Policy Gadgets Mashing Underlying Group Knowledge in Web 2.0 Media

Deliverable 3.4

PADGETS Fully Integrated Platform

Workpackage:	WP 3 – Design and Implementation of the PADGETS platform
Authors:	Iosif Alvertis , George Gionis (NTUA), Robert Kleinfeld, Hannes Gorges, Lukasz Radziwonowicz (Fraunhofer FOKUS), Michele Punti (WHL), Anna Triantafillou (ATC), Leonidas Kallipolitis (ATC), Luis Benavidis (GOOGLE), Manolis Maragoudakis, Yannis Charalabidis, Euripides Loukis (AEGEAN)
Status:	Final
Due Date:	31/12/2011
Preparation Date:	31/12/2011
Version:	1.00
Classification:	Confidential

Document History

Version	Date	Author (Partner)	Remarks
Draft v0.1	02/12/2011	R. Kleinfeld (Fraunhofeur), Lukasz Radziwonowicz (Fraunhofer FOKUS), Hannes Gorges (Fraunhofer FOKUS)	Document Structure, Contents Draft, Comments and Assignments
Draft v0.2	23/12/2011	I.Alvertis , G.Gionis (NTUA), R. Kleinfeld (Fraunhofer FOKUS) ,H. Gorges (Fraunhofer FOKUS), Lukasz Radziwonowicz (Fraunhofer FOKUS), Michele Punti (WHL), Anna Triantafillou (ATC), Leonidas Kallipolitis (ATC), Luis Benavidis (Google), Manolis Maragoudakis (AEGEAN)	Integrated contributions for deployment and hostig, key features and improvements of PADGETS dashboard and extensions for monitoring and analytics with focus on social media metrics, indicators for awareness, interest and acceptance as well as opinion mining.
Draft v0.5	26/11/2011	All partners	Internal Review
Final v1.0	31/12/2011	V. Diamantopoulou, A.Androutsopoulou, Y. Charalabidis, E. Loukis (AEGEAN)	Final Review

TABLE OF CONTENTS

1 Introduction 4
1.1 Purpose 4
1.2 Structure of the Document 5
2 PADGETS Platform Components 6
3 PADGETS Platform Deployment And Hosting 7
4 PADGETS Dashboard 9
4.1.1 Opinion Mining Module Setup 10
4.1.2 Create Policy Messages 10
4.1.2.1 Text 11
4.1.2.2 Video 11
4.1.2.3 Survey And Polls 12
4.1.2.4 Conclusion 13
4.1.3 Scheduler 14
4.1.4 Evaluate And Take Action On Feedback Stream 15
4.1.5 Notifications 16
4.1.5.1 Publisher And Subscribers 18
4.1.5.2 Notification Format 19
4.1.5.3 Components 21
4.1.5.4 Interfaces 22
4.1.6 Creating And Evaluating Reports 24
4.1.6.1 Social Media Metrics 26
4.1.6.2 Awareness, Interest And Acceptance 27
4.1.6.3 Opinion Mining And Sentiments Analysis 28
5 Conclusion 32
List of Figures 33

1 Introduction

1.1 Purpose

PADGETS policy making and modelling software is a powerful platform (Figure 1) to grow, engage and shape the civil audience across social media networks like Facebook, Blogger, YouTube and Twitter. PADGETS is rather an instrument of designing political processes more transparently and reducing their complexity. It empowers policy makers to promote their policies via social networks. Via reports and analytics policy makers are able to monitor citizens' feedback in a fine granular manner like target groups, time frame, interest, acceptance and awareness. From the citizen perspective end-user are qualified to interact directly with the policy maker and to influence the policy making process.

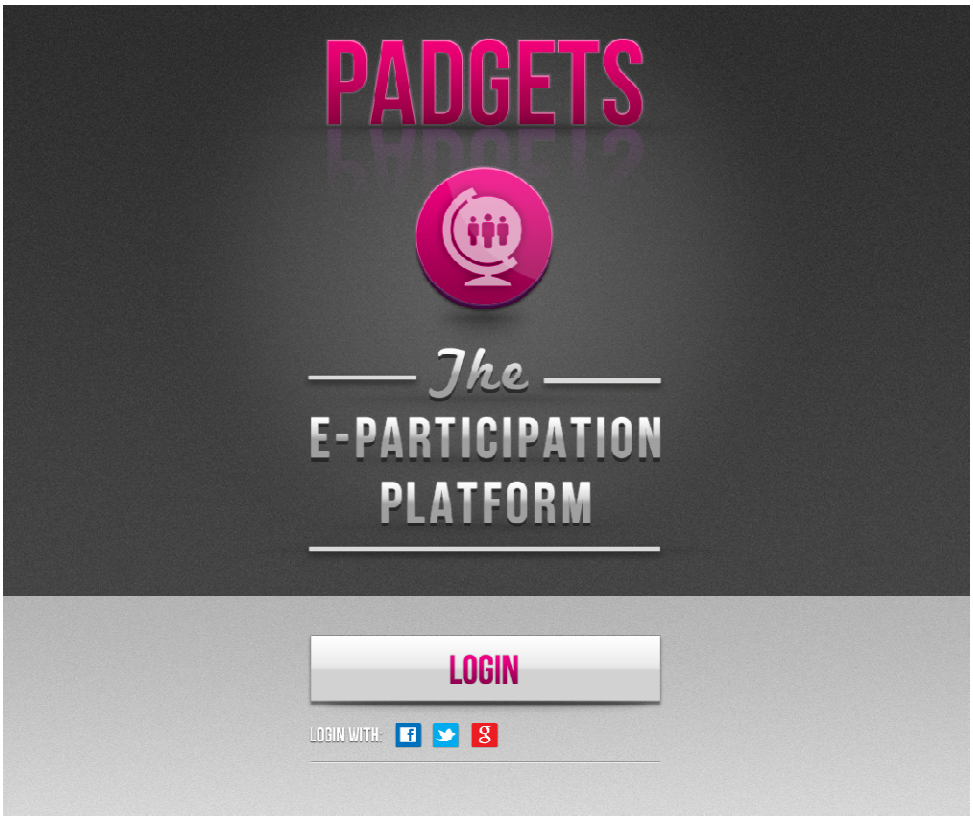

Figure 1: PADGETS Platform

This deliverable addresses the final description of the integrated PADGETS platform prototype. The underlying chapters pursue the following objectives:

- Description of deployment and hosting environment for pilot activities
- Presentation of PADGETS platform suite: campaign manager, scheduler, publisher, monitor and analytics – 3rd party tools like mobile applications are already described in the last deliverable
- Brief overview of PADGETS platform components
- User guideline to perform political opinion-forming and decision-making processes

The PADGETS platform is built up on loosely-coupled web service components which are deployed on the AEGEAN university cloud infrastructure for testing and initiating pilot activities. Core components of the PADGETS platform are the dashboard and the application server. Dashboard represents the graphical user interface for policy makers to initiate, monitor and evaluate policy campaigns via web browser and mobile device. Application server acts as broker between social media networks and dashboard. Processes like social media cross-publishing, tracking, opinion mining, decision support computation and visualization is performed by the application server. PADGETS components are built up on state-of-the-art (web) technologies like REST, JavaScript, CSS, Play framework[1], jQueryUI[2], KnockOutJS[3], MySQL and J2EE. PADGETS system design follows current policies regarding multilingualism, user interface design principles, cross-platform and cross-device support and distributed systems.

1.2 Structure of the Document

The deliverable introduces the final PADGETS platform components and their capabilities. Interactions between components and users will be explained. Based on this PADGETS system design, the following chapter will explain the PADGETS deployment and hosting approach for upcoming pilot activities. The last chapter is dedicated to the PADGETS dashboard suite introducing campaign manager, scheduler, publisher, analytics and monitor. It should be noted that the dashboard will be discussed as policy maker tool. All mentioned statements are related to the integrated PADGETS platform prototype which is deployed on the AEGEAN cloud infrastructure.

[1] http://www.playframework.org/
[2] http://jquery.com/
[3] http://knockoutjs.com/

2 PADGETS Platform Components

The PADGETS platform is an aggregation of different components responsible for different tasks of the project, as described in the DoW. In Figure 2 below all the different parts of the platform are visible with all the connections among them.

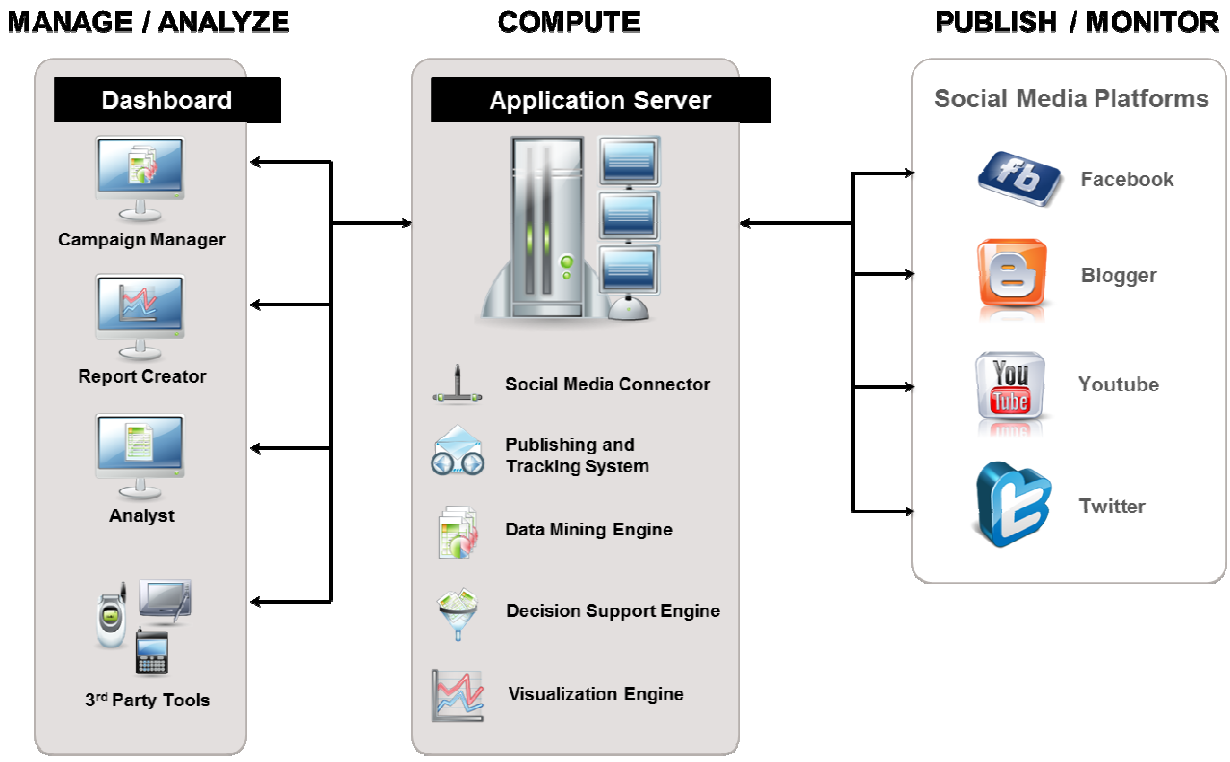

Figure 2: PADGETS Platform Components

In explain, PADGETS consists of the following components:

Dashboard: Dashboard is the web interface of the platform, based on JQuery UI and NockOutJS, where a policy initiator can setup and manage a policy campaign. Evaluation of feedback streams like comments, surveys and polls are supported by monitor and report capabilities. Dashboard has a direct communication with the application server.

