Wizualizacja grafów

Szymon Matejczyk

Szymon Matejczyk

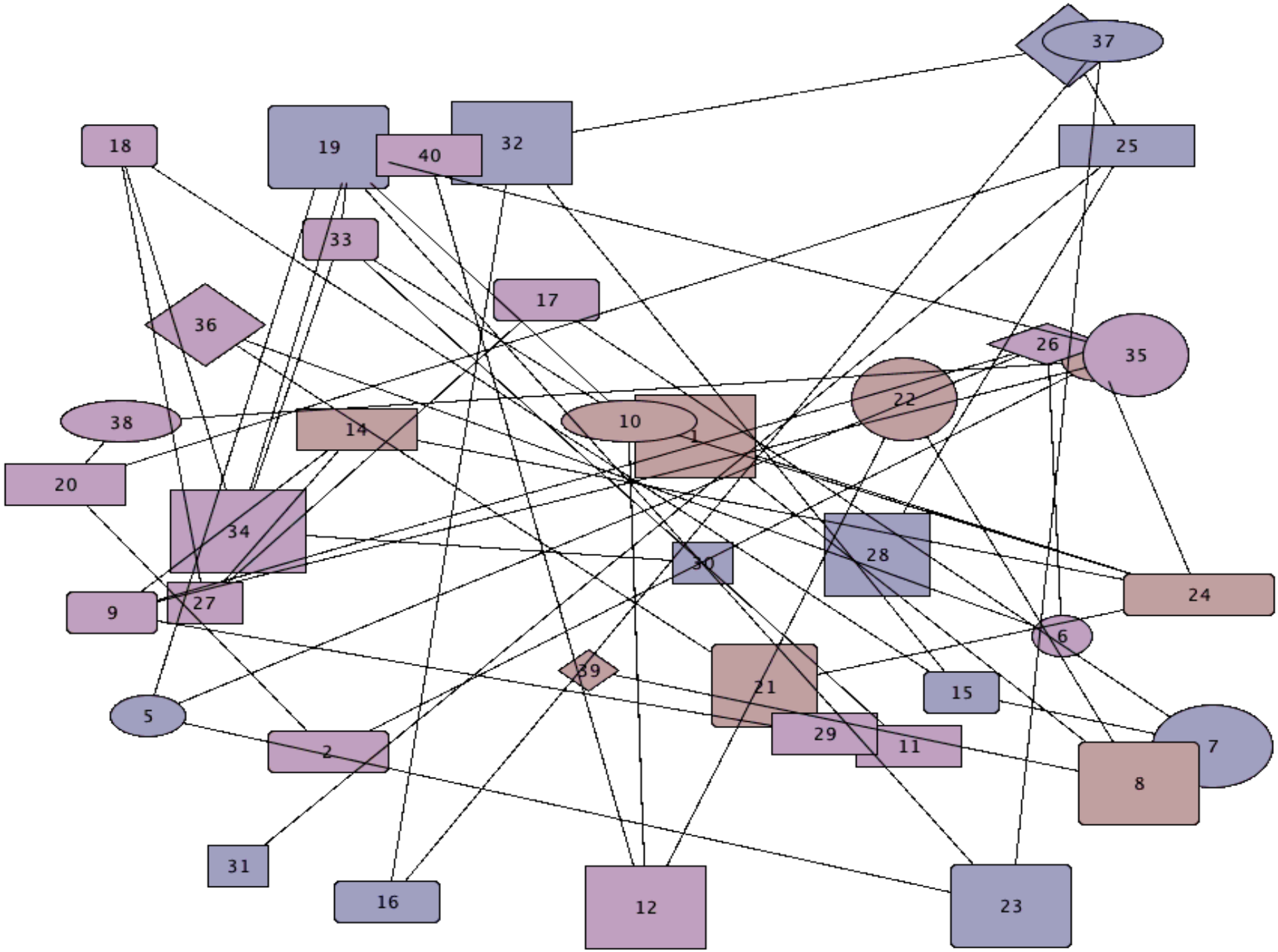
Plan prezentacji

1. Wprowadzenie
2. Zarys teorii
3. Reprezentacja grafów
4. Programy do rysowania (nasze wymagania)
    1. graphViz
    2. diagram.ly
    3. yEd
5. Biblioteki
    1. jgraph (mxGraph)
    2. yFiles
    3. JUNG
6. Podsumowanie

Szymon Matejczyk

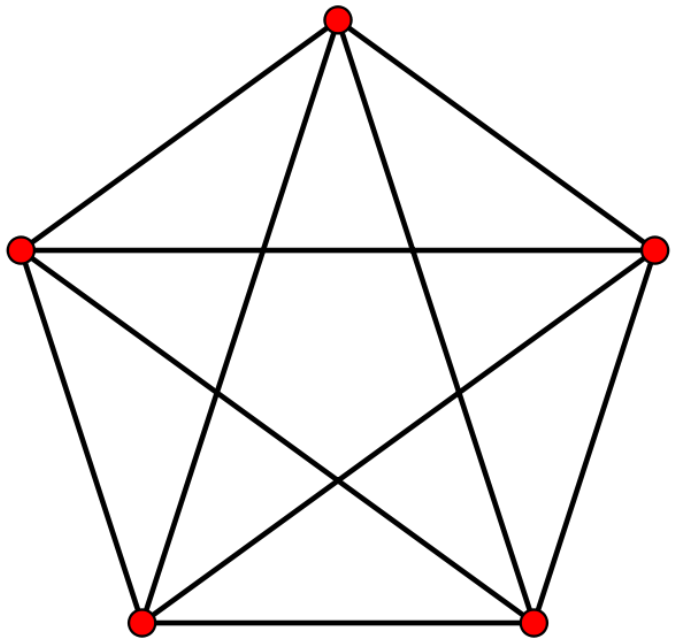
Po co?

