RSS v2.0: Spamming, User Experience and Formalization

Thrushna Nalam

University of Tampere
School of Information Sciences
Computer Science
Supervisor: Dr. Eleni Berki
July 2015

University of Tampere
School of Information Sciences
Computer Science / Software Development
Thrushna Nalam: RSS v2.0: Spamming, User Experience and Formalization
M.Sc. thesis, 61 pages, 14 index and appendix pages
July 2015

RSS, once the most popular publish/subscribe system is believed to have come to an end due to reasons unexplored yet. The aim of this thesis is to examine one such reason, spamming. The context of this thesis is limited to spamming related to RSS v2.0. The study discusses RSS as a publish/subscribe system and investigates the possible reasons for the decline in the use of such a system and possible solutions to address RSS spamming. The thesis introduces RSS (being dependent on feed readers) and tries to find its relationship with spamming. In addition, the thesis tries to investigate possible socio-technical influences on spamming in RSS.

The author presents the idea of applying formalization (formal specification technique) to open standards, RSSv2.0 in particular. Formal specifications are more concise, consistent, unambiguous and highly reusable in many cases. The merging of formal specification methods and open standards allows for i) a more concrete standard design, ii) an improved understanding of the environment under design, iii) an enforced certain level of precision into the specification, and also iv) provides software engineers with extended property checking/verification capabilities. The author supports and proposes the use of formalization in RSS.

Based on the inferences gathered from the user experiment conducted during the course of this study, an analysis on the downfall of RSS is presented. However, the user experiment opens up different directions for future work in the evolution of RSS v3.0 which could be supported by formalization. The thesis concludes that RSS is on the verge of death/discontinuation due to the adverse effects of spamming and lack of its development which is evident from the limited amount of available research literature.

RSS Feeds is a perfect example of what happens to a software if it fails to evolve itself with time.

Key words and terms: RSS, RSS feeds, RSS spam, Feed readers, Socio-technical influence, Formal Specification, Open standards, Publish/Subscribe, CafeOBJ.

Acknowledgement

I dedicate this thesis work to my beloved mother, whose words of encouragement and love has allowed me to be who I am today. I am happy for having chosen University of Tampere, Finland for my Master's degree. I have found a very good atmosphere and learning facilities that have contributed towards my interest in the field of research. I express my sincere gratitude to my parents and friends who motivated and supported me to pursue a master degree.

I would like to thank Dr. Eleni Berki, my thesis supervisor and my mentor for helping me choose this topic. During the course of this thesis, I have received constant support and motivatination from her. She has always inspired me to conduct excellent research in the field of security and Internet safety. This thesis work would have not been possible without the proper guidance and feedback from her. I would like to express my gratitude towards my research team headed by Dr. Eleni Berki for the excellent research on open standards and formalization.

I would also like to thank Prof. Erkki Mäkinen, for his critical comments and giving right direction to the thesis work. A special thanks to Saila Ovaska, Lecturer, University of Tampere for assisting me in the conduct of the user experiment for this thesis. I would also like to thank the participants of the user experiment conducted for the study. They were very cooperative and supportive throughout the experiment.

Contents

1. Introduction

A publish/subscribe system provides the subscribers the ability to interact or express their interest in an event or be notified of the publisher, when an event is generated or matches their interest. In simple words, publishers 'publish' information and subscribers 'subscribe' to the information that they want to receive. The information published by the publisher is termed as event and the act of delivering is denoted by notification [Eugster et al., 2003].

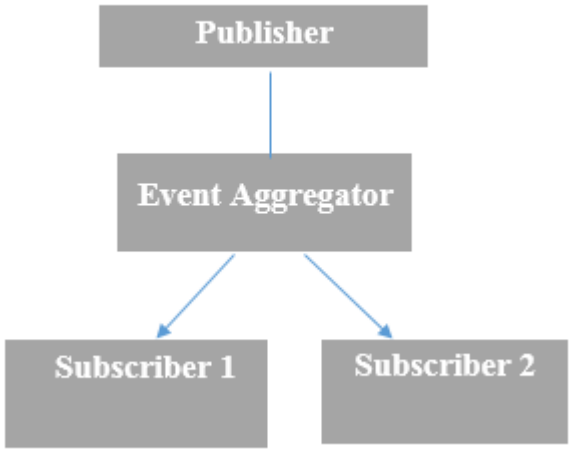


Figure 1. A simple representation of publish/subscribe system

Rich Site Summary (RSS) is an XML[1] code and is often referred to as RSS feeds which is used for distributing and aggregating web content. The design and model of RSS is quite simple: consumers subscribe to feeds of their interest by polling for the feeds periodically, to stay up to date [Liu et al., 2005].

Users determine their favorite websites or blogs they want to read and a properly configured RSS reader or an RSS integrated web browser will fetch the selected information or hyperlinks and display the contents on the user's desktop in regular intervals.

[1] XML stands for Extensible Markup Language

RSS feeds were once seen as the most essential tool to distribute web content and to build traffic to a site. The demise of RSS feeds offers a very important case study of how a technology could fail due to the lack of necessary improvements with the changing times. The decline of RSS can be traced down to the following reasons:

i) RSS is a dependable software meaning it cannot function without the use of an RSS feed reader or aggregator.

ii) Insufficient development made, failed to evolve.

One of the motivating factors behind this thesis is to apply formalization (formal specification technique) to RSS v2.0 to investigate if it can be used to prevent spamming or help it to evolve as RSS v3.0. By formalization, we mean to use a formal specification technique. In context of this thesis, CafeOBJ, an algebraic specification language has been used to demonstrate how formalization can be beneficial. The author has been actively involved in research activities related to formalization and its application to open standards.[Barlas et al., 2014]

A user experiment is conducted to track down the user experience of using RSS in comparison to technologies such as Twitter[2], Facebook[3], etc. Furthermore, we would would like to observe with the help of the user experiment, if there is a possible socio-technical influence on RSS spamming. In other words, what could be the possible socio-technical attributes related to RSS spam. The term Socio-technical systems was introduced by Emery and Trist [1960] to stress the reciprocal interrelationship between machines and humans.

1.1 Research Questions

The study focuses on RSS feeds and spamming. Since this thesis discusses about RSS feeds as a publish/subscribe system, it is important to discuss the existing literature in RSS feeds related to publish/subscribe systems. This thesis analyzes the following research questions:

[2] http://www.twitter.com
[3] http://www.facebook.com

Q1. How RSS differs from other publish/subscribe systems?

Q2. What is RSS spam and how does spam look like in RSS feeds?

Q3. How was the overall experience of using RSS feeds with feed readers?

Q4. What is the possible socio-technical influence on spamming in RSS feeds?

Q5. What are the benefits of using formalization in RSS feeds?

1.2 Research Methodology

The research methodology for this thesis work is systematic literature review followed by the analysis based on the user experiment.

The systematic review consists of a scientific methodology that goes one step further than an overview. It is used to refer to a certain methodology used to gather and evaluate the available information for a research. When performing systematic review a researcher is able to select and quantify the results. Systematic review conduction process needs planning, execution, and result analysis. During the planning phase the objectives and a review protocol are defined. Usually the research questions and methods that need to be used are discussed. The execution phase involves the initial studies and research leading to the selection and evaluation of the already found knowledge. This involves the inclusion and exclusion criteria of the data selected based on the review protocol made earlier. Once the papers have been selected based on the review protocol and the research questions the data can be extracted to the result analysis phase [Mian et al., 2007].

An experimental evaluation is generally divided into two parts where exploration takes place in the first and evaluation takes place in the other part. The exploration identifies what questions should be asked about the subject/system under discussion and the evaluation attempts to answer those questions. The inferences of the user experiment provide proof of concept and illustrate potential but they may not provide solid evidence.

1.3 Thesis Structure

The structure of the thesis proceeds as follows: Chapter 2 discusses the temporal and contextual limitations of this thesis, elaborating about the review protocol, data sources, search strategy and study selection. Chapter 3 explains the concepts of publish/subscribe system and their types. Chapter 4 describes the characteristics of RSS feeds and also an overview of the feed registries and aggregators. Chapter 5

concentrates on the spamming in publish/subscribe system. This chapter also focuses on the various types of publish/subscribe spam, namely blog spam and email spam. Chapter 6 describes about RSS, spam and socio-technical attributes. Chapter 7 presents a research paper about formal specification of RSS. Chapter 8 is concerned with a user experiment and its findings. Chapters 9 and 10 conclude the work and discuss the limitations and future work.

2. Literature Review

Literature review is always the initial step of any research work as it illustrates the importance of the research area and establishes the relevance of the research topic with the existing research. The review focuses on the literature published in the last 10 to 15 years in order to have relevance with the current trends in the field of RSS feeds and its relationship with publish/subscribe, spam and formalization.

2.1 Review Protocol

In order to find answers to the research questions defined in Section 1.1, a number of tasks or steps have to be completed. These tasks are referred to as review protocol. The review protocol helps to achieve consistent and consolidated results.

The components of review protocols are defined as follows:

- Data sources – refer to the scientific databases or other sources of information which are to be used for searching the required literature.
- Search strategy or search process – shows how the data sources are queried to obtain the desired search results.
- Study selection – defines the criteria for selection of the literature available after the search process.
- Selection of final articles - involves the extraction of relevant literature according to the research questions.

2.1.1 Data Sources and Search Strategy

The following data sources have been selected for searching the literature:

- Springer Link[4]
- ACM Digital Library[5]
- IEEE Digital Library[6]
- Wiley Online Library[7].

[4] http://link.springer.com
[5] http://dl.acm.org
[6] http://ieee.org/ieeexplore
[7] http://onlinelibrary.wiley.com

The search strategy has been divided into two phases:

- Search using individual keywords
- Using search strings.

The relevance of the second phase is considered to narrow down the number of articles or to find articles more specific to the research questions. If the number of results for individual keywords is high, the search string is to be used.

Keywords used are the following: *RSS feeds, publish/subscribe, spam, feed readers, formalization* and the search strings used are: *RSS feeds and spam, RSS feeds and publish/subscribe, RSS feeds and formalization.*

2.1.2 Study Selection

Inclusion and exclusion criteria have been defined for study selection. We used the following inclusion and exclusion criteria.

Inclusion Criteria:

1) The literature is related to publish/subscribe systems with focus towards RSS feeds.

Reason: The study investigates RSS feeds as a publish/subscribe system.

2) The literature is related to spam in content-based publish/subscribe systems.

Reason: There has been much research on content-based systems and their evolution and how they stand apart but very little about the reasons of spam in content-based publish/subscribe systems.

3) The literature is related to formalization in RSS feeds.

Reason: The study tries to find a relationship between formalization and RSS feeds.

4) The literature is in the form of scientific publications, i.e., journal articles or conference proceedings.

Reason: Journal articles and conference proceedings of high reputation and recognition help to ensure that the literature review is unbiased and minimizes the possibility of favor to a particular practitioner or approach to be followed.

Exclusion Criteria:

1) The literature is related to RSS feeds but does not handle spamming or any content-based publish/subscribe system.

Reason: A study on content-based publish/subscribe system not involving spamming may not be relevant to the actual research questions being discussed.

2) The literature does not relate to all type of content-based publish/subscribe systems, as the area is vast and irrelevant to the focus of this study.

Reason: Not all content-based publish/subscribe systems are RSS feeds. A study not related to this is thus not related to the research questions.

3) The literature belongs to formalization but does not address RSS feeds.

Reason: The context of formalization is limited to RSS feeds as per the scope of this thesis.

4) The literature is not in the form of scientific publication.

Reason: Unrecognized publications or company reports may be biased in nature and may lack scientific evidence.

2.2 Search Results

The search for the literature has been performed in September 2014.

The selected keywords have been searched using the selected data sources. Table 1 shows the number of articles for each keyword in different data sources.

Keywords	Springer Link	ACM Digital Library	IEEE Digital Library	Wiley Online Library
RSS Feeds	5110	1642	166	3313
Spam	8962	5083	1467	2698
Publish/Subscribe	3425	2400	711	17
RSS Feed readers	1543	556	15	960
Formalization	37,506	7040	1596	14,072

Table 1. Keyword search results

Since the number of articles for individual keywords in each data source was high and low, the second phase of search strategy has been used.

The search strings "RSS feeds and publish/subscribe **",** "RSS feeds and spam**",** "RSS feeds and formalization" have been used for different data sources. Table 2, Table 3, and Table 4 show the results for the search strings respectively.

Database	Total search results	Article publications	Other publications
Springer Link	122	119	3
ACM Digital Library	200	159	41
IEEE Digital Library	4	4	0
Wiley Online Library	19	5	14

Table 2. Search results for 'RSS feeds and publish/subscribe'

Database	Total search results	Article publications	Other publications
Springer Link	273	41	228
ACM Digital Library	134	34	100
IEEE Digital Library	2	2	0
Wiley Online Library	102	41	61

Table3. Search results for 'RSS feeds and spam'

Database	Total search results	Article publications	Other publications
Springer Link	116	111	5
ACM Digital Library	37	31	6
IEEE Digital Library	0	0	0
Wiley Online Library	0	0	0

Table 4. Search results for 'RSS feeds and formalization'

2.3 Selection of Final Articles

After performing the study selection, the titles and abstracts have to be read to find their relevance to the research questions. Ultimately, final articles for discussion have to be selected.

As per the analysis made after reading the titles and abstracts a total of 17 articles were selected to study the relationship between RSS feeds and publish/subscribe and RSS feeds and spam. Although the number of articles published with the keyword 'formalization' was around 60,000, none of them addressed to a possible relationship between RSS feeds and formalization. However, only one publication was found relevant to 'RSS feeds and formalization' search string. This publication has been co-authored by the author of this thesis and has been presented in Section 7.2.

The list of the 17 articles selected as per the result of systematic literature review has been presented in Table 5.

Item Title	**Publication Title**	**Publication Year**	**Database**
The many faces of publish/subscribe	ACM Computing Surveys	2003	ACM
Enriching topic-based publish-subscribe systems with related content	Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data - SIGMOD '08	2008	ACM
Securing publish-subscribe overlay services with EventGuard	Proceedings of the 12th ACM Conference on Computer and Communications Security - CCS '05	2005	ACM
Cobra: Content-based Filtering and Aggregation of Blogs and RSS Feeds.	NSDI'07 Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation	2007	ACM
A Two-Phase Approach to Subscription Subsumption Checking for Content-Based Publish/Subscribe Systems	2010 24th IEEE International Conference on Advanced Information Networking and Applications	2010	IEEE

Security aware content-based publish/subscribe system	Proceedings - IEEE Symposium on Computers and Communications	2009	IEEE
A software infrastructure for RSS deployment and linking on the web	WebMedia '05: Proceedings of the 11th Brazilian Symposium on Multimedia and the Web	2005	ACM
A Semantic Map of RSS Feeds to Support	Lecture Notes in Computer Science	2012	Springer
RoSeS: A continuous content-based query engine for RSS feeds	Lecture Notes in Computer Science	2011	Springer
Characterizing the splogosphere	Proceedings of the 3rd Annual Workshop on Weblogging Ecosystem: Aggregation, Analysis and Dynamics.	2006	ACM
Towards Spam Detection at Ping Servers.	International Conference on Weblogs and Social Media'07	2007	ACM
TrackBack spam	Proceedings of the 2009 ACM Workshop on Cloud Computing Security - CCSW '09	2009	ACM
Preventing Spam in Publish/Subscribe	26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06)	2006	IEEE
Modeling publish/subscribe communication systems: Towards a formal approach	Proceedings - International Workshop on Object-Oriented Real-Time Dependable Systems, WORDS	2003	Springer
Caching in Content-Based Publish/Subscribe Systems	GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference	2009	IEEE
Formal Specification of Open Standards and the Case of RSS v2.0	Proceedings of the 18th Panhellenic Conference on Informatics - PCI '14	2014	ACM

Table 5. Final list of selected articles.

3. Publish/Subscribe Systems

The basic model of a publish/subscribe system relies on the interaction between the event publishing publishers and the event subscribing subscribers through an event notifying service provider. The event here is the information passed by the publisher through the event managing service provider. In order to generate an event a publisher uses the operation *Publish ()* and passes the event while the service provider propagates the events to the subscribers. Every subscriber gets notified of the event according to their interest. Subscribers choose the events they want to receive based on their interest. Publishers have the ability to send out advertisement about the nature of their future events using the *advertisement ()* operation to keep the subscribers interested [Eugster et al., 2003].

The event service between publishers and subscribers can be categorized into three dimensions:

Space – The interacting parties do not need to know each other, as the publishers publish the events through an event service provider and the subscribers receive these events through the event service manager. Neither of the parties keeps track of the references between them.

Time – The interacting parties do not necessarily need to be active at the same time. In particular, the publisher might publish some events while the subscriber might not get notified of the same at the original occurrence of the event.

Synchronization – The subscribers might not be getting notified of the event at the exact occurrence of the event published but rather at different pace. The production and the consumption of the events might not necessarily take place in a synchronous manner.

In a publish/subscribe system, the subscribers are usually interested in particular type of events, and not all events published by the publisher. A subscriber can choose from a variety of schemes for this purpose. In this section, the two most common types of publish/subscribe systems are discussed.

3.1 Topic – Based Publish / Subscribe

The first publish/subscribe system was based on topics and subjects. It extents to the channels used for communicating with methods for characterizing and classifying the events content. Keywords are used as identification for subscribers to identify the topic of their interest [Eugster et al., 2003]. The topic name is usually specified, every topic is viewed as an event and is identified by a unique name, with an interface offering publish/subscribe operations. Topic based publish/subscribe system is perfect fit for event service schemes as the messages are classified into groups corresponding to the users interest.

Topic based publish/subscribe systems had risen to popularity due to their simplicity. When a user subscribes to a set of topics of his/her interest there is also a chance the subscriber is completely unaware of the existence of similar topics of his/her interest. This leads to loss of information making the whole purpose of publish/subscribe system useless [Boim and Milo, 2008].

The interfaces of a topic based publish/subscribe system share operations such as *create, publish, subscribe and unsubscribe.* To send messages, the publishers first create topics, each topic is recognized by a unique id (topic ID) and serves as a mediator between publisher's side and subscriber's side. To publish a message for a given topic, the publishers call *publish* () operation with reference to its topic ID. This message is propagated to the topic subscribers. To become a subscriber of a given topic, interested users call *subscribe ()* operation, with the appropriate topic ID. The corresponding *unsubscribe* () operation removes the subscription.

In topic based publish/subscribe system subscribers specify their interest by subscribing to a feed published by the publisher. In this case, the publishers and subscribers have to agree upon a certain set of topics which would be covered by the channels. Producers who generate content related to those topics, publish the content on the corresponding topic channels to which the users are subscribed and the users receive asynchronous updates via these channels.

The drawback with topic based publish/subscribe system is that all topics need to be predefined and further classified into topics. Although the infrastructure of the topic based publish/subscribe systems are simple, it still becomes difficult to connect

publishers and subscribers with predefined topics. This makes the whole purpose of using a publish/subscribe system irrelevant. A content-based publish/subscribe system fixes these constraints where publishers and subscribers specify their interest through event filters and event contents [Liu et al., 2005].

3.2 Content – Based Publish / Subscribe

The most commonly used publish/subscribe system type after the topic-based systems is the content-based publish/subscribe system. In content-based systems, the subscribers have the ability to describe attributes of the content of their interest using an expressive query language. The system then filters the matching content generated by the publishers based on the subscribers queries [Rose et al., 2007].

In content-based publish/subscribe systems, subscribers can express more in detail and specify constraints over the content they receive from publishers, network brokers evaluate these constraints provided by the subscribers and deliver the interested publications to the subscriber [Zhang et al., 2012].

An interesting feature of the content-based publish/subscribe systems is that they provide a subscription scheme for the subscriber based on the content published [Chaabane and Jmaiel, 2009]. In content-based publish/subscribe systems, the subscribers have more hold over the content they receive as they have the ability to filter the information published by the publisher. This has been the major reason for the popularity of the content-based system.

Despite various improvements in topic-based publish/subscribe systems, they offer only a static scheme, meaning a restricted or less expressive [Eugster et al., 2003]. Whereas content-based publish/subscribe systems are more versatile and flexible for the subscriber, letting the subscriber be more expressive and have more control over the content received. Subscribers subscribe to only selective events based on their interests by providing filters and are only notified upon occurrence of those events.

4. RSS Feeds

This chapter gives a detailed description about RSS feeds and an overview on the characteristics of RSS feed registries and aggregators.

4.1 RSS Feeds: An Introduction

RSS commonly known as Really Simple Syndication and Rich Site Summary is based on an XML query language that allows the syndication of lists of hyperlinks, along with other information like publishing date and author's name. This helps the subscribers to decide whether they want to follow the link or not. RSS is mainly a content-based publish/subscribe system used to syndicate news like information from news like websites, community like blogs, and many more. Any information which can be broken into discrete items can be syndicated via RSS. To enable RSS, a website owner needs to provide a standard XML format that is mostly compatible with many programs and machines through a channel or server. This XML page helps in fetching the most recent information for the subscriber through the list of hyperlinks provided [Camacho-Guerrero and Macedo, 2005].

RSS document is frequently called an RSS feed. RSS feeds involve publishers, subscribers and a software called "RSS Feed Reader" or "RSS Feed Aggregator". The main purpose of RSS is that it allows the subscribers to receive the information that they are interested in without having to visit the website manually each and every time. Subscribers subscribe from one or many websites using the feed reader either using a URL or by clicking the RSS feed icon available on the website. The feed reader fetches the information regularly keeping the subscribers up to date with the latest news.

The basic structure of an RSS document is self-describing and uses a simple syntax. RSS is written in XML which makes the elements case sensitive and must be properly nested. Listing 1 shows the basic structure of RSS.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
  <title>abc homepage</title>
  <link>http://www.abc.com</link>
  <description>abc content</description>
  <item>
    <title>RSS </title>
    <link>http://www.abc.com/rss</link>
    <description>New RSS structure on abc</description>
  </item>
  <item>
    <title>RSS as XML</title>
    <link>http://www.abc.com/RSSXML</link>
    <description>New RSS XML on abc</description>
  </item>
</channel>
</rss>
```

Listing 1*:* RSS

The basic structure of RSS includes a *channel* element which consists of the complete information of the website or the blog like ‘link’, ‘title’ and ‘description’. The <*channel*> element contains three required *child* elements namely <*title*> (defines the title of a channel), <*link*> (defines the link to a channel) and <*description*> (provides the description of the channel). Each <*channel*> element can have one or more <*item*> elements. Each <*item*> element describes the article. The <*item*> element consists of three required *child* elements namely <*title*>, <*link*> and <*description*> [Montgomery, 2003].

Table 6 lists the elements and their description in standard RSS v2.0.

Element	**Definition**
Title	The title of the channel
Link	The link of the website or blog
Description	The description of the feed
Language	(Optional) Specifies the language of the feed
Copyright	(Optional) Copyright material
managingEditor	(Optional) Specifies the managing editor of the content
pubDate	(Optional) Defines the last publication date
lastBuildDate	(Optional) Defines the last build date
Image	The image information of the feed

Table 6. RSS 2.0 Elements

Table 7 shows the required and optional elements of the <*item*> element.

Element	**Definition**
<author>	(Optional) Specifies the mail address of the author of the feed
<category>	(Optional) Defines the category of the feed
<comments>	(Optional) Allows an item to link to comments about that item
<description>	Required. Describes the feed
<enclosure>	(Optional) Allows a media file to be included with the item
<link>	Required. The link of the website or blog
<pubDate>	(Optional) Defines the last publication date
<source>	(Optional) Specifies the source of the feed
<title>	Required. Defines the title of the feed

Table 7. RSS 2.0 <*item*> Elements

Table 8 shows the required and optional elements of the <*channel*> element.

Element	**Definition**
<category>	(Optional) Defines the category of the feed
<copyright>	(Optional) Information on Copyright material
<description>	(Required) Describes the channel
<docs>	(Optional) Specifies a URL to the documentation format used in the feed
<generator>	(Optional) Specifies the program used to generate the feed
<image>	(Optional) Allows an image to be displayed
<language>	(Optional) Specifies the language the feed
<lastBuildDate>	(Optional) Defines the last build date
<link>	Required. The link of the website or blog
<managingEditor>	(Optional) Specifies the managing editor of the content
<pubDate>	(Optional) Defines the last publication date
<skipDays>	(Optional) Specifies the days where feed aggregators should skip updating the feed
<skipHours>	(Optional) Specifies the hours where feed aggregators should skip updating the feed
<textInput>	(Optional) Specifies a text input field that should be displayed with the feed
<title>	Required. Defines the title of the channel
<webMaster>	(Optional) Defines the email address to the webmaster of the feed

Table 8. RSS v2.0 <*channel*> elements

4.2 The Drawbacks of RSS

RSS is changing in the world of publish/subscribe, one must understand that it goes beyond news publishing and searching news. It is one step ahead by allowing information providers to communicate to their subscribers. Although RSS is a free and easy to use, there are difficulties in it which have not been addressed so far. The first one is the difficulty to identify and choose relevant data sources that match their interest

and to know how to subscribe to them. In RSS, there is no standard method or an agreed upon method to locate the feeds and subscribe to them. In simple terms, RSS feeds are just subscribed by entering the link or URL to the aggregator or feed reader. The second one is that RSS cannot work without a compilation of a feed reader or aggregator thereby making it dependable [Hochard et al., 2012].

Each RSS document consists of both static and dynamic information of the site. The <item> tag defines the story of the website subscribing to, with information like headline, title, URL and description. An example of the same can be found in Listing 1. Although the process of subscribing to a web feed is not as simple as visiting the website, there are three ways to subscribe to RSS feeds:

1. The easiest and the most common method is to subscribe directly through the web browser, called RSS auto discovery method, as most internet browsers include a RSS feed aggregator.
2. The second method is to use an online service provider, such as "Google Reader".
3. The third method is to use a desktop based feed aggregator to subscribe to feed, were users need to locally install the client application.

In the latter two methods, the subscribers need to manually enter feed URL of the websites or blogs they are interested onto the feed reader in order to receive feeds. The difficulty arises when the subscriber must manually search for the feed stream they are interested in and decide on a subscription process provided by the feed aggregator [Hochard et al., 2012].

4.3 Security Constraints of RSS

When discussing about RSS feeds it is very important to discuss its security constraints. RSS feeds can deliver any type of content, the publishers can include any type of executables or documents in the enclosure field of their feed. It is highly possible that these files can contain viruses or other type of unwanted programs. The developers of RSS feed readers usually take precautions when creating the program to ensure that if a feed contains suspicious file types, in such a case the programs provides a warning to

the user viewing the feed. The other problems concerning RSS is the potential exploits in both online and offline RSS aggregators or RSS readers. Some RSS feeds can contain HTML which include scripting language like JavaScript, exploits could occur if an infected RSS feed is viewed. The danger lies in the fact that many RSS readers, news aggregators automatically download the information contained in the enclosure field regardless of the file type or the source. Unfortunately, not all RSS readers and aggregators consider the possible security threats associated with RSS feeds. Some users will automatically download enclosures without warning or any thoughts of security.

As RSS feeds became more and more popular, security threats grew large. Even though publishers are finding new and innovative methods for RSS feeds, hackers are taking notice of the vulnerability that can be caused. Housley [2010] quoted, *"The power and extendibility of RSS in its simplest form is also its Achilles heel."* The vulnerabilities lie in the expansion capabilities of RSS specifications especially in the enclosure field, which is used to launch the podcasting scheme. Though the enclosure field in particular is not the problem as most RSS feeds do not use the enclosure tag. The enclosure tag is mainly used to link file types, images, documents, mp3 files and executables that could be found in most email spam. [Housley, 2010]

Problems may arise when a subscriber wants to subscribe to an individual section of a website or blog, instead of the all the sections in it. The subscriber cannot subscribe to the individual sections of a website or blog unless the corresponding section's URL and/or XML code is available. The reasons of which and why are discussed in the user experiment conducted for the purpose of this thesis in Chapter 8.

4.4 RSS Feed Readers and Aggregators

RSS being the most convenient software to receive news and updates is interestingly a dependable software. In order to use RSS one needs a special software called "feed reader" or "feed aggregator". There are a variety of open source and commercial readers available on the internet, some of which have been explained in Table 9.

Feed aggregator	Classification	Feed search	Type
Google Reader	topic hierarchy	category, description	social network, recommendation
Digg	category list	content	social network, recommendation
Netvibes	category list	keyword	widgets, mashup
Feedzilla	topic hierarchy	category, keyword	widget generator
NewsIsFree	category	keyword	feed discovery
Feedsee	topic	topic, keyword	keyword discovery
Search4rss	-	keyword	feed discovery
Syndic8	topic hierarchy	keyword	feed discovery

Table 9. RSS feed registries and aggregators (Adapted from [Hochard et al., 2012])

GoogleReader[8] is a RSS feed registry and feed reader that allows Google users to create a personalized hierarchy of RSS feeds. Feeds from websites or blogs can be searched via keywords and have to be added manually. One of the reason why Google Reader was popular was because it allows the integration of social network in its feed registry letting the users share and recommend feeds online among friends and family. Feedzilla[9] is a RSS feed registry that helps in collecting and categorizing the contents of the RSS feeds. Feedzilla lets the users filter and categorize their feeds, and also supports in building a user interface widget that enables in publishing the chosen news feeds. NewsIsFree[10] is a feed registry and also a feed reader where users have the ability to choose the feed contents based on feed category, feed name, feed date and feed language. Feeds can also be searched using name and description of the feed, Feedsee[11] lets the users search feeds in blogs and websites using topic and keyword search. This is unlike the above feed registries and readers which are mostly content-based. In Search4RSS[12] users can discover feeds from websites and blogs using a feature called 'discover feed'. This feed registry also has its own search engine for searching web

[8] http:// www.google.com/reader
[9] http:// www.feedzilla.com
[10] http:// www.newsisfree.com
[11] http:// www.feedsee.com
[12] http:// www.search4rss.com

feeds published by web pages. Syndic8.com[13] is a popular RSS registry mainly built and maintained for gathering syndicated news. It provides a number of RSS syndication tools to evaluate and validate various RSS services [Hochard et al., 2012].

RSS cannot function without a feed reader or aggregator, making it a dependable software. The feed aggregator can run on a local computer or can be online based. So what does the feed reader do exactly? The feed aggregator or reader regularly checks the websites or blog files that the users are interested in and when a new item has been added by the publisher or the website owner, the software notifies the user in a similar way a user is notified when a new email arrives. In the online version of the feed aggregator or reader, a website performs the same functionalities as the client application. When the user wants to view the information, he/she logs into the website and avails the information gathered by the feed aggregator or reader.

Figure 2 is a diagrammatic representation of how the websites, RSS feed and a local computer having a feed aggregator are related [Software Garden Inc, 2004]. It shows that a web browser is first used to read the files from the first website and then the second website. While the RSS XML file monitors the websites simultaneously by the RSS feed aggregator.

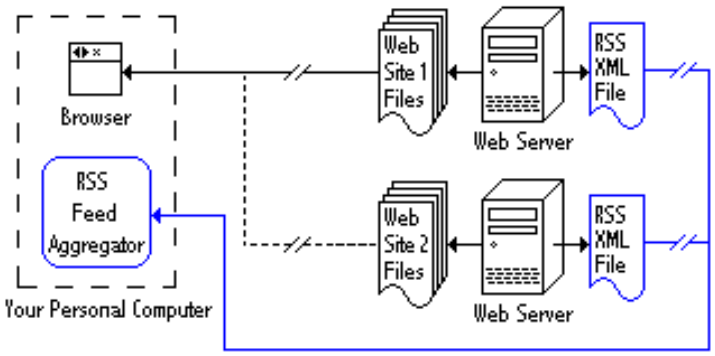


Figure 2. RSS feeds and feed aggregators

[13] http:// www.elsindi8.com

4.4.1 RoSeS-Content-based RSS Aggregator

RoSeS (Really Open Simple and Efficient Syndication) [Tomàs et al., 2011] is a content-based RSS feed aggregation system which allows individual users to create personalized feeds by defining content-based aggregation queries on selected collections of RSS feeds. RSS query language allows users to define personalized RSS feeds. The result of each query is a new feed that can be accessed locally and, if necessary, be published for other users. The RoSeS language can be explained in two parts, *the publication language and the subscription language.*

A publication language query contains three clauses:

– A mandatory *from* clause, which specifies the input feeds that produce output items, called main feeds.

– Zero, one or several *join* clauses, each one specifying a join with a secondary feed. Secondary feeds only produce annotations (no output) to main feed elements.

– An optional *where* clause for filtering conditions on main or secondary feeds.

A subscription language query allows defining subscriptions to existing publications or source feeds. A subscription specifies a feed, a notification, a periodicity and possibly a transformation.

For example, a user wants to create a new feed PubSubRoSeS.rss which aggregates all articles about RoSeS published by Tomàs et al. This can easily be translated into the aggregation query in Figure 3 which applies a simple conjunctive filtering condition on the union of the corresponding feeds.

register feed PubSubRoses
as http://www. PubSub.com/Roses/feed/ChanKey = PubSubRoSeS
Where item contains "RoSeS" and item.author contains " Tomàs*"*

Figure 3. An example of RoSeS query language as per Tomàs et al. [2011]

Tomàs et al. [2011] stated that "*RoSeS is a large scale RSS feed aggregation system based on a continuous multi-query processing and optimization"*. RoSeS is a simple and expressive aggregation language for RSS Feeds, when compared to centralized

server-based feed aggregators like GoogleReader, YahooPipes! And Cobra, RoSeS advocates a distributed client-based aggregation infrastructure which allows users to install and personalize their local feed aggregator.

4.4.2 Cobra-Content-based RSS Aggregator

Cobra (Content-Based RSS Aggregator) provides its users a distributed and scalable system for a personalized view of articles from millions of RSS feeds. Cobra consists of a three tiered network of crawlers that extract data from web feeds, filters that match articles against user subscriptions, and reflectors that serve matching articles on each subscription as an RSS feed, which can be using any RSS reader [Rose et al., 2007].

The most interesting part about Cobra is that it makes use of the offline service provisioning technique, using minimal amount of resources and supporting a given number of source feeds. The number of crawlers, filters, and reflectors, and the interconnectivity between these services helps the technique to determine the configuration of the network.

Crawlers, filters, and reflectors constitute the three-tier architecture of Cobra. Crawlers periodically crawl web feeds, such as blogs, news sites, and other RSS feeds which are collectively termed as *source feeds.* Cobra crawlers make use of different techniques to reduce polling load. They check for updates in a lightweight manner in the source feeds while a typical blog or news feed will present the most recent articles only. The content of new articles is sent to the filters which match it using a case-insensitive and index-based algorithm against the content of those selected by user subscriptions. An appropriate reflector receives the articles matching a given subscription by push mechanism and then presents a personalized RSS feed to the end user which can be browsed using a feed reader. The reflector caches the last ‘n’ matching articles for the feed (where n is typically 10), requiring that the user poll the feed periodically in order to detect all the matching articles. Many existing RSS feeds that limit the number of articles included in the feed show same similar behavior. Although the reflector must be polled by the user (as required by the current RSS standards), yet this polling traffic is very low as compared to the requiring users to poll many thousands of source feeds [Rose et al., 2007].

5. Spamming in Publish/Subscribe Systems

This chapter deals with the kind of spam encountered by the internet users while using publish/subscribe systems such as blogs and email. An overview of caching in content-based publish/subscribe system is also presented.

Spam is a well known problem related to internet based applications. The spammers find internet a haven for creation and distribution of plagiarized content. Spamming in blogs and in email applications has not been studied well though the past research has advocated that spam can be minimized on the World Wide Web (WWW) [Kolari et al., 2006].

Spam blogs, or splogs feature plagiarized or auto generated content. They create link farms to promote affiliates, and are motivated by the profitability of hosting ads. Splogs are generated with two often overlapping motives. The first is the creation of fake blogs, containing gibberish or hijacked content from other blogs and news sources with the sole purpose of hosting profitable context based advertisements. The second, and better understood form, is to create false blogs that realize a link farm [Kolari et al., 2007].

Spam blogs or splogs are identified by plagiarized or auto generated content and creation of link farms to promote advertisements. They are motivated by by their affliates and profitability of hosting ads. The motivation could be the creation of fake blog that may contain hijacked content from other sources with the purpose of hosting profitable ads [Kolari et al., 2007]. A detailed description about link farms, fake blogs and other modes of spamming has been presented in the following sections.

5.1 Blogs

Blogs and the blogosphere are characterized by the following features: (i) the blog hosts host them freely (ii) content syndication for distribution is provided by them, (iii) remote web service applications for publishing are supported by them [Kolari et al., 2006].

The term ‘Blog’ is a “*contraction of the term weblog*”. Blogs can be used for any topic, but are usually used by bloggers to share and exchange information on various subjects such as personal life to food and video games. They also provide personal opinions on the same. Due to their informal tone of language, blogs are used by companies to build

and maintain relationships with their customers. Blog articles are often referred to as posts. [Bursztein and Mitchell, 2009].

Kolari et al. [2006] indicated that spam ranges as high as 75% at ping servers, to around 20% at blog search engines. A similar trend can be noticed in web search engines in general where a user searches for a particular string and the results are flooded by irrelevant content and back links.

5.1.1 TrackBack Spam in Blogs

Cross-references between blogs can be inserted using the TrackBack mechanism. The TrackBack interface can be used by a new blog post citing an older one to insert a link in the older post automatically. According to Bursztein and Mitchell [2009], TrackBack is important because it provides link reciprocity.

TrackBack mechanism works well for two reasons. First, it is more time consuming and error prone for a blogger to notify each blog cited in a post, and second it is tedious for a blogger to add manual notification to all the blogs that cite it. These features of TrackBack attracted malicious users to use it soon after it appeared. TrackBack mechanism may be utilized to perform search engine optimization in addition to lure users to malicious sites. A small quantity of TrackBack holds a potential of providing large amount of internet traffic. Thus, spammers might lure thousands of blog readers to a particular site using one spam TrackBack [Bursztein and Mitchell, 2009].

5.1.2 Splogs and Ping Servers

Blogs may use standard interfaces defined by ping servers to notify new or updated posts. The pings usually are associated with the blog title, homepage and sometimes with syndication feed location. Pings are restricted only by their frequency. Being restriction free in nature and by providing an improved exposure to search engines, ping servers are frequently buzzed by Splogs.

Ping servers face two kinds of spams - (i) pings from non-blogs, and (ii) pings from splogs, both of which could be referred to as spings. Splogs constitute around 88% of all pinging URLs but they account for only 75% of all pings. This follows from the fact that many splog pings are one-time pings [Kolari et al., 2006].

Subsequent pings do not use the same URL but specify arbitrary pages as blog homepages even though they have no relationship with blogs or the blogosphere.

Zombie pings, spings that exist even though the splog (or page) they represent is non-existent (or is already eliminated) in the blogosphere are one of the favorites of the spammers. Most of the popular web search engines give particular importance to the URL tokens of page. Splogs exploit the ranking criteria of search engines with the help of similar but fake URLs by hosting blogs in the info domain, where domain registrations are less expensive and easily available [Kolari et al., 2006].

5.2 Email Spam

Tarkoma [2006] explained about email spam which comes in two forms inbound and outbound spam, with inbound spam originating from and outbound being sent to a foreign network. Spam originates from networks infested with a host machine (zombie machine) that has been taken over by spammers or their helpers, e.g., using Trojans or viruses. Prevention of spam is not possible with any one particular technique. Spammers are difficult to get identified and located as they use several techniques such as open relays, spoofing and zombie machines.

In a similar manner, publish/subscribe spam may also be classified into inbound and outbound types. Publish/subscribe systems are typically multi-hop, filters describe their end points and have a static or configured topology to support efficient online filtering. In the inbound case, an active filter that matches a lot of unwanted messages signifies the presence of spam in a client's queue. The client has the ability to change the filter to a more concise using spam detection filters that are available. The client has also the option to use a combination of techniques such as sender verification, white listing, black listing, computational puzzles, grey listing, sender verification, and content filtering. In short, the client application is responsible for the reduced performance impact of unwanted messages [Tarkoma, 2006].

Tarkoma [2006] estimates client interests through filters. Spammers operate with email address in email spam and in publish/subscribe they operate with filters. For spammers to maximize their throughout, the author emphasizes on two ways such as "out-of-band"

and “online”. For out-of-band the spammers have external information of the clients through long term monitoring of the users interests and application behavior. In online, the spammers try to compromise the servers of the publish/subscribe networks with the help of brokers. This way the spammer has complete understanding of the filters used which helps them to reconstruct the subspaces through frequently used filters [Tarkoma, 2006]. Since publish/subscribe systems have a different architecture than email, therefore the ways of prevention of spam should also be different.

5.3 Caching in Content-Based Spam

Any message is supposed to reach all interested destinations in a publish/subscribe system [Baldoni et al., 2003]. This holds good for all the active clients and therefore their subscriptions are available in the system at publish time. However, it is possible that a client joins the network after the publishing of an interesting message in a dynamic environment. A new subscriber in publish/subscribe system cannot retrieve messages matching to the subscription that have been published earlier. Hence, the retrieval of previously published content with the help of caching is one of the most challenging problems in publish/subscribe. A large part of network traffic is redundant despite extraordinary volumes. Multiple users, at any given site, request for almost the same content. Caching facilitates replication of the content and serve identical requests locally, and prevent them from over utilizing the network resources. The content is stored on a storage device that is physically or logically closer to the user by a cache.

Sourlas et al. [2009] have described some key points through the *caching points and request/response mechanism* with which they have proposed the enhancement of retrieval of previously published content in publish/subscribe systems.

In caching points, caches are installed in brokers and a request/response mechanism is introduced with the aim to provide a publish/subscribe system with the ability to ensure the availability of old information for future clients.

Furthermore, Sourlas et al. [2009] quoted the following to elaborate the caching points:

"In their system each broker is selected as a candidate caching point for a message as long as it has in its subscription table at least one client subscribed in this message. A published message is transferred to all brokers with client subscribers. Also, a broker with a client subscriber is easily reached by a request message."

In request/response mechanism when a client requests for a previously published content from the network, he/she makes a request message apart from subscribing. The request message works similarly to the publish message, but in this case it carries along with it a series of broker identifiers. Brokers upon receiving the request message check for a subscription filter matching it. For every matching subscription filter a request message is passed on, when there is no subscription filter, the request message is dropped. Only when a matching is found a response message is initiated. The response message carries the previously published messages as well as a sequence of nodes. Once a broker receives a response message, it pops off its identifier from the sequence and forwards it to the first broker of the remaining sequence. Using the above procedure, every new subscriber and only that one will receive every old message matching its filter. Although the mechanism proposed by the author helps in retrieving the old content, it lacks in addressing the problem of event replication and also fails to discuss about the spamming caused due to the mechanism.

6. RSS Spam and Socio-Technical influence

In the context of the this thesis, we refer RSS spam to be any content that has not been subscribed to by the user.

RSS spam is not the same as what you get in your email inbox. With email spam you are getting unwanted messages. RSS spam targets directories and search engines rather than the end user itself. RSS feeds are indexed by search engines and directories and can be considered as news. This, however, has not gone unnoticed by the spammers who are using RSS to spam search engines. RSS spam largely consists of *four* main types most often found in RSS search engines. The first type is keyword stuffing, the second type involves RSS feed link farms, the third type is the creation of fake RSS feeds and the fourth is event replication.

Key word stuffing involves in filling each RSS feed article with high value keywords for a specific topic. The articles are not for the end users (human visitors) but instead are for the search engine robots to direct traffic to a target website. This technique is nothing more than an adaptation of typical keyword stuffed web page, often banned by major search engines.

The RSS feeds involving RSS feed farms contain very little content, mostly a simple keyword. Their main attraction is their feed title. When one enter the title it is routed to a blog containing tens or hundreds of other blogs and RSS feeds, each of which redirecting to more other links within the farm. The goal is to trick the user to clicking advertisements or directing them to a product website.

Fake RSS feeds appear to be legitimate but are often duplicated article content. Whether the content is valuable or not is debatable. These feeds are usually created in mass, using automated scripts, and appear to be similar to that of link farms. By attracting the users to seemingly valuable content, they hope to gain advertisement clicks or product website traffic.

Event replication is the main issue in publish/subscribe spam. In simple terms, event replication means the forwarding and replication of a published event until it has been circulated throughout the network. Event replication is by in itself one of the most scalable techniques to disseminate spam. Filters inferred by the spammer may generate

notifications generated and delivered to the neighboring brokers. However, this does not guarantee the spammer a 100% circulation of notifications [Tarkoma, 2006].

6.1 Socio-Technical Influence of Spam

A socio-technical system is a social system built upon a technical base (hardware and software). A social system may arise from technology (social media) or the physical world (human interactions) [Whitworth, 2004].

Whitworth and Liu [2009] redefined spam as ‘tragedy of the commons’ in new technical clothes. They advocated that socio-technical problems can be solved only by socio-technical solutions. This is because neither technological answers like filters nor social answers like laws can solve them. Social or technical responses alone seem powerless against sociotechnical problems like spam. Spam is just a face for a whole genre of antisocial acts that threaten online society, including spyware, phishing, spoofing, scams, unwanted pornography, identity theft, libel, privacy invasions, piracy, plagiarism, and online harassment.

Social or technical responses alone seem powerless against socio-technical problems like spam. spam is just a face for a whole genre of antisocial acts that threaten online society, including spyware, phishing, spoofing, scams, unwanted pornography, identity theft, libel, privacy invasions, piracy, plagiarism, and online harassment.

Social engineering is a non-technical means of intrusion used by hackers to interact with humans which often involves tricking people into furnishing private information. The term ‘social engineering’ meant to represent smart methods that solve the social problems. Due to the positive ideas related to the word ‘engineering’ it was appropriated for various social problems of the time. The use of cultural tactics, social disguises and tricks to facilitate illegal use of computer systems and networks may be defined as social engineering [Erbschloe, 2004; Hansson, 2006]

6.2 Socio-technical attacks

In this section, we briefly explain major socio-technical attacks, i.e., the techniques used by attackers/spammers in socio-technical aspect. Most of the attacks are directly or indirectly related to malware infection and gathering user information in unethical ways.

Malware

Several types of invasion programs acting as parasites are designed to install and maintain themselves on a computer without the permission of the user. They track the activities and the usage details of the computer once they get installed. The information is gathered and sent across the internet to malicious user(s). In addition, unauthorized websites may install desktop items or plug-ins to the web browser with the intention of collecting information or infecting the computer. Some programs are inspired by social engineering and phishing techniques. On the other hand, some programs force the users to follow a particular set of steps or instructions before allowing them to access a particular program on their computer or login to the computer itself.

These kind of programs, malicious software, plug-ins, web bugs, worms, viruses, Trojans can be collectively be called as malware [Abraham and Chengalur-Smith, 2010; Erbschloe, 2004; Ivaturi and Janczewski, 2011; Luo et.al., 2009].

The malware attacks are considered to be most successful of all types of socio-technical attacks because malwares are persistent and pervasive. The malware involves both technical and psychological tactics to intrude and maintain itself into a computer [Abraham and Chengalur-Smith, 2010; Ivaturi, 2011]. The attackers may use social skills in order to persuade the victim to perform an action that is beneficial to them. They try to exploit anything related to the users to bring greed, fear or curiosity in them and then make them their prey. Another reason for which malware attacks have been successful is their availability on the internet in different forms and platforms [Ivaturi and Janczewski, 2011].

Pop-ups

Computer users usually encounter unwanted alert messages while browsing on the internet or using a web based application/software. These messages open in a new window of the web browser with the intention of online advertising and marketing

[McCoy et.al., 2007; Palmer, 2005]. A number of instances have fake messages, obscene images and/or graphic content in these messages. These kind of new window messages or alert boxes are called pop-ups. The attackers present these messages to lure or scare the users to convince them to download a particular software which is a malware in disguise [Abraham and Chengalur-Smith, 2010; Erbschloe, 2004; Ivaturi and Janczewski, 2011].

Search engine poisoning

Search Engine Optimization (SEO) is a collective term for a number of tricks and techniques to elevate the rank of a web link on a search engine i.e. to facilitate easy searching of a web link using particular keywords and increasing the number of visitors. The major search engines viz. Google, Yahoo, Bing issue support guidelines on how to improve search result rankings. Search engines have become the first choice of the users on the internet to dig out the most relevant form of information. Most of the traffic (number of visitors) on a website is governed by the search engines. As such, website owners strive to boost the number of visits on their websites by optimizing their exposure in relevant search results. [Howard and Komili, 2010; Leontiadis et.al., 2014; Lu and Lee, 2011]

Even though legitimate SEO techniques are used and encouraged by the search engines, dishonest web developers and attackers may choose to abuse these techniques to get a favorable ranking which is referred to as ‘blackhat’ SEO. The attackers lure the users to their websites using unethical practices and blackhat SEO. Lu and Lee [2011] studied malicious search engine redirection with a deeper analysis on blackhat SEO. This practice of conducting SEO attacks luring the web users to visit malicious websites, fake links, etc. is called Search Engine Poisoning (SEP). [Howard and Komili, 2010; Ivaturi and Janczewski, 2011; Lu and Lee, 2011]

Howard and Komili [2010] discussed different types of SEO attacks including keyword stuffing, farms, fake web links, etc. They explained how hackers/attackers automate search engine poisoning attacks to distribute malware. The users have trust in the search results provided by the search engines. The attackers exploit this trust to launch malware attacks. SEP is becoming popular as it doesn’t require social skills as in a typical social engineering technique.

The terms keyword stuffing, fake feeds, link farms, event replication have been introduced in the beginning of this chapter in the context of RSS feeds. As such there is a similarity between the techniques used for spamming in RSS and different types of socio-technical attacks. Due to this similarity, we can say that there might be a possible socio-technical influence on RSS spamming. We will try to investigate further on this possible influence as per our research question with the user experiment.

7. Formal Specification of Open Standards and the Case of RSS v2.0

In this chapter, the relevant research content with respect to formalizing RSS is presented. The author along with *five* other members of a research group at University of Tampere tried to establish a relation between open standards and formalization and presented the same at *18th Panhellenic Conference on Informatics - PCI '14* in Greece.

7.1 Open Standards and Formalization: An Introduction

An 'Open Standard' refers to a format or a protocol that is subject to full public assessment without any usage constraints. Open standards allow people to freely use and transfer data through fidelity. In the open source software community open standard means that it is open and can be freely adopted, implemented and extended. Usually open standards are either un-owned or owned by a collective body. There are many definitions of open standards from national IT agencies, Interoperable Delivery of European Government Services to Public Administrations, Businesses and Citizens (IDABC), and World Trade Organization (WTO). Open standards specifying formats are sometimes referred to as open formats [Barlas et al., 2014].

Formalizing industrial standards and communication protocols is not a new idea. Recent research and development results have outlined the importance of formalizing standards in various industrial areas and production lines. In brief, a formalized standard can i) enhance communication and understanding among various stakeholder groups, ii) be a management tool for various management teams, and iii) standardize production and production lines.

7.2 Formal Specification of Open Standards

Formal specification involves investing a lot more effort upfront by mathematically modelling the constructs in the early stages of software development. This reduces requirements errors as it forces a detailed analysis of the requirements. Incompleteness and inconsistencies can be discovered and resolved. Therefore, savings are made as the amount of re-work due to requirement problems is reduced. The algebraic approach of formal specifications focuses on specifying a system in terms of its operations and the relationships between those operations. Types of data are formally specified along with operations on those data types. The implementation details, such as the size of representations are quite abstract in nature.

Benefits of using the formal specification method

- Fewer ambiguity issues: Natural language specifications are informal and usually contain ambiguities. Even if written very carefully, no one can ensure that when an individual reads this specification in order to make use of the standard, his/her understanding of how things should work matches exactly what the designers of the standard had in mind. While the natural language specification can not really be eliminated, as it is way more natural for humans to start with a formal specification, its involvement can be minimized: Instead of using a natural language specification all the way through the standard building process, it can be used to begin with (requirements part) and then use the formal version. Using natural languages to carry out requirements during phases can lead to misinterpreting errors, due to the obvious linguistic ambiguities. Formalizing that means of communications reduces that factor, since mathematical specifications can only be interpreted in one way

- More concrete system design: A standard's interoperability depends on the precision of its requirements. The better the requirements of a standard are specified the easier it is to make full, correct, use of the standard, especially when it is a part of an interoperable system. Besides, there might even be a financial gain, as a standard that has been formally specified early on requires less maintenance costs and is easier to upgrade.

- Important to have a means of specification where you are going to be able to verify formally and ensure that you can go from phase to phase without losing track of the original requirements.

- Verbosity of the specification: Formal specifications of standards are significantly more compact than the ones written in natural languages. One example of that is the specification for the format of ARPA Internet text messages [Crocker, 1982] that is almost 40 pages. A formal specification of that could be just a few pages. Also, a

well written specification of a small module can be applicable in other bigger systems as well, so re-usability shrinks the size even more.

- Under circumstances (using an algebraic specification methodology), we can extend the specification, allowing for property checking and verification. Verification most often makes sure that the standard is working as designed regardless of the implementation. Of course the formal specification may not be always executable, so there is transformation needed towards programming language. If the transformation is not algorithmically and automatically done, with correctness preserving transformations, nothing is gained [Dijkstra, 1981]. That is, if human activity is needed in the transformation, the correctness of the specification does not prove that the implementation does the same as what was specified.

7.3 Formalization of RSS v2.0 with CafeOBJ

RSS' v2.0 specification files can be found in RSS Advisory Board [2014]. The specification files provided there are quite different from the kind of documents you would expect to find because, while every single parameter of RSS is explained, sometimes the explanation can be quite messy and sometimes unclear. The RSS v2.0 specification files provided in RSS Advisory Board [2014] are quite verbose, as all natural language specifications, but surprisingly less verbose than usual specifications.

CafeOBJ is a new generation algebraic executable, industrial strength algebraic specification language/system. The main underlying logics of CafeOBJ are order-sorted algebras [Diaconescu and Futatsugi, 2000; Goguen and Meseguer, 1992] (used to specify abstract data types) and hidden algebras [Barr et al., 1998; Diaconescu and Futatsugi, 2000] used to specify abstract state machines, providing support for object oriented specifications. As a direct successor of OBJ, it inherits all its features (flexible mix-fix syntax, powerful typing system with sub-types, and sophisticated module composition system featuring various kinds of imports, parametrized modules, etc.) but it also adds combinations of rewriting logic and hidden algebra. Listing 2 displays a sample CafeOBJ module. Keyword *mod!* declares the module with tight semantics. List is the visible sort of that represents a list, and sort Elt is a subset of that, representing an element of that

list, both declared by enclosing [and]. *pr(NAT)* denotes protecting import of module NAT (built-in module specifying natural numbers). The keyword op is used to declare operators. The operator *nil* is a constant denoting the empty list. Operator _|_ takes an element and a list and returns the new list with that item merged and _//in_ searches if a given element is inside a given List. Keyword var (vars) declares a CafeOBJ variable(s), while the equations defining axioms of the specification begin with *eq* or *ceq* when the equation is conditional. The equations on this module define the behavior of the operators we declared. The CafeOBJ system uses declared equations as left-to-right rewrite rules and reduces a given term. Usually, we write equations for each operator that observe a system's state (observational operator) over each operator that changes the system state (transition operators). We usually write equations for the operators that observe a system's state (observational operators).

```
mod! LIST (X : : TRIV) {
pr (NAT)
[ Elt < Li s t ]
op n i l : −> Li s t .
op | : Elt Li s t −> Li s t .
op // in : Elt Li s t −> Bool
op = : Li s t Li s t −> Bool {comm} .
vars L L1 L2 : Li s t . vars E1 E2 : Elt .
eq (L = L) = true . eq ( n i l = (E2 | L2 ) ) = f a l s e .
eq ( (E1 | L1) = (E2 | L2 ) ) = (E1 = E2) and (L1 = L2) .
ceq (E1 // in E2) = true i f (E1 = E2) .
ceq (E1 // in (E2 | L) ) = true i f (E1 = E2) or (E1 // in L) .
eq (E1 // in n i l ) = f a l s e .
```

Listing 2. A Sample of CafeOBJ Module

The *channel* is the most important block of an RSS feed. It contains three mandatory elements (link, title, description) and a number of other, optional elements. To declare such a module in CafeOBJ, we import the modules that correspond to those three required elements (*pr* command) and a module that contains all the other optional elements

(*CHANNELOPTIONAL*). Then we introduce the sort *Channel*, the transitional operators that create an empty channel (*createchannel*) and then set the channel details. Operator *setchannel* is declared twice since a channel can be created with or without the optional elements. Observational operators get$_{XXX}$ return the link, title and description of a given channel. Observational operator *optionalcontent-exists?* is true if the channel contains any of the optional elements and if so, observational operator *getoptionalfrom-channel* returns this content.

Channel module protects other modules, like *link, title*, etc. Those modules have to be declared before the channel, as CafeOBJ follows this bottom-top approach. To define the module title, we protect CafeOBJ's built in module, string, that introduces the string sort and also provides useful utilities for string manipulation. Sort title is a sub sort of String, as all titles are strings but not the other way around. Then we have transitional operators *createtitle* (creates an empty title) and *settitle* (sets the contents of the title) and observational operators *gettitle* (returns the title) and **c-***propertitle* that is only true if we have used the *settitle* operator to create and set the contents of a title. An important benefit of formal specifications is the re-usability of the modules. For instance, both the *channel* and *item* blocks use the *link, title* and *description* entities. So, if we specify those three modules then both the channel and item blocks can import and make use of those modules. In fact, since all of those three modules are quite similar,we can declare a generic module (Listing 3) and use CafeOBJ's module term importing/renaming to create the three different modules (TITLE, DESCRIPTION and LINK) out of just one module declaration (BUILDINGBLOCK) [R.ăzvan Diaconescu et al., 1999]. So, Listing 3's title reference, creates the *Title* module by importing BUILDINGBLOCK and renaming sort *MS1* to *Title*, operators *create* to *createtitle*, *set* to *settitle*, *get* to *gettitle* and *c-proper* to *c-propertitle*. The * in the module's declaration tells us that this module is declared in loose semantics as it can be used as a constructor for other modules.

```
mod! CHANNEL {
pr (LINK + TITLE + DESCRIPTION + CHANNELOPTIONAL)
[ Channel ]
op createchanne l : −> Channel
op setchanne l : Link Title Description −> Channel
```

```
op setchanne l : Link Title Description
        Channeloptional −> Channel
op getlinkfromchannel : Channel −> Link
op gettitlefromchanne l : Channel −> Title
op getdescriptionfromchannel : Channel −> Description
op optionalcontent−exists ? : Channel −> Bool
op getoptionalfromchannel : Channel −> Channeloptional
var C : Channel .
var L : Link .
var T : Title .
var D : Description .
var OPT : Channeloptional .
eq getlinkfromchannel( setchannel (L,T,D) ) = L .
eq getlinkfromchannel( setchannel (L,T,D,OPT) ) = L .
eq getlinkfromchannel ( setchannel (L,T,D) ) = T .
eq getlinkfromchannel ( setchannel (L,T,D,OPT) ) = T .
eq getlinkfromchannel ( setchannel (L,T,D) ) = D .
eq getlinkfromchannel ( setchannel (L,T,D,OPT) ) = D .
eq optionalcontent−exists ?(C) = if  C = setchannel (L,T,D,OPT) then true
                                    else false          fi.
ceq getoptionalfromchannel (C) = OPT
if optionalcontent−exists ?(C) .  }
```

Listing 3. The Channel module.

The *Image* module is quite similar to *Channel* module; an image has to include a url, a title, a link and optionally some other elements too. So we have the appropriate module imports, the new sort introduction (Image), transitional operators to create and set the image details (twice declared as we do have optional elements) and observational operators that return the title, url, link and if present, the optional elements given an image. Listing 4 displays the Image module.

What's interesting is the IMAGEOPTIONAL module that contains all of the optional elements that can accompany an image declaration. Those elements are < width > and < height > numbers, indicating the width and height of the image in pixels and < description > that contains text that is included in the TITLE attribute of the link formed around the image in the HTML rendering [RSS Advisory Board, 2014]. To model this requirement we use CafeOBJ's record structure. A record consists of fields and an element of the type is completely determined by the value of each field (slot). When we declare a record: i) A sort with the record name is declared, ii) a mix-fix operator is declared, so that a term of the form record − nameslot − name1 = value1, slot − name2 = value2, ... is an element of record-name. Moreover, the slot-value pairs in the braces may be written in whatever order, and you may even omit some of them (helpful since all of those three elements are optional). Finally, for each slot, two access functions are declared and defined; one that returns a slot-name given the record-name and one that sets a value to a slot-name of a record-name [Nakagawa et al., 1999]

```
mod IMAGE {
pr (LINK + TITLE + URL + IMAGEOPTIONAL)
[ Image ]
op createimage : −> Image op setimage : Url Title Link −> Image
op setimage : Url Title Link Imageoptional −> Image
op gettitlefromimage : Image −> Title
op geturlfromimage : Image −> Url
op getlinkfromimage : Image −> Link
op optionalcontent−exists ? : Image −> Bool
op getimageoptional s : Image −> Imageoptional
var U : Url
var T : Title
var L : Link
var I : Image
var IOPT : Imageoptional
eq gettitlefromimage ( setimage (U,T,L) ) = T .
ceq gettitlefromimage ( setimage (U,T,L, IOPT) ) = T

    if ( properimage ?(IOPT) ) .
```

```
    eq geturlfromimage ( setimage (U,T,L) ) = T .
    ceq geturlfromimage ( setimage (U,T,L, IOPT) ) = U
        if ( properimage ?(IOPT) ) .
    eq getlinkfromimage ( setimage (U,T,L) ) = L .
    ceq getlinkfromimage ( setimage (U,T,L, IOPT) ) = L
        if ( properimage ?(IOPT) ) .
    eq optionalcontent−exists  ?( setimage (U,T, L, IOPT) ) = true .
    eq optionalcontent−exists ?( setimage (U,T, L) ) = false .
ceq getimageoptional s ( I ) = IOPT if optionalcontent−exists ?( I ) .  }
```

Listing 4. The Image module.

RSS limits the dimensions of an image; width can not be more than 135 pixels and height can not be more than 400 pixels. To model that, we create three new observational operators; *properheight?* becomes true if the height is within range, same as *properwidth?*. Operator *properimage?* becomes true if both image dimensions are within range

An item can contain a link, a title and a description. All elements of an item are optional, however at least one of title or description must be present [RSS Advisory Board ,2014]. To model that, we declare the *setitem* operator five times, with the possible combinations that can take place, just like *setchannel* operator from the Channel module. A channel though may contain many items and in order to model that properly, after we declare the Item module, we declare a module called C-ITEMS, that acts as a list of items. To do that we use the LIST module that we have declared in Listing 2, with some sort and operator renaming. C-Items can hold an arbitrary number of Items and this is exactly what a Channel may contain. We've modeled the RSS v2.0 standard in CafeOBJ, preserving the exact same structure (entities containing other entities), ensuring that some elements are required (or that entity won't be accepted as proper) while others are optional (including them does not make a difference, just as long as they have been properly set) and made sure that some properties of the specification are held, e.g., image dimensions have to be within some limits, date restrictions. That concludes the formal specification of RSS v2.0 standard with the help of CafeOBJ.

7.4 Research Results

The author with the research group presented the idea of applying formal specification techniques to open standards specifications and demonstrated what we support and mean by formally specifying RSS v2.0. This is a novel and original work. No-one else ever made a formal specification of an open standard, and this is the reason that there is no related research work to compare and contrast our work. To our knowledge and until now, there has been no similar published work in scope, aims and results in the research field.

Among the reasons for not attempting formalization of open standards and other standards in general might be the feasibility of the approach. One might ask the general question: are formal methods applicable to all types of standards? RSS looks like a suitable case, but in other cases (e.g. rather richer and more complicated standards like the Creative Commons or other) might require a different specification approach and formal method. The latter might be more or less known to different groups of people. We chose CafeOBJ, an inhouse specification tool that we are very familiar with in modeling and specifying concepts like those of open standardization.

Unfortunately many software engineers are not educated enough in the use of formal methods, or they apply them very rarely. Although formal methods have a slow learning curve, they are also easy to forget and, as mentioned earlier, standards' readers could be alienated if not sufficiently prepared when reading a formally specified standard.

8. Investigation through User Experiment

This chapter elaborates the course of the user experiment which includes the user task, the target group, the material collection process (the interview method) followed by the inferences. An overview of how RSS is in today's world is presented. This is important in terms of understanding the overall experience of using RSS.

Since the research conducted for this thesis does not have a specific or well formed background to compare or analyze, it is not appropriate to formulate specific or detailed hypothesis. We decided to use a free text form with generic questions about RSS feeds, RSS spam, RSS feed readers and aggregators and the overall user experience.

8.1 The Target Group

By nature, the research area of this thesis is related to the experience of the users, namely how end users (students in our case) consider using RSS in today's world compared to better technologies like Twitter or Facebook. This requires that the users participating in the user experiment are able to think creatively and are open to technology.

In general, the target group used in this user experiment were students using social network systems. This was because, the study conducted for this thesis required people to be using the on going social network systems such as, Facebook and Twitter. The students were from an undergraduate program majoring in interactive technology from the University of Tampere. Furthermore the goal was to have students with long experience of social media as the research tasks required students to have knowledge of how to subscribe to websites or blogs etc., using RSS feeds. The target group was not limited to certain nationality of age or language. Both genders were covered. About 30 people had agreed to participate in the study among which only five participants had previous experience in using RSS.

8.2 Guidelines for the User Experiment

The user task was supposed to be performed individually and required users to perform the task for a period of *two* to *four* weeks. The task list first consisted of a manual providing basic information about RSS feeds, feed aggregators and their 'how to use'.

The students had to choose from a list of websites or blogs provided in the task list along with a feed aggregator which could be either web based or client application on the local computer to subscribe to feeds. They had the oppurtunity to choose any feed reader or aggregator of their choice. The task list along with the manual was distributed to the students via email.

The choice of websites and blogs for this thesis were choosen based on their popularity and the frequency of polling. The list of website and blogs are mentioned in the list below:

- TechCrunch
- Simply Recipes
- BoingBoing
- Mashable
- Read/Write web
- John Battelle's Searchblog
- 43Folders
- 37signals
- DumbLittleMan
- Interesting Thing of the Day
- CrunchGear.

The most important part of this user experiment was to subscribe to individual sections of websites or blogs, an example of which can be seen in Figure 4.

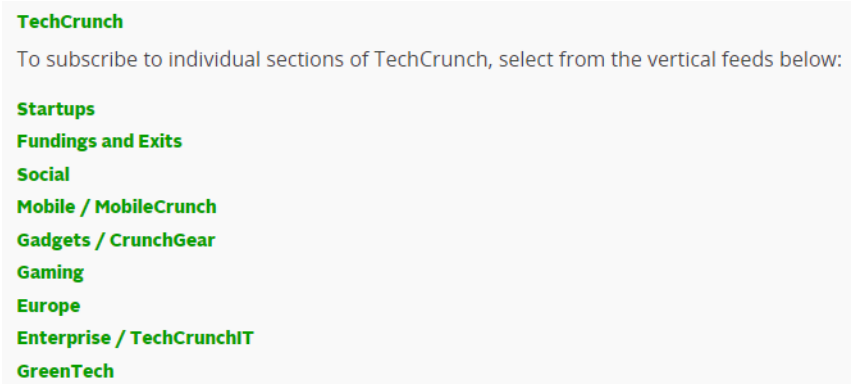


Figure 4. An example of subscribing to individual feeds (TechCrunch)

At the end of the *two* to *four* weeks period the students were advised to submit a free text form addressing the following questions:

a) How relevant was the content you received with respect to the content you subscribed to in your feed aggregator ?

This is necessary for analyzing the content they recieve in terms of spamming. According to this thesis we define any content recieved by the user that has not been subscribed to as RSS spam.

b) How was your user experience with RSS and RSS feed reader? Did you face any difficulties ? If any, specify.

The aim of this question is to understand the ease of use in RSS. RSS being a dependable software which means not just mastering one but two or more softwares in order to use it.

c) What was the choice of RSS feed reader and why ?

This helps in undertanding the various feed readers and analyzing the functionality of different feed readers. This also helps in understanding what makes each and every feed reader unique and popular among users.

d) How do you rate your overall experince using RSS feeds? Will you continue to use RSS feeds?

This explains the future of RSS among users and provides users experience of the product when compared to today's technologies such as Twitter.

8.3 The Material Collection Process

For the process of material collection, the interview method was choosen for getting comments and feedback on the task. The purpose was to also test if the task was understandable and if it had changed their views about RSS. The interview method proved to be a good choice as the participants had the ability to explain more in detail about their user experince and the secuirty issues related to RSS. After the task was submitted online, the participants were called for a personal interview.

The research material used in this user experiment consists of the inputs recieved via the free form text and the interview process. About 30 students participated in the user experiment. Other tools used in the study were the RSS feed readers.

8.4 Demographics

The user experiment was succesfully completed by 23 males and *seven* females. Ages of participants varied from 20 to 30 years. Most of the participants were Finns, the others were from India, Ukraine, and Russia. Only five out of 30 participants had used RSS before the experiment. This indicated that the participants may find difficulties in using RSS feeds. The rest had some distant knowledge of RSS but have not used it until the study.

8.5 Inferences

In this section, the research objectives and task results are addressed, and conclusions are made based on the results derived from the user experiment.

There were a number of interesting findings as an outcome of the user experiment conducted. The inference are illustrated with the help of Figures 5-8.

a) *How relevant was the content you received with respect to the content you subscribed to in your feed aggregator* ?

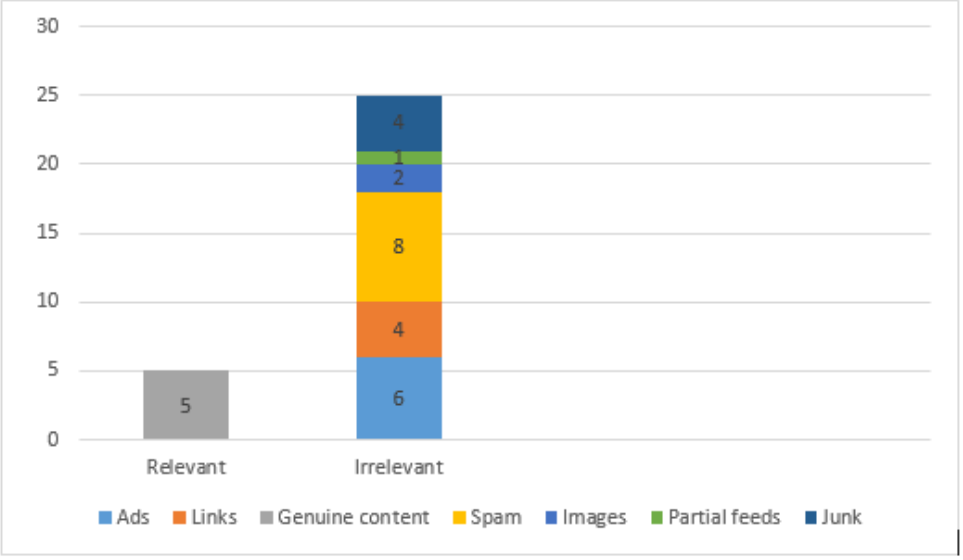


Figure 5. Relevance of the content received.

For this thesis we have defined RSS spam based on the relevance of the content received as per the subscritions of the user. It was observed that 25 out of 30 participants who took up the user experiment, had been directly or inderctly affected by spamming. During the interview it was noticed that participants were unable to even identify the kind of content they recieved (spam or not).

The contents the partipants received via their feed reader included advertisements, incomplete content with links to web pages, auto downladable links, images (obscene images), partial feed content leading to spammed links and search engines. The content received by the participants belonged to the kind of RSS spam explained in Chapter 6.

b) *How was your user experience with RSS and RSS feed reader? Did you face any difficulties ? If any, specify*.

About 90% of the participants found it very tedious to use RSS feed readers. The participants felt that the subscription part of the RSS feeds is the easiest, provided that the RSS feed readers are well organized. They observed that there was flooding of the inbox when they did not use appropriate filters. Users found it very difficult to master the feed reader. Since RSS feeds cannot function without a feed reader, participants felt demotivated to use RSS. Subscribing to individual sections of websites or blogs were intense as most websites or blogs did not have the option, leaving the users to depend on the filters provided by the feed reader. Most participants felt convenient to visit the website directly rather than use RSS feeds due to this.

Among the participants, 13 of them used an online feed reader such as Feedly[14], Feedbucket[15] and Diggreader[16] etc, and the rest used a client version of feed reader which they had installed locally on their computers for this study. The participants who used a client feed reader complained of many feeds which have been pre-loaded into the reader already.We found that it was difficult for the users to omit the pre-loaded feeds from the feed readers. On further exploration, we have noticed that only the professional version has the feauture to delete the pre-loaded feeds. These pre-loaded feeds had reportedly flooded the inbox of the RSS feeds users with content they

[14] http://www.feedly.com
[15] http://www.feedbucket.com
[16] http://www.digg.com/reader

were not interested or had not subscribed to. If we take notice, this is also a kind of spamming through feed readers and not RSS. This makes us question if it is the feed reader that is to blame for the spamming in RSS.

c) What was the choice of RSS feed reader and why ?

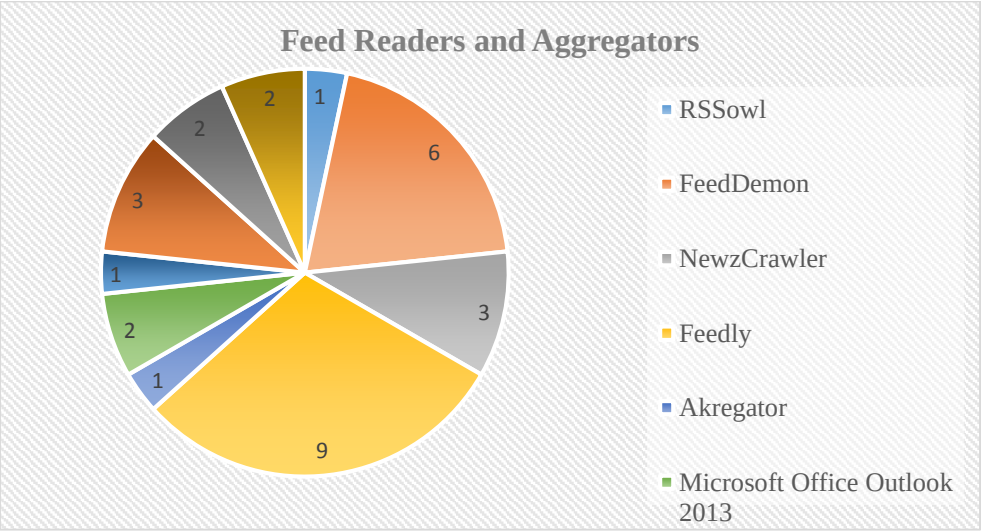


Figure 6. Different types of Feed readers and aggregators used for the experiment.

Based on the inputs recieved from the participants, Feedly was the most popular RSS feed reader among the participants followed by FeedDemon[17] and Diggreader.

Feedly is an online feed reader, much like the Google Reader itself which has been discontinued. The reasons why Feedly was very popular among the users is the ease of use. Many participants mentioned that the tool is very light and does not send a lot of irrelevant feeds. But what is interesting is that it is tolerable and acceptable for many users to allow irrelevant feeds to an extent.

FeedDemon is a client application which does not support any updates due to the discontinuation but this does not affect the users from using it. Even after the discontinuation, it is still popular among the users. One of the reason being its filters. Feed demon provides easy filters that can be applied which helps in minizing the spam content.

[17] http://www.feeddemon.com

Among the participants Diggreader is the third most popular feed reader. The main feature that interest users is the social-media connect available. It lets the users connect to Facebook,Twitter or Google+[18] and share the content directly without much difficulties.

d) How do you rate your overall experince using RSS feeds? Will you continue to use RSS feeds?

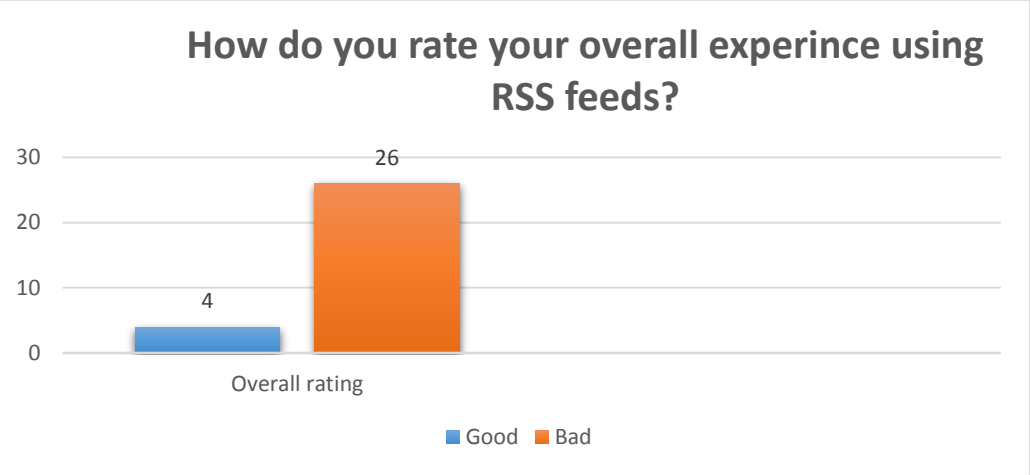


Figure 7. An overall rating of the experince using RSS feeds

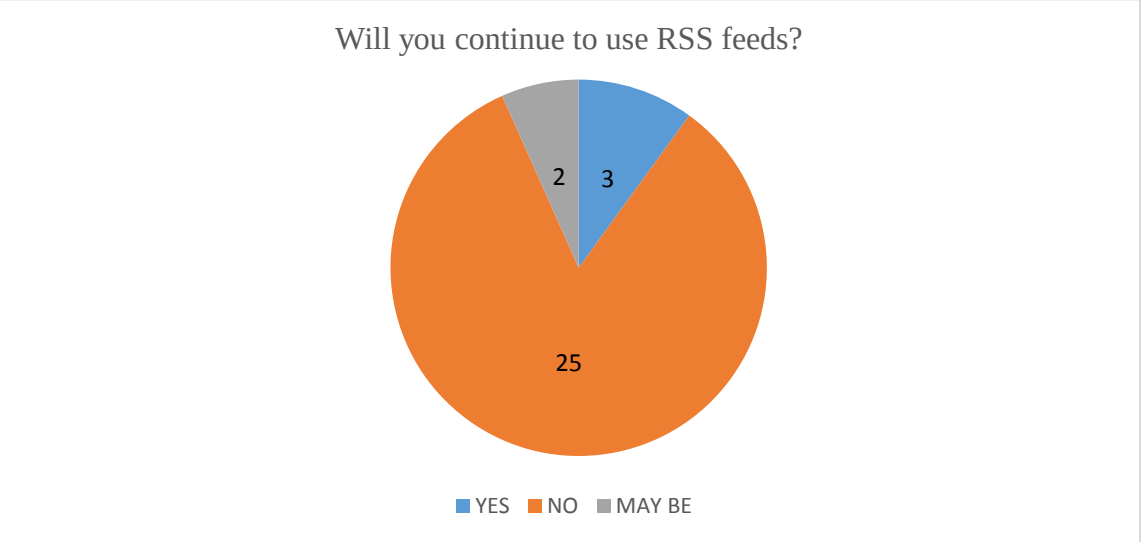


Figure 8. Responses on continued usage of RSS feeds.

[18] https://www.plus.google.com

Figure 7 and 8 clearly explains the fate of RSS in today's world. The participants did not feel very motivated to use RSS feeds beyond one week of the user experiment. Although they found it useful and interesting at the begining, they were not happy with the content they received in the form of feeds. And preferred visiting the website itself as it saves them from being spammed. Most websites have RSS button because the publishers of the website cannot loose the existing users who prefer using RSS.

Table 10 presents a summary of the comments from participants and the types of spam encountered by them during the user experiment. The RSS spam identified from the comments of the participants fall under the kind of RSS spam discussed in Chapter 6.

Participants	**Notable comments from participants**	**RSS spam identifier**
Participant 1	Partial feed content	RSS feed link farms
Participant 2	Obscene images	Spam blogs or Splogs
Participant 3	Auto downloadable links	Fake RSS feeds
Participant 4	Flooded inbox	Event replication
Participant 5	Repeated feeds	Event replication
Participant 6	Pre-subscribed feeds in the feed reader	Spam blogs or Splogs
Participant 7	Unwanted images	Plagiarized content
Participant 8	Advertisements	Auto generated content
Participant 9	Malicious software	Fake URLs
Participant 10	Duplicates of feeds	TrackBack mechanism
Participant 11	Unwanted feeds	Keyword stuffing
Participant 12	Links leading to advertisements	RSS feed link farms
Participant 13	Feeds contain obscene images	Spam blogs or Splogs

Table 10. RSS spam techniques identified from the user experiment

8.7 RSS in Today's World

RSS has been widely accepted by the masses, and is still very useful. And there's been a contrast between RSS and Twitter. Twitter is and was never a competitor for RSS. RSS users may variably argue that social media services such as Twitter, Facebook or

Google+ is no substitute. They may be right but in true form of things the two never competed, but for many Twitter is believed to the substitute for RSS.

According to many, RSS is considered to be the root of today's Twitter(tweets). The main difference between the two is the time relevancy, also the social media connect that attracts the masses. The reasons for the popularity of Twitter are many, one of them is the instantness factor meaning real time news based on the persons interest. In Twitter one can follow accounts that tweet their articles or stories, so you still have the latest content updated but here people are more into following their friends or influncers. The social media connect that Twitter offers its users attracts them the most. Through tweets users get real time up to date information of the people or any type of news in a more instanttaneous way.

Having said all that about Twitter, one must understand that RSS is different, and most importantly not a competitor. RSS is not instant, yet it is contemporary. RSS is not 100% real time though it is a good news platform customized by you, with only the things you care about.Average RSS users have drifted towards Facebook or Twitter rather than subscribing to RSS feeds. Especially after Google shut down its much popular Google Reader.

The death of Google Reader does not mean the death of RSS, although it does show that the experience was not satisfactory for it evolve. If Google Reader had not taken over the RSS reader market and then failed to innovate, perhaps an RSS reader would have offered a more compelling experience for the non-information readers and more of mainstream users.

9. Conclusion

In this thesis, the author has discussed the role of RSS as a publish/subscribe system in terms of spamming, user experience and formalization. The research questions are examined based on the available literature and a user experiment.

Q1. How RSS differs from other publish/subscribe systems?

The systematic literature review performed in this thesis explains RSS as a publish/subscribe system. However, RSS cannot be completely defined as a publish/subscribe system as RSS is an XML code that relies on a RSS feed reader. The client behavior needs to be monitored as RSS cannot function without a feed reader.

Q2. What is RSS spam and how does spam look like in RSS feeds?

We have answered this research question based on the user experiment as the available scientific literature on RSS spam was very limited. RSS spam is a kind of spam that affects the RSS feeds. RSS spam targets directories, search engines rather than the end user. This is due to the fact that RSS feeds are indexed by search engines and directories and can be considered news. It is an unfortunate side effect of free communication. While RSS users can typically unsubscribe to feeds they consider as spam, some users fail to identify whether the content they receive from RSS feeds is spam or not. This was mainly identified in the user experiment conducted for this thesis. In some cases, browsing with keywords in an RSS search engine is where the problem arises. RSS spam is not the same as the spam that you get in your email inbox.

The reasons behind RSS spamming are yet to be studied extensively, but the author wants to emphasis that the major reason for not being able to control or prevent RSS spam is lack of evolution. By lack of evolution, the author means that adequate time and efforts have not been given to sustain RSS in today's world. Unlike Twitter and Facebook, RSS has failed to evolve itself with the changing times. It is no surprise that the user experience with RSS has not captured the imagination of people over the years. Especially after the advent of technologies like Facebook, Twitter and Google+.

Q3. How is the overall experience of using RSS feeds with feed readers ?

The overall user experience of RSS feeds can be deduced from the user experiment conducted for this thesis. According to the results mentioned in Chapter 8, 26 out of 30 participants found the overall user experience of using RSS feeds as bad. They found it tedious to set up and use a feed reader. The participants from the user experiment expressed their disregard over RSS mainly due to the feed readers and the content they received through them.

As per the scope of this thesis, the participants were limited to university students. The author wants to establish a critique on the target group as it only included students of a particular level of expertise and familiarity to the use of RSS and feed readers. The same has been explained in Section 8.1. Had the target group consisted of participants regularly involved with RSS, there would have been a better knowledge of RSS and familiarity with feed readers. Consequently, the results of the user experiment might have varied.

During the course of the user experiment, the participants were not able to identify or detect RSS spam. *An assertion can be made that most number of internet users require a basic learning about internet safety and spam detection in order to fight spam*. In a survey conducted by a research team (including the author), it was found that there has been very limited education and training on security and privacy awareness.

The author also wants to bring forward the importance of training facilities on awareness on internet safety and security related issues. Moreover, the author wants to highlight, security and spamming issues and the role of awareness about internet safety in course curriculum while students are pursuing their higher education.

Q4. What is the possible socio-technical influence on spamming in RSS feeds?

RSS feeds may be treated as a socio-technical system considering the technical and social implications of this web syndication. A range of elements are linked together to achieve the functionality of a socio-technical system. These elements include user practices, cultural meanings, technology, markets, maintenance networks, content platform, etc. [Geels, 2005]. The intention of a user, type of feed readers, and

subscribed content (keywords) constitute the same type of elements in the case of RSS feeds.

Chapter 6 brought forward a possible relationship between socio-technical attributes and RSS feeds. We observed that there is a socio-technical influence on spamming in general. Section 6.2 particularly pointed out three major types of socio-technical attacks, i.e., malware, pop-ups and SEP. The ways in which these attacks are executed were found to be quite similar to the type of spam received by the subscribers during the user experiment. The sample responses from the participants as presented in Table 10 confirm a significant similarity between socio-technical attacks and the spamming in RSS feeds. However, an exploratory study involving socio-technical attributes and feed readers would be needed to investigate how the attackers or illegitimate internet users use RSS feeds for sending irrelevant content to their subscribers.

Q5. What are the benefits of using formalization in RSS feeds?

Chapter 7 presented a research paper that has been co-authored by the author of this thesis. We are able to find the benefits of using formal specification method based on the test case (RSS v2.0 and CafeOBJ) presented in the research paper. Formalizing RSS helps in reducing the ambiguity issues and is most beneficial in early stages of software development cycle. Well defined and frozen requirements are the basis of a successful software development. This is helpful in reducing requirement errors as it provides a detailed analysis. Formalization helps in identifying the incompleteness and inconsistencies of a standard. Formalization is not effective on RSS v2.0 in its current version. This is in terms of spam control or prevention. We propose RSS V3.0 to be an evolved version of RSS V2.0 i.e. formally specified version of RSS v3.0. This proposed version could be more effective.

While formalization is beneficial, it is important to mention the difficulties of using formalization technique. Unfortunately many software engineers are not educated enough in the use of formal methods, or they apply them very rarely. Although formal methods have a slow learning curve, they are also easy to forget and, as mentioned earlier, open standards' readers could be alienated if not sufficiently prepared when reading a formally specified standard.

The research conducted in this thesis is a novel and unique work and has not been attempted yet. Therefore, it is very difficult to compare and contrast this thesis work with any other related research. The inferences of the user experiment provide proof of concept and illustrate potential but they cannot provide solid evidence. An experimental evaluation is generally divided into two parts where exploration takes place in the first and evaluation takes place in the other part. The exploration identifies what questions should be asked about the subject/system under discussion and the evaluation attempts to answer those questions.

10. Limitations and Future Work

Although the author has been very critical of the thesis work based on contextual and temporal limitations yet the results of the user experiment and the inferences drawn from them are worth applauding. In spite of inadequate number of citations/research material and primary references, the study has addressed all the research questions.

However, there have been temporal and contextual limitations. Since the literature in RSS is very limited, this study limits according to following:

- The study considers literature published in the last *two* decades and focuses on the literature published in last 10-15 years in order to have relevance with the current trends in the field of publish/subscribe systems.
- The research made on RSS for this thesis included publish/subscribe systems, types of publish/subscribe system and spamming in publish/subscribe system. There were a significant number of journals related to publish/subscribe system but none related to RSS and spamming.

The context of this thesis is very niche and therefore it could serve as a basis of future work on RSS in different directions namely formalization, anti-spamming and internet safety. A focused and empirical research approach involving case studies in research, academia and industrial practice is needed to evolve RSS v2.0 and define standardized characteristics of open standards.

Future research would identify explanatory cases to justify and validate certain points or issues related to spamming in RSS. Furthermore, it would deeply investigate the problems related to spamming in RSS and the possible measures to counter this problem. This study provided important directions to look for solutions to the problems relating to RSS spamming.

One of the possibilities to extend this study is to evolve RSS v3.0 based on formalization which could be accomplished by formalization of RSS as an independent web news syndicator. This would enable RSS to function without being dependent on Feed Readers or Aggregators. A similar inspiration of RSS in today's world is Twitter which promises to be a replacement of RSS but can never be. However, Section 8.7 of this thesis has already established that RSS and Twitter cannot be competitors.

References

Abraham, Sherly, and InduShobha Chengalur-Smith. 2010. “An Overview of Social Engineering Malware: Trends, Tactics, and Implications.” *Technology in Society* 32(3): 183–96. http://dx.doi.org/10.1016/j.techsoc.2010.07.001.

Baldoni, R., M. Contenti, S. T. Piergiovanni, and a. Virgillito. 2003. “Modeling Publish/subscribe Communication Systems: Towards a Formal Approach.” *Proceedings - International Workshop on Object-Oriented Real-Time Dependable Systems, WORDS*: 304–11.

Barlas, Konstantinos, Eleni Berki, Iulia Adomnita, Thrushna Nalam, Golnaz S. Nejad, and Jari Veijalainen. 2014. “Formal Specification of Open Standards and the Case of RSS v2.0.” In *Proceedings of the 18th Panhellenic Conference on Informatics - PCI ’14*, 1–6. http://dl.acm.org/citation.cfm?doid=2645791.2645809.

Barr, Michael, Charles Wells, and Category Theory. 1998. “BACK MATTER.” http://www.worldscientific.com/doi/abs/10.1142/9789812816108_bmatter.

Boim, Rubi, and Tova Milo. 2008. “Enriching Topic-Based Publish-Subscribe Systems with Related Content.” *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data - SIGMOD ’08*: 1327. http://portal.acm.org/citation.cfm?doid=1376616.1376767.

Bursztein, Elie, Peifung E. Lam, and John C Mitchell. 2009. “TrackBack Spam.” In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security - CCSW ’09*, http://portal.acm.org/citation.cfm?doid=1655008.1655010.

Camacho-Guerrero, José Antonio, Macedo, Alessandra Alaniz, and Alessandra Alaniz Macedo José Antonio Camacho-Guerrero. 2005. “A Software Infrastructure for RSS Deployment and Linking on the Web.” In *WebMedia ’05: Proceedings of the 11th Brazilian Symposium on Multimedia and the Web*, 1–9. http://dl.acm.org/citation.cfm?id=1114236 (October 16, 2014).

Chaabane, Amina, and Mohamed Jmaiel. 2009. “Security Aware Content-Based Publish/subscribe System.” In *Proceedings - IEEE Symposium on Computers and*

Communications, IEEE, 538–43. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5202383.

Creus Tomàs, Jordi, Bernd Amann, Nicolas Travers, and Dan Vodislav. 2011. "RoSeS: A Continuous Content-Based Query Engine for RSS Feeds." In *Lecture Notes in Computer Science*, , 203–18. http://link.springer.com/chapter/10.1007/978-3-642-23091-2_19 (November 6, 2014).

Crocker, David. 1982. "Standard for the Format of ARPA Internet Text Messages." *Dept. of Electrical Engineegring,University of Delaware, Newark, DE,19711*. https://tools.ietf.org (June 6, 2015).

Diaconescu, R.ăzvan, Kokichi Futatsugi, and Shusaku Iida. 1999. "Component-Based Algebraic Specification and Verification in cafeOBJ." In *FM'99 — Formal Methods*, , 1644–63. http://link.springer.com.ezp-prod1.hul.harvard.edu/chapter/10.1007/3-540-48118-4_37.

Diaconescu, Răzvan, and Kokichi Futatsugi. 2000. "Behavioural Coherence in Object-Oriented Algebraic Specification." *Journal of Universal Computer Science* 6(1): 74–96.

Dijkstra, E.W. 1981. "The Correctness Problem in Computer Science." *Academic Press*.

Emery, F.E, and E.L Trist. 1960. "Socio-Technical Systems." *Management Science Models and Techniques, Oxford, UK* 2: 83–97.

Erbschloe, Michael. 2004. *Trojans, Worms, and Spyware: A Computer Security Professional's Guide to Malicious Code*.

Eugster, Patrick Th., Pascal a. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. "The Many Faces of Publish/subscribe." *ACM Computing Surveys* 35(2): 114–31. http://portal.acm.org/citation.cfm?doid=857076.857078.

Geels, F.W. 2005. *Technological Transitions and System Innovations: A Co-Evolutionary and Socio-Technical Analysis*. Edward Elgar Publishing.

Goguen, Joseph a., and José Meseguer. 1992. “Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations.” *Theoretical Computer Science* 105(2): 217–73.

Hochard, Gaïane, Zoé Lacroix, Jordi Creus, and Bernd Amann. 2012. “A Semantic Map of RSS Feeds to Support Discovery.” In *Lecture Notes in Computer Science*, 122–33.

Housley, Sharon. 2010. “RSS Security.” http://www.feedforall.com/rss-security.htm (March 15, 2015).

Howard, Fraser, and Onur Komili. 2010. “Poisoned Search Results : How Hackers Have Automated Search Engine Poisoning Attacks to Distribute Malware .” *Sophos Technical Papers*: 1–15.

Ivaturi, Koteswara, and Lech Janczewski. 2011. “A Taxonomy for Social Engineering Attacks.” *Proceedings of CONF-IRM*.

Kolari, Pranam, Tim Finin, Akshay Java, and Anupam Joshi. 2007. “Towards Spam Detection at Ping Servers.” *International Conference on Weblogs and Social Media’07* (iv): 1–2. http://aisl.umbc.edu/resources/342.pdf (October 16, 2014).

Kolari, Pranam, Akshay Java, and Tim Finin. 2006. “Characterizing the Splogosphere.” In *Proceedings of the 3rd Annual Workshop on Weblogging Ecosystem: Aggregation, Analysis and Dynamics, 15th World Wid Web Conference. University of Maryland, Baltimore County, 2006*, 1531–36.

Leontiadis, Nektarios, Tyler Moore, and Nicolas Christin. 2014. “A Nearly Four-Year Longitudinal Study of Search-Engine Poisoning Categories and Subject Descriptors.” *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*: 930–41.

Liu, Hongzhou, Venugopalan Ramasubramanian, and Emin Gün Sirer. 2005. “Client Behavior and Feed Characteristics of RSS, a Publish-Subscribe System for Web Micronews.” *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*: 3–3. http://portal.acm.org/citation.cfm?doid=1330107.1330111.

Lu, Long, and Wenke Lee. 2011. “SURF : Detecting and Measuring Search Poisoning Categories and Subject Descriptors.” *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS ’11)*: 467–76.

Luo, Weimin Luo Weimin, Jingbo Liu Jingbo Liu, Jing Liu Jing Liu, and Chengyu Fan Chengyu Fan. 2009. “An Analysis of Security in Social Networks.” *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*: 648–51.

McCoy, Scott, Andrea Everard, Peter Polak, and Dennis F. Galletta. 2007. “The Effects of Online Advertising.” *Communications of the ACM* 50(3): 84–88.

Mian, Paula, Tayana Conte, Ana Natali, Jorge Biolchini, and Guilherme Travassos. 2007. “A Systematic Review Process for Software Engineering.” *Empirical Software Engineering* 32: 1–6. http://portal.acm.org/citation.cfm?id=1241572.1241584.

Montgomery, Molly. 2003. “RSS Tutorial.” *Lone Star lIbrarian* 56(2). http://www.sla.org/chapter/ctx/lsl/lslv56n2.pdf (April 4, 2015).

Nakagawa, A.T., Toshimi Sawada, and Kokichi Futatsugi. 1999. “CafeOBJ User’s Manual -- ver.1.4.2 --.” http://www.ldl.jaist.ac.jp/cafeobj/doc/.

Palmer, Daniel E. 2005. “Pop-Ups, Cookies, and Spam: Toward a Deeper Analysis of the Ethical Significance of Internet Marketing Practices.” *Journal of Business Ethics* 58(1): 271–80.

Rose, Ian, Rohan Murty, and PR Pietzuch. 2007. “Cobra: Content-Based Filtering and Aggregation of Blogs and RSS Feeds.” *NSDI’07 Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*: 29–42. http://static.usenix.org/legacy/events/nsdi07/tech/full_papers/rose/rose_html/ (October 16, 2014).

RSS Advisory Board. 2014. “RSS Advisory Board.” http://www.rssboard.org/rss-specification (July 5, 2015).

Software Garden Inc. 2004. “What Is RSS?” http://rss.softwaregarden.com/aboutrss.html (May 30, 2015).

Sourlas, Vasilis, Georgios S. Paschos, Paris Flegkas, and Leandros Tassiulas. 2009. "Caching in Content-Based Publish/Subscribe Systems." *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*: 1–6. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5425532.

Sven Ove Hansson. 2006. "A Note on Social Engineering and the Public Perception of Technology." *Technology in Society* 28(3): 389–92.

Tarkoma, S. 2006. "Preventing Spam in Publish/Subscribe." *26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06)*: 21–21. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1648911.

Whitworth, Brian. 2009. "The Social Requirements of Technical Systems." *Handbook of Research on Socio-Technical Design and Social Networking Systems*: 3–22.

Whitworth, Brian, and Tong Liu. 2009. "Channel E-Mail: A Sociotechnical Response to Spam." *Computer* 42(7): 63–72.

Zhang, Kaiwen, Vinod Muthusamy, and Hans-Arno Jacobsen. 2012. "Total Order in Content-Based Publish/Subscribe Systems." *2012 IEEE 32nd International Conference on Distributed Computing Systems*: 335–44. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6258006 (March 12, 2014).

Appendices

Sample responses collected during the user experiment are presented as per the following:

I'm using Akregator program (v. 1.5.4) for RSS feeds. It is a part of KDE suite and it is installed in many Linux computers by default.

I've subscribed to the following blogs for this experiment:

1. TechCrunch // Enterprise stream
2. Mashable // Tech
3. ReadWrite
4. Dumb Little Man - Tips for Life
5. Interesting Thing of the Day

I'm been subscribed to these blogs 13 - 20 October.

I don't have received any spam or other content that I've not subscribed. I don't have received any spam in mail email, which I've used for subscriptions.

Difficulties with my RSS reader:

1. I'm not sure, do I've subscribed to individual streams. In some cases the Akregator shows an individual stream with >> signs (for example TechCrunch >> Enterprise), in some cases doesn't show anything about streams.
2. Some posts require the Adobe Flash, and the Akregator warns about the missing plugin. Fortunately I can disable this warning.
3. Sometimes the function "open in tab" displays an empty page (The problem is with critical HTML errors in RSS posts...It seems like the Akgregator is very tight about HTML errors).

The best experience was when I've opened all posts in the Chrome browser.
My experience with this program was positive. It was very easy to start using this program, to subscribe to feeds etc. The program is minimalistic, without any unnecessary functionality. I don't know, maybe I've missed something because of so minimalistic user interface.

RSS Feed Experiment

RSS Feed Reader used: Feedly

Subscribed blogs: Interesting Thing of the Day
Simply Recipes
Read/Write Web - Social
TechCrunch - Social
Mashable - Social Media

Subscribing time: Tuesday 14.10.2014 - Sunday 19.10.2014

I didn't receive any spam to my reader or my email address. The individual streams that I subscribed also offered the kind of content that you would expect to get. I got few updates daily from all these feeds.

I couldn't use any of the feed readers suggested because I don't have Windows, only Mac OS X. I liked using feedly as my RSS reader because it can be used via browser. There is also app available for my phone and my tablet so it's easy to use also in them. It's easy to create different categories in feedly so the content can be organized nicely. There is also a working search feature. I didn't encounter any particular problems with the feed reader, everything seemed to work as it should.

2. Microsoft Office Outlook 2013
3. TechCrunch
4. Friday to Monday → 3 days
5. Not any spam mails or anything, just news or commersials
6. Overall Experience:

I have used this kind of services also before, but I get frustrated because there was so much news en just a few pictures. If I go to the Internet there is much more pictures and graphics. But now here in TechCrunch there was picture in every RSS feed and that was nice even though there was also some commercial among these picture. I think it is no appropriate to put commercials feeds like this, because people don't want more spam to their email box. Accordingly it was not great to read just commercial feeds. Most common commercial was Apple Inc. commercial, it was annoying.

I would say that RSS Feed is old school thing, there are several reasons for that. First of all almost every people have broadband connecting on their own computer. I think – I don't know – but maybe RSS Feed is made to modem computers to get news more quicker and low costs. Maybe its more suitable for mobilephones, I don't know.

I don't personally anymore use RSS Feeds on my computer. Its just so much more fun to go web to check the latest news with pictures and everything. Maybe I just don't read enough specific theme to use RSS Feed more often. I get so much email already that I don't want anymore read RSS Feeds, because I think that those are like e-mails!

I don't know anything else to say...

I chose FeedDemon as my RSS reader and when I set it up on Tuesday evening I subscribed to following feeds: 43 Folders, Interesting Thing of the Day, John Battelle's Search Blog, Mashable and TechCrunches Gadgets and Gaming sections. I added ReadWrite, Boing Boing, Dumb Little Man and Muropaketti (finnish tech site) on Thursday. Most sites didn't offer an easy / obvious way to subscbribe only to particular topics, TechCrunz was only one to have the feeds nicely separated. ReadWrite had their feed very cleverly hidden and I spent atleast half an hour before I found it from an public IFTTT recipe by accident.

I checked my feeds two times a day, in the morning after waking up and sometime in the evening. I made notes after almost every check of the feed, and there's two topics on almost every note. Mashable being so much more active than the other feeds that it pretty much floods the whole thing and Mashables non-tech related news being something that I'm not really interested in. I actually made a filter that made FeedDemon to mark news from Mashables "US & World" section read, and thus hiding them from the feed. After applying the filter my feed looked a lot better, but I'd probably still need to hide a couple of their sections (Watercooler and Entertainment to be precise).

I didn't get any spam or other stuff that I hadn't subscribed to either to my email or FeedDemon, but as mentioned before only TechCrunz has easy links to feeds of individual sections. So as I subscribed to e.g. Mashable, I subscribed to the unwanted sections too so the feed worked as promised. The filter I set for the "US & World" section needed a bit of tweaking before it worked so I really would have liked if all the sites offered individual feeds like TechCrunz does.

On Thursday evening, when Apple announced new iPads and other products I noticed a problem / annoyance in having multiple tech sites news in same feed. The amount of overlapping content was huge when a couple of the sites actively pushed content about Apples new products and it was somewhat annoying to try find something that I hadn't read already.

I'm going back to my old style of reading news, visiting the sites individually. This eliminates the problem from the last paragraph and is just an old habbit of mine. Using an specialized reader program is somewhat annoying, I have my browser open all the time so it would be so much easier to just open a new tab, write a letter or two of the address of my feed reading site and let the browser autofill it and have it open in a second. With an external reading program I either have to have it on all the time to have be able to get the news as quickly and I really don't like the sound of that.

When iGoogle, the google frontpage that could be personalized with widgets was still a thing I had a couple of RSS widgets there, each showing only one sites news. I was really disappointed when iGoogle was closed and actually asked a friend who was in a web design school to make me equivalent site and actually offered to pay him. He wasn't interested and I looked up some sites that could do the job but none of them seemed satisfying. I really just want a website that can show a couple of news feeds in their individual streams and has a Google searchbar.

Program: Newz Crawler

Blogs subscribed to: TechCrunch; Gaming, Mashable, Interesting Thing Of The Day, Dumb Little Man, John Battelle's Search blog

I have been subscribing to the previous blogs since 15.10 via Newz Crawler. I have received appropriate content in the feed reader. The only problem was that the feed reader had loads of pre-subscribed feed channels that flooded my feed readers inboxes. To get rid of this annoyance, I had to manually delete every feed channel that I didnt want to get feeds from, which took a while since there was at least a 100 of them.

The usage of this feed reader was overall very easy. The setup was easy and fast, and all the needed features were easy to find and learn. It was very convenient that all the news and articles that i wass interested in came right into my feed reader and i could read them there. I found it annoying that there were so many feed channels already installed, and they kept spamming my inboxes until i deleted them. Also, the fact that there were so many different inboxes already installed made the program look a lot more complicated than it actually was.

RSS feeds are an overall good way of getting the news that you're interested in, but i must say that after this experiment, I will not keep on using RSS feeds. The reason for this is the fact that it just doesn't feel good enough to me, and I'd rather visit the web page than read all the news from the feed reader. The feed readers version of the article is so simplified and sometimes doesn't contain all the content that the actual article in the website has.

I used FeedDemon in the experiment. I chose it based on the recommendation from the list because I didn't have much knowledge about RSS feed readers prior to the experiment. I subscribed to Dumb Little Man, John Battelle's search blog, Mashable, Simply Recipes and TechCrunch. I couldn't find the feature that would allow me to subscribe to individual streams from FeedDemon. Later I discovered that I should have subscribed to the blogs from their sites to find the individual streams and not directly from Feed Demon. I subscribed to the feeds on October the 13th and I've kept the subscriptions to this date (20.10). I haven't received any spam to my knowledge on the feed reader. I didn't connect the feed reader to the email because that part wasn't in the second set of instructions that I received.

FeedDemon seems easy enough to use. The interface is quite simple and efficient. The feed itself is clear and it's easy to find older posts. I found the little pop ups that appear when a feed updates with new articles to be quite useful. As I mentioned before I couldn't find the option to subscribe to individual streams so I tried using a filter function. The filter doesn't seem to work tough so I probably did something wrong.

The biggest problem I had wasn't with the program itself but with its installer. When I installed FeedDemon I was a bit sloppy and didn't read all the windows in the installer and ended up getting some malware on my computer. To be specific the malware program was called NeuroNetwork or something like that. I used malwarebytes to remove the malware.

I used FeedDemon to follