HORIZON 2020
ICT - INFORMATION AND COMMUNICATION TECHNOLOGIES

PROFIT
Promoting Financial Awareness and Stability

PROMOTING FINANCIAL AWARENESS AND STABILITY

H2020 – 687895

VISUALIZATIONS AND INFORMATION WIDGETS

WORK PACKAGE NO.	**WP2**	WORK PACKAGE TITLE	**LINKED DATA LIFE CYCLE**
TASK NO.	**T2.3**	TASK TITLE	**DATA PUBLICATION, VISUALISATION AND SHARING**
MILESTONE NO.		**8**	
ORGANIZATION NAME OF LEAD CONTRACTOR FOR THIS DELIVERABLE		**SWC**	
EDITORS		**MARTIN SCHAUER, ARTEM REVENKO (SWC)**	
CONTRIBUTORS		**HEIDELINDE HOBEL (SWC)**	
REVIEWERS		**PETER HANECAK (EEA), AIKATERINI KATMADA (CERTH)**	
STATUS (F: FINAL; D: DRAFT)		**F**	
NATURE		R – REPORT	
DISSEMINATION LEVEL		PU – PUBLIC	
PROJECT START DATE AND DURATION		**JANUARY 2016, 36 MONTHS**	
DUE DATE OF DELIVERABLE: ACTUAL SUBMISSION DATE:		**JUNE 30, 2017** **JUNE 30, 2017**	

REVISION HISTORY

VERSION	DATE	MODIFIED BY	CHANGES
0.1	19-05-2017	A. REVENKO (SWC)	FIRST VERSION OF TOC
0.2	22-05-2017	A. REVENKO (SWC)	DRAFTED EXECUTIVE SUMMARY, INTRODUCTION, CONCLUSION
0.3	24-05-2017	A. REVENKO (SWC)	ADDED SECTION 2
0.4	26-05-2017	A. REVENKO (SWC)	ADDED SECTION 3
0.5	29-05-2017	A. REVENKO (SWC)	ADDED SECTION 4
0.5.1	30-05-2017	A. REVENKO (SWC)	FINALIZED EXECUTIVE SUMMARY, INTRODUCTION, CONCLUSION
0.6	01-06-2017	A. REVENKO (SWC)	FINAL OFFICIAL PRELIMINARY DRAFT VERSION OF D2.4
0.7	09-06-2017	A. REVENKO (SWC)	REVIEW FROM AIKATERINI KATMADA RECEIVED. THE CONTENTS REVISED ACCORDINGLY.
0.7.1	16-06-2017	A. REVENKO (SWC)	REVIEW FROM PETER HANECAK RECEIVED. THE CONTENTS REVISED ACCORDINGLY.
1.0	26-06-2017	A. REVENKO (SWC)	FINAL REVIEW, PROOFING AND QUALITY CONTROL. READY FOR SUBMISSION

List of Abbreviations

API Application Program Interfaces. 7

GUI Graphical User Interface. 7

ID Identifier. 11

JPEG Joint Photographic Experts Group. 15

JS JavaScript. 7

PDF Portable Document Format. 15

PNG Portable Network Graphics. 15

SKOS Simple Knowledge Organization System. 9

SVG Scalable Vector Graphic. 15

UI User Interface. 8

URI Uniform Resource Identifier. 11

List of Figures

Table of Contents

Executive Summary

The deliverable presents the results of preparing the visualization tools for the PROFIT platform. It reports about the following achieved results:

Section 2 Creation of a widget for visualizing the knowledge graph. The tool is especially suitable for visualizing large complex graphs. The user identifies a node of interest and receives the visual representation of the complete local structure and some aggregate representation of the global structure: number of transitive children, number of concept scheme and distance to the top nodes. Moreover, the visualization is interactive, so the user is able to explore the knowledge graph in an intuitive manner.

Section 3 Creation of a widget for visualizing numerical trends. The interactive tool allows for visualization of a time series. The user is able to trigger several trends and change the time span. Moreover, on hovering the mouse over data points the user gets additional information about their values.

