Prefuse – dynamiczna wizualizacja danych

Zimowisko Linuxowe 2011

Konrad Brodzik

Podstawowe pojęcia

- Graf
  - w uproszczeniu – zbiór wierzchołków, które mogą być połączone krawędziami, w taki sposób, że każda krawędź kończy się i zaczyna w którymś z wierzchołków […] Wierzchołki grafu zwykle są numerowane i czasem stanowią reprezentację jakichś obiektów, natomiast krawędzie mogą wówczas obrazować relacje między takimi obiektami. (Wikipedia)
- Drzewo
  - graf, w którym nie ma cykli (pętli)
- Minimalne drzewo rozpinające (MST)
  - łączy wszystkie wierzchołki grafu tak, aby suma długości krawędzi była jak najmniejsza

Podstawowe pojęcia

- Efekt ”łał”
  - efekt, powodowany przez system, gdzie dużo elementów daje się klikać

Po co bawić się grafami?

- na studiach każą
- wizualizacja sieci komputerowych i innych struktur pseudohierarchicznych
- czasem można stworzyć ciekawy rysunek

Sprytny Wiki Nawigator

- Narzędzie do graficznego przeglądania Wikipedii, z podziałem wyników wyszukiwania na kategorie wiedzy
- Zgrupowane wyniki wyszukiwania udostępniane poprzez Web Service

