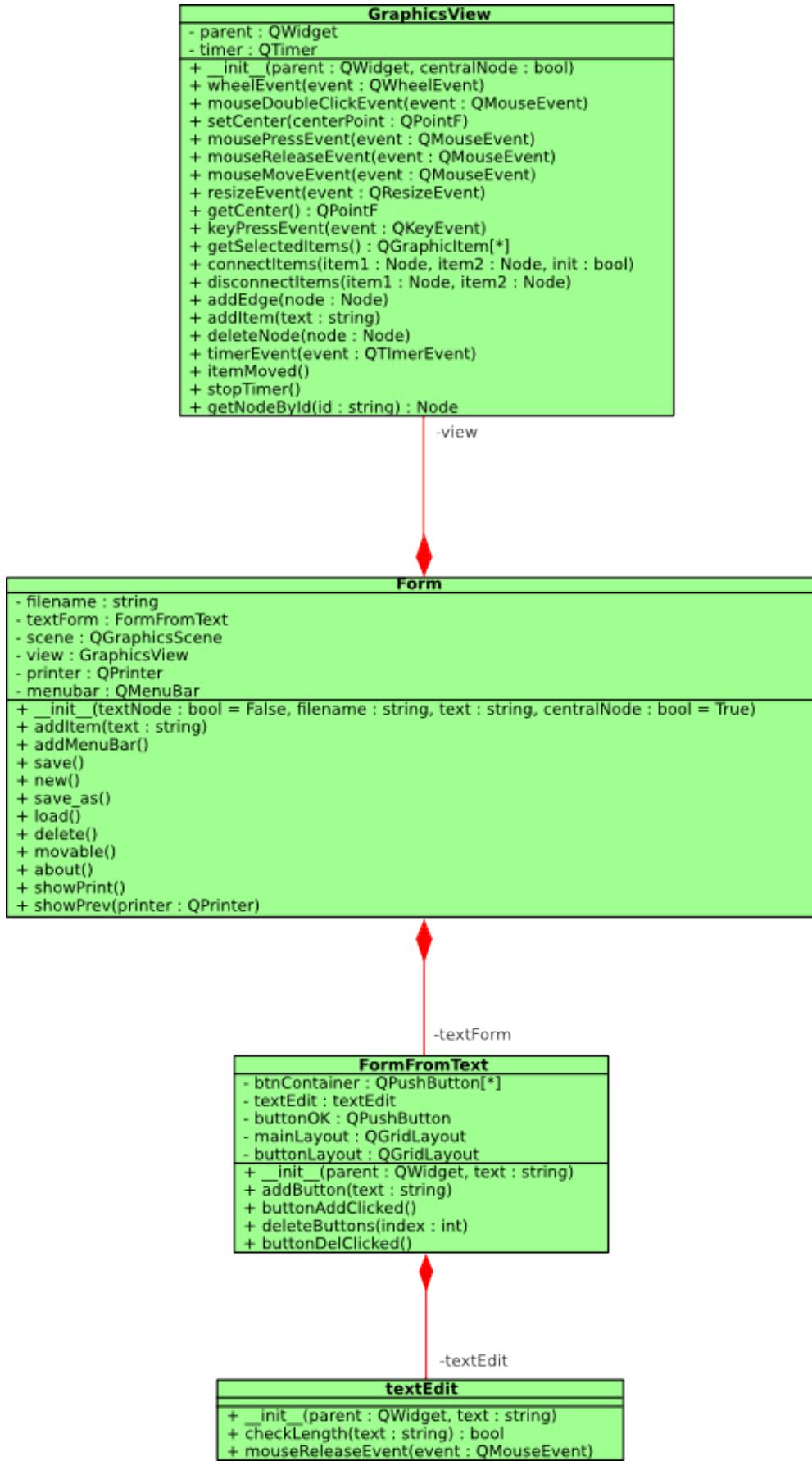
Łączenie obiektów odbywa się na podstawie wykrywania kolizji. W metodzie mouseMoveEvent, przy przesuwaniu obiektu sprawdzane jest czy obiekty nie nachodzą na siebie i są łączone. Tworzy się nowy obiekt klasy Edge, który zostaje dodany do sceny a w obu łączonych obiektach klasy Node, do pola edgeList dodawana jest referencja na obiekt Edge.

Animacje w mapie pamięci odbywają się przy pomocy zegara QTimer zarządzanego przez klasę GraphicsView. W momencie gdy dany obiekt klasy Node zmienił swoją pozycję, zegar zostaje uruchomiony. Przy każdym impulsie zegara obliczane są nowe pozycje każdego elementu na podstawie odległości i połączeń między nimi, następnie pozycje są aktualizowane uwzględniając połączenia między nimi (obiekty klasy Edge). W momencie gdy po obliczeniu sił żaden z elementów nie zmienia swojego położenia, zegar zostaje zatrzymany by oszczędzić czas procesora.

Rysowanie i geometria poszczególnych elementów jest zaimplementowane w klasie Node i Edge. Za geometrię odpowiada metoda shape, która przekazuje do sceny informację jaki obszar zajmuje dany obiekt. Metoda paint rysuje element, który składa się z elipsy wypełnionej kolorem oraz tekstu. W momencie gdy funkcja paint jest uruchamiana, każde połączenie (Edge) pomiędzy obiektami zostaje również przerysowane, by ich linie płynnie się aktualizowały przy zmianie pozycji węzłów.