3rd Party Tools (Mobile application): Policy initiators may also have access on a campaign through other interfaces, as long as the application server exposes RESTful interfaces. On that phase of the project, an Android application has been developed to support running a campaign via a mobile device. The broad, easy adapted in any technology (e.g. XMPP communication) and the open character of the Android platform have been the main reason for that decision.

Visualisation Engine: Visualisation engine is responsible to export campaign data on the web interface. Google Chart Tools have been used to give a "Google Analytics"-like feeling of mass data visualization. This component communicates through the application server with the Clients, to support the decisions made on a campaign. Visualisation Engine provides social media platform driven metrics, awareness, interest and acceptance of target groups, trend topics and opinions.

Decision Support Engine: Decision Support Engine runs simulations based on data both coming from social media and the data mining engine. Via the application server it has an interface on the client to manage simulations. Results of decision support engine are clustered data sets for awareness, interest and acceptance of citizens and their performed interactions with policy messages.

Data Mining Engine: RapidMiner has been used to extract data from raw social data (e.g. comments, likes, views etc.). It has a bidirectional connection with the Application Server and delivers data also to the decision support engine for further processing.

Application Server: Application server is responsible to manage the communication both with social media and with all different components. It is "heart" of the platform where data are stored and information is rooted on the proper channels. It is connected to every other component inside the system. Applications server provides RESTful interfaces for other components especially the social media metrics API for raw social media data and computed results of data mining engine and decision support engine.

Publishing and Tracking System: XMPP server is responsible to deliver notifications on the clients for any new social activity coming from social media. It has a client-plugin on the application server and another one the dashboard in order to manage real-time communication. Based on XMPP server the application server provides features for cross-publishing of policy messages across social media platforms. Policy messages in form of a Twitter message, Facebook status update or Blog post are published to various social media channels. The application server tracks simultaneously end-user feedback for instance a comment to a Facebook status update.

Social Media Connector: Social Media Connector is the gateway between Application Server and Social media platforms. The connector utilizes abstract API to exchange data between social media platform APIs for publishing and tracking of policy messages as well as extracting raw Social media data. Social media platform APIs are mapped to generic features and categories of the abstract API.

PADGETS platform components follow the concept of policy life cycle. Thus, supported actions include: creation of policy campaign, cross-publishing of policy message, monitoring of end-user feedback, computation of raw social media data and final presentation of reports and analyses. This set of features empowers policy initiators to run all-embracing policy campaigns across social media platforms.

3 PADGETS Platform Deployment and Hosting

PADGETS platform is based on technologies that support a distributed architecture, like web services, JQuery clients and XMPP components. As represented in Figure 3 even the components responsible for data analysis do communicate through RESTful interfaces, thus there is interoperability and lack of dependency concerning the communication format.

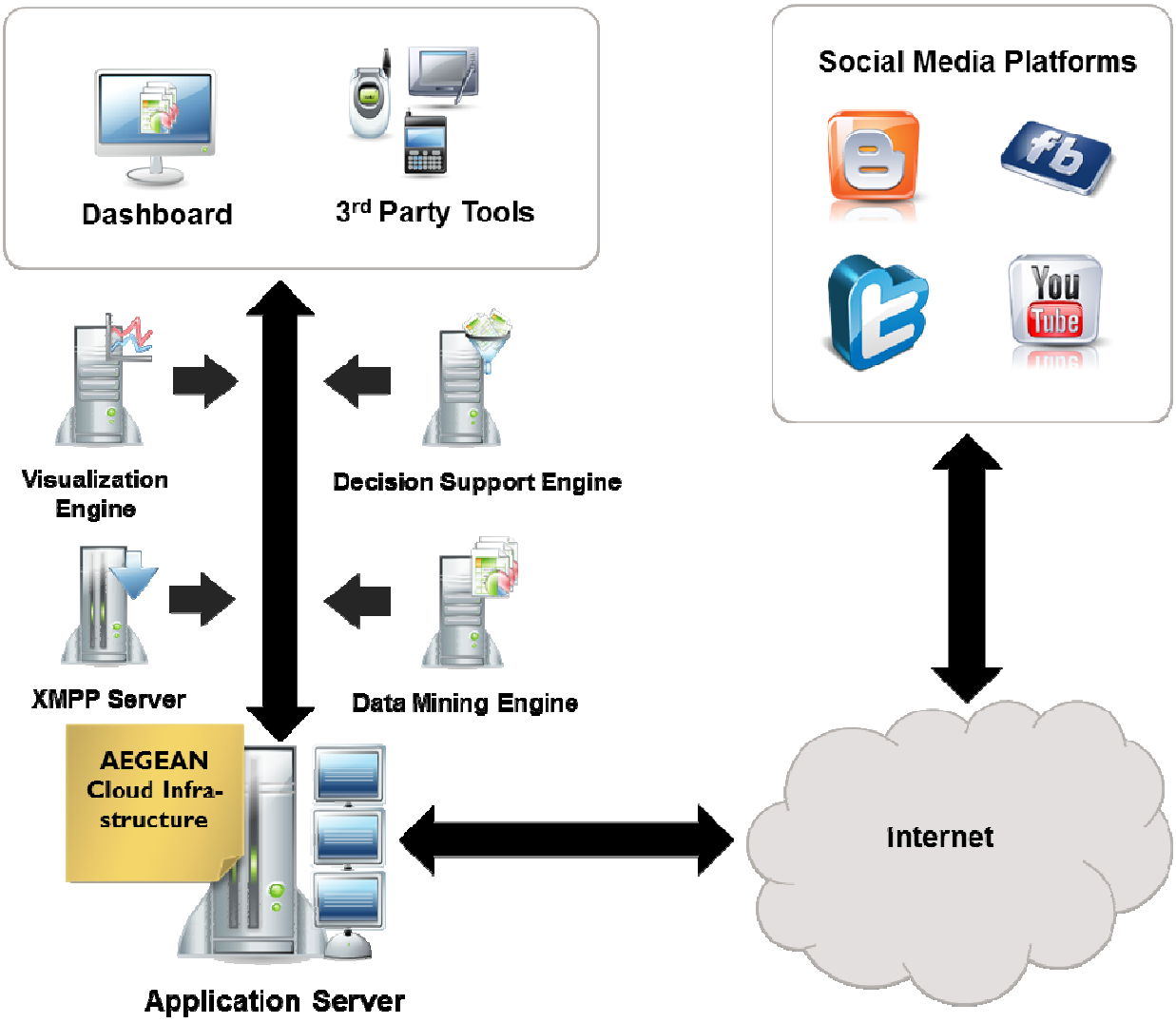

Figure 3: PADGETS Platform Deployment

Nevertheless PADGETS handles sensitive data, even if the majority of them are public (e.g. Twitter messages, Blogger posts or Facebook Pages). For that reason, for the first version of the platform, it has been decided to run every component under the same domain, and setup a localhost communication among the components, except for the interface towards the clients and Social media of course.

The whole PADGETS platform is being deployed on the AEGEAN cloud infrastructure, providing a Windows Server 2008 RC machine for deployment. Inside this server there is the Application Server (Glassfish), Ignite OpenFire XMPP Server, RapidMiner, Decision Support Engine, Apache Web Server and all the other components of the platform.

4 PADGETS Dashboard

PADGETS dashboard facilitates policy makers and their consultants with various features for managing policy campaigns. Figure 4 summarizes the basic features of the PADGETS dashboard: campaign manager, publisher, analytics, monitor, social media platform integration and 3rd party tools support.