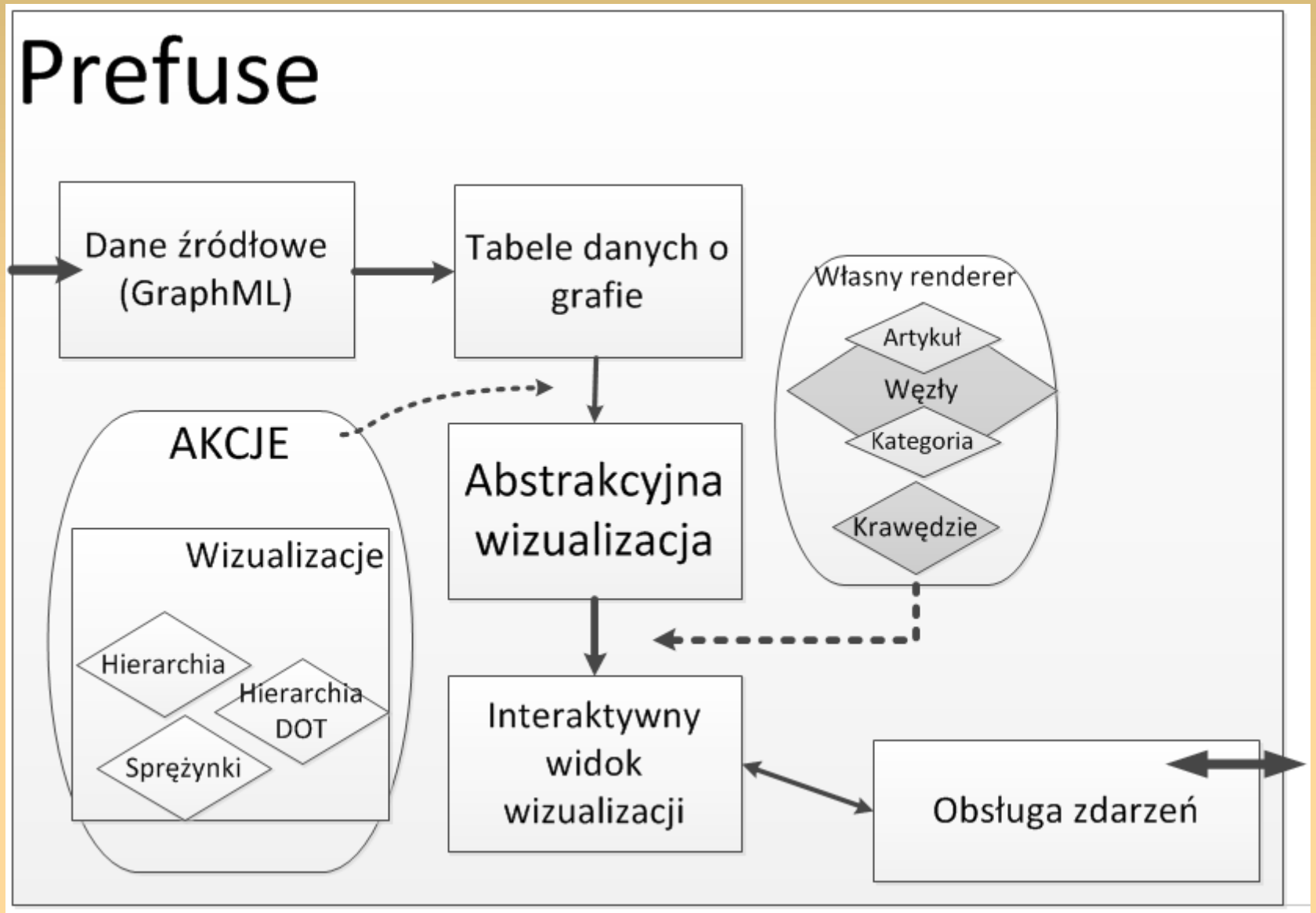
Różne rozwiązania do wizualizacji

- Gossamer (http://gossamer.eti.pg.gda.pl/)
  - Napisany we Flashu
  - powolny!
- Graphviz (http://www.graphviz.org/)
  - napisany w C
  - szybki, prosty i statyczny
- Open Graph Drawing Framework (http://www.ogdf.net/)
  - bardzo rozbudowana biblioteka w C++

Różne rozwiązania do wizualizacji

- Prefuse Visualization Toolkit (http://www.prefuse.org)
  - ostatnia wersja 2007.10.21
  - napisany w Javie, działa na JRE 1.6.0u23
- Flare (http://flare.prefuse.org/)
  - funkcjonalny odpowiednik Prefuse we Flashu

Prefuse - architektura



Dostęp do danych

- Wszystkie atrybuty węzła, zadeklarowane w relacji bazodanowej lub GraphMLu są automatycznie dostępne za pomocą słabo lub silnie typowanych getterów
- Elementy są indeksowane z użyciem wysokowydajnych struktur – trie i drzew czerwono-czarnych

Źródło danych - GraphML

- Węzły i krawędzie opisane tagami <node> i <edge>
- każdy element może posiadać dodatkowe atrybuty w tagach <data> - muszą być zdefiniowane na początku
- obsługiwany przez klasę GraphMLReader

Źródło danych - GraphML

Przykładowy plik:

```
<graphml>
   <graph edgedefault="undirected">
      <!-- definicja atrybutów -->
      <key attr.name="x" attr.type="double" for="node" id="x"/>
      <key attr.name="y" attr.type="double" for="node" id="y"/>
      <key attr.name="name" attr.type="string" for="node" id="name"/>
      <key attr.name="typ" attr.type="string" for="node" id="typ"/>
      <node id="Lama">
         <data key="name">Lama</data>
         <data key="typ">artykul</data>
         <data key="x">343.74</data>
         <data key="y">247.5</data>
      </node>
      <node id="Ssaki_Ameryki_Po%c5%82udniowej">
         <data key="name">Ssaki Ameryki Południowej</data>
         <data key="typ">kategoria</data>
         <data key="x">382.91999999999996</data>
         <data key="y">217.5</data>
      </node>
      <edge source="Ssaki_Ameryki_Po%c5%82udniowej" target="Lama"/>
      </graph>
</graphml>
```

Źródło danych - SQL

- Dowolny driver JDBC
- Pobieranie wybranych z użyciem Prefuse Expression Language elementów przez DatabaseDataSource bezpośrednio do tabeli

Prefuse Expression Language

- podobny do SQLa
- zapytania wstępnie kompilowane
- funkcje typowo grafowe (ISNODE, TREEDEPTH) i wizualizacyjne (VISIBLE)
- Przykład:
  pobierz wszystkie widoczne węzły na warstwie 0:

  Iterator currentLayer = vg.tuples(
  ExpressionParser.predicate("[warstwa] = 0 && VISIBLE()"));

Widok (prefuse.Display)

- Najważniejszy obiekt w wizualizacji
- Agreguje:
  - tablicę danych
  - wizualizację
  - akcje związane z wizualizacją
- Dodawanie danych:
  - visualisation.addGraph(graph)

Wizualizacja (prefuse.Visualisation)

- zawiera graficzne atrybuty elementów
  - położenie
  - widoczność
  - kolor, kształt
  - mapa sąsiadów

Tabela danych (prefuse.data.Table)

- zawiera właściwości elementów pobrane ze źródła danych, wraz z metodami canGet i get, typowanymi jako Object lub String/Boolean/Integer itp..