## 9.2 System sygnałów i slotów

Biblioteka QT umożliwia na bardzo proste zarządzanie zdarzeniami na obiektach. Posiada ona system sygnałów i slotów, który implementuje wzorzec projektowy obserwatora (ang. *Observer pattern*). Obiekty mogą emitować sygnały (również z przekazywanymi wartościami) i inne obiekty mogą reagować na te sygnały podłączające je do slotów. Podejście takie jest bardzo elastyczne, gdyż do jednego slotu można podłączyć wiele sygnałów jak również sygnał może być obsługiwany przez wiele slotów. Każdy sygnał ma swoją nazwę po której jest identyfikowany i jest możliwe również przekazywanie wartości podczas emitowania sygnału. Niektóre obiekty zawierają predefiniowane sygnały i sloty, dzięki czemu można przekierować zdarzenia poszczególnych obiektów. Na przykład klasa QPushButton która implementuje przyciski na widżecie emittuje sygnał "clicked()" w momencie kliknięcia na niego [PYQT].



Rys 9.2 Schemat UML interfejsu użytkownika

System sygnałów i slotów jest wykorzystywany głównie w klasie *FormFromText* oraz *textEdit*. Obiekt *pierwszej klasy* nasłuchiwa sygnału o nazwie “addItemToList” który emitowany jest w momencie gdy zostaje zaznaczony tekst. Następnie tworzy w oknie dynamicznie dwa przyciski. Pierwszy zawiera tekst który został zaznaczony, drugi napis “X”. Od razu do obu przycisków są podłączane sygnały kliknięcia, które są obsługiwane przez dwie metody w klasie *FormFromText*. Jeżeli użytkownik naciśnie pierwszy przycisk, zostaje emitowany sygnał “addItem” który jest połączony pośrednio do klasy *GraphicsView*, która otrzymuje informacje jaki tekst został wybrany i tworzy nowy element na scenie razem z tym tekstem. Jednak gdy użytkownik naciśnie drugi przycisk, oba przyciski zostają usunięte z okna. Schemat UML prezentuje rysunek 9.2.

### 9.3 Interakcja z przeglądarką internetową

W celu zintegrowania przeglądarki internetowej Firefox z programem, koniecznym było stworzenie rozszerzenia. Jest to możliwe dzięki językowi Javascript, językowi opisu interfejsów użytkownika XUL oraz bibliotece XPCOM. Dodaje ono do menu kontekstowego nową pozycję z tekstem “Add to MindMap” za pomocą definicji w języku XUL. W momencie gdy użytkownik kliknął tą pozycję menu, uruchamia się funkcja w języku javascript, która sprawdza czy tekst został zaznaczony, następnie uruchamia aplikację do map pamięci z parametrami “–text” oraz ciągiem znaków zawierającym zaznaczony tekst. Aplikacja zostaje uruchomiona jako proces równorzędny, dzięki czemu przy zamknięciu przeglądarki internetowej możliwa jest dalsza praca z mapą pamięci. Jest to możliwe dzięki bibliotece XPCOM.

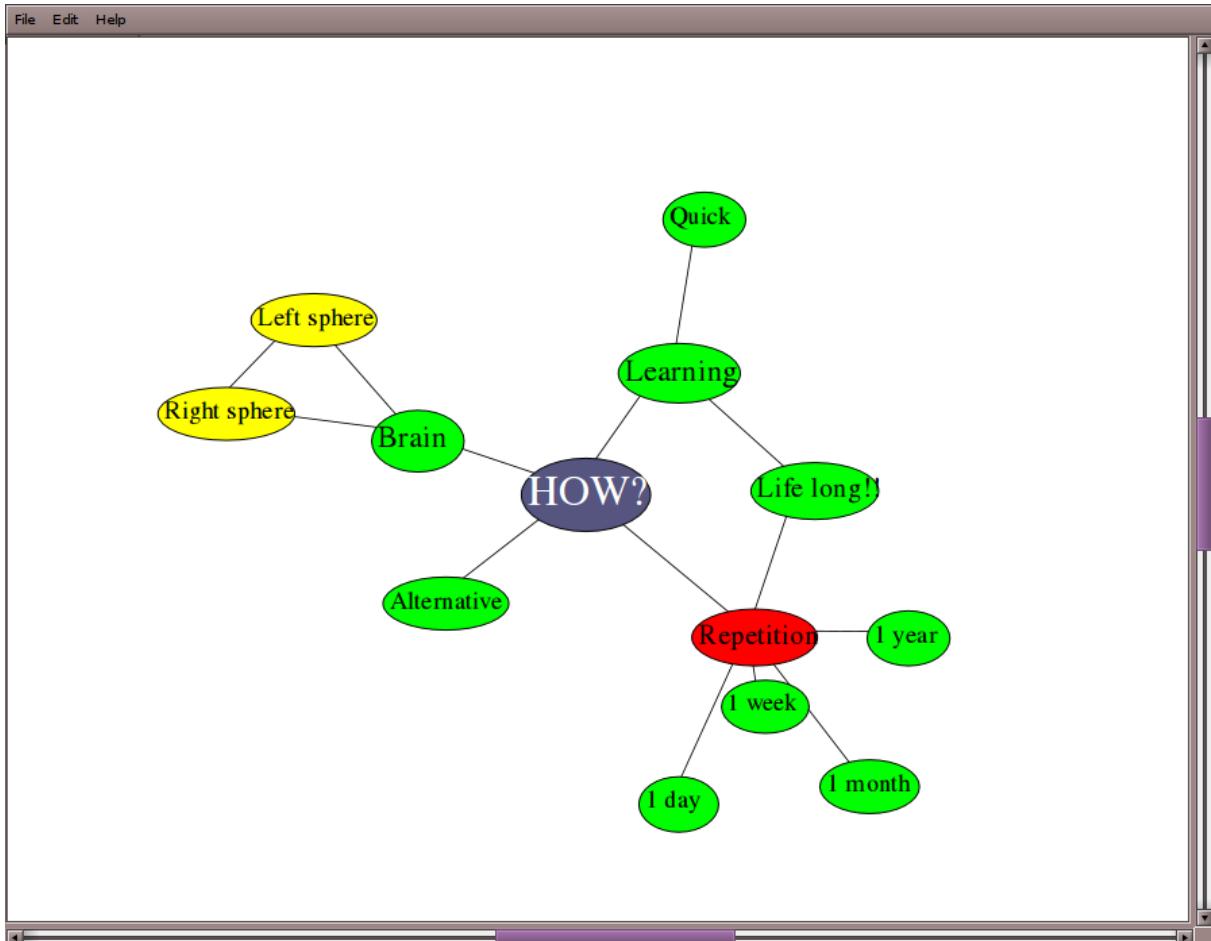
### 9.4 Konstrukcja kodu źródłowego aplikacji