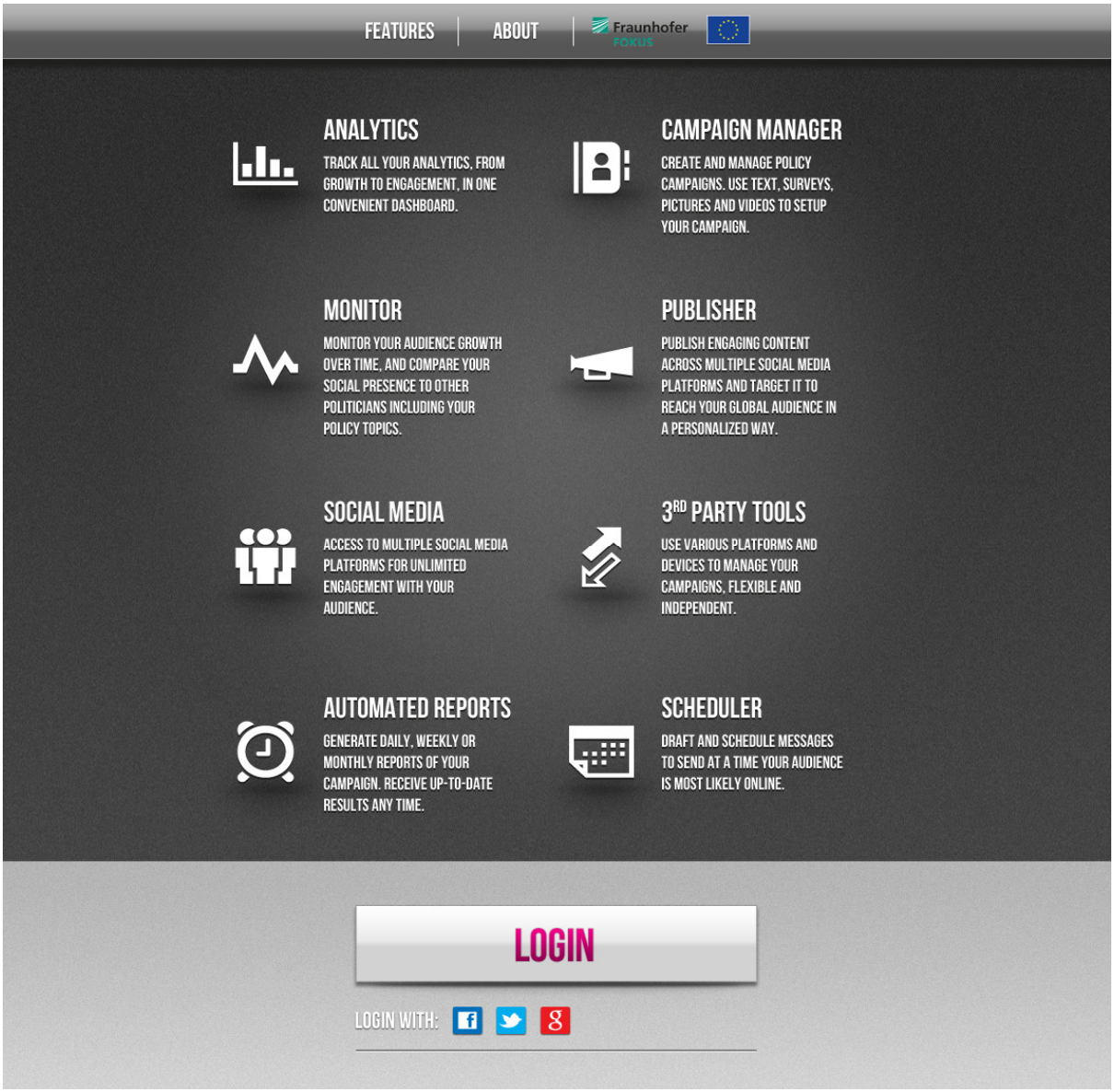

Figure 4: PADGETS Platform Features

With this broad feature set, PADGETS is the most comprehensive policy campaign management platform for politicians, policy makers, consultants and the public sector in general. Through PADGETS this target group is able to:

1. grow politicians audience
2. engage politicians audience

3. drive decisions derived from citizens engagement

In the following explanations we define regulations, policies, draft laws and publications as so called policy message which is published across social media networks. As a result, a policy message can be a blog post or a Facebook status update. In the further policy life cycle of a policy campaign the spreading and end-user interaction with such a policy message is tracked and analysed by data mining and decision support computations. The following subsections will explain these policy life cycle related features of the dashboard.

4.1 Managing Policy Campaigns

Policy campaigns are the heart of PADGETS. Therefore a policy maker and his consultants need a tool-set to manage the campaigns. This task is performed by the campaign manager which is represented in Figure 5.

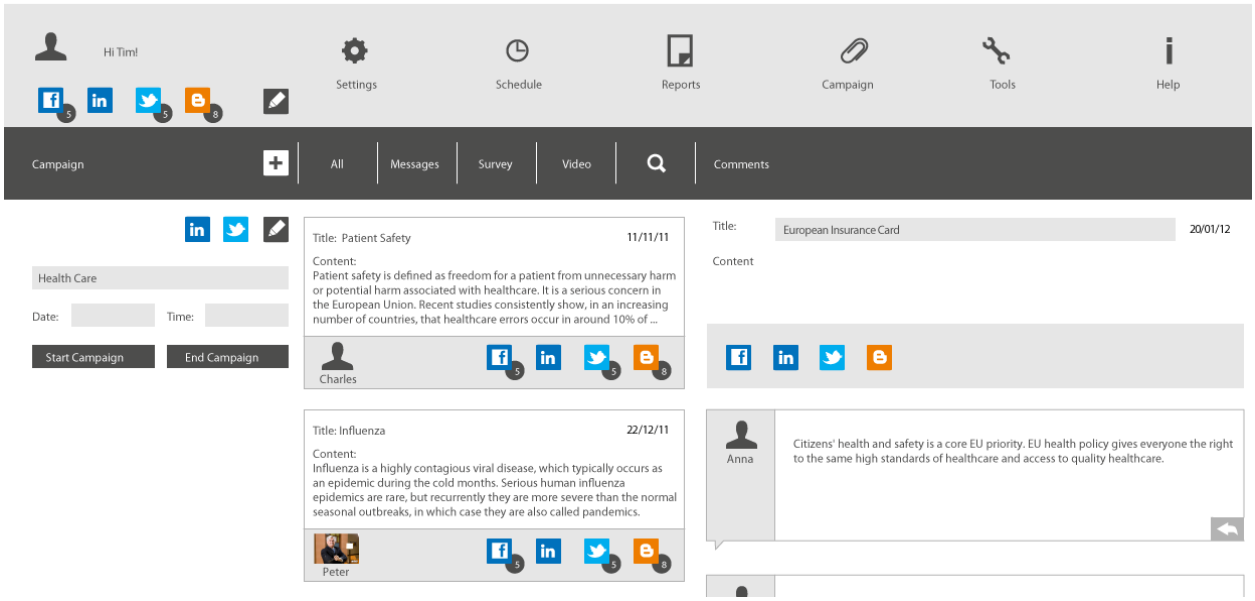

Figure 5: Campaign Manager

The campaign manager is divided into three columns: campaign information (start, end, hashtag, region, target group), policy message (text, image, video, survey and polls) and feedback stream (comments, likes, policy maker's feedback). Following paragraphs will explain the features of the campaign manager.

4.1.1 Opinion Mining Module Setup

A policy maker has the rights to create a new policy campaign. Policy maker chooses first an appropriate name for the campaign. In the next step the dates for the beginning and the ending of the campaign is needed. In the beginning the campaign is not public by default. The start date turns the campaign to public and the first policy messages are published. The policy maker chooses also the main language and a region for the circle of influence of the campaign. Another important parameter is the hashtag which has to be defined. Via hashtag the diffusion of policy campaign related topics can be tracked across Twitter.

4.1.2 Create Policy Messages

Policy maker creates policy messages via the dashboard as illustrated in Figure 6. There are three different kinds of messages: text, (image), video, survey and polls. After creating, the policy maker decides where the message is published in the social media networks. It is possible to publish a policy message in more then one social networks in parallel.

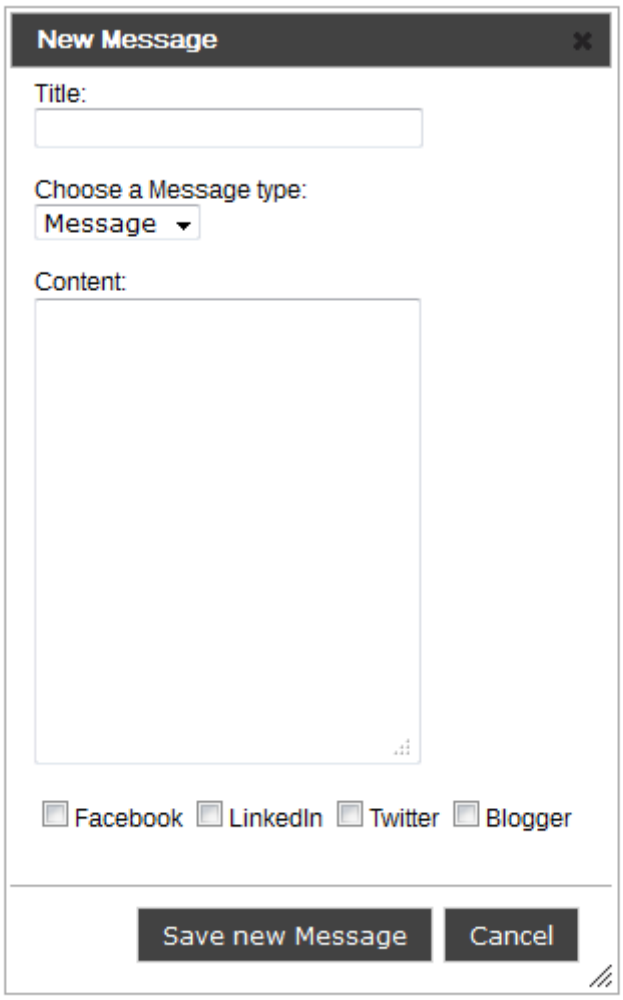

Figure 6: Policy Message Form

4.1.2.1 Text

Text messages are the most common form of policy messages. The policy maker chooses a title for the text message and chooses the target social media networks, where it should be published. Some properties like the length of the text or the usage of a title are dependent from the aspired channel. Twitter for example except only short messages (140 characters) without a title or any media (images, videos) inside. Text messages can be published as Tweet, Facebook status message or Blog post.

4.1.2.2 Video

Beside of text messages it is also possible to publish videos via PADGETS. The policy maker types the link of a clip in the input field and chooses the target social media networks, where the video should be published. It depends on the social media network and the video platform provider (e.g. YouTube) how the video will be showed. Blogger and Facebook for example have a YouTube player inside the posts. Twitter is in lack of such a

feature, therefore PADGETS publishes only the link of the video in a Tweet. After a click of this link, the end-user will be forwarded to the video page of the video platform provider.

4.1.2.3 Survey and Polls

PADGETS dashboard gives the policy maker the opportunity to create simple polls and more complex surveys with more then one question for a campaign. As illustrated in Figure 7 the creator of the survey chooses a title, a start and an end date for the survey.

The order of the questions can explicitly be determined or the questions will be displayed in a random order. Every question comes along with options. The creator chooses the number of response options and how many votes an end-user can give. PADGETS supports different types of questions, which influence the results of a survey. After creating a survey or poll, the creator publishes it on one or more Social media networks.