Section 4 Creation of a widget for visualizing conceptual trends (topics). The interactive tool allows for visualizing the temporal distribution of concepts or topics. The user is able to trigger several topics and change the time span. On hovering the mouse over data points the user gets additional information about their values.

1 Introduction

1.1 Main Notions

A *widget* in general is an encapsulated part of a GUI that allows users to interact with it in one or more ways to provide them with advanced functionalities. A few examples of usages of widgets would be:

- providing additional information to what already can be found in the application embedding the widget;
- integrating some external service by using its API;
- allowing a user to easier unterstand a set of data by visualising it in a user-friendly way, e.g. a chart.

As already mentioned one typical characteristic of a widget is that it allows a user to interact with it. Whatever a widget's purpose is, this interaction gives users the possibility to adapt it to their needs, and therefore allowing each of the users to benefit from the widget in their own customized way. This is one of the reasons why charts are a popular instrument of representation inside widgets. Well implemented charts offer a wide variety of possibilities to interact with them to make a chart exactly show what a user wants to see in the way user wants to see it.

Implementing charts that support multiple ways of interacting with them while still looking appealing to the users is a time-consuming piece of work. That's why nowadays so-called "libraries" are an integral part of a software developer's life. These libraries are collections of classes, functions or any other programming language specific code allowing a software developer to reuse them instead of having to implement some specific behaviors (e.g. charts) from scratch. The widgets created in the context of the visualization tools of the PROFIT platform are written in JS, which is a client-side script language interpreted at the run-time and executed directly by a user's web browser. The libraries used for working with charts are:

Highcharts / Highstock (`https://www.highcharts.com/`): Highcharts is a JS library containing all the functionality for working with a set of interactive charts, highly customizable by the software developer. Highstock is the pendant used for charts showing timelines with more advanced ways of interacting with the results.

D3.js (`https://d3js.org/`): "D3.js" basically is a JavaScript library binding data to the document object model allowing the developer to do all sorts of transformations with it. In the context of the visualization tools it is used to build a custom chart type displayed as an SVG container and map the data directly to this chart.

Since JS is a client-side script language most of the interaction with a chart doesn't require sending requests to the server, which allows the chart to react to user input (e.g. mouse clicks) without any delay. The only time it is necessary for the widgets to contact the server is when a different data point has to be displayed. These requests are done

via Ajax, which allows contacting the server and processing its result without having a load a new page in the web browser, so even then an interruption of the user experience is avoided by using graphical transitions, which fluently guide the user to the new set of data.

1.2 Scope of the Document

The document describes three different visualization tools. The description of each tool covers the following properties:

- the purpose of the tool;
- the expected input;
- the UI;
- the requirements;
- an example of usage.

1.3 Relation to PROFIT Project

In frames of Deliverable 2.4 a knowledge graph was delivered. The knowledge graph has multiple usage, one of the usages is to help the users explore the structure of the domain. Moreover, in PROFIT project we aim at crowd-sourcing the creation of the knowledge graph. To facilitate both usages we develop a visualization tool that is capable of visualizing the knowledge graph in an intuitive manner.

In order to visualize the outcomes of WP4 we develop tools to represent numerical trends and the outcomes of the sentiment analysis assessment.

Finally, for representing the outcomes of the trend analysis in the news articles we develop a tool for visualizing the distribution of conceptual and topical trends over time.

Development Procedure

All the tools were designed in an agile way. The initial design was presented to the consortium and the feedback was collected. The design was reworked according to the collected feedback and the first demonstrator of the tools was prepared. The demonstrator was shown to the consortium and the feedback was again collected and incorporated into the development of the final implementations. This process allows us to state that the tools were developed in a tight cooperation with the whole consortium and the tools are supposed to cover the needs of all the partners.