W aplikacji poszczególna funkcjonalność została od siebie odseparowana, umożliwiając łatwiejsze zrozumienie kodu źródłowego i dalszą rozbudowę. Większość klas dziedziczy po generycznych klasach biblioteki QT. Poszczególne funkcjonalności zostały zaimplementowane w następujących klasach:

- Główne okno programu - klasa Form
- Serializacja - funkcje serialize i deserialize w pliku serialize.py
- animowanie, rysowanie poszczególnych elementów mapy pamięci- klasy Node i Edge
- zarządzanie interakcją z użytkownikiem - klasa GraphicsItem
- tworzenie mapy pamięci z tekstu - klasa FormFromText oraz textEdit
- interakcja z przeglądarką internetową - pliki pluginu do przeglądarki

## 10 Interfejs użytkownika

Z uwagi na lekką odmiennosć interfejsu użytkownika aplikacji od standardowych programów, w poniższym rozdziale zostanie omówiona interakcja programu z użytkownikiem.



Rys 10.1 Wygląd interfejsu użytkownika programu

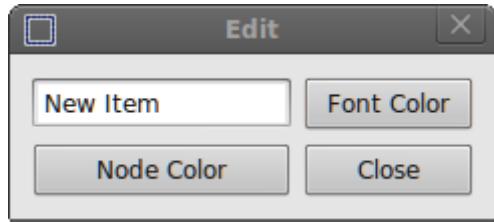
Aplikacja ma bardzo prosty interfejs z uwagi na dostarczenie użytkownikowi jak największej przestrzeni do oglądania mapy pamięci. Ma to na celu umożliwienie pełnego skupienia się przy tworzeniu mapy. Główną część okna zajmuje widok mapy pamięci, wraz z paskiem menu programu. Rysunek 10.1 ilustruje wygląd interfejsu użytkownika. Interakcja z programem odbywa się za pomocą klawiatury i myszki. Zaleca się korzystanie z myszy zewnętrznej lub touchpada skonfigurowanego do naciskania środkowego przycisku myszy i funkcji rolki.

### 10.1 Poruszanie się w widoku mapy pamięci

Przesuwanie widoku mapy pamięci odbywa się za pomocą naciśnięcia prawego lub środkowego przycisku i poruszania myszy. Zwiększanie lub zmniejszanie widoku wielkości mapy pamięci odbywa się za pomocą używania rolki na myszce. Należy zwrócić uwagę by nie był zaznaczony żaden element przy używaniu rolki.

## 10.2 Manipulacja węzłami

By dodać nowy węzeł do mapy pamięci, należy kliknąć dwukrotnie na pustym miejscu w widoku mapy pamięci. Stworzy to nowy element i otworzy okno edycji tekstu i stylu węzła. Widok okna edycji pokazuje rys 9.2.



Rys 10.2 Wygląd okna edycji węzła

Po zamknięciu okna edycji uaktualnia się wygląd węzła. Przywołać to okno można ponownie klikając dwukrotnie na obiekcie. By edytować wielkość węzła, należy zaznaczyć element i użyć rolki w myszce.

Zaznaczając dowolne elementy można je usunąć naciskając przycisk DELETE lub wybierając z górnego menu Edit->Delete.