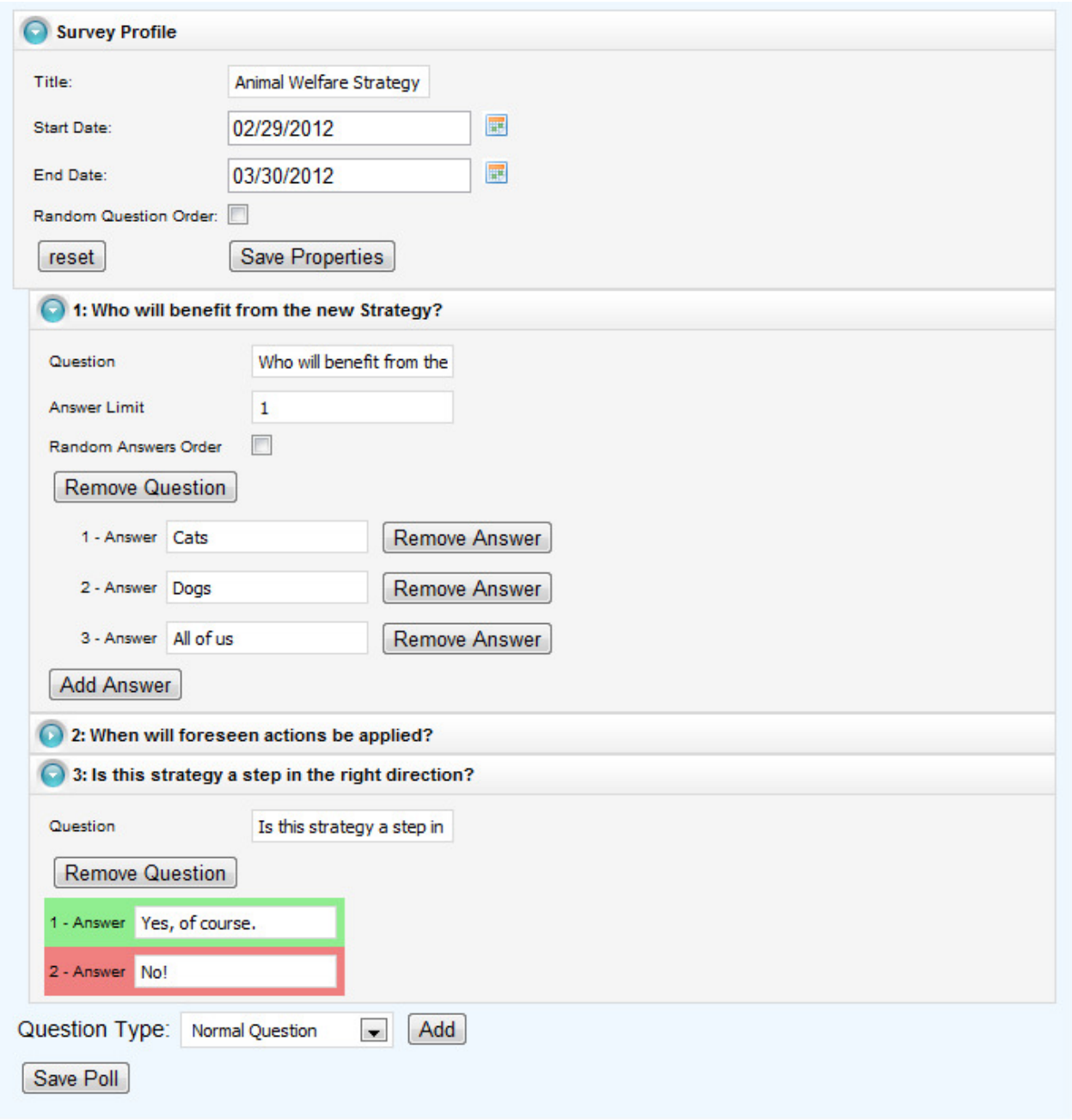

Figure 7: PADGETS Surveys and Polls

4.1.2.4 Conclusion

Summarizing published policy messages appear sequentially in the second column of the dashboard (Figure 8).

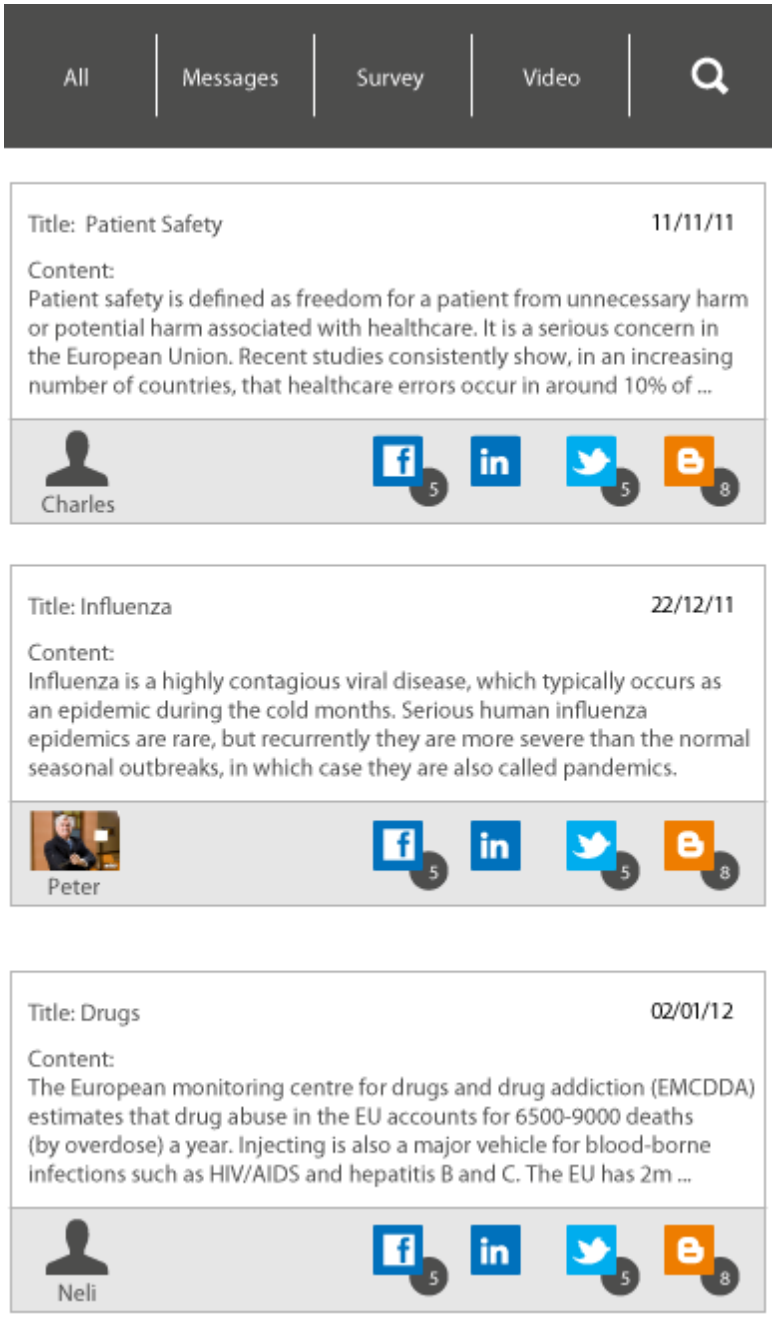

Figure 8: Policy Message

Messages can be searched and filtered by content, type and social media network. Furthermore, each policy message is marked with a feedback notification for each social media network. The dashboard supports seamless updates of a policy message by typing directly in the input form.

4.1.3 Scheduler

It is not necessary to create and publish policy messages instantly. Via the scheduler, a policy maker has the ability to plan when policy messages should be published. The message creator chooses a date in the future;

PADGETS stores the policy message until this date and publish it autonomous across social media networks when this date comes.

4.1.4 Evaluate and Take Action on Feedback Stream

The third column of PADGETS dashboard (Figure 9) addresses feedback streams of end-users. Feedback streams are mainly comments which are related to a Blog post or any status message.

These feedbacks are visualized sequentially as stream. Filters can be defined by social media network. Based on the XMPP server as part of the tracking engine, feedbacks will be displayed in real-time after an end-user provides interaction on a Social media platform. The most important feature is the feedback button for policy makers. Each feedback offers the opportunity to take action as reply to a comment. Policy maker presses the reply button and can formulate a statement to the comment of an end-user. This feature empowers the chance that policy makers can interact directly with their target group in one common tool without managing various social media network accounts.

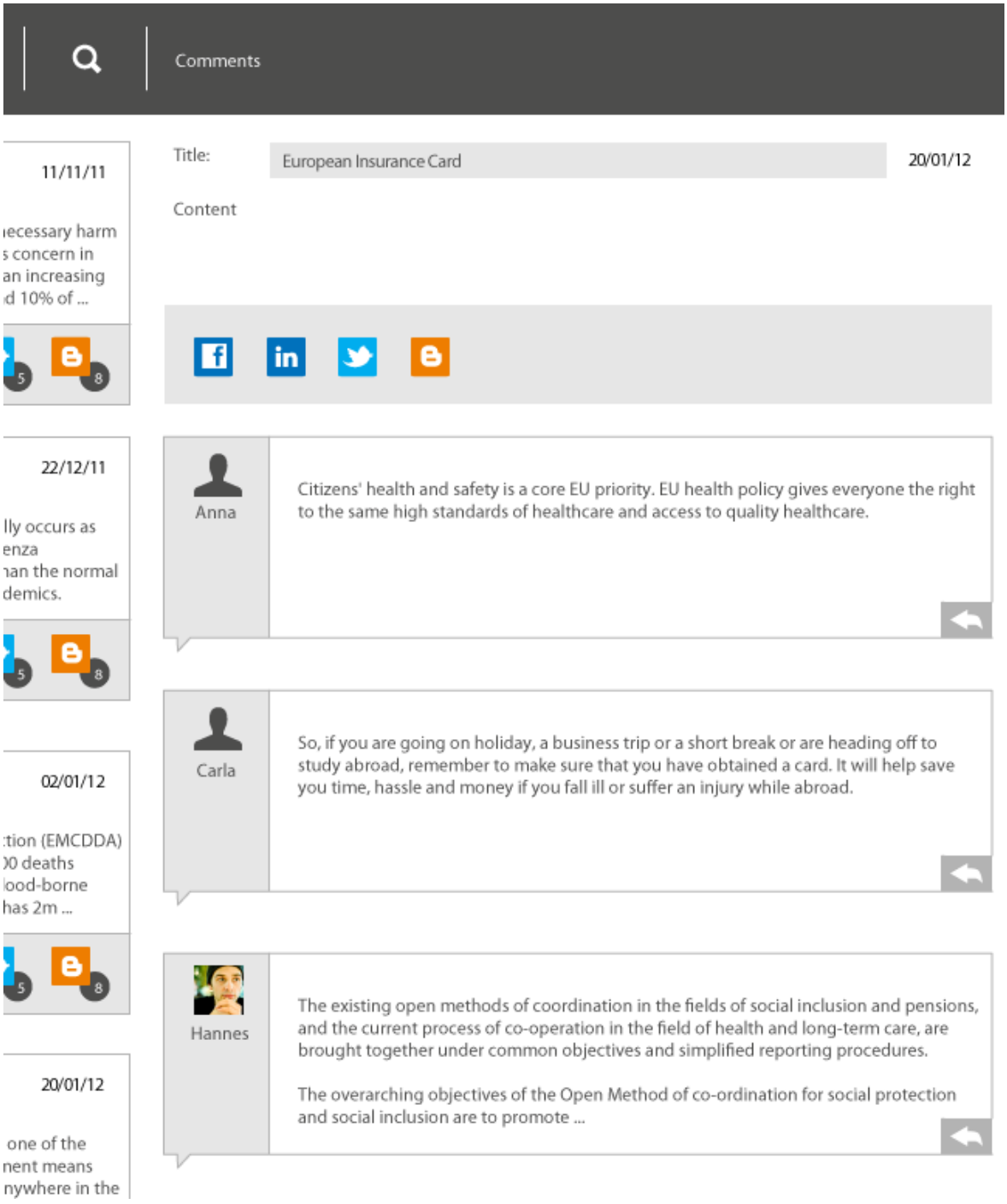

Figure 9: Feedback Stream

4.1.5 Notifications

PADGETS dashboard will be responsible to integrate different social media network accounts for a campaign initiator; thus for accounts characterised by high traffic it is very important to receive notification for any new social activity from the network of the policy maker (e.g. comments, likes etc.). Thus, real-time notifications are valuable to support a social dialog with citizens around social media.

For PADGETS the notification system as part of the tracking engine is based on two core concepts: (a) the XMPP protocol to deliver notifications and (b) Activity Streams format for the data of the notifications.

(a) XMPP is a real-time communication protocol and it is ideal to deliver messages as soon as they are generated, from the server to the client. To the extent, that PADGETS focuses in sending messages to platforms (HTPP POST messages), XMPP is not necessary on the publishing side. On the tracking side now, as long as the social media connectors are polling (with HTTP) on a specific time to get new activities, PADGETS database can be updated as soon as new feed is available. On the contrary, this new social activity cannot be delivered real-time on the proper client (the one currently used) unless the XMPP protocol is used; at least until HTML5 protocol is standardized and supported from browsers, something expected for year 2021. This is the main reason that the XMPP protocol, with a great supporting community, has been selected.

Additionally, many platforms including Twitter and Facebook have started supporting some real-time APIs, of little functionality at the moment. As soon as there are vastly supported, the tracking engine of PADGETS will be able to deliver activity on the client as soon as it is generated from the citizen on the platform. The distributed architecture of XMPP allows PADGETS to be independent from any change on the API, and no further change on the notification system will be needed in that case.

(b) Activity Streams is a data format that helps developers to describe objects and activities in social media with a common structure and vocabulary. Even Facebook's Graph API used terms, words and concepts from Activity Streams in order to describe their Open Graph API.

In other words, Activity Streams is a promising format to follow, it includes even some light semantics (namespaces) and helps not lose any information with its Schema, if an application copes with social media concepts. With a future perspective, Activity Streams is the only data format yet that moves towards an interoperable social web (for social data specifically).

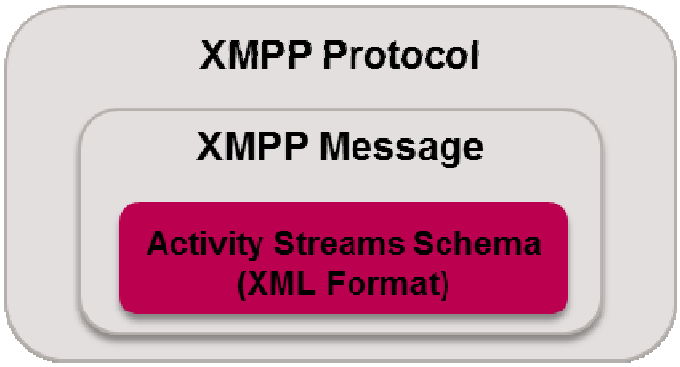

Figure 10: Activity Streams Schema

But, where XMPP is absolutely required is on the notification side of PADGETS. The clients of the architecture cannot be aware of a new message as soon as it is available, because the concept of HTTP is that the client polls the server. Only mid-layer approaches can send notifications to users, but still we can talk for heavy traffic, polling and not that real-time communication (add 15-20 minutes additionally to the time that takes for the social media connectors to track content). In other words, when new feed is available, clients will be informed without needing to poll or refresh (Ajax solution on client side to establish a light XMPP connection).

Queuing and subscribing is also supported through XMPP, very useful feature for receiving notifications after logging in.

To sum up, the XMPP server goes just after the database of the application server (see diagram below). When a new activity is tracked from social media connectors the database is updated and this activity triggers an XMPP message towards the XMPP PubSub service. The PubSub service is running on the server, and can be extended with a XMPP component. Then, XMPP clients, who have subscribed after publishing on specific campaigns, get notifications for feed on a campaign they have subscribed.

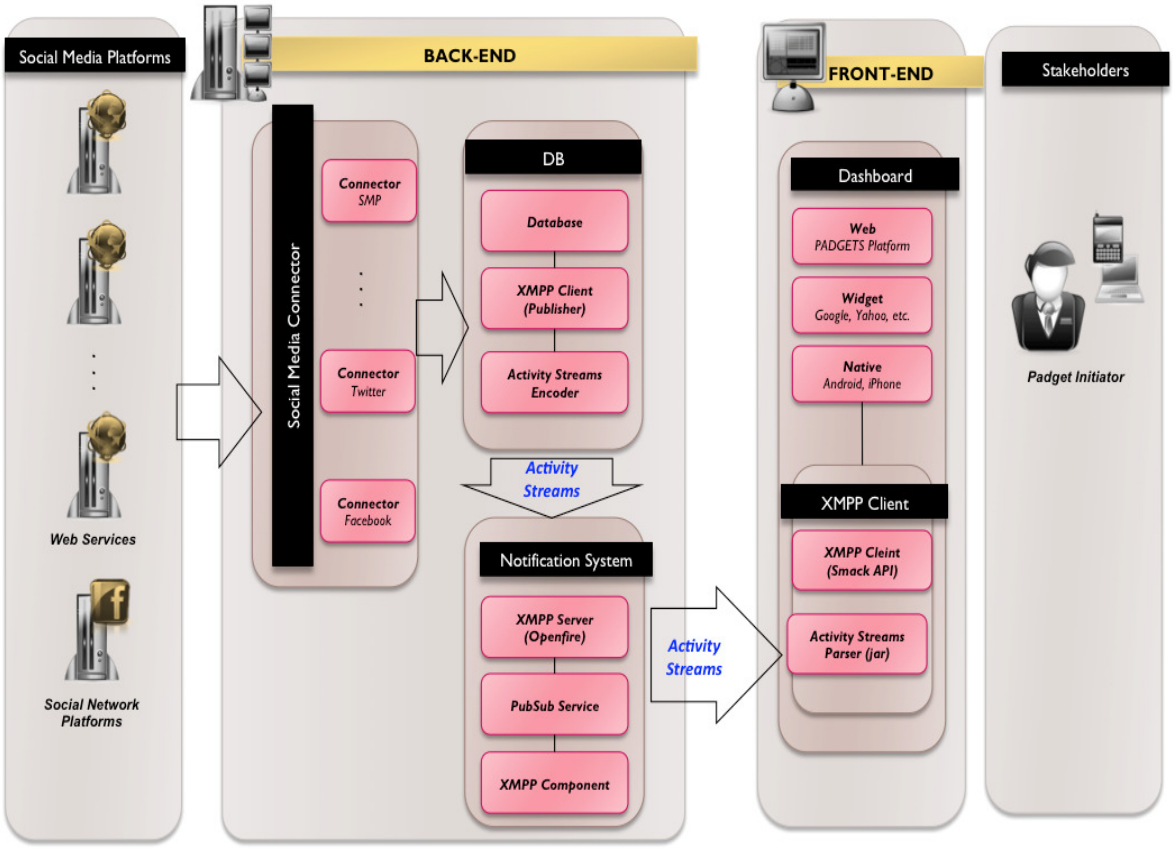

Figure 11: Activity Streams Architecture

4.1.5.1 Publisher and Subscribers

For every campaign that is being created inside the PADGETS platform, a new campaign node is going to be created automatically so that all updates of this can be posted there. In a campaign like this the administrators can register authorized publishers. Those publishers can publish to the specific nodes and the message is automatically distributed to the entire list of subscribers.

Subscribers can either find a node to subscribe on or the administrator itself can subscribe individuals or a group of individuals directly. The procedure to succeed this is rather simple through the XMPP-component API with a complete interface of commands.

The systems works with every XMPP client (web, desktop, mobile, etc) and the authorization depends on the server only. This way every client (administrator, publisher, subscriber) can complete the corresponding task securely, without the need of a specific interface.

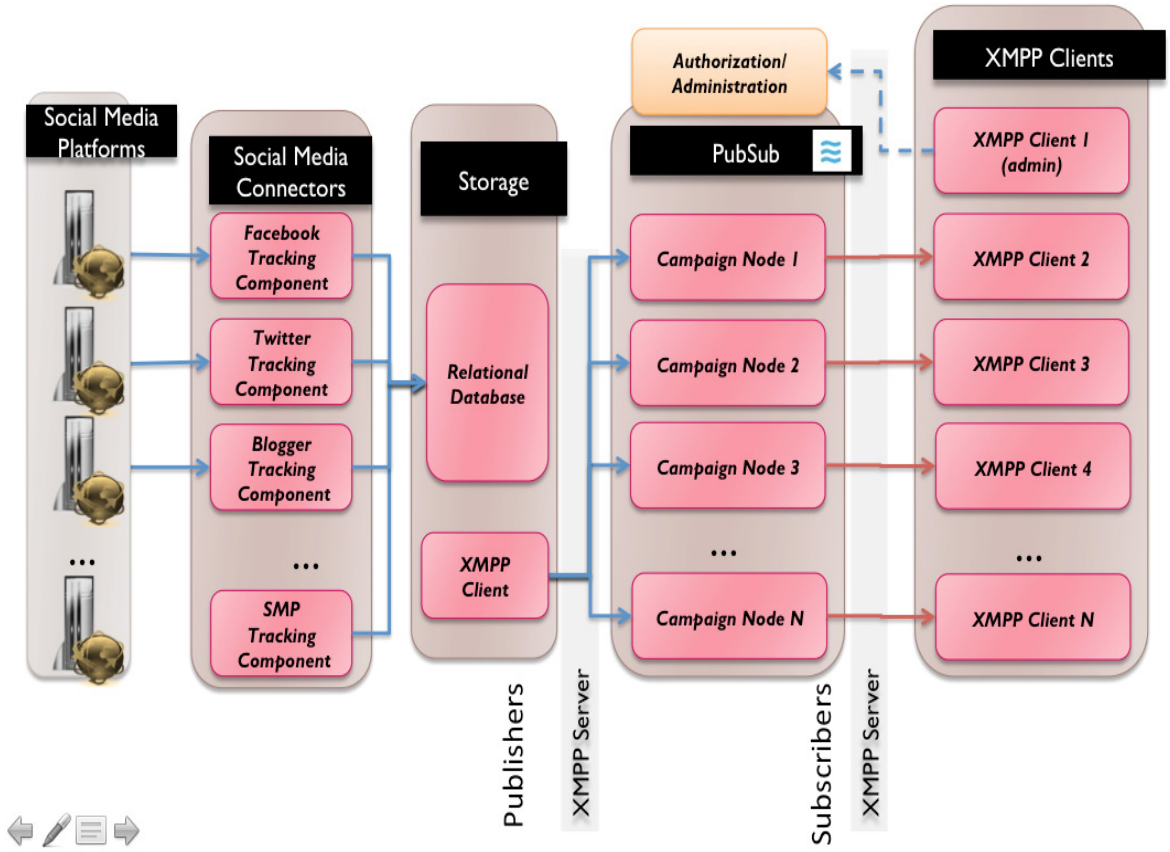

Figure 12: Publisher and Subscribers Architecture

4.1.5.2 Notification Format

An example notification is shown below. This is formatted in the activity streams style and it is being delivered through XMPP between the components.

```
<entry xmlns:thr="http://purl.org/syndication/thread/1.0"
xmlns="http://www.w3.org/2005/Atom"
xmlns:activity="http://activitystrea.ms/spec/1.0/"
xmlns:padgets="http://padgets.eu/spec/1.0/"
xmlns:v="http://www.w3.org/TR/vcard-rdf/">
<id>idofEntry</id>
<title>NameofEntry</title>
<published>17/6/2011</published>
          <author>
             <name>Username</name>
             <uri>http://example.com/username</uri>
             <activity:object-type>person</activity:object-type>
             <link rel="alternate" type="text/html"
href="http://example.com/username"/>
         </author>
<activity:object>
      <activity:object-type>comment</activity:object-type>
      <content type="text/html">
                     <div>This is the content of the message</div>
               </content>
      <id>statusID:221212</id>
      <link rel="alternate" type="text/html" href="http://commentURL.com"/>
</activity:object>
<activity:verb>post</activity:verb>
<activity:target>
      <activity:object-type>status</activity:object-type>
      <link rel="alternate" type="text/html" href="http://statusURL.com"/>
</activity:target>
<padgets:campaign id="idCampaign" name="campaignName"/>
<padgets:platforms>
      <padgets:platform>
            <name>facebook</name>
            <padgets:page id="page1ID" name="My Page name"/>
                     <link rel="alternate" type="text/html"
href="http://profileURL.com"/>
      </padgets:platform>
              <!--encoder enables more platform tags, but for notification
only feed from one platform will arrive at time-->
      </padgets:platforms>
</entry>
```

Layout

The notification format is in the specific way:

```
@Username[link] verb @Object [link] @Account[link]
[content]
#Time
```

example:
Anna posted a **comment** on **Facebook**
PADGETS is a really interesting project
18:23 Wednesday 14 January 2012

Below there is a mock-up of how the dashboard notification system is designed

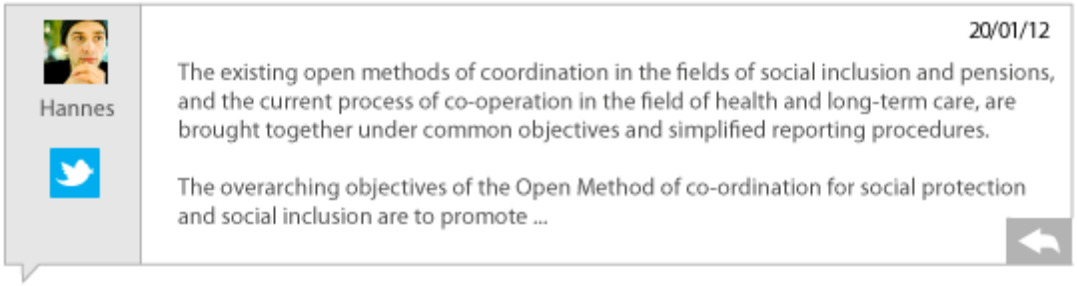

Figure 13: Dashboard Notification System

4.1.5.3 Components

The notification system consists of three main subsystems:

- **The XMPP Component**

The XMPP component is the most important part of the whole notification system. It works localhost with the openfire server for security reasons, but it can be easily extensible for scalability reasons and work in many distributed machines and servers. Its main task is to create and manage the campaign nodes, publishers, subscribers and all the logic for the layer of notification system.

- **Activity Streams Encoder/Decoder**

This specific component is responsible to create the activity stream format messages that are necessary, for the various components to communicate. The encoder/decoder is written completely in python, but it also supports Java APIs through the Jython project. Jython programs can import and use any Java class. Except for some standard modules, Jython programs use Java classes instead of Python modules. Jython includes almost all of the modules in the standard Python programming language distribution, lacking only some of the modules implemented originally in C. Jython compiles to Java bytecode (intermediate language) either on demand or statically.

- **Bosh Client**

The bosh client is responsible for the dashboard real-time communication, implementing the Bosh protocol. Bidirectional-streams Over Synchronous HTTP (BOSH) is a transport protocol that emulates a bidirectional stream between two entities (such as a client and a server) by using multiple synchronous HTTP request/response pairs without requiring the use of polling or asynchronous chunking. It is a draft standard of the XMPP Standards Foundation.

The related standard XMPP Over BOSH defines how BOSH may be used to transport XMPP stanzas. The result is an HTTP binding for XMPP communications that is intended to be used in situations where a device or client is unable to maintain a long-lived TCP connection to an XMPP server. To implement this architecture it was perquisite to first install an apache server.

Apache needs some special configuration with the Openfire-Xmpp server to allow the http-bind (BOSH), because it needs to load specific modules and later to set up a new proxy.

4.1.5.4 Interfaces

The interface of the system consists of three main categories:

The interface of the publisher is as follows:

Publisher connects to Publish-Subscribe	***connect***
Publisher creates a publishing node (i.e. node3). Automatically a new node (i.e. node4) is generated for feedback where publisher also subscribes to.	*create new node:node3*
Publisher posts a new message on a node (existing by the admin or publisher)	*publish:node3 <text to be published>*
Component publishes on PubSub a new tracked interaction (XMPP <publish> message)	
Publisher disconnects from Publish-Subscribe	*disconnect*

The interface of the subscriber is as follows:

Subscriber searches for available nodes to subscribe	***search***
Subscriber connects to a node to receive all the new updates	*connect node*

The interface of the administrator is as follows:

Admin connects to Publish-Subscribe	***connect***
Admin creates a publishing node	*create new campaign node:node1*
Admin creates a tracking node (The Social Media connectors are set publishers)	*create new tracking node:node2*
Admin allows a user to publish on a node	*add publisher:node1 <JID of publisher>*
Admin allows the publisher to be a subscriber (generally)	*add sub to whitelist:<JID of publisher>*
Admin allows publisher of node1 to subscribe node 2	*subscribe:node2 <JID of publisher>*
Admin disconnects from Publish-Subscribe	*disconnect*

Below, a more detailed description of some of the capabilities of the system is provided.

1. Campaign Initiator registers PADGETS ⇒ an XMPP account (JID and Password) is created on the XMPP server

    - The user is capable to connect with real-time interfaces on PADGETS

    Implementation: XMPP supports specific method [XEP-0133] to create and edit a user http://xmpp.org/extensions/xep-0133.html#add-user. Nevertheless, there is additionally a plugin available on XMPP server (User Service) that allows direct HTTP requests to the server to manage users http://www.igniterealtime.org/projects/openfire/plugins.jsp

2. Campaign Manager creates a new campaign in PADGETS ⇒ XMPP Server creates a corresponding campaign node.

    - Any Client Software (i.e. the PADGETS mobile app or a 3rd party website that re-publishes data streams of the campaign) can subscribes to the campaign node in the XMPP Server to receive real-time notifications.

    Implementation: the XMPP client who is an authorized publisher connects to the pubsub component with a message, and sends it a message of the format:
    create new node: [nodeID]

    Then it may disconnect, even if it is not necessary if it is expected high traffic, as XMPP is suitable for long connections (mainly should disconnect for development and maintenance purposes). The command is the following:
    `disconnect`

3. Campaign Owner publishes a new message to a campaign in PADGETS (this is done through Application Server) ⇒ Application Server publishes the new message to the XMPP Server's campaign node

    - All Clients that have subscribed (and been authorized by the campaign manager) to the campaign node get notified in real-time of the newly posted message – no polling the Application Server or the Social Media Platforms is needed by the Clients to get the update.

    Implementation: the XMPP client, who is an authorized publisher connects to the PubSub component, send a message to publish on a new campaign, with the following message:
    `publish:[nodeID] <text to be published>`

4. Application Server retrieves a user comment made on a campaign message from a social media platform ⇒ Application Server publishes the retrieved comment to the XMPP Server's campaign node

    - All Clients that have subscribed (and been authorized by the campaign manager) to the campaign node get notified in real-time of the newly retrieved comment.