- bioinformatyka
- business process modeling
- data mining
- bazy danych
- zarządzanie sieciami
- sieci społecznościowe
- projekty informatyczne
- wizualizacja WWW
- visual programming

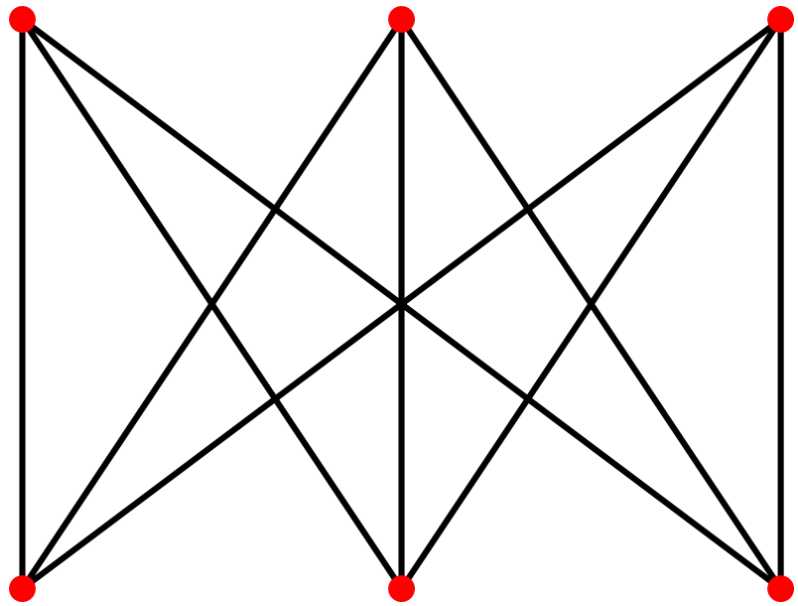

37
18
19
40
32
25
33
17
36
26
35
22
38
14
10
1
20
34
28
30
24
9
27
6
39
21
15
5
29
11
7
2
8
31
16
12
23

Szymon Matejczyk

Planarność



K_5



$K_{3,3}$

Graf skończony jest planarny wtedy i tylko wtedy, gdy nie zawiera podgrafu homeomorficznego z grafem K_5 ani z grafem $K_{3,3}$.

Szymon Matejczyk

Planarność

Wzór Eulera

Jeżeli G jest grafem spójnym i planarnym, to $|V| + |S| - |E| = 2$, gdzie V - zbiór wierzchołków, E - zbiór krawędzi, S - zbiór ścian dowolnego rysunku płaskiego grafu G.

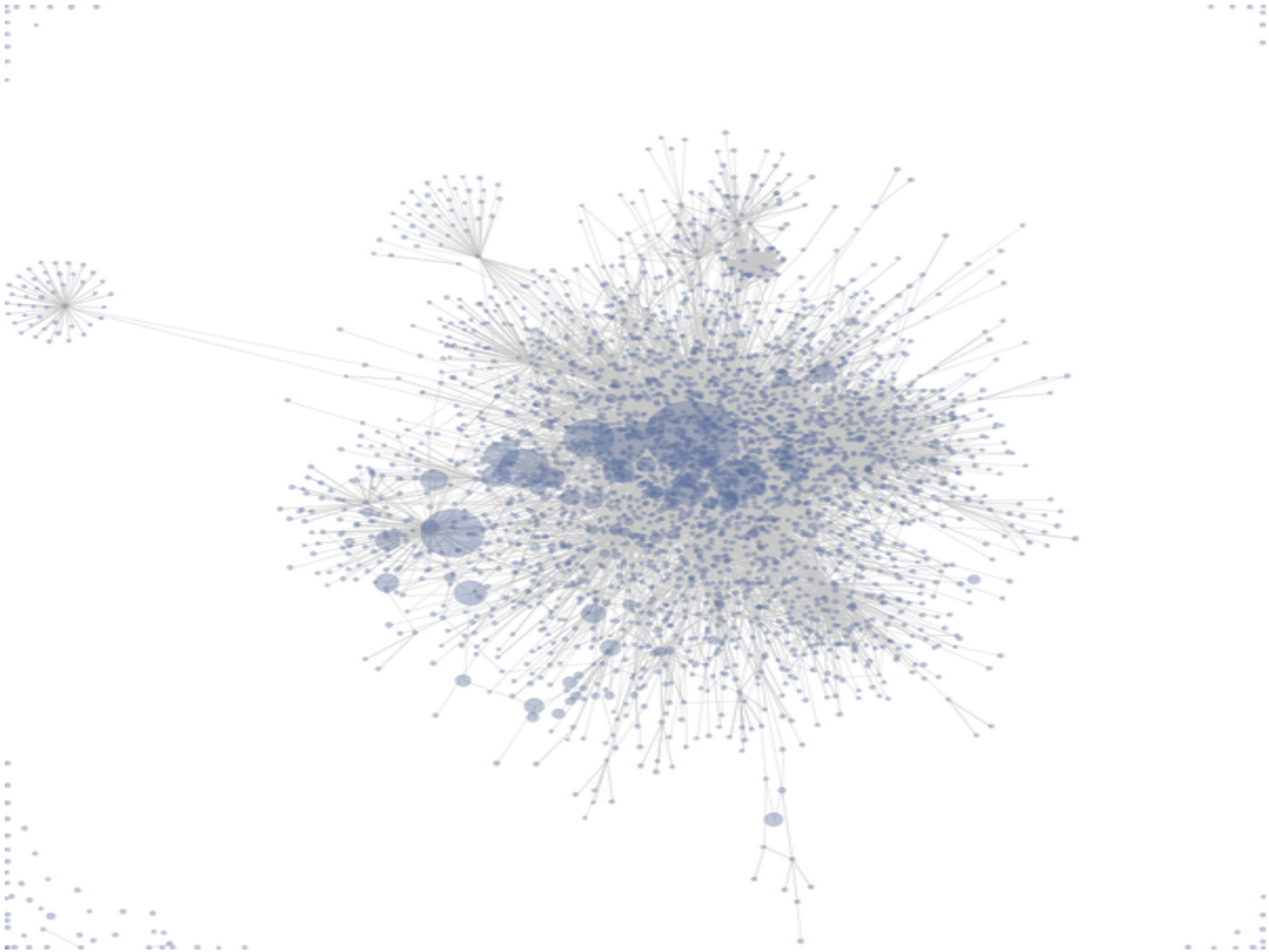
Wnioski:

- Jeżeli G jest planarny, to $|E| \leq 3 \cdot |V| - 6$
- Jeżeli G jest planarny, to wierzchołek o najmniejszym stopniu jest stopnia co najwyżej *5*.

Wizualizcja grafów

Szymon Matejczyk

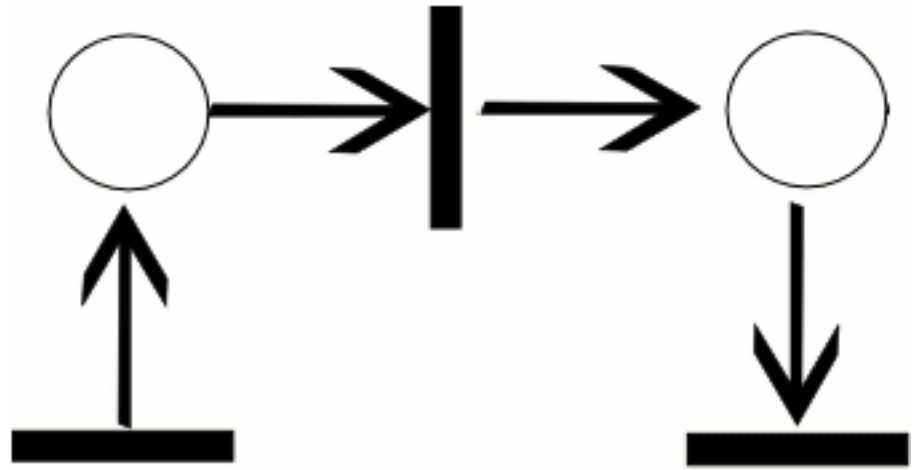
Force-directed algorithm



Szymon Matejczyk

Inne grafy

- sieci Petriego
- hipergrafy
- grafy 3d?
- etykietowanie?



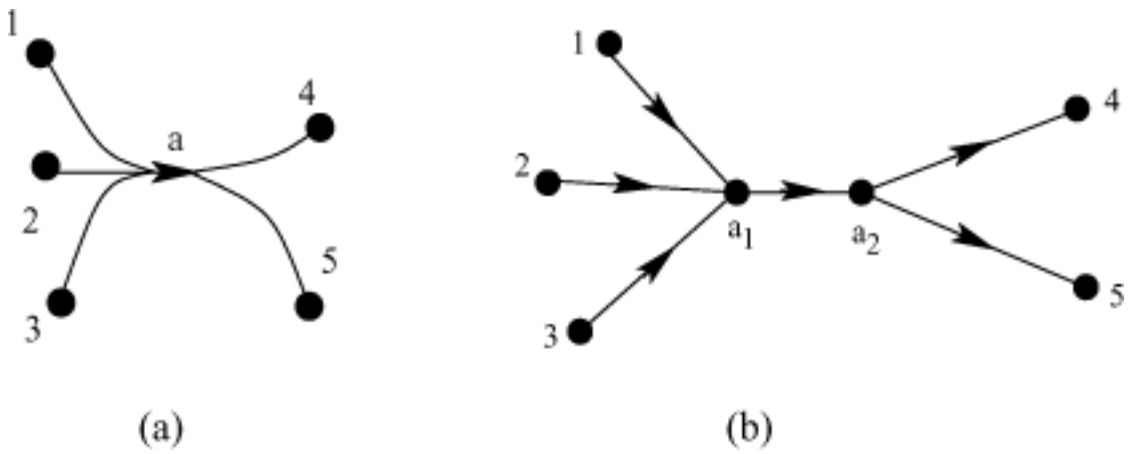
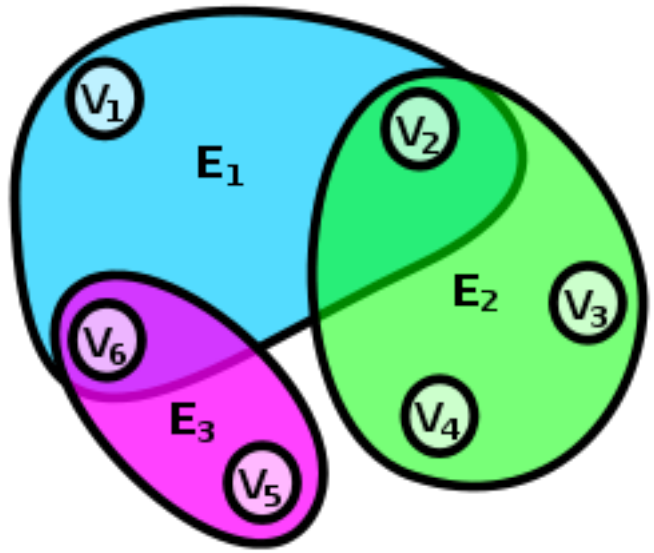


Figure 3 – Grouping origin and destination of a hyperarc.



Szymon Matejczyk

Reprezentacje grafów: graphml

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml ...>
<key id="d0"for="node" attr.name="color" attr.type="string">
<default>yellow</default>
</key>
<key id="d1" for="edge"attr.name="weight" attr.type="double" />
<graph id="G"edgedefault="undirected">
<node id="n0">
<data key="d0">green</data>
</node>
<node id="n1" />
<node id="n2">
<data key="d0">blue</data>
</node>
<edge id="e0" source="n0" target="n2">
<datakey="d1">1.0</data>
</edge>
<edge id="e1" source="n0" target="n1">
<data key="d1">1.0</data>
</edge>
<edge id="e2" source="n1"target="n3">
<data key="d1">2.0</data>
</edge>
</graph>
</graphml>
```

Szymon Matejczyk

Reprezentacje grafów: gml

```
graph
[
hierarchic 1
label ""
directed 1
node
[
id 0
label "1"
graphics
[
...
                ]
]
node
[
id 1
label "2"
]
edge
[
source 25
target 4
]
```

Szymon Matejczyk

Reprezentacje grafów: DOT language

```
digraph unix {
size="6,6";
node [color=lightblue2, style=filled];
"5th Edition" -> "6th Edition";
"5th Edition" -> "PWB 1.0";
"6th Edition" -> "LSX";
"6th Edition" -> "1 BSD";
"6th Edition" -> "Mini Unix";
"6th Edition" -> "Wollongong";
"6th Edition" -> "Interdata";
"Interdata" -> "Unix/TS 3.0";
"Interdata" -> "PWB 2.0";
"Interdata" -> "7th Edition";
"7th Edition" -> "8th Edition";
"7th Edition" -> "32V";
"7th Edition" -> "V7M";
"7th Edition" -> "Ultrix-11";
```

Szymon Matejczyk

Reprezentacje grafów

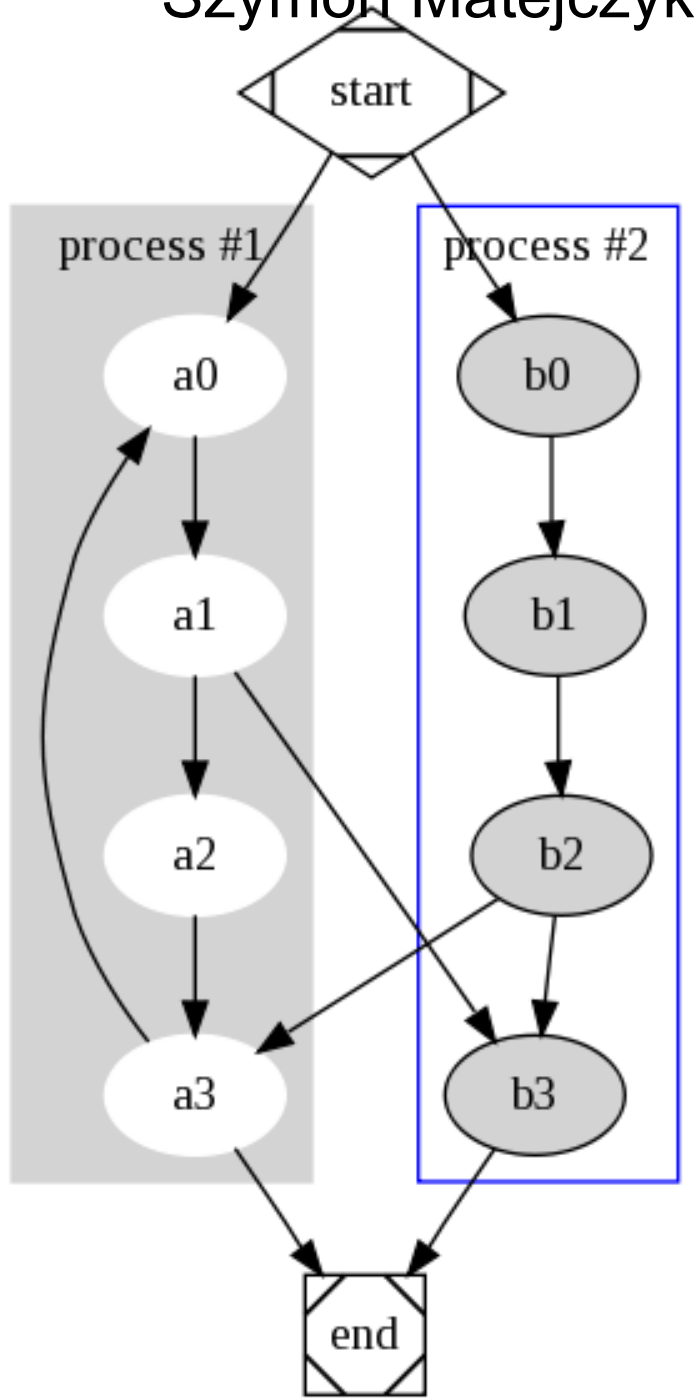
Inne:

- gxl
- trivial graph format
- XGGML
- ...

Szymon Matejczyk

Graphviz

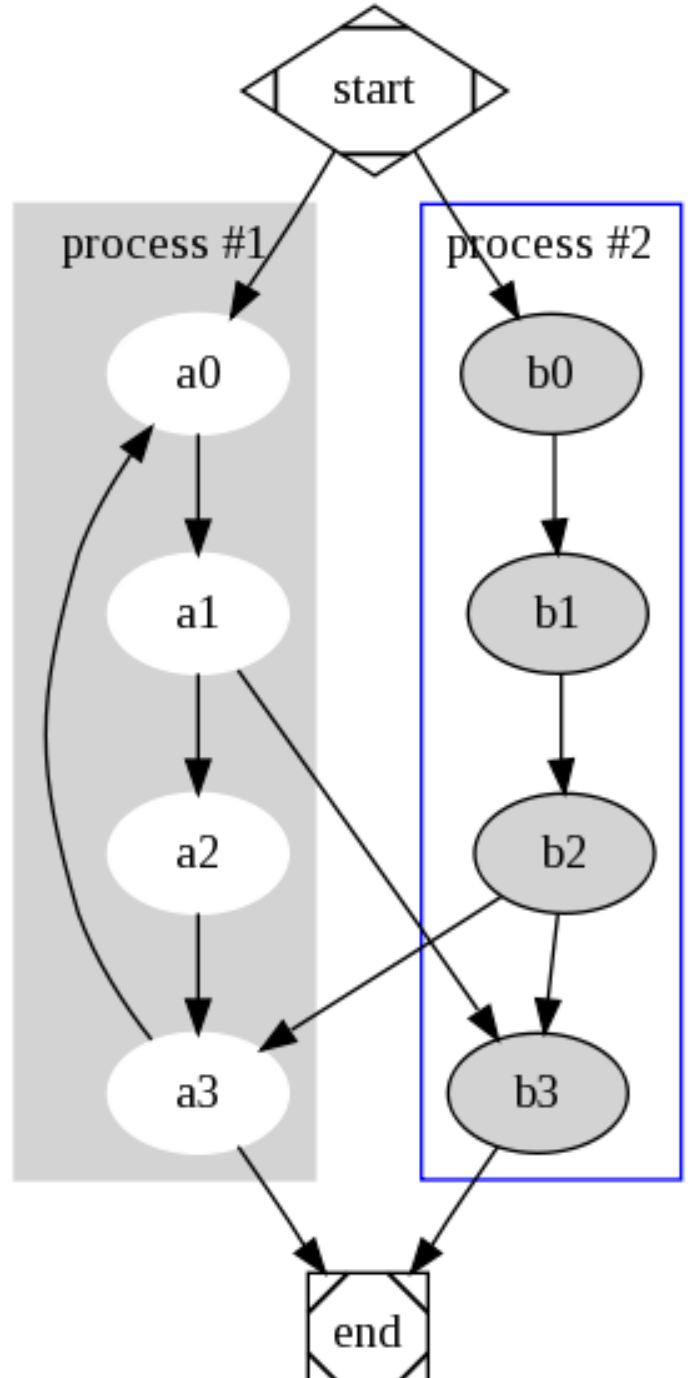
- Open source
- input: DOT language
- obsługa z linii poleceń
- brak edycji
- biblioteki:
  - C
  - Tcl/Tk
- output: svg, PDF, PS
- layouty - pluginy



Szymon Matejczyk

Graphviz - neato

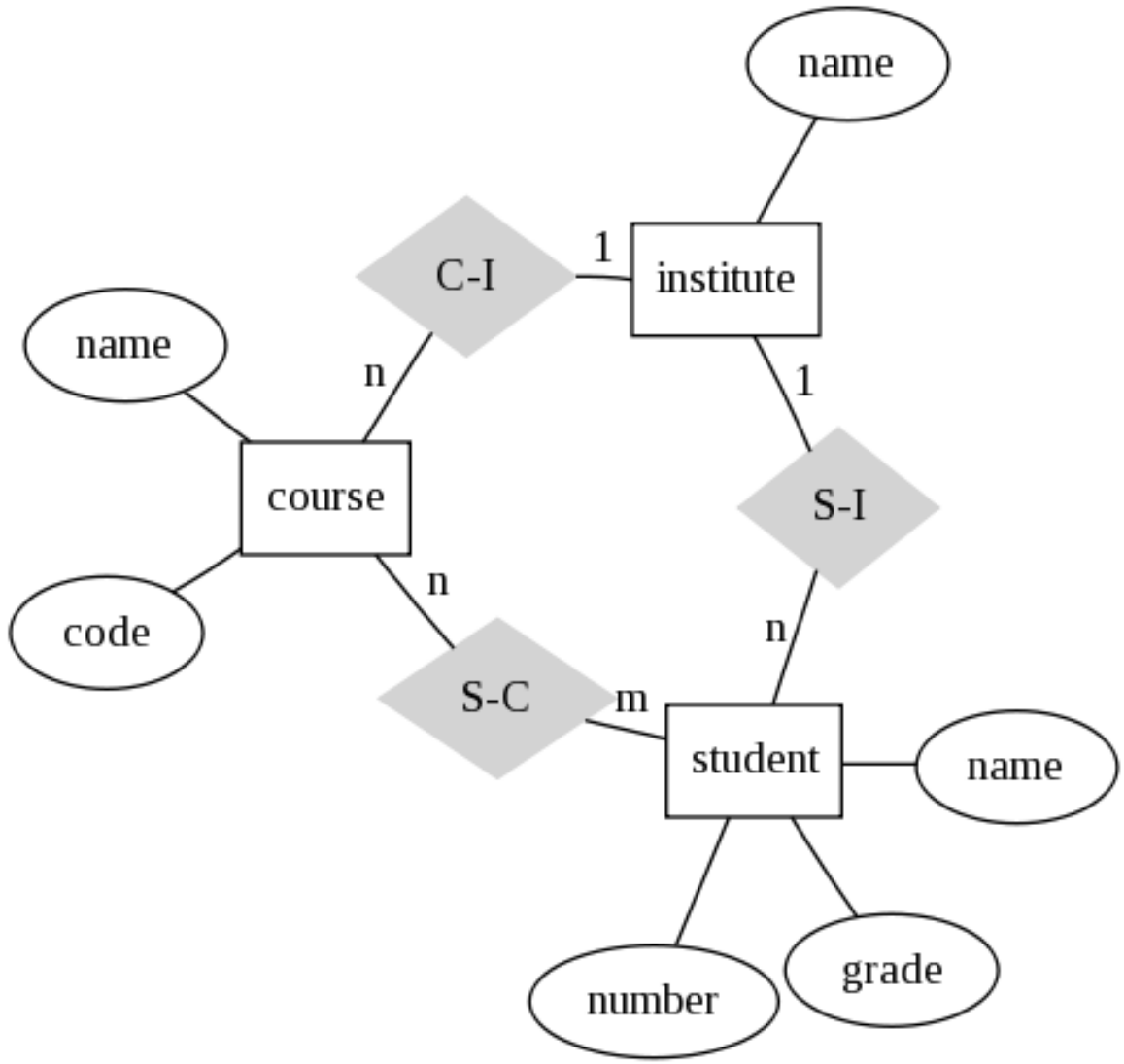
dot - "hierarchical'' or layered drawings of directed graphs. This is the default tool to use if edges have directionality. dot aims edges in the same direction (top to bottom, or left to right) and then attempts to avoid edge crossings and reduce edge length.



Szymon Matejczyk

Graphviz - neato

neato - This is the default tool to use if the graph is not too large (about 100 nodes) and you don't know anything else about it. Neato attempts to minimize a global energy function, which is equivalent to statistical multi-dimensional scaling.

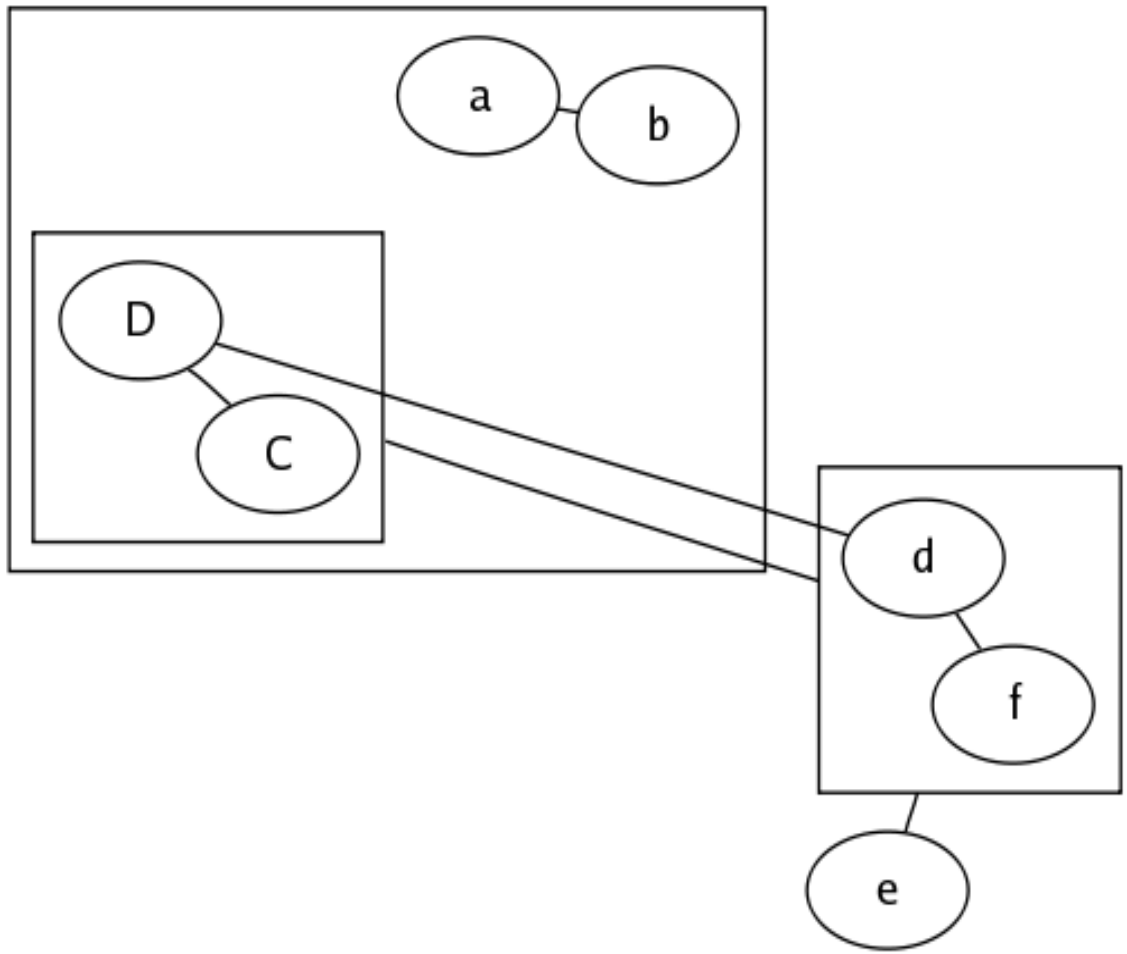


Entity Relation Diagram

Szymon Matejczyk

Graphviz - fdp

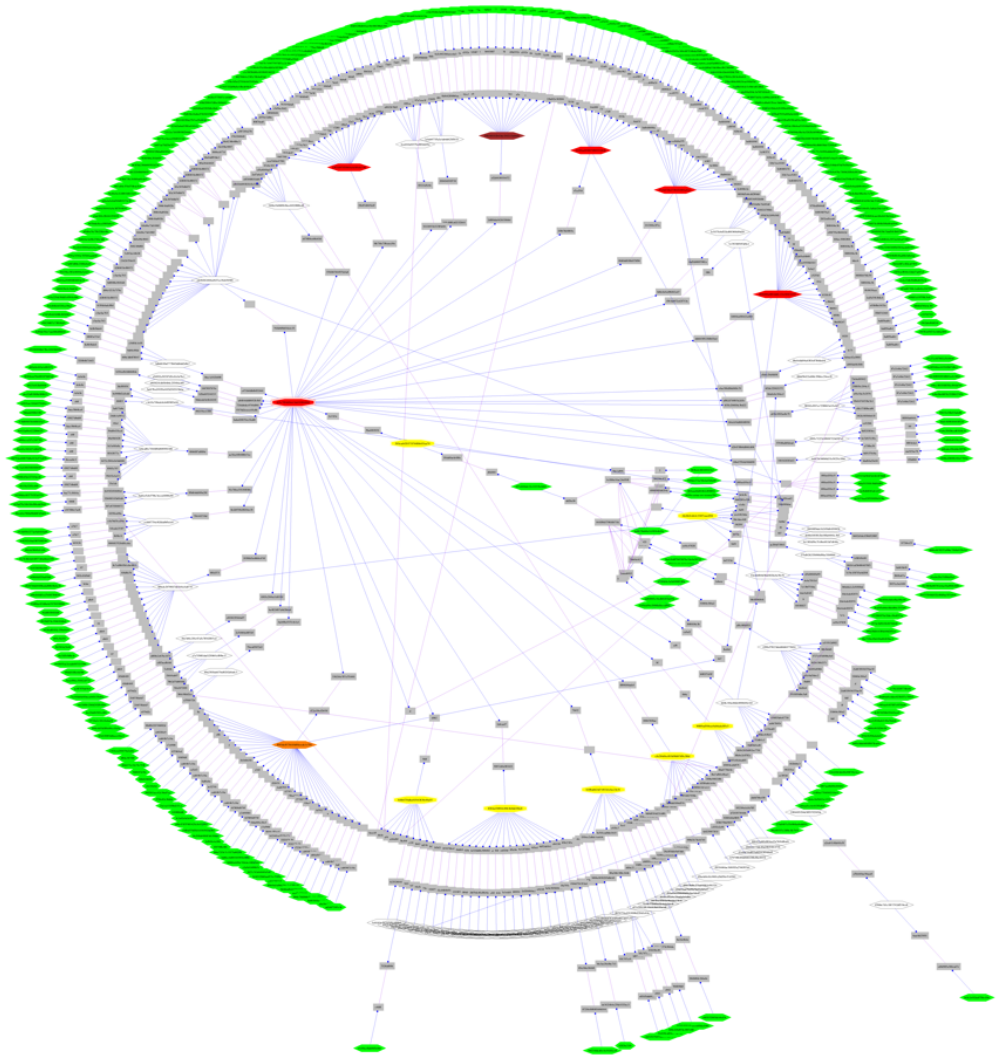
fdp - "spring model'' layouts similar to those of neato, but does this by reducing forces rather than working with energy. Fdp implements the Fruchterman-Reingold heuristic including a multigrid solver that handles larger graphs and clustered undirected graphs.



Szymon Matejczyk

Graphviz - wopi

wopi - radial layouts, after Graham Wills 97. Nodes are placed on concentric circles depending their distance from a given root node. You can set the root node, or let twopi do it.



Szymon Matejczyk

Graphviz - circo

circo - circular layout, after Six and Tollis 99, Kauffman and Wiese 02. This is suitable for certain diagrams of multiple cyclic structures, such as certain telecommunications networks.

Szymon Matejczyk

diagram.ly

- Java script
- input/export: własny xml
- export: jpg, png, svg
- bogate opcje graficzne
- grupy/porty
- różne typy diagramów
- wszystkie przegladarki
- integracja z php/.Net/AJAX

Wizualizacja grafów

Szymon Matejczyk

yEd

Szymon Matejczyk

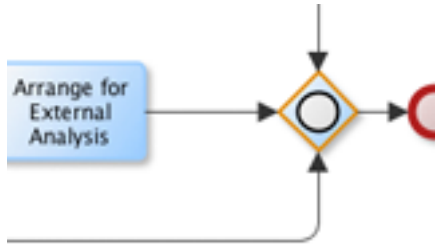
yEd

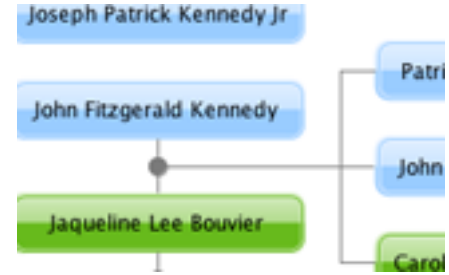
- free
- import: xml, excel, graphml
- export: graphml, graficzne
- automatic layouts
- Windows, Unix, MacOs

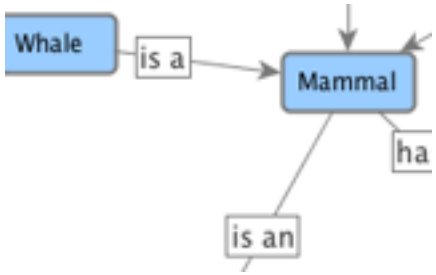
Szymon Matejczyk

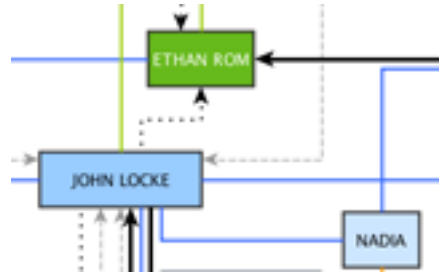
yEd - supported diagrams

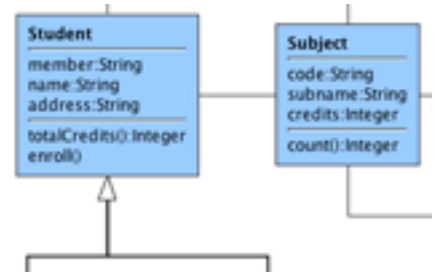
yWorks











Szymon Matejczyk

Biblioteki - wymagania

- import/export do popularnych formatów
- wspierane typy grafów, grupowanie
- przybliżanie i oddalanie, przesuwanie, itp.
- wbudowany edytor
- reakcja na zdarzenia użytkownika
- dodatkowe dane (labele, opisy, obrazki)
- layouty
- algorytmy grafowe
- wbudowane typy wyświetlania grafu
- wsparcie, dokumentacja
- otwartość kodu
- wspierane systemy
- technologie webowe

Szymon Matejczyk

jgraph

- Free, open-source (BSD)
- SWING
- wczytywanie z ~CSS
- bogaty zbiór wbudowanych symboli
- import export do własnego xml
- walidacja
- framework JavaScript, diagram.ly
- komercyjne wsparcie
- bardzo skromny manual + przykłady
- tranzakcje

Szymon Matejczyk

jgraph - grafy

- multigrafy
- porty
- grupy
  - rozwijanie i zwijanie, ukrywanie
  - wspólne wartwy
  - widok tylko danej grupy

Szymon Matejczyk

jgraph

```
// Adds cells to the model in a single step graph.getModel().beginUpdate(); try { Object v1 = graph.addVertex(parent, null, "Hello,", 20, 20, 80, 30); Object v2 = graph.addVertex(parent, null, "World!", 200, 150, 80, 30); Object e1 = graph.addEdge(parent, null, "", v1, v2); } finally { // Updates the display graph.getModel().endUpdate(); }
```

Wizualizcja grafów

Szymon Matejczyk

jgraph

Szymon Matejczyk

yFiles

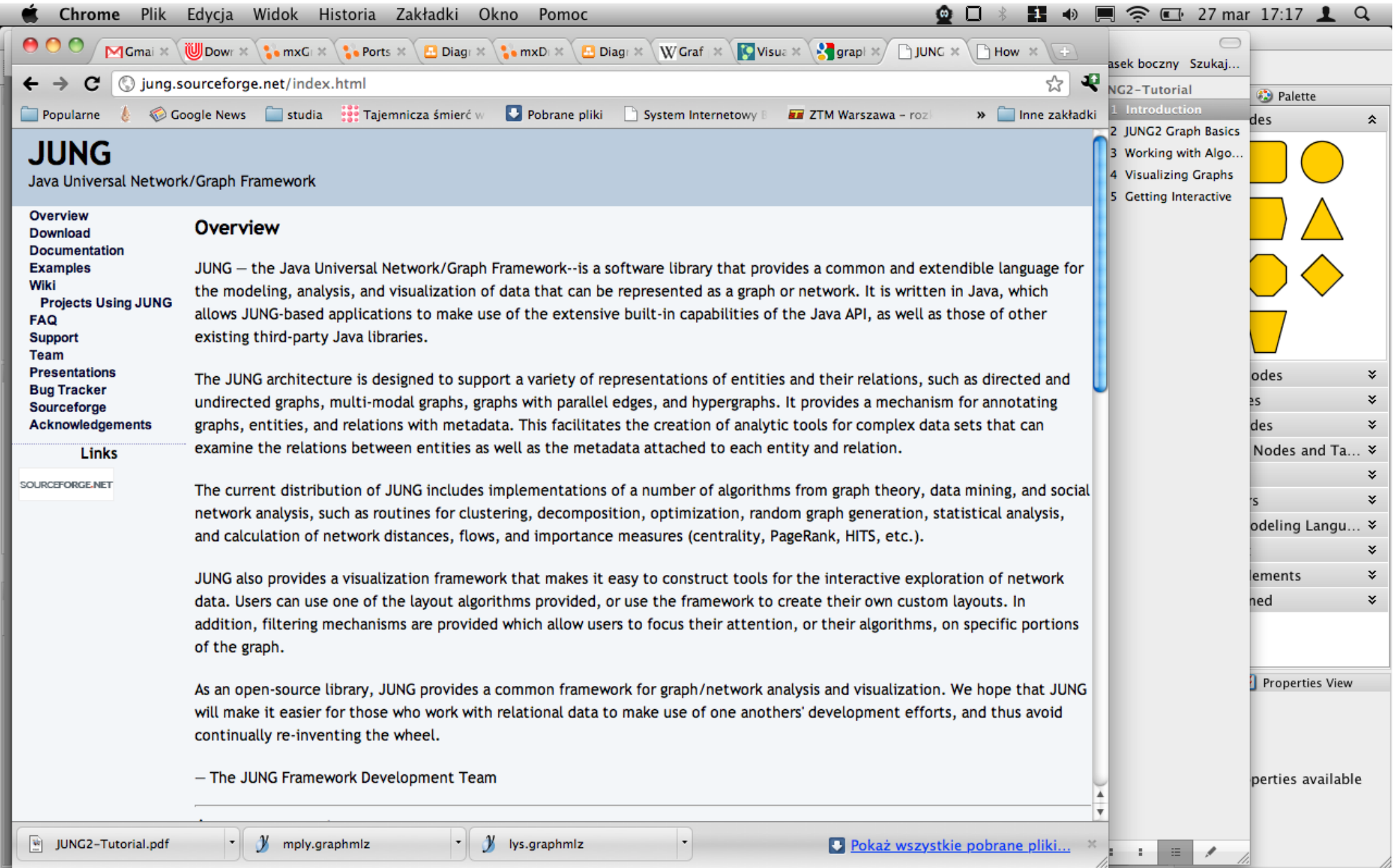
- komercyjny (30 dniowy trial)
- także w .NET
- import/export z graphML, gml
- export do formatów grafiki rastrowej i wektorowej
- notacja UML, BPMN, wbudowane style całych diagramów
- wspracie dla JS, Silverlight, Flex
- bogaty manual i przykłady

yWorks

Wizualizacja grafów

Szymon Matejczyk

JUNG



Szymon Matejczyk

JUNG

- open source
- input: graphml, PAJEK
- layouts
- export: png, jpg, eps
- lekki
- brak przykładów i tutoriala - źródła i API dostępne

Wizualizacja grafów

Szymon Matejczyk

JUNG

```
Graph<Integer, String> g = new SparseMultigraph<Integer, String>();
// Add some vertices. From above we defined these to be type Integer.
g.addVertex((Integer)1);
g.addVertex((Integer)2);
g.addVertex((Integer)3);
// Add some edges. From above we defined these to be of type String
// Note that the default is for undirected edges.
g.addEdge("Edge-A", 1, 2); // Note that Java 1.5 auto-boxes primitives
g.addEdge("Edge-B", 2, 3);
System.out.println("The graph g = " + g.toString());
```

Szymon Matejczyk

JUNG

```
public static void main(String[] args) {
    SimpleGraphView sgv = new SimpleGraphView(); //We create our graph in here
    Layout<Integer, String> layout = new CircleLayout(sgv.g);
    layout.setSize(new Dimension(300,300));

    BasicVisualizationServer<Integer,String> vv =
        new BasicVisualizationServer<Integer,String>(layout);
    vv.setPreferredSize(new Dimension(350,350));

    JFrame frame = new JFrame("Simple Graph View");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().add(vv);
    frame.pack();
    frame.setVisible(true);
}
```

Przykłady

- jgraph
  - HelloWorld
  - Port
  - walidacja
- yFiles
  - layout.withoutview.GroupingLayoutWithoutAView
  - layout
    - genealogy
    - orgchart
    - view.realizer
    - io.graphml