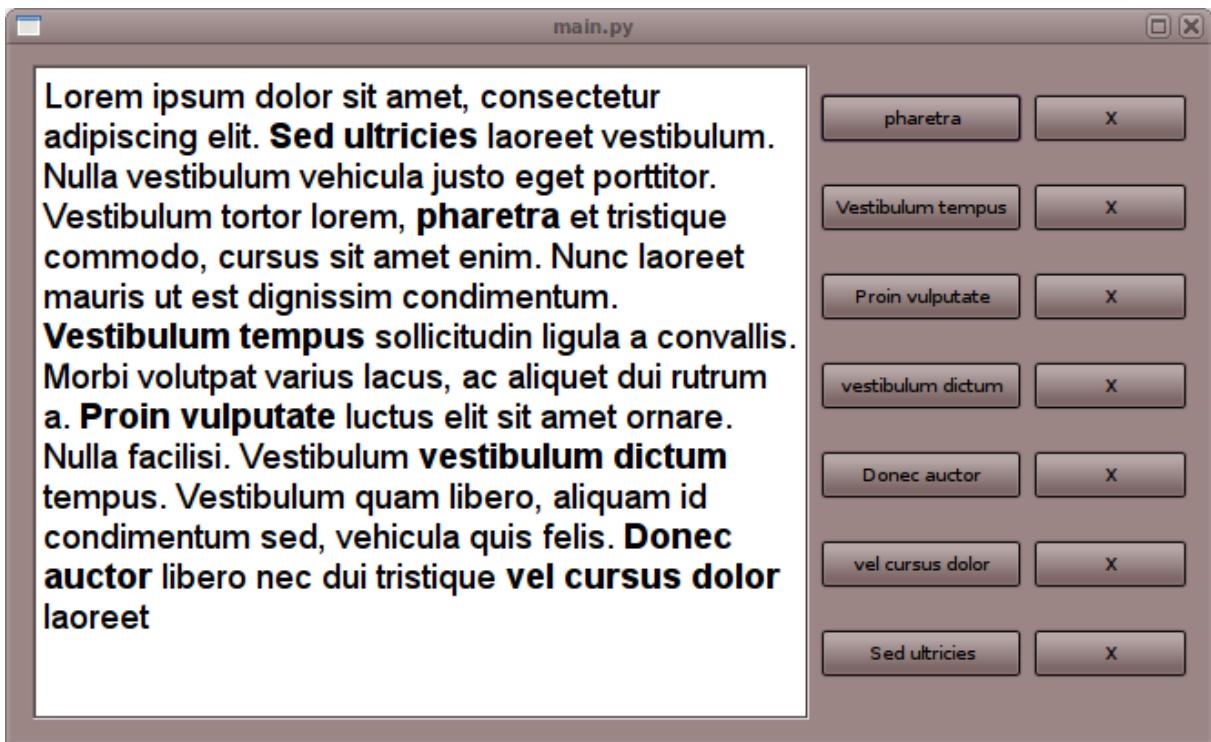
Łączenie węzłów odbywa się poprzez przeciągnięcie dowolnego elementu na inny.

Rozłączanie węzłów odbywa się w analogiczny sposób co łączenie, tylko należy przy tej operacji trzymać wcisnięty przycisk CTRL i najechać elementem na inny który chcemy rozłączyć.

Zaznaczając element i naciskając środkowy przycisk myszki można sprawić, by element się nie animował i nie przemieszczał. Umożliwia to dokładne pozycjonowanie elementów jeżeli automatyczne nie spełnia wymagań użytkownika.

## 10.3 Dodawanie węzłów z tekstu

Kolejnym elementem interfejsu jest tworzenie mapy pamięci z tekstu ze strony internetowej. Jeżeli zainstalowany został dodatek do przeglądarki Firefox (proces instalacji jest omówiony w rozdziale 12), można zaznaczyć w przeglądarce dowolny tekst, i z menu kontekstowego kliknąć pozycję "Add to Mindmap". Pojawi się wtedy okno tworzenia węzłów z tekstu. Interfejs tej funkcjonalności pokazuje Rys 9.3.



Rys 10.3 Wygląd okna dodawania węzłów z tekstu

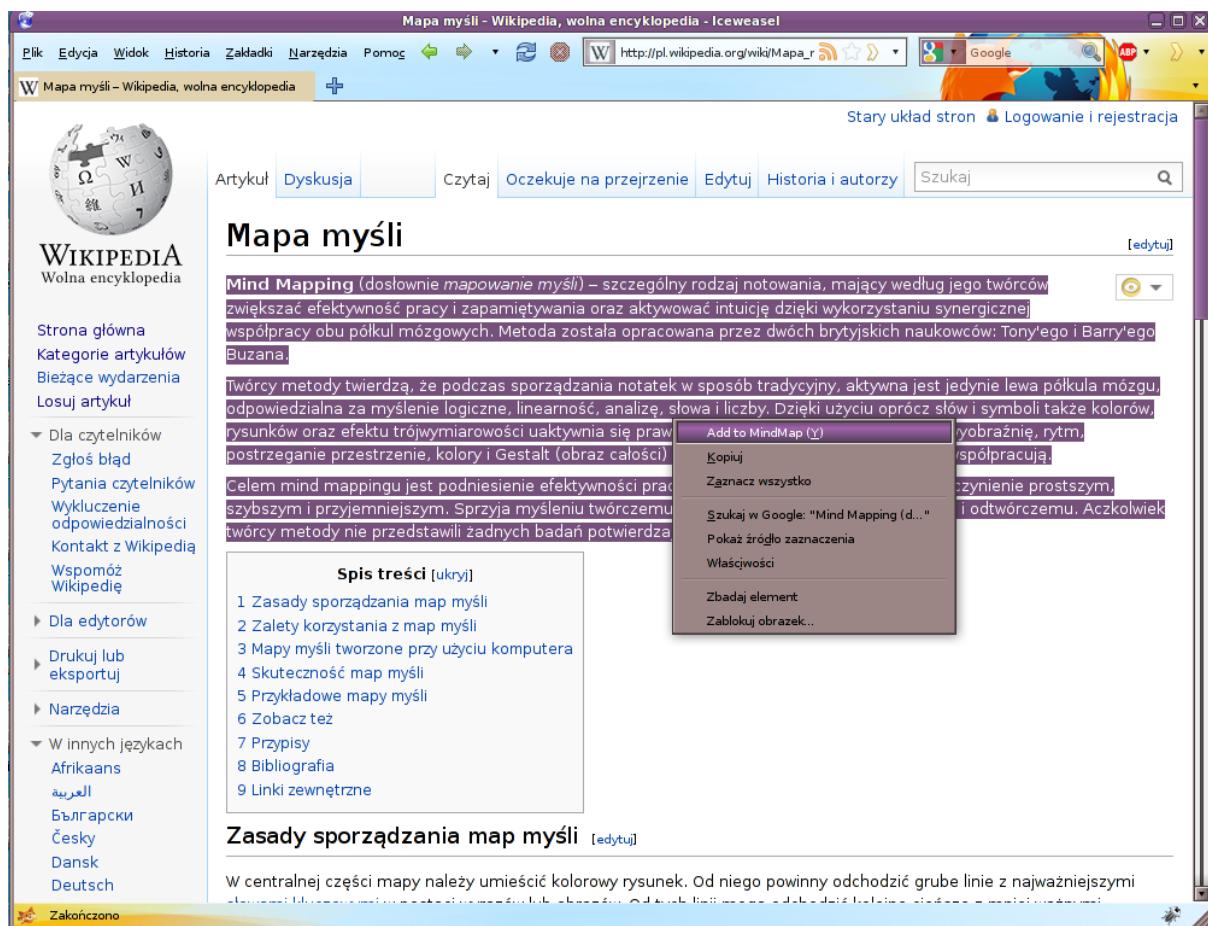
By przekształcić dany fragment tekstu na węzeł w mapie pamięci, należy za pomocą myszki zaznaczyć dany tekst. Po zaznaczeniu pojawią się dwa przyciski. Jeden (z zaznaczonym tekstem) służy do dodania go do mapy pamięci, drugi do usunięcia zaznaczenia tekstu. Po kliknięciu na przycisk z tekstem znikają oba przyciski i pojawia się węzeł na mapie pamięci. Po zdecydowaniu co zrobić z wszystkimi zaznaczeniami tekstu zostanie przycisk zamknięcia okienka.