Akcje (prefuse.action)

- Uniwersalne komponenty, służące do przetwarzania danych lub rysowania
- Standardowe akcje:
  - rozplanowanie przestrzenne węzłów
  - kolorowanie wierzchołków
  - rysowanie etykiet
  - kolorowanie krawędzi
  - powiększanie i przesuwanie widoku

Akcje (prefuse.action)

```
ColorAction text = new ColorAction(treeNodes,
        VisualItem.TEXTCOLOR, ColorLib.gray(0));
// stworz akcje, ktora bedzie uruchamiana cyklicznie w nieskonczonosc
ActionList color = new ActionList(Activity.INFINITY);
color.add(text);
// dodaj akcję do kolejki
m_vis.putAction("color", color);
// koloruj. teraz.
m_vis.run(”color”);
// wylacz kolorowanie
m_vis.removeAction(”color”);
```

Akcje ułożenia

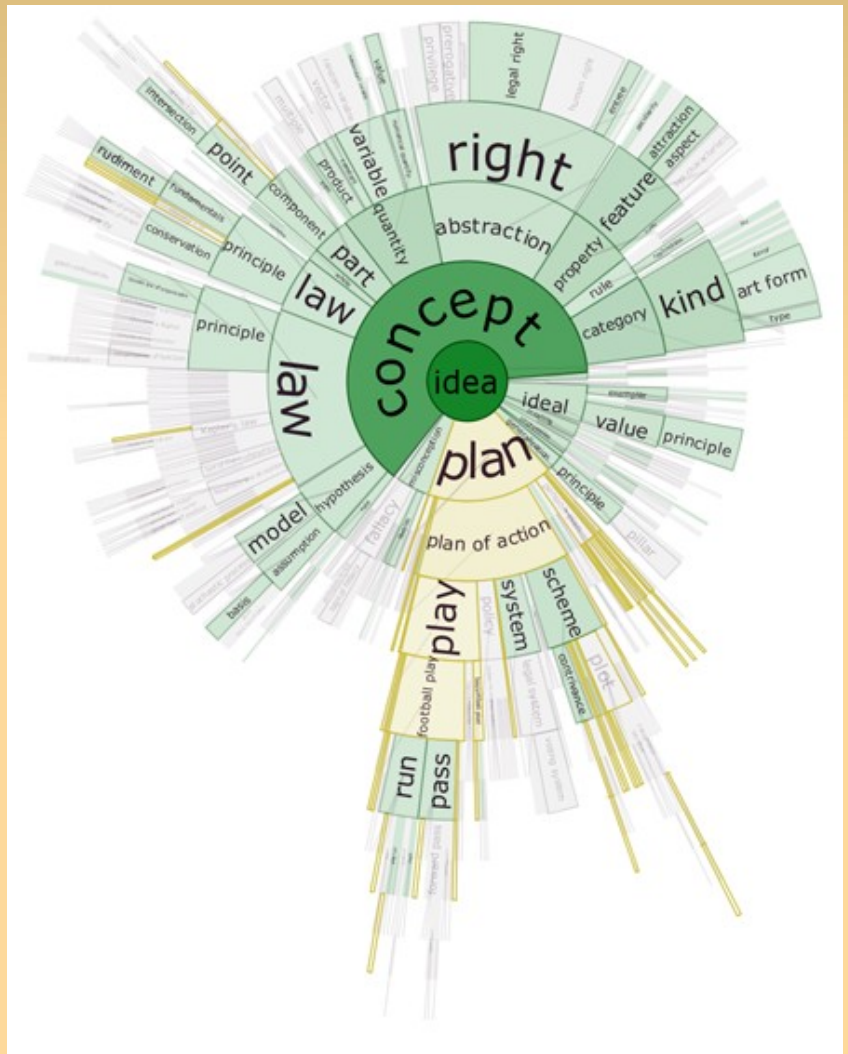
- Rozplanowują węzły według określonego porządku
- Pułapki:
  - jeśli rozplanowanie działa w tle, może nam uniemożliwiać drag & drop
  - niektóre ułożenia wymagają drzewa – jeśli podany graf nim nie jest, rysowane jest jego MST
  - niektóre ułożenia wymagają dodatkowych parametrów, przy ich braku narysują wszystko w jednym punkcie

Dostępne ułożenia

- Symulacja fizyczna (węzły połączone wirtualnymi sprężynami)
- Drzewo
- Oś
- Okrąg
- Rozmieszczenie losowe
- Własny layout z podaniem współrzędnych