2 Knowledge Graph Visualization

2.1 Introduction

This section describes a tool to visualize a knowledge graph, based on semantic web principles. It contains notions from the technical and mathematical areas. Below we introduce some of the notions that are essential for understanding the contents of the current section.

A *thesaurus* is a set of concepts with three relations defined between concepts: broader, narrower, related. The related relation is symmetric, i.e. if concept c_1 is related to c_2 then c_2 is related to c_1. The relations broader and narrower are

1. inverse of each other, i.e. if c_1 is broader than c_2 then c_2 is narrower than c_1,
2. transitive, i.e. if c_1 is broader than c_2 and c_2 is broader than c_3 than c_1 is broader than c_3.

The *top concepts* are the concepts that do not have any broaders. Leaves are the concepts that do not have any narrowers. We also use the notion of the *concept scheme* in order to be able to define hierarchies of different "nature" in a single thesaurus. Formally the concept schemes are not different from top concepts.

An *ontology* as we use it here consists of three sets: classes, relations, attributes. Later the individual concepts from a thesaurus may be assigned to classes from an ontology. We may define a class A to be a subclass of a different class B. This would imply that any individual i belonging to class A also belongs to class B. Relations define possible "links" between individuals of different classes. The relations may be symmetric or inverse. Also we may define a relation to be functional, i.e. an individual may have only one another individual that is in the specified relation to it. Attributes of individuals "link" the individuals to the values of fixed types, e.g. dates, integers, url.

For storing the thesauri we use the SKOS ontology[1]. In addition to the three described relation on the concepts of the thesaurus the SKOS defines some attributes as, for example, primary and alternative labels of the concepts in different languages. This way SKOS thesauri are appropriate for storing multi-lingual thesauri. Moreover, SKOS thesauri use *matching* links to other thesauri to be able to define links between different thesauri.

A *knowledge graph* as we use it here is an application of an ontology to a set of individuals, assigning individuals to classes, defining relations between individuals, defining attributes of individuals. We typically use concepts from a thesaurus as individuals.

[1] `https://www.w3.org/TR/2008/WD-skos-reference-20080829/skos.html`

2.2 Description

This type of chart shows a detailed view of a concept of a thesaurus including all its directly related concepts and the concept scheme it belongs to. In the center of the chart the currently selected concept is displayed as a circle containing the concept's label and right below the total number of transitive narrower concepts. Direct "skos:narrower" (i.e. child) concepts are displayed as ellipses below the root concept, direct "skos:broader" (i.e. parent) concepts are displayed as ellipses on top of the root concept. Every broader concept can be connected to one or multiple concept schemes displayed in the center on the very top of the chart. The lines connecting a broader concept and a concept scheme show the number of concepts on the shortest path from the center concept to the concept scheme in a small circle. For example if this number says "1" concept scheme and the broader concept are directly connected, if it says "3" the shortest path to the concept scheme would include two more concepts between broader concept and concept scheme.

On the left and right of the center concept all the other related concepts are displayed, which includes concepts connected via the "skos:related"-property as well as all custom relations. Since the relation type can vary the title of the relation gets displayed on the line connecting the center concept with one of these related concepts.

There exist following possibilities to interact with the chart:

1. By clicking one of the concepts, which shows the user additional information on this concept in the details area to the right or at the bottom of the visualisation and highlights the path from the selected concept up to the concept scheme(s).
2. By dragging a concept around the graph (this works for all concepts except the center concept) to e.g. rearrange all the displayed concepts or quickly read a relation label, which was covered by another concept before. After releasing the mouse button the dragged concept will jump back to the cluster of concepts it belongs to.
3. By double clicking one of the concepts, the concept will become the new center concept and all of its relation data gets loaded on the fly and displayed in the visualisation.
4. By using the search bar on top of the visualisation. After writing a few characters into this search box an autocomplete mechanism will suggest a list of concepts matching the entered search string. By clicking on one of the suggestions or selecting it via the arrow keys on the keyboard followed by a click on the enter-button this suggestion becomes the new center concept and all of its relation data gets loaded on the fly and displayed in the visualisation.