# 11 Przykład użycia programu

By dobrze zrozumieć sposób działania aplikacji, należy przytoczyć przypadek użycia tworzenia kompletnej mapy pamięci. Zostało to przedstawione w tym rozdziale krok po kroku.

## 11.1 Wybór tekstu

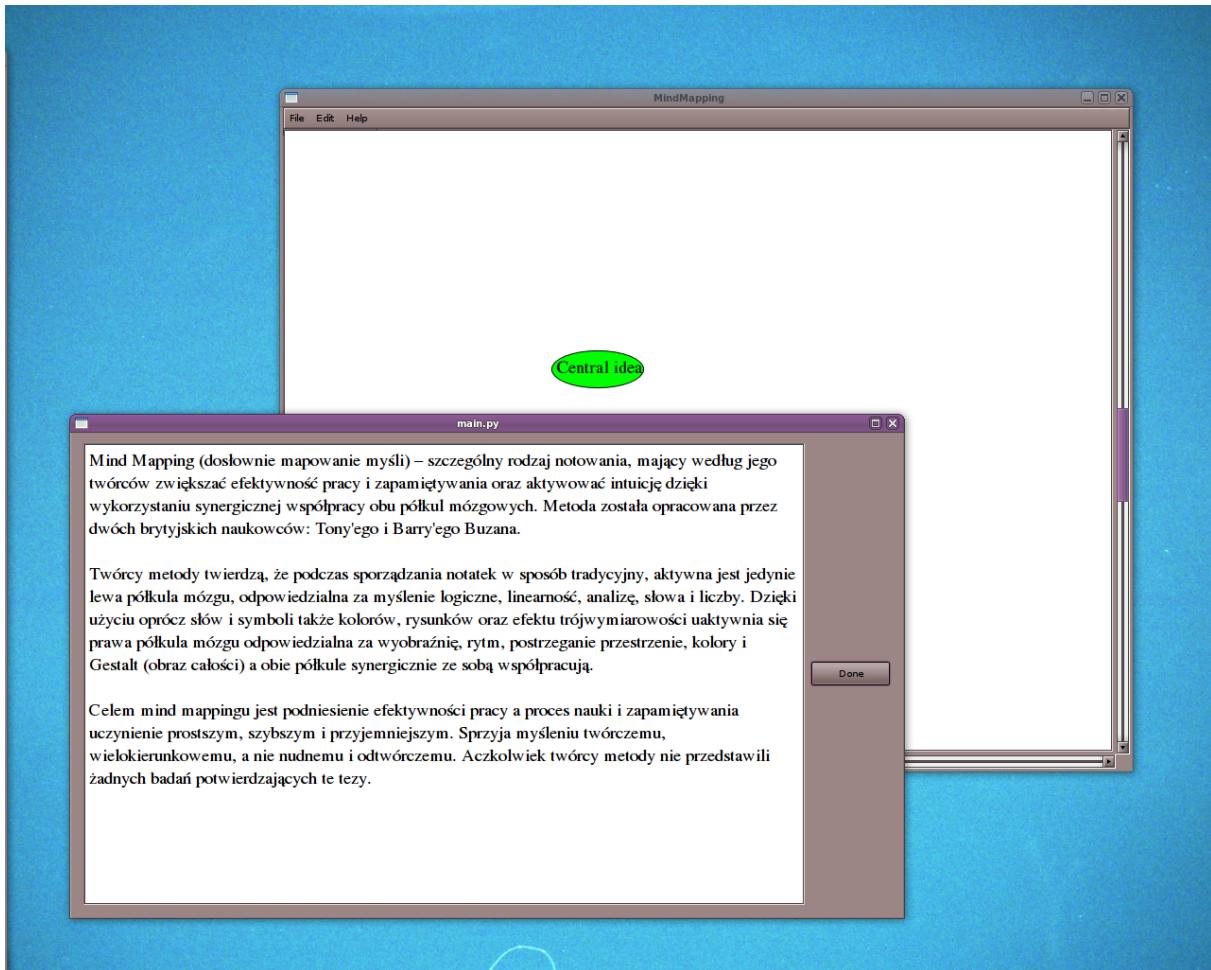
Najłatwiej tworzy się mapy pamięci na podstawie istniejącego tekstu. Ucząc się przy pomocy komputera, najczęstszym źródłem informacji są strony internetowe. Zakładamy, że użytkownik poprawnie zainstalował dodatek do przeglądarki internetowej. Pierwszym krokiem jest znalezienie strony internetowej z interesującą nas zawartością. Następnie zaznaczamy tekst z którego chcemy stworzyć mapę pamięci, klikamy prawym klawiszem nad zaznaczonym tekstem i wybieramy pozycję “Add to MindMap”. Powyższe czynności obrazuje rysunek 11.1.