Dostępne ułożenia

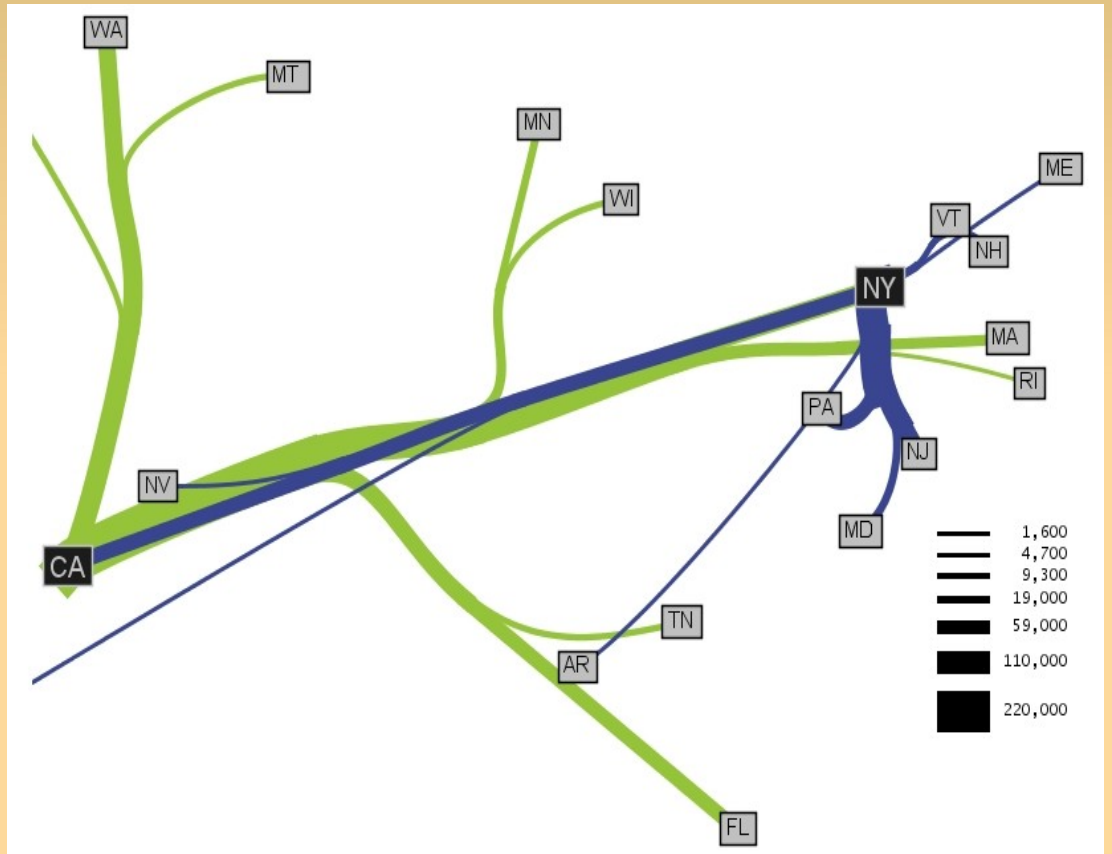
- Bliskoznaczność słów – im bliżej środka – tym bliższe znaczenie