2.2.1 Requirements

- jQuery v3.0+

- d3js v4.5+

2.3 Input Interface

The data displayed in the chart needs to be an object representing the center concept. This object needs to have following structure, in which all properties are required except properties marked optional:

uri: The URI of the root concept. This has to be a unique ID.

label: The label of the root concept.

definition (optional): The description of the root concept if one is available.

relations: An object containing all the information about concepts somehow related to the center concept, mapped by their relation type.

narrower (optional): An array of narrower concept objects, each of the objects has following properties:

uri: The URI of the current narrower concept. This has to be a unique ID.

label: The label of the current narrower concept.

definition (optional): The description of the current narrower concept if one is available.

broader (optional): An array of broader concept objects, their structure is the same as in "narrower".

... (optional): Any property name beside "narrower" and "broader" will be treated as a custom relation. These concepts will be displayed to the left and the right of the center concept, their structure is the same as in "narrower".

conceptSchemes (optional): An array of the concept schemes the root concept lies within. Each of the concepts schemes is an object containing following properties:

uri: The URI of the current concept scheme. This has to be a unique ID.

label: The label of the current concept scheme.

definition (optional): The description of the current concept scheme if one is available.

shortest_path: The shortest path from the center object to this concept scheme. For example if this number says "1" concept scheme and the broader concept are directly connected, if it says "3" the shortest path to the concept scheme would include two more concepts between broader concept and concept scheme.

narrower_trans_count: The total number of transitive narrower concepts of the center concepts, which includes the narrower concepts of the center concept, the narrower-

relations of the narrower concepts and so on.

classInfo (optional): An array of custom classes of the center concept if available. Every custom class is an object containing following properties:

label: The label of the custom class, which is displayed in the information box to the right of the visualisation.

In Listing 1 an example of input is presented.

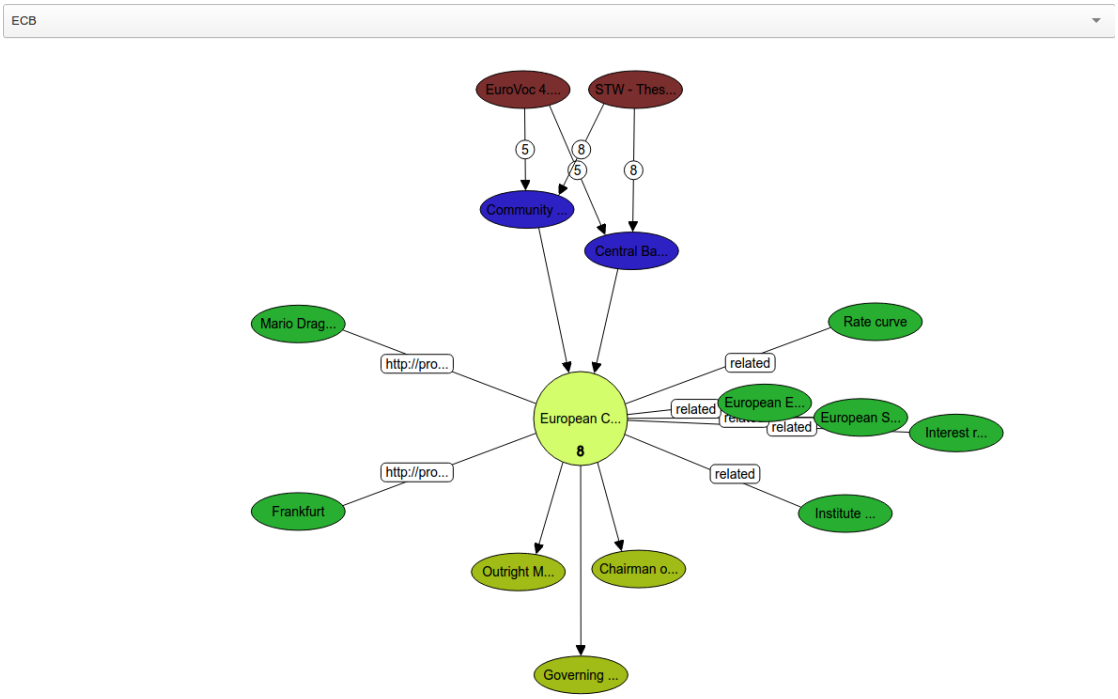
```
{
    'uri': 'http://example.com/04534d05-047b-4603-8c66-
      ↪ ea4f1f8e5f4bLEL',
    'label': 'Root Concept',
    'definition': 'The definition of the root concept.',
    'relations': {
        'narrower': {
            'concepts': [
                {
                    'uri': 'http://example.com/04534d05-047b
                      ↪ -4603-8c66-ea4f1f8e5f4b',
                    'label': 'Narrower Concept 1',
                    'definition': 'The description of
                      ↪ Narrower Concept 1.',
                },
                {
                    'uri': 'http://example.com/04534d05-047b
                      ↪ -4603-8c66-ea4f1f8e5f4c',
                    'label': 'Narrower Concept 2',
                    'definition': 'The description of
                      ↪ Narrower Concept 2.',
                }
            ]
        },
        'http://www.w3.org/2004/02/skos/core#custom_a': {
            'concepts': [
                {
                    'uri': 'http://example.com/04534d05-047b
                      ↪ -4603-8c66-ea4f1f8e5f4b11',
                    'label': 'Custom Concept 1',
                    'definition': 'The description of Custom
                      ↪  Concept 1.',
                }
            ]
        }
    },
```

```
    'conceptSchemes': [
        {
            'uri': 'http://example.com/8d052dfc-44bf-4985-8
              ↪ ce3-4564570a161b',
            'label': 'Concept Scheme 1',
            'definition': 'The description of Concept Scheme
              ↪  1',
            'shortest_path': 3
        }
    ],
    'narrower_trans_count': 15,
    'classInfo': [{
        'label': 'Some Class'
    }]
}
```