Rys 11.1 Wygląd okna przeglądarki internetowej

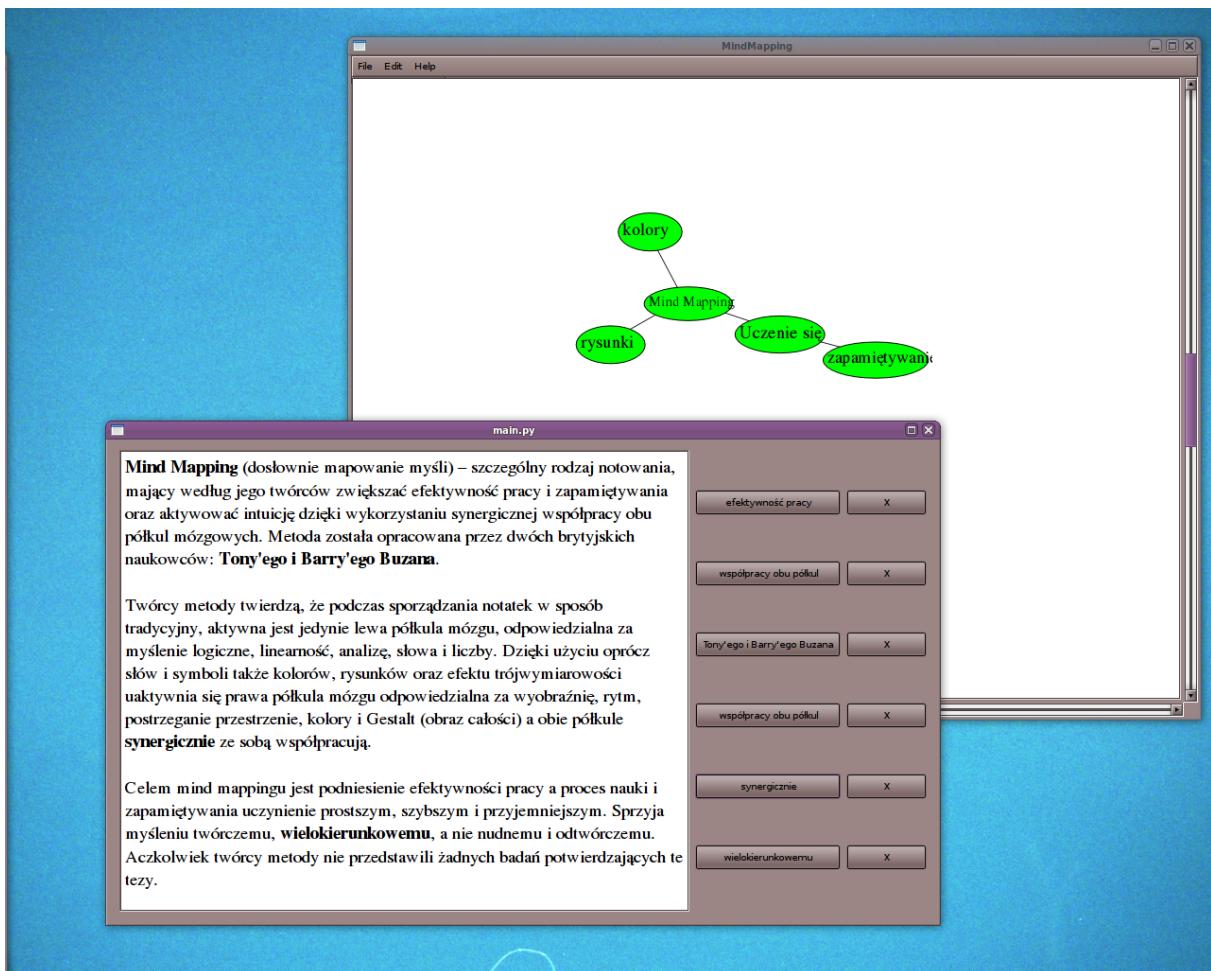
## 11.2 Wybór słów-kluczy

Następnym krokiem jest wybranie najważniejszych słów czy fraz które występują w tekście. Powinny być charakterystyczne i jednoznacznie kojarzyć się z tematem danej treści. Początkowy wygląd okna tworzenia mapy pamięci z tekstu prezentuje rysunek 11.2.



