MIPRO 2014, 26-30 May 2014, Opatija, Croatia

Alert Notification as a Service

Marjan Gusev, Sasko Ristov
University Ss Cyril and Methodius, FCSE,
Rugjer Boshkovic 16, Skopje, Macedonia
Email: {marjan.gushev, sashko.ristov}@finki.ukim.mk

Goran Velkoski, Pano Gushev
Innovation LLC,
Belasica 2, 1000 Skopje, Macedonia
Email: {goran.velkoski, pano.gushev}@innovation.com.mk

***Abstract*—Digital revolution offers instant information on millions of web sites. In plenty of available data, customers are interested to find information in context of appearance of an object or a certain property as soon as it is published by a given site. Some sites offer push notification to customers if they are subscribed to such a service. However, not all sites implement push service, and in these cases customers frequently visit the web site and wait for an occurrence, such as sport result, rank list, etc. In this paper we present how an intelligent software agent can realise this task instead of users that are manually visiting the web site and send information as if the web site has implemented a push service.**

This novel idea is good for both customers and web site providers. Customers would subscribe to get the information as soon as it is available, and web site providers would like to enable push service without changes in the web site software, that might be very costly. Provision of this kind of notification can be realised via sophisticated mobile phone alerts, text messages, twitter or Facebook notification, or even supported by new operating systems like Maveriks OS or web sites such as Google notifications. In this paper we present details and benefits of a such system.

***Index Terms*—push service, notification, alert service, cloud computing, intelligent agent**

I. Introduction

Remote access to web content is of utmost importance. Searching for information on the Internet can sometimes be the most straightforward job, but on other occasions a laborious and frustrating task for both experienced and casual users [1].

The fact that being well informed is key to almost everyone can surely be counted amongst the reasons for the expansion of social networking.

Borgatti et al. [2] indicate that people are gathering a considerable amount of information from shared informations by their friends and/or acquaintances.

Additionally, Lerman [3] expresses the need of aggregated news on a single website or software makes the news aggregators very popular and a successful business possibility. Powered by additional social information this process may provide an effective social recommendation system.

Furthermore, people are creatures of habit when it comes to web activity, which is very similar to the actual "autopilot" navigation that is present in nature [4]. We all have our daily routine when it comes to starting the day on the web. Some do it when they wake up, others at their workplace. What is in common is that we all do it, except that we just have different habits such as opening the mail inbox, replying emails, checking the daily news and friends' activities, etc.

Not all sites provide push services and enable notifications. In fact, there are only few sites, such as sport news or similar information sites that send push information to customers. The new digital era initiates situations where, a lot of people are waiting for a notification to appear on certain web pages. For example, one would like to find out if there is a change in a given property associated to a certain key phrase. In this case the user is frequently visiting all those web sites looking for a given text or keyword.

Yet, overwhelmed by more important daily activities, the web occupation is often limited. This is one of the reasons for the success of feed gathering systems that provide data from multiple sites on a single web site or software based on the user required feed. Understandably, this kind of systems are expected from the popular web search engine providers such as Google, Microsoft, Yahoo etc. Clearly, the reason for this is their engine that is able to provide quick news indexing and availability to the users.

Indeed, Google Reader [5] was one of the most popular solutions that provided this functionalities to the users by using the popular RSS [6] (Rich Site Summary) news feed to gather information across the Web.

The problem occurs because not all sites provide RSS feed for their content. Our motivation is to build a cloud based system that provides an engine for site indexing and automatic feed creation for all of the sites. The overall idea is to realise this as a Service.

The expected affliction here is the performance issue since we are not just gathering RSS feeds but also crawl sites that do not offer this possibility. Therefore, in order to provide a scalable and elastic service, we find cloud computing to be a valid solution to do so.

We call the new service Alert Notification as a Service (ANS) because of its notification nature, and place it in the Cloud as a novel cloud service. Additionally, the service provides a possibility of a variety of notification services such as: Email, Facebook Notification, Skype Notification, Google Cloud Messaging (GCM) [7], SMS, Twiter, LinkedIn etc.

This novel cloud service can be offered to the end user, but also to the cloud service providers in order to provide better indexing for their hosted sites, and they can also offer this as a Service. By doing this, the sites will not be burdened with the RSS feed requirement and in the same time the users may get to the much needed results quicker and in the form they require.

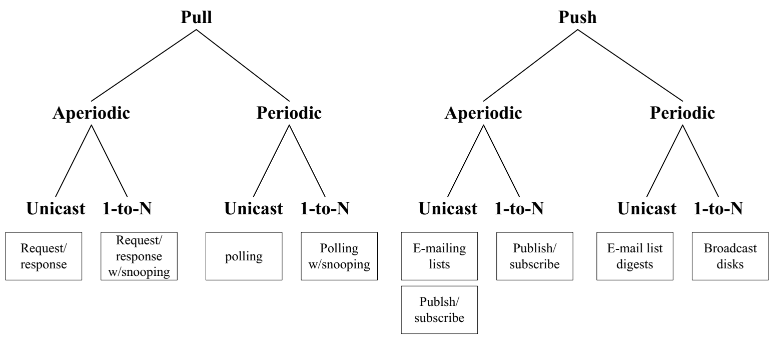

Fig. 1. Data delivery options for notification [8]

This paper concentrates on the development of a web service that will automatically visit all these web sites and alert if there is a change in the existing text or appearance of a certain keyword phrase. It will act as a kind of an intelligent agent and will periodically visit the pages instead of users to find a particular occurrence of their interest.

The rest of the paper is organised as follows. Related work in the area of push notification services and news feed services is presented in Section II. Section III describes the concept of ANS. The details of the system requirements and initial system architecture are discussed in Section IV. In Section V, we present the proposed system architecture intended for push notifications delivery. Section VI discusses the notification as a service system. The conclusion is specified in Section VII.

II. Related Work

In this section we will discuss some similar currently popular services and software solutions and compare it to our approach. The huge variety in offer of similar solutions confirms the popularity of this kind of systems.

We already started discussing similar software possibilities such as Google Reader in Section I. This is just an honourable mention to the software that Google released. Yet, the solution was discontinued and as of July 15, 2013 all subscription data is permanently and irrevocably deleted [5]. Google reader was an RSS feed aggregator and since it was deprecated a lot, for other solutions emerged to replace it.

Just by using the alternative to list [9] we can find more than 200 applications present that will offer similar services [10]. Some notable mentions are: Feedly [11], NewsBlur [12], Feedspot [13], etc. Almost all of these solutions offer a way to organise and read web content that is automatically aggregated on a single website or a software. Additionally, some of the solutions even offer statistics and synchronisation among multiple devices or even between friends when integrated with social networks.

Recently, Google released a new popular solution called Google Alerts that allows users to monitor the web for interesting new content based on search queries [14]. Besides the initial popularity of the system, one cannot miss out the more than few issues that the system has [15].

Therefore a lot of professional level and entry level applications are available on the market as both open source and proprietary solutions. Dembak [16] identified popular alternatives for this system. We will just mention some of them and summarise their functionalities. Talkwalker, Social Mention, Mention, Trendrr, Bloomberg Professional, Meltwater News, etc. are some of the popular solutions that provide functionalities for easy tracking information on specific pages or on the Web. Functionalities such as alerts in real time from web via email or in a RSS feed reader, social engine tracking, news importance ranking, analytics are available in these solutions.

There are various technologies that can be used to develop a notification service. The basic technology used was the push concept. Push technology stems from a very simple idea with ”in-your-face” nature [8]. Rather than requiring to explicitly request (i.e., ”pull”) the information that they need, data can be sent to users without having them specifically ask for it.

Push technology has evolved in last 15 years, and even the WWW consortium is trying to establish a description of Push API [17]. It defines how a web application can use an API via the user agent that will communicate to the push server and application server. HTML5 also supports the push technology [18] by defining Server-Sent Events (SSE) and standardising Comet for all standards-compliant web browsers. The Server-Sent Events specification ”defines an API for opening an HTTP connection for receiving push notifications from a server.” [19].

Push and Pull technologies use different data delivery options [8], as presented in Figure 1. Aperiodic pull is using the request response communication, while the periodic one is using polling. Push technology pushes the information without request or polling. In this case the user has to subscribe for a service. If a site would like to realise a push service, then the corresponding technology has to be used and this requires more resources.

Push notifications may be delivered via various methods which include standardised protocols (e.g. Server-Sent Events [SSE], the GSM Short Message Service [GSM-SMS], SIP MESSAGE [RFC3428], or OMA Push [OMA-PUSH]), or via

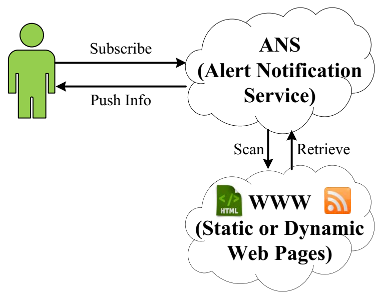

Fig. 2. Concept of ANS

browser-specific methods.

RSS is a realisation of push technology or a web feed, usualy used for syndicated news service when one would like to read all the content that is pushed by a given server. Restful services are those Resource oriented architecture [20] and Representational State Transfer (REST) [21] can be used to realise push by concentrating on the roles of the components, the constraints of their interaction with other components and their interpretation of significant data elements. One of the alternatives is using SOAP while building web services. The main idea is to change the standard client server communication with a stateless communication by sending a representation of each application state with appropriate links.

Pushlets are another example of a push notification technology [22], where the server never terminates the response and the client is automatically notified about new events.

III. Concept of ANS

Our new solution helps all those sites that would not like to invest a lot of resources to implement various push services. They will still remain the same static or dynamic web sites. The concept of the ANS is presented in Figure 2. The user subscribes to the service and chooses the notification delivery method. Then the ANS is performing all the tasks, by frequent scanning the existing web site and finding a specific condition for an alert to occur.

In our solution we build all technology resources, similar to those described in W3C specification of a Push API [17]. However, we design a creation of a full service and do not require that the user has to change their concept of realisation the web application. Our service is realised as a web application where a user can register and subscribe using any web browser and can specify the way the notification can be received by choosing any conventional methods like e-mail, text message, social network or operating system notification.

Main functionalities can be summarised as in Table I.

TABLE I
Main functionalities of the Alert Notification Service

Functionality	Description
Definition	Specification of a web page, search key phrase, a property or occurrence of a keyword as an alert
Scanning	Crawling, visiting the web page and retrieving the required web content
Triggering	Controlling the crawling frequency, comparing the content with alert constraints and triggering the alert
Delivery	Activating the delivery channel and transmitting the alert message

Upon registration one can login to the main admin page. The main admin page can allow to define the alerts by selecting web pages and corresponding text to appear or to be changed from the last occurrence. Two main functionalities are possible within this activity:

- Select a web page where there is a change in text occurrence. For example, one would be interested to know when a new published paper will be indexed in a particular database, such as ieeexplore, and which paper, if more papers are waiting for publishing; and
- Select a web page with occurrence of a given text phrase. For example, my name has appeared in a blog or news.

The next functionality is selection of delivery method. The user can select among various delivery channels, such as personal message delivery via electronic mails or text messaging on a mobile device, personal messaging on social networks, or recently as a functionality of the operating system, such as Maveriks OS, or functionality of the web browser, such as Google Chrome.

The main functionality is realisation of scanning the existing web sites. Actually, the system should be able to do what a typical user is doing when visiting a web page, looking for an information. It means not just visiting a certain web page and finding an occurrence of a text within a paragraph, but also clicking on certain buttons or entering a value in certain fields. It will simulate a human entering data on an interactive web page, with a possibility to select a certain radio button or an item in a drop down list, to enter a value in a certain text field, or to click an activation button. Then the system will be able to trace the file and find a predefined keyword phrase, by detecting if there is a change in the context or in the occurrence of the keyword phrase.

This new system will enable the web sites that do not have push notification, alerting system or use corresponding API to realize alert notification. It will actually build push notification for any web page.

The alerting system can be realized conventionally by e-mail, skype, twitter or facebook messaging, or can be realized as a web page where one can see notifications, or as an add-on tool for the web browser. An example of occurrence is the alert bell on Google chrome.

It can be thought of as a system for users who would like to be notified for a certain change, or for cloud service providers who would like to build push functionality for classical web sites that do not support this technology.

IV. System Requirements

In this section we will discuss details of the system requirements and define the initial system architecture.

The system can be a service for users who would like to be notified about occurrence of a key phrase on a certain web page. The user interface should enable definition of the alerts and browsing of the results. Additionally, the ANS system may be offered to cloud service providers who would like to build push functionality for classical web sites that do not support this technology.

The main system functionalities are specified in Table I. Here we elaborate details of each functionality.

A. Definition

The system should allow user friendly definition of alerts. An alert can be defined by providing the following information:

- **key phrase** The user is looking for a certain key phrase in the web content. It may be one word or a sequence of several words. For example, one might look for a certain ranking to find if a name of a given person appeared in the list;
- **keyword as property of the key phrase** The user is looking for a certain property of the key phrase or just defining an occurrence of the keyword. For example, one might likes to find out the ranking of the person in the list and retrieve the position number;
- **alert constraint** The user should be able to define when to trigger an alert. For example, it may be an occurrence of a key phrase, change in its property, such as position number, combination with other keywords, or when the occurrence is changed etc.;
- **web page** The user is providing the url of the required web page;
- **defining a macro** The user should be able to define a macro of provided keyboard or mouse actions that leads to a occurrence of a specific web page content. For example, one might click on a certain radio button, select an option in a drop-down list, enter a specific text, click on action button in the given web page to filter the web content and dynamically choose the web content;
- **scanning frequency** The user must be able to select how often the site should be crawled for updates; and
- **delivery channels** The user should be able to select delivery channels for alert notification. For example, one might select e-mail and SMS text message on a mobile device as notification delivery channel.

B. Scanning

Scanning is realised as a crawling engine that will frequently visit a certain web page and execute various selection or activation actions. The final output is retrieving the web content on the visited web page, avoiding as much junk information as possible.

The prerequisite for this requirement is different from all existing software solutions that mostly use RSS feed from a specific site to retrieve a corresponding web content. This service must be able to extract information from any web site by crawling through its content. The crawling frequency is a parameter that can be defined by the user.

Additionally a user can select a web site that is offering RSS feeds and use this service as a kind of filter that selects a key phrase occurrence as defined alert. In this case the crawling is not started, and this functionality is actually a typical news aggregation functionality.

C. Triggering

Triggering refers to implementation of the main core control in the web service. Once the web content is retrieved by the scanning service, it is analysed and cleared from junk information. The selected information is then compared to the definition of the alert constraint. A positive outcome of the alert constraint condition triggers an alert message which is then scheduled for transmission via delivery channels.

This functionality is also novel compared to other present similar systems. The innovative approach is required in order to extract just a predefined selected part of the web content that the users need. Let's say that we want to follow the stock information on various stock exchanges. We can just simply follow the web page RSS feed with plenty of junk information and then extract the required information. The system must provide the user means to select a specific information. Definition of alert constraint is the realisation of an agent that checks if a certain condition is fulfilled triggering an appropriate alert. Then this triggers a push notification via selected delivery channels.

Additionally the user must be able to record the user interface input to filter the web content. It consists of several mouse and keyboard actions for a given web page that result in presentation of a specific content. For example, the user may be a scientist who is interested when a new journal paper is referenced on IEEEXplore database. The user should be able to navigate to the IEEEXplore search enter the search query he desires and check if in the list of the papers there is some change. The whole navigation and input on the page should be recorded in order to enable repeated query of the same request.

The whole functionality must have a web interface i.e. all the activities should be recorded using just web browser. This provides the means to use the same functionality across multiple platforms and additionally across multiple devices.

D. Delivery

When the alert is triggered the corresponding delivery channel is activated.

The system can activate a push notification to the end users as a kind of a real time notification using various messaging channels. The users can define some of the following delivery channels to get alert notifications:

- personal messaging via Email;
- personal messaging via text message (SMS) on a mobile device;
- personal messaging via social networks, like Facebook, Twiter, LinkedIn, etc.;

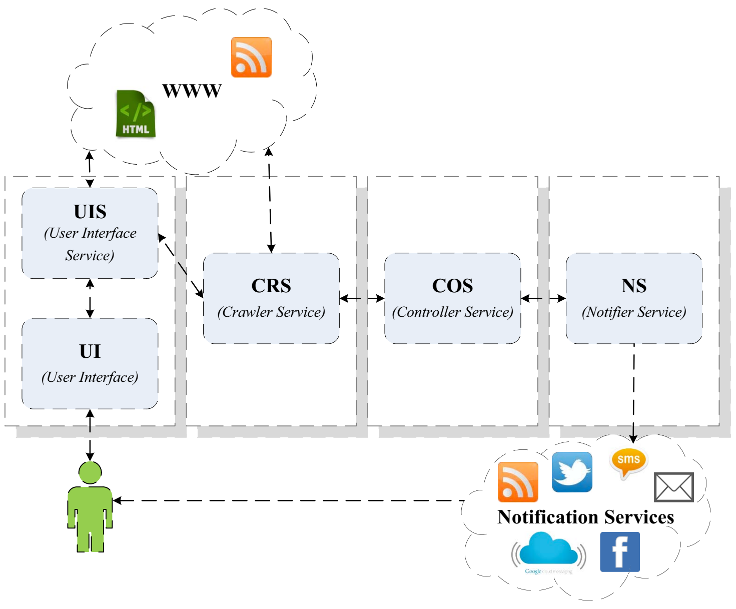

Fig. 3. Architecture of ANS

- cloud messaging, like Google Cloud Messaging (GCM);
- operating system notification, such as Maveriks OS, IOS, etc.;
- web browser notification, such as Google Chrome; and
- news feed delivery RSS.

This is also a functionality, which is not provided by the other similar solutions.

E. Other functionalities

Next we present other functionalities that may define the user interface of the complete ANS service. We start with content organisation and grouping. This functionality enables better user experience and is performance bound. The system is expected to be used by multiple users and often some will request similar if not even the same queries. Therefore, the system must organise and group the queries based on a required web page, or a specific query. This way the system should visit one single page and index it for information ones to cover as much as possible user queries.

The next functionality concerns the statistics of usage and alert generations. It may be important for future use to analyse typical behaviour and schedule corresponding resources for cloud service providers and site owners. The final goal is to realise an intelligent agent that will predict user activities and provide an information faster, and with greater accuracy.

The overall system should be compatible with different notification methods. It should support all existing push notification technologies and provide an API that can communicate with other web pages, push notification systems, RSS feeds, or similar scheduling systems, such as calendars. Realisation of an API that communicates with other systems and can integrate with operating systems and web browsers from one side, and web and push servers from other side is another feature of our service compared to the other similar systems.

V. ANS System Architecture

The main architecture of the new defined ANS system is presented in Figure 3. Core modules are realising the main functionalities. The main User Interface Service (UIS) brought by a web browser realises the functionality to define the alerts, including recorder and simple web interface components.

The Crawler Service (CRS) realises the scanning functionality and outputs appropriate content to the Controller Service (COS), which realises the triggering functionality. COS activates the CRS on a specific timing to realise frequent web page scanning. It also analyses the provided content, filters the relevant information from the junk and stores the relevant data. If the condition defined by the alert constraint is fulfilled it triggers the Notifier Service (NS) which realises the delivery functionality. NS activates corresponding notification channels to the users.

Additionally the system requirements must be performance and power aware. Therefore the system must be scalable and flexible in order to provide virtually unlimited scalability.

VI. Discussion

The advantages of push technologies are straightforward [8]. The traditional pull approach requires that users know a priori when and where to look for data or that they spend an inordinate amount of time polling known sites for updates and/or hunting on the network for relevant sites. Push relieves the user of these burdens.

There are several problems with the implementation of the push technology. The web technology changes a lot and the crawler system might need constant update to enable quality

TABLE II
AUTONOMOUS SOFTWARE AGENT PROPERTIES THAT OUR SYSTEM HAS

Property	Description
reactive	responds to a change in required information for a given content
autonomous	controls the process of triggering without direct human interaction
continuous	runs as continuous process over time
trustworthiness	adheres to learned behaviour and is truthful

scan of the web page and allow transfer of the related content from the web site to the system. Thus the system can make a good decision to find out if a condition for triggering occurs.

This system acts as an autonomous software intelligent agent according to the definition given in [23]. It scans the web page instead of the users and finds if a certain condition is fulfilled. The environment is the web, sensing correlates to scanning, and the action that produces change of state is comparison. The algorithm of realisation of the intelligent agent in this case follows a simple principle to learn by an example given by the user and sense the relevant information from the collected content.

The described online learning system with adaptive testing has several important agent's properties as presented in Table II. Our intention was not to present all characteristics and agent features, as given by Badica et al. [24] but to cover the most essential and specific properties analysed by our system.

Another problem is the realisation of the decision making algorithm that triggers an action. It is not a simple comparison of two strings, but it is realised by a rather complex actions. This happens since various technologies can hide the relevant information. For example, the information can be positioned it in tables, frames, in-line text, or various modules, so the realisation of the web scanner needs a careful algorithms to include all relevant cases. Various browsers present the relevant information with different algorithms, so when the user learns the crawler, has to be careful to use appropriate browser, which is supported by the crawler.

VII. CONCLUSION

In this paper, we have introduced a new system that actually builds on all those classical web sites, where the user is pulling information and realises the push of required information. In this case the user specifies what can trigger and push the information to the user. The system is built as a software as a service cloud solution.

One might think that this system is just an agent that provides push functionality to the conventional web sites. Instead of recompiling the software on enormous dynamic web sites and realise push technology for them, this system enables adding just a service with the same functionality that one can subscribe. In reality it repeats the actions, one frequently performs when looking for a certain information, and acts if a human reads the site and finds out a change in information he/she is looking for. This is represented as an alert and the user will be informed as soon as there is a change in information he/she is looking for.

Modern operating systems, like Maveriks, IOS or others include alert notification features if there is a change in some system information, like calendar events, received mail, finished actions, etc. Some modern web browsers also build notification alerts, such as Google Chrome, which is looking for a news on different web sites which correlates to the one the user is interested. This is a part of the conventional Google sprawling action, which compares users keywords with new scanned web pages. However, this is quite different from our service, which looks for a change in information in a predefined web page.

Adding functionalities of our newly introduced system may be part of all new releases of modern operating systems and web browsers. They are beneficial for both the users looking for a change in information or by providing push technology to conventional web site, beneficial for enormous number of cloud and web site providers.

REFERENCES

[1] C. Hölscher and G. Strube, "Web search behavior of internet experts and newbies," *Computer networks*, vol. 33, no. 1, pp. 337–346, 2000.

[2] S. P. Borgatti and R. Cross, "A relational view of information seeking and learning in social networks," *Management science*, vol. 49, no. 4, pp. 432–445, 2003.

[3] K. Lerman, "Social information processing in news aggregation," *Internet Computing, IEEE*, vol. 11, no. 6, pp. 16–28, 2007.

[4] J. Prendergrast, B. Foley, V. Menne, and A. K. Isaac, "Creatures of habit?" *The art of behaviour change, London*, 2008.

[5] "About Google Reader." [Online]. Available: http://www.google.com/reader/about/

[6] "RSS 2.0 Specification (version 2.0.11)." [Online]. Available: http://www.rssboard.org/rss-specification

[7] Google. [Online]. Available: http://developer.android.com/google/gcm/index.html

[8] M. Franklin and S. Zdonik, "data in your face: push technology in perspective," in *ACM SIGMOD Record*, vol. 27, no. 2. ACM, 1998, pp. 516–519.

[9] "AlternativeTo.net." [Online]. Available: http://alternativeto.net/

[10] "Google Reader Alternatives on AlternativesTo.net." [Online]. Available: http://alternativeto.net/software/google-reader/

[11] "Feedly, Homepage." [Online]. Available: http://feedly.com/#welcome

[12] "NewsBlur, Homepage." [Online]. Available: http://www.newsblur.com/

[13] "Feedspot, Homepage." [Online]. Available: http://www.feedspot.com/

[14] "Google Alerts." [Online]. Available: http://www.google.com/alerts

[15] R. Grant, "Google Alerts 'broken,' 'useless,' and slowed to a 'trickle'," 2013. [Online]. Available: http://venturebeat.com/2013/03/21/google-alerts-broken-useless-and-slowed-to-a-trickle/

[16] Y. Dembak, "7 apps to help you replace Google Alerts," 2013. [Online]. Available: http://venturebeat.com/2013/05/13/google-alerts-replacements/

[17] W3PUSH. (2013, Aug.). [Online]. Available: http://www.w3.org/TR/push-api/#dfn-appserver

[18] W3HTML5. (2013, Aug.). [Online]. Available: http://www.w3.org/TR/html5

[19] TodayJava. (2010, Mar.). [Online]. Available: https://today.java.net/article/2010/03/31/html5-server-push-technologies-part-1#still

[20] L. Richardson and S. Ruby, *RESTful web services*. O'Reilly, 2008.

[21] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[22] J. van den Broecke, "Pushlets white paper," *Just Objects BV August*, vol. 6, 2002.

[23] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," *Intelligent Agents III Agent Theories, Architectures, and Languages*, pp. 21–35, 1997.

[24] C. Bădică, Z. Budimac, H.-D. Burkhard, and M. Ivanović, "Software agents: languages, tools, platforms," *Computer Science and Information Systems/ComSIS*, vol. 8, no. 2, pp. 255–298, 2011.