Dostępne ułożenia

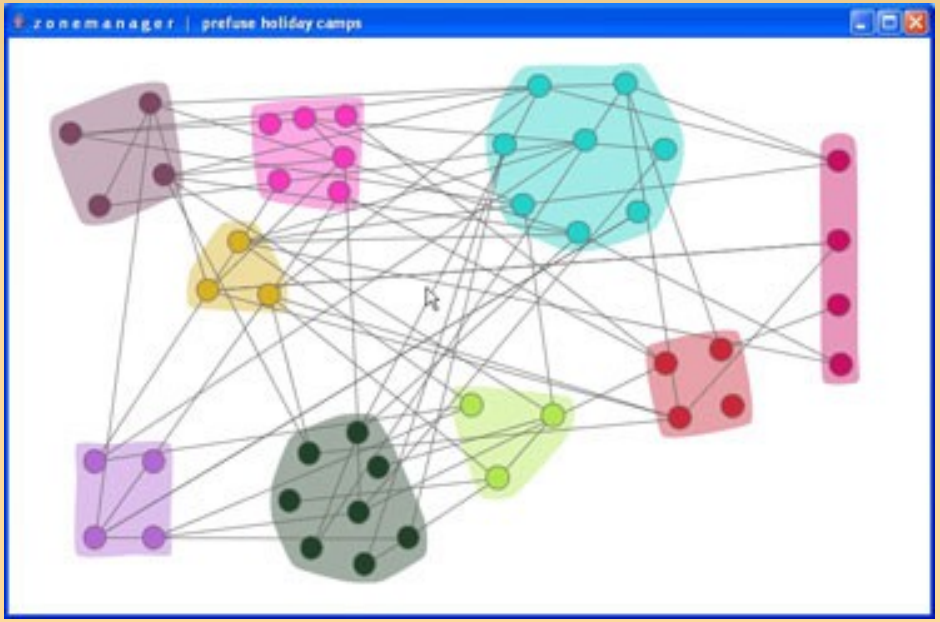
- ”Wędrówki ludów” między stanami

  USA
  - grubość linii reprezentuje wielkość przepływu



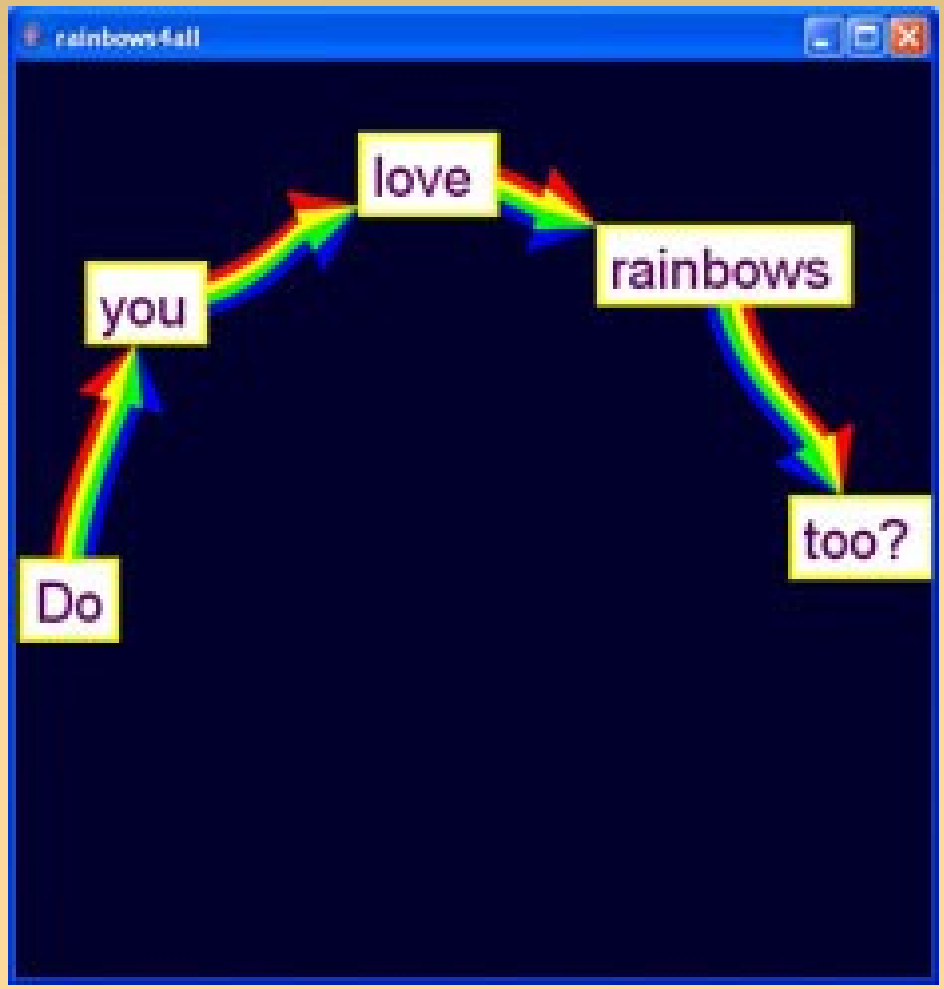
Gadżety

- ZoneManager – grupowanie sprężynujących węzłów



Gadżety

- RainbowEdges – bo grafy wcale nie muszą być nudne!



Gadżety

- Poza tym:
  - filtr ”rybie oko”
  - podświetlanie krawędzi
  - centrowanie na klikniętym węźle
  - cdn.

Dynamiczne dodawanie węzłów

- Teoretycznie nie powinno się, ale jak się bardzo chce..

```
Node rootNode,myNode;
Graph t;
Display disp;
synchronized (disp) { // kluczowe! inaczej wizualizacja zniszczy nasze zmiany
        Node myNode = t.addNode();
        t.addEdge( rootNode, myNode);
}
```

Demonstracja

- Sprytny Wiki Nawigator

  http://swn.eti.pg.gda.pl/swn2/Uruchom.aspx

Podsumowanie

- Zalety Prefuse z punktu widzenia projektu SWN
  - płynne rysowanie kilkuset węzłów
  - łatwa integracja z Web Services
  - gotowe komponenty, które można sukcesywnie podmieniać
  - możliwość szybkiego zbudowania interfejsu (Swing)
  - możliwość pracy bez serwera WWW
- Pytania?