Rys 11.2 Wygląd interfejsu programu po dodaniu tekstu

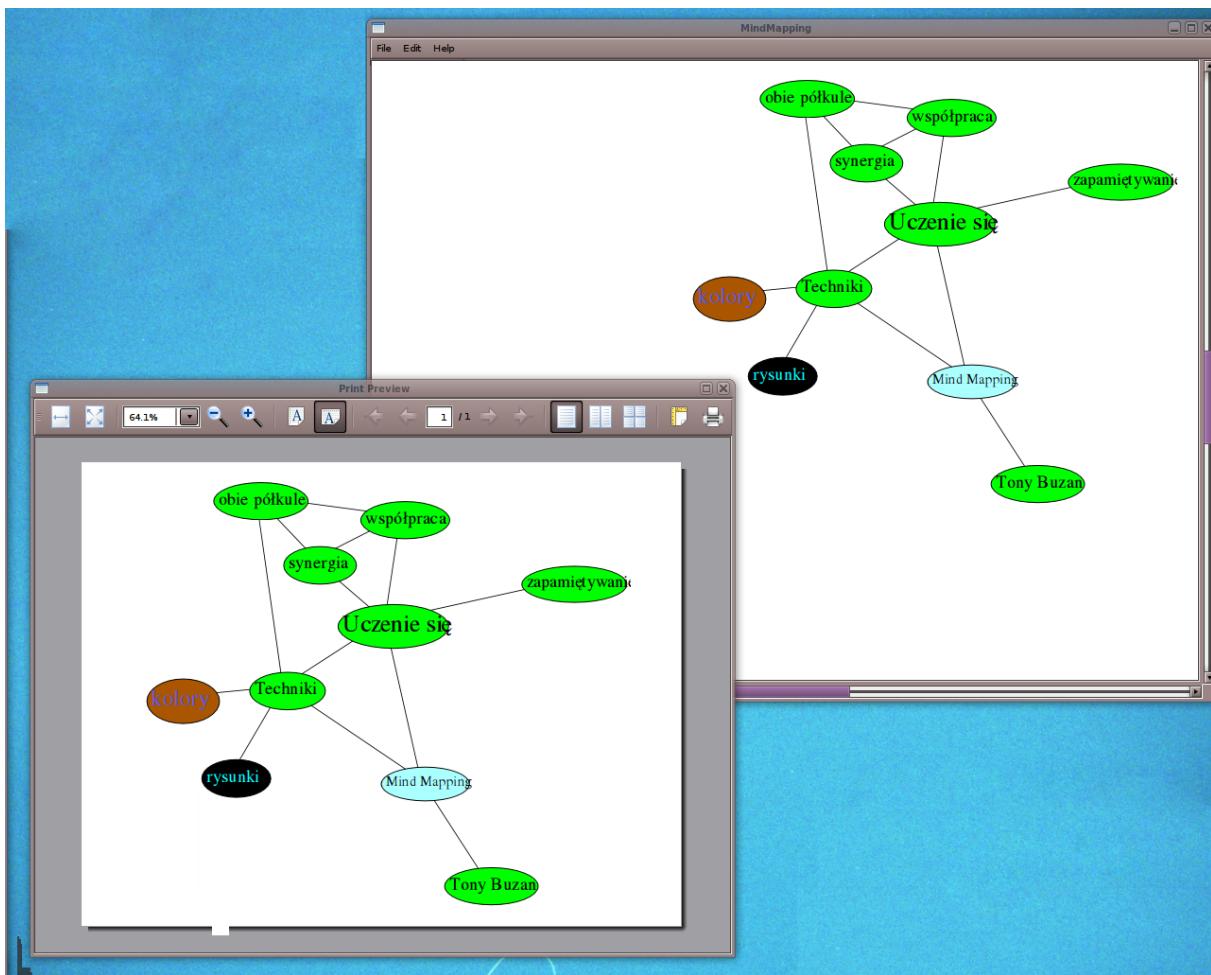
Wybieramy odpowiednie słowa z których stworzymy węzły. W tym celu za pomocą myszy zaznaczamy odpowiednie słowa lub frazy, które pojawiają się z prawej strony okna. By dodać dane zaznaczenie do mapy pamięci, klikamy na przycisk z tekstem. By usunąć zaznaczenie, klikamy na przycisk z napisem "X". Rysunek 11.2 prezentuje ten proces.



Rys 11.3 Wygląd interfejsu programu po dodaniu paru węzłów

### 11.3 Dostosowanie mapy pamięci

Po wybraniu wszystkich elementów, możemy zamknąć okno z tekstem. Ostatnim krokiem jest dostosowanie i rozbudowanie swojej mapy pamięci do własnych potrzeb. Używając technik manipulacji węzłami opisanyimi w poprzednim rozdziale układamy interesującą nas strukturę, kolory i wielkość mapy. Po zakończonej pracy możemy ją zapisać i wydrukować. Jest to przedstawione na rysunku 11.4.



Rys 11.4 Drukowanie mapy pamięci

## 11.4 Uczenie się z mapy pamięci

Po stworzeniu mapy pamięci, oglądając ją po dłuższym czasie bardzo szybko można sobie uzmysłować na podstawie jakich materiałów czy myślała była ona tworzona. Dzieje się tak dzięki relacjom pomiędzy węzłami. Ważnymi elementami na mapie są również wielkość i kolor danego elementu.

Jednak największą zaletą tworzenia mapy jest umiejętność osoby tworzącej powiązania procesu myślowego towarzyszącego tworzeniu z daną mapą. W zależności od czasu tworzenia mapy pamięci, możliwe jest przypomnienie sobie szczegółów podczas jej powstawania. Czym więcej czasu użytkownik poświęci na daną mapę, skupi się i przemyśli jej strukturę, tym większe są szanse na wyciągnięcie maksimum z niej.