Listing 1: Input Example for Knowledge Graph Visualization

2.4 Example

In Figure 1 an example of visualization is presented. The concept "European Central Bank" is visualized. On the figure one can see the broaders, the narrowers, the related concepts, and two custom relations. Below the chart one finds a textual representation of the attributes of the "European Central Bank" concept.



European Central Bank

http://profit.poolparty.biz/PROFIT_Finance/foundedIn

1998-06-01

http://profit.poolparty.biz/PROFIT_Finance/publishes_2

http://www.ecb.europa.eu/pub/projections/html/index.en.html,http://www.ecb.europa.eu/explainers/html/index.en.html,http://www.ecb.europa.eu/pub/html/index.en.html

http://profit.poolparty.biz/PROFIT_Finance/geoPoint

50.1095 8.674

http://profit.poolparty.biz/PROFIT_Finance/homepage

http://www.ecb.europa.eu

http://profit.poolparty.biz/PROFIT_Finance/abstract

The European Central Bank (ECB) is one of the seven institutions of the European Union (EU) listed in the Treaty on European Union (TEU). It is the central bank for the euro and administers the monetary policy of the 17 EU member states which constitute the Eurozone, one of the largest currency areas in the world. It is thus one of the world's most important central banks. The capital stock of the bank is owned by the central banks of all 27 EU member states. The bank was established by the Treaty of Amsterdam in 1998, and is headquartered in Frankfurt, Germany. The current President of the ECB is Mario Draghi, former governor of the Bank of Italy. The primary objective of the European Central Bank is to maintain price stability within the Eurozone, which is the same as keeping inflation low and preventing deflation. The Governing Council aims to keep inflation (as measured by the Harmonised Index of Consumer Prices) below, but close to, 2% over the medium term. Unlike other central banks, for instance, the Federal Reserve System, the ECB has a single primary objective, with other objectives subordinated to it. The basic tasks of the ECB are to define

Figure 1: Visualization of the knowledge graph: “ECB” concept

3 Numerical Visualization

3.1 Introduction

This type of chart shows a numerical time series. For every time series the time is displayed on the x-axis and the y-axis shows the value on a specific date. Below the main graph there is an additional line showing for which time frame data is available.

There exist following possibilities to interact with the chart:

- By clicking on an object in the legend below the x-axis one can hide or show it in the chart.
- There are multiple ways of changing the period of time displayed in the chart:
  - by clicking on one of the predefined time periods in the upper left corner;
  - by manually changing the from- or to-date in the upper right corner. The entered date has to be in the "YYYY-MM-DD"-format;
  - by manually changing the start or end of the time period by dragging the left or right handle next to the marked part of the timeline-overview below the chart;
  - by shifting the currently selected time interval (e.g. 30 days) into the future or the past by dragging the handle below the marked part of the timeline overview below the chart;
- By moving the mouse over one of the dots in the chart one can see a comparison of the values on this day over all displayed concepts.
- By clicking on the icon in the top right corner the user gets a list of possible ways to export the chart, which include downloading the currently displayed chart as an image (PNG, JPEG or SVG), a PDF document or to directly print the chart.

3.1.1 Requirements

- jQuery v3.0+
- Highcharts v5.0+ (including the "exporting" module)

3.2 Input Interface

The series data displayed in the chart needs to be an array of objects resembling the concepts or topics. Each of these objects requires following properties:

name: The name of the time series, which will be used as its label throughout the chart

data: This property contains an array of data points resembling the dots in the chart. Every dot needs to come as an an array containing two elements.

1. The Unix timestamp of the data point in milliseconds, which gets used as the value for the x-axis.
2. The number of concept tags of this topic on this specific data point, which will be displayed on the y-axis.

In Listing 2 an example of input is presented.

```
[
    {
        'name': 'Concept 1',
        'data': [
            [Date.UTC(2017, 0, 1), 30],
            [Date.UTC(2017, 0, 2), 10],
            [Date.UTC(2017, 0, 3), 15]
        ]
    },
    {
        'name': 'Concept 2',
        'data': [
            [Date.UTC(2017, 0, 1), 40],
            [Date.UTC(2017, 0, 2), 30],
            [Date.UTC(2017, 0, 3), 50]
        ]
    }
]
```

Listing 2: Input Example for Numerical Visualization

3.3 Example

In Figure 2 an example of numerical visualization is presented. Some time series is visualized, the time span start at May 2013 and spans till May 2017. The visualization is interactive, on mouse hover action additional information is shown.

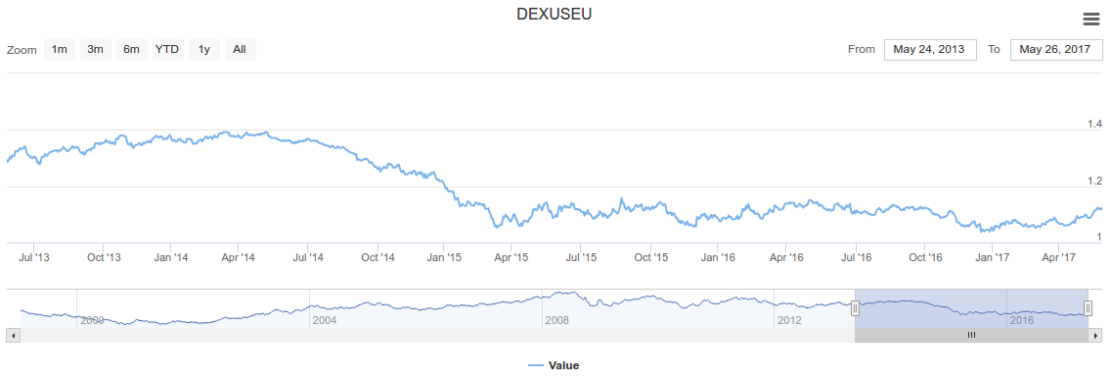


Figure 2: Numerical Visualization Example

4 Conceptual Visualization

4.1 Introduction

This chart trends of individual concepts or of topics (i.e. sets of concepts). For every concept or topic an area is displayed, the time is displayed on the x-axis and the y-axis shows the weight of the object (concept or topic).