    Implementation:
    `subscribe:[nodeID] <JID of publisher>`

    - The PADGETS Front End is notified real-time about the newly retrieved comment.
    - Again no polling the Application Server or the Social Media Platforms is needed by the Clients to get the update.

4.1.6 Creating and Evaluating Reports

PADGETS supports policy initiators by tracking all analytics from growth to engagement in one convenient dashboard. Reports can be generated by the dashboard for social media metrics, indicators like awareness, interest and acceptance as well as opinion analysis for end-user feedbacks. Figure 14 illustrates the reports section with social media metrics for trend topics, interactions by region and top referrers.

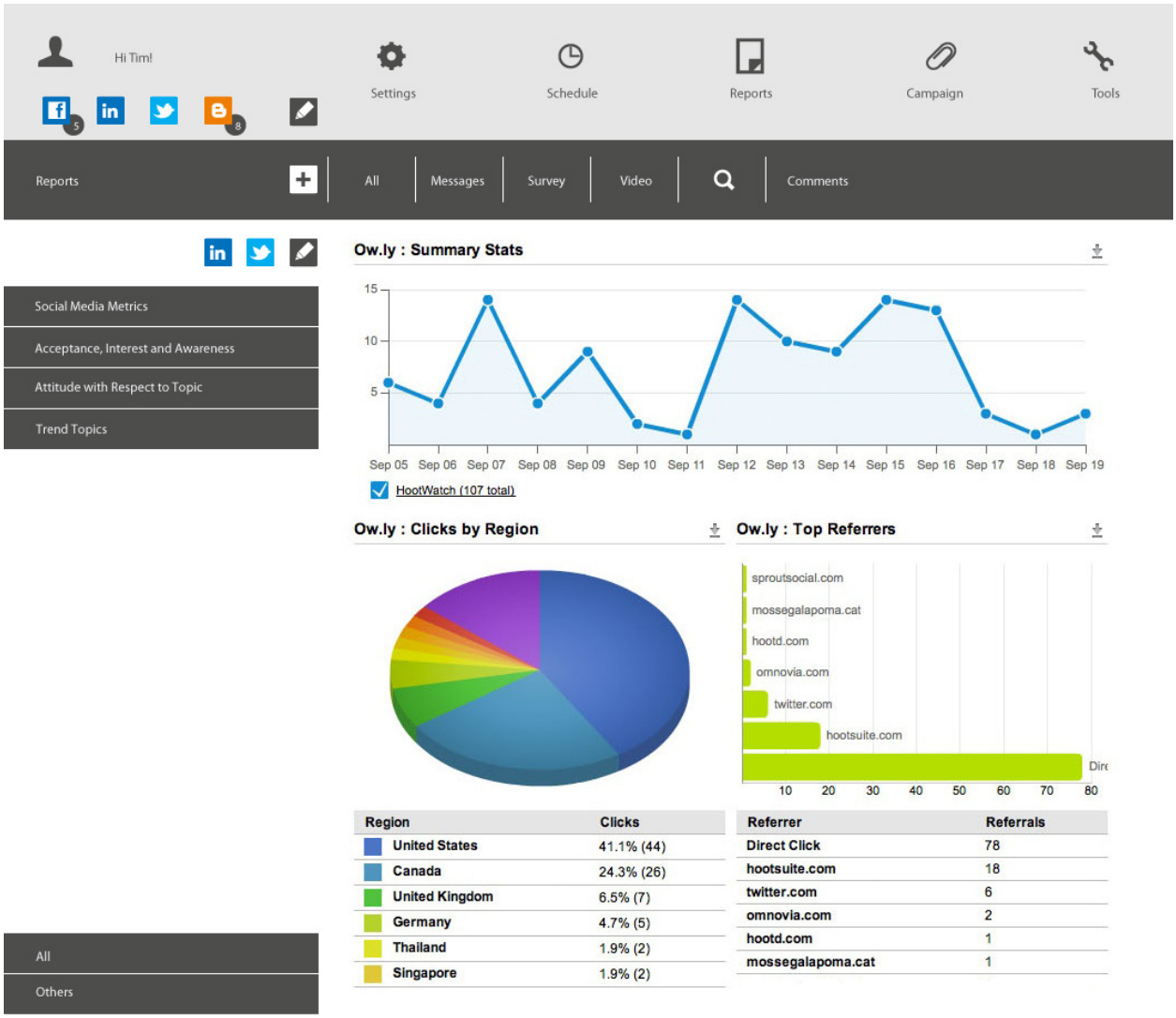

Figure 14: PADGETS Dashboard Reports

PADGETS delivers following features in the analytics module of the dashboard:

- Full integration of Facebook Insights
- Presentation of metrics on campaigns which are currently running
- Presentation of fan and follower growth over time
- Support the daily performance of campaigns over time
- Provide engagement data on each policy message made to a social media network
- Request entrant demographics by surveys and polls
- Measure end-user engagement by indicators like interest, acceptance and awareness
- Track topic distribution by hashtags
- Presentation of opinions for policy message related comments

The next sections will explain in detail social media metrics, indicators for awareness, interest and acceptance as well as opinion analysis.

4.1.6.1 Social Media Metrics

PADGETS requests and processes raw social media data from social media platform APIs like Facebook, Blogger, Twitter and Youtube. From these data sets, following social media metrics are deduced:

- Number of reaches per social media platform (SMP)
- Number of view per SMP
- Number of replies per SMP
- Number of shares per SMP
- Number of posts per SMP
- Number of likes / dislikes SMP
- Number of comments per SMP
- Number of tweets
- Results of opinion polls
- Number of followers/friends per SMP
- Number of daily posts
- Number of daily tweets

These metrics are mostly based on conversational data with the objective to have a clear statement regarding user engagement in social media platforms related to a policy campaign. It is to be noted that not every social media platform API supports the just mentioned metrics. These metrics are visualized as reports based on the before mentioned concepts of the last deliverable.

4.1.6.2 Awareness, Interest and Acceptance

The following section provides an overview of the approach we propose to adopt for the measurement of each of the indexes the DSC is supposed to generate (acceptance, interest and awareness).

Acceptance

The acceptance index is the least impacted by the privacy policies of social media platform. As a matter of fact, such index is calculated starting from a specific question asked via questionnaire resident in a PADGETS environment not influenced by norms imposed by each social media platform. It will thus be possible to add two further questions to the questionnaire requiring the user to state his/her age and gender.

The DSC will thus be in the position of provide:

- Current level of acceptance index (aggregated and broken down by age & gender).
- Resampled level of acceptance index.
- Projected level of acceptance index.

Interest

The interest index tries to estimate the number and the characteristics of the people who, by interacting with the policy message, contribute to its viral diffusion across social media platforms.

Starting with the data coming from Facebook it will be possible to create two clusters (Man and Women) based on gender containing a significant amount of the total interactions (70-80%). This is possible since Facebook allows retrieving users' gender via API. Once we have a first understanding of how users are distributed between the two genders, we consider all data coming from the other three platforms (about 20-30% of total) as missing values (as far as gender is concerned) that need to be reclassified. The reclassification is made using the distribution emerged by the analysis of Facebook data.

Once the two clusters M & W are filled with all the interactions occurred over the four social media platforms under consideration, the break down by age ranges will be carried out using information provided on a daily basis by Facebook insights (insights/page_active_users_gender_age/day) for the age distribution (given the gender) of users that have interacted in an active way with the Fan Page. At the end of the process it will be possible to generate a total of 12 clusters (6M + 6W). The variability of the numerosity in these clusters will be used by the system dynamics model to predict the evolution in the near future of the interest index level. Having fixed age ranges (i.e., those imposed by Facebook) will also facilitate the resampling of the index in order to improve its coherence with the actual distribution of the population resident in the targeted territory.

In so doing the DSC will be able to provide all the metrics that were introduced:

1. Current level of interest index (with gender & age breakdown).

2. Resampled level of interest index.

3. Projected level of interest index.

Awareness

The awareness index is the most impacted by the lack of disaggregated data. One palliative strategy we propose to adopt for estimating the awareness index takes advantage of the complementary role that the YouTube platform may play. We in fact address to insert links to YouTube videos in Twitter, for which no tools (neither native nor provided by third parties) exist in order to measure impressions. The data collected will represent an underestimation of the awareness since it will measure clickthroughs in lieu of impressions. Regarding Blogger domains, Google Analytics functionalities will be used for collecting information pertaining to visualizations et similia. In order to refine the estimation, before-mentioned views will be complemented with data provided by Facebook on Page Stream views per day (insights/page_stream_views_unique/day).

The paucity of information on the awareness index will not allow to use the system dynamics simulation model to forecast the evolution of the index levels over time. Such functionality may be substituted by a more traditional analytical method aimed at forecasting time series (i.e., regression models).

4.1.6.3 Opinion Mining and Sentiments Analysis

The Opinion Mining component is the result of a synergy between the data mining core module and a powerful ETL (Extract, Transform, Load), data analysis and predictive reporting engine. The former is trained from past user comments and uses sophisticated Machine Learning algorithms for feature selection and classification such as Support Vector Machines, Principal Component Analysis and Naive Bayes, while the latter is utilized based on RapidAnalytics, which can run upon existing, standard application servers. RapidAnalytics is able to transform the opinion mining tasks into web services, which can be accessed either through the Application Server or separately through a Web Interface in order to return information to users in a machine readable format such as XML or JSON and even produce reports using web visualization techniques such as Adobe Flash.