## 12 Uruchamianie programu

Aplikacja posiada następujące wymagania systemowe:

- Interpreter języka Python w wersji 2.6 - 2.7
- Biblioteki QT w wersji 4.7 lub nowszej
- Nakładki PyQt w wersji 4.7 lub nowszej
- Przeglądarki internetowej Firefox w wersji 3.5 - 4.0 beta 10
- W przypadku systemu operacyjnego Windows, należy zainstalować pakiet *VC++ 2008 Redistributable Package*. Jest on dostępny na stronie firmy Microsoft

Aplikacja jest dostępna na płycie CD dołączonej do pracy inżynierskiej.

By uruchomić program na systemach opartych z grupy Linux, należy zainstalować interpreter python, biblioteki QT oraz PyQt. W systemach opartych o dystrybucję Debian (np. Ubuntu, Mint) wystarczy wydać polecenie z terminalu z dostępem administracyjnym:

```
apt-get install python python-qt4 libqt4-core libqt4-gui
```

Następnie uruchamiamy program za pomocą skryptu shellowego z flagą uruchomieniową *runner.sh* znajdującego się w folderze */linux*.

Do uruchomienia aplikacji pod systemami operacyjnymi z rodziny Microsoft Windows wystarczy odpakować archiwum *mindmapping-win32.zip* znajdujący się w folderze */windows* na dysku lokalnym i uruchomić plik wykonywalny *mindmapping.exe*.

### 12.1 Instalacja rozszerzenia do przeglądarki Firefox

Aplikacja posiada dwa osobne rozszerzenia, w zależności od systemu operacyjnego na którym zostaje uruchomiona. Znajdują się one odpowiednio w folderach */linux* i */windows*. By zainstalować rozszerzenie, należy uruchomić przeglądarkę Firefox i wpisać w pasku adresu ścieżkę do pliku z rozszerzeniem.

W systemie operacyjnym Linux, gdy aplikację mamy w folderze */home/user/mindmapping*, wpisujemy:

```
file:///home/user/mindmapping/mindmapping-linux.xpi
```

Następnie po zainstalowaniu rozszerzenia i uruchomieniu ponownym przeglądarki musimy skonfigurować ścieżkę z której będzie uruchamiania aplikacja. W tym celu uruchamiamy Narzędzia->Dodatki, i w preferencjach rozszerzenia podajemy bezwzględną ścieżkę do pliku wykonywalnego. W systemie Linux musimy podać plik *runner.sh*, natomiast w systemie Windows *mindmapping.exe*.

## 12.2 Uruchamiania aplikacji z argumentami

Domyślnym sposobem uruchomienia aplikacji jest uruchomienie go bez argumentów. Jest zalecany dla większości użytkowników. Tryb ten tworzy automatycznie nową, pustą mapę pamięci.

### 12.2.1 Tryb tworzenia mapy pamięci z tekstu

Ten tryb jest potrzebny przy integracji z przeglądarką internetową. Przy uruchomieniu należy dodać dwa argumenty:

np. *mindmapping.exe -text "Przykładowy tekst z którego chemię stworzyć mapę pamięci"*

Dzięki temu po uruchomieniu pojawią się dwa okna: główne okno z pustą mapą pamięci oraz dodatkowe z kreatorem węzłów z tekstu. Opis używania tego okna znajduje się w rozdziale 10.8. Wygląd aplikacji po uruchomieniu w tym trybie prezentuje rysunek 11.2.

### 12.2.2 Tryb Wczytania pliku przy uruchomieniu

Przy integracji programu z systemem operacyjnym lub innym oprogramowaniem potrzebne jest również automatyczne wczytywanie aplikacji z zapisaną mapą pamięci. W tym przypadku należy dodać dwa argumenty:

*mindmapping.exe -file /ściezka/do/pliku.mindqt*

Wszystkie zapisane pliki tego programu mają rozszerzenie \*.mindqt, jednak możliwe jest wczytanie dowolnego pliku. Ważne jest tylko, by wczytywany plik został wcześniej zapisany w tej aplikacji, w przeciwnym razie aplikacja pokaże błąd wczytywania pliku.

## 13 Podsumowanie pracy

W ramach projektu pomyślnie ukończono implementację aplikacji do tworzenia map pamięci spełniającą postawione wymagania. Podstawowym było stworzenie aplikacji, która umożliwi tworzenie dowolnych połączeń pomiędzy elementami na mapie. Kolejną rzeczą było stworzenie łatwego w operowaniu interfejsu użytkownika, pozwalającego na tworzenie i operowanie elementami na mapie. Następnym ważnym elementem aplikacji było umożliwienie użytkownikowi szybkiego stworzenia mapy pamięci z dowolnego tekstu na stronie internetowej. Ostatnią rzeczą było stworzenie aplikacji dostępnej na wiele platform. Jest to możliwe dzięki językowi programowania Python i zestawie bibliotek QT.

Aplikacja zostanie w przyszłości rozwijana by uzyskać w pełni funkcjonalny edytor map pamięci, mogący konkurować z istniejącymi produktami na rynku. Planowane jest również dostosowanie aplikacji na platformy mobilne.

## Bibliografia

- [BUZAN] Buzan Tony, *Mapy twoich myśli*, Ravi, Łódź 2003
- [PSYCHO] Nęcki Edward, Orzechowski Jarosław, Szymura Błażej, *Psychologia poznawcza*, PWN, Warszawa 2006
- [COGNI] Stilling Neil, Weisler Steven, Chase Christopher, *Cognitive Science an introduction*, MIT Press, London 2005
- [PYQT] Summerfield Mar, Rapid GUI programming with Python and QT, Prentice Hall, New York 2008
- [ISO] ISO/IEC 9126-1, Software engineering — Product quality — Part 1: Quality Model, June 2001