There exist following possibilities to interact with the chart:

- By clicking on a time series in the legend below the x-axis one can hide or show it in the chart.
- By moving the mouse over one of the dots in the chart one receives additional information, which includes details on the trend and the value in this time period.
- By clicking on the icon in the top right corner one gets a list of possible ways to export the chart, which include downloading the currently displayed chart as an image (PNG, JPEG or SVG), a PDF document or to directly print the chart.

4.1.1 Requirements

- jQuery v3.0+
- Highcharts v5.0+ (including the "exporting" module and the "drilldown" module)
- Highstock v5.0+

4.2 Input Interface

The series data displayed in the chart needs to be an array of objects resembling the concepts/topics. Each of these objects requires following properties:

name: The name of the object that will be used as its label throughout the chart.

data: This property contains an array of data points resembling the dots in the chart. Every dot needs to come as an array containing two elements.

1. The Unix timestamp of the data point in milliseconds, which gets used as the value for the x-axis.
2. The weight of this object on this specific data point, which will be displayed on the y-axis.

concepts: An array of all the concepts included in this topic. Since only a label of every concept gets used in this chart every concept is only a string value.

In Listing 3 an example of input is presented.

```
1 [
```

```
    {
        'name': 'Topic 1',
        'data': [
            [1483228800000, 30],
            [1483315200000, 10],
            [1483401600000, 15]
        ],
        'concepts': [
            'concept 1', 'concept 2'
        ]
    }, {
        'name': 'Topic 2',
        'data': [
            [1483228800000, 40],
            [1483315200000, 30],
            [1483401600000, 50]
        ],
        'concepts': [
            'concept 1', 'concept 2', 'concept 3'
        ]
    }
]
```

Listing 3: Input Example for Conceptual Visualization

4.3 Example

In Figure 3 an example of conceptual/topical visualization is presented.

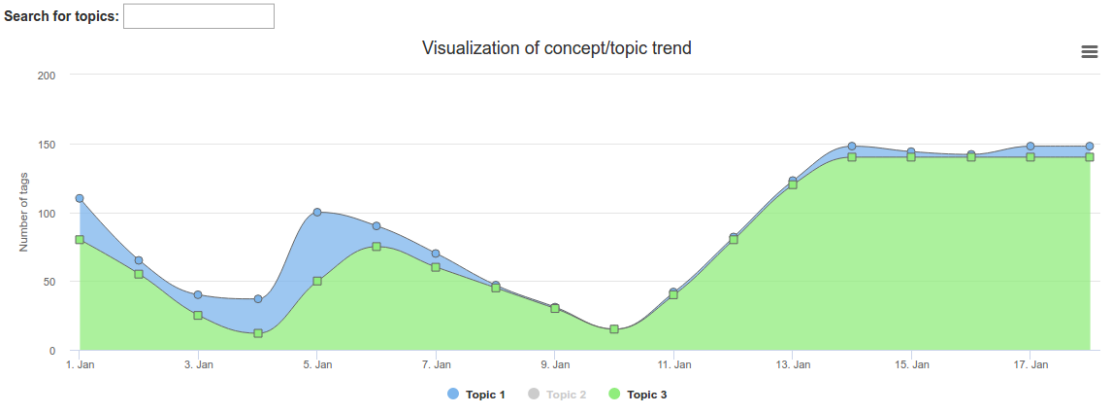


Figure 3: Conceptual Visualization Example

Conclusion

In frames of the current deliverable 3 visualization tools were developed. The development of the tools was agreed with all the interested partners including intermediate demonstrations, therefore the developed set of tools is expected to cover the needs of all partners and will be used in PROFIT platform prototypes later on.