A graphical depiction of the Opinion Mining implementation framework is illustrated below:

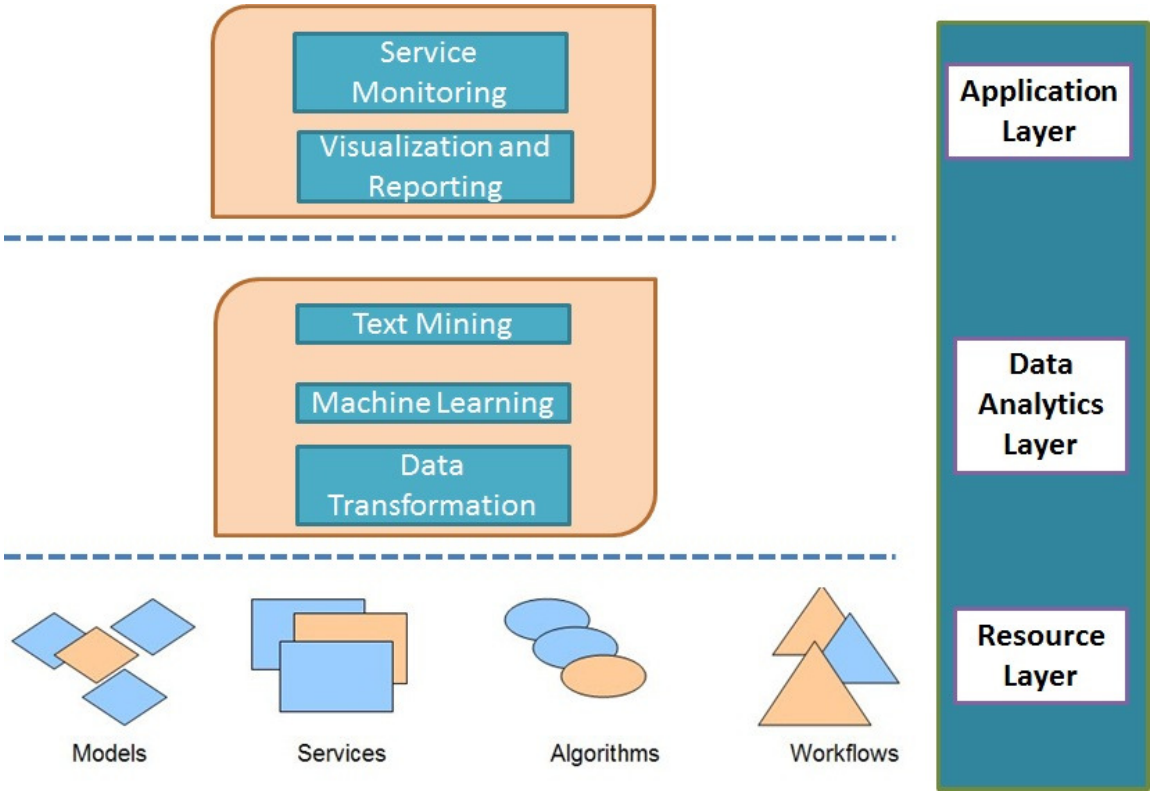

Figure 15: Opinion Mining implementation framework

There are three main layers, the lower one dedicated to built-in models for sentiment classification, services that extract outcomes from the previous task into machine readable information, algorithms and work flows that deal with text mining of comments of different languages. All of the above communicate through RapidMiner and Java with the Data Analytics layer, which contains the processes that either train or perform classification of sentiments. Finally, there is the application layer, in which policy makers can supervise the analytics outcome through visualization techniques

Capabilities of the Opinion Mining Component

The basic OM framework allows administrators to connect a MySQL database into it in order to retrieve a massive amount of comments from various social media.

Upon retrieval, administrators can proceed to either

- the training stage for a user's language or
- the extraction of the sentiment of a single comment or a pool of comments.

As regards to the former, we have implemented a hybrid system, based on RapidMiner and Java EE, in which the administrator manually annotates the sentiment (e.g. positive or negative) of a comment, provides an additional list of trivial words (it is called a "stopword list") for each language and then uses the built in PMML (Predictive Modeling Markup Language) with Java to create a prediction model.

Concerning the latter, our implementation uses again a PMML process with RapidAnalytics, in which the JSON format is selected as output. Upon requesting an Opinion Mining task to the Server, a JSON file will be broadcasted containing a special attribute named as “prediction” which contains the label of the sentiment the system considers as most probable, given an unknown comment. Since it is more useful to classify a set of comments (e.g. arriving to the Application Server on a per day basis) the RapidAnalytics also produces a collection of prediction, which can either be visualized internally through its own visualization capabilities or forwarded to the visualization component of the platform.

It is noteworthy to state that the web service is also connected to a scheduled analysis and reporting module that is able to be programmed to provide all of the desired tasks on a scheduled basis. The scheduler is implemented internally within RapidAnalytics and is able to connect with RapidMiner processes store locally on the Opinion Mining Server.

The following sample screenshot depict the opinion mining classification process, developed in PMML/XML using RapidMiner and Java as well as the service-oriented programmable capabilities of the previous process using RapidAnalytics:

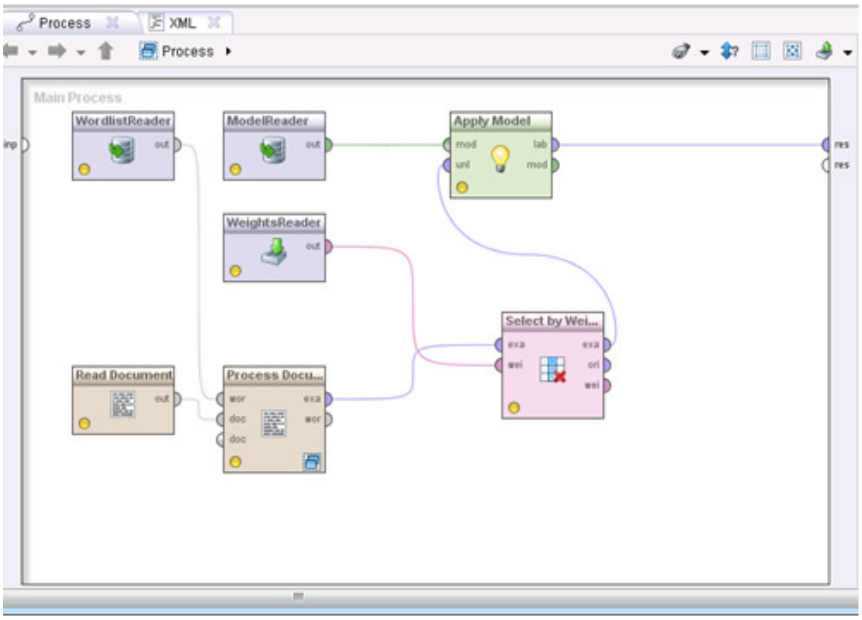

Figure 16: Opinion Mining classification process

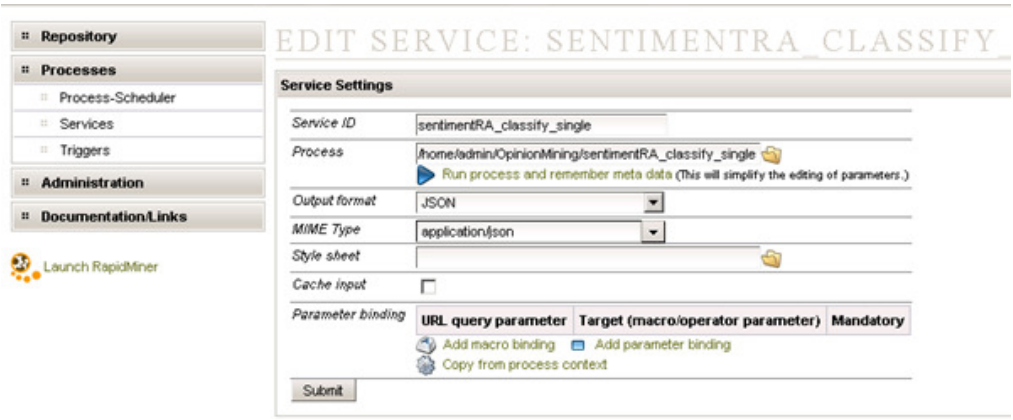

Figure 17: RapidAnalytics service exposure

A useful addition to the OM web service, is the ability for policy makes to visualize not only the sentiment of comments but also which terms are mostly associated with positive or negative sentiments. This is again implemented as a pattern recognition task within RapidMiner and the exported to policy makers as a web service through the RapidAnalytics Engine.

Our general approach in the reports we are going to design in each pilot, in cooperation with the corresponding policy makers, will be to have a common structure, consisting of:

One screen for each social media platform used, with three categories of metrics concerning the three abovementioned perspectives: awareness, interest and acceptance respectively. This will necessitate for each social media platform employed to define which of the metrics it provides give information on each of the above three perspectives.

And also one 'synthetic' screen, again with three categories of 'synthetic' metrics, which combine information from all employed social media platforms, concerning awareness, interest and acceptance respectively; this will necessitate to combine in an appropriate manner metric of similar orientation (awareness, interest, acceptance) from all employed social media platforms. A first indicative table with such metrics of these three categories for Facebook, YouTube, Twitter and Blogger (= the main social media platforms to be used in the pilots) is provided below

	Facebook	**YouTube**	**Twitter**	**Blogger**
Awareness	visits of main page	video views	clicks on a link included in a tweet	view of posting
Interest	comment	comment	replies - retweets	enter a new posting
Acceptance	likes – positive comments	likes – positive comments	positive replies - retweets	likes- positive postings

This table is going to be further elaborated in the pilots. As we can see it the last row of the above table, we are going to integrate the results of opinion mining – sentiment analysis in the above metrics, e.g. for the calculation of the acceptance metrics (for each social media platform and for the synthetic one) will be taken into account the number of the comments classified by opinion mining – sentiment analysis as 'positive'

5 Conclusion

The underlying Deliverable 3.4 accompanies the Integrated PADGETS Platform Prototype which is hosted on AEGEAN cloud infrastructure. Publishing and Tracking System, Decision Support Engine and Data Mining Engine are already integrated in the Application Server. Deliverable 3.4 addresses a brief description of the PADGETS platform prototype. The focus of the deliverable is on key features as campaign management, monitoring and analytics of the PADGETS dashboard. Furthermore it covers hosting and deployment facts for pilot activities.The current challenge is to adapt the integration tasks to the pilot activities in order that pilot requirements will be reflected in the PADGETS Platform.

List of Figures

Figure 1: PADGETS Platform 4
Figure 2: PADGETS Platform Components 6
Figure 3: PADGETS Platform Deployment 8
Figure 4: PADGETS Platform Features 9
Figure 5: Campaign Manager 10
Figure 6: Policy Message Form 11
Figure 7: PADGETS Surveys and Polls 13
Figure 8: Policy Message 14
Figure 9: Feedback Stream 16
Figure 10: Activity Streams Schema 17
Figure 11: Activity Streams Architecture 18
Figure 12: Publisher and Subscribers Architecture 19
Figure 13: Dashboard Notification System 21
Figure 14: PADGETS Dashboard Reports 25
Figure 15: Opinion Mining implementation framework 29
Figure 16: Opinion Mining classification process 30
Figure 17: RapidAnalytics service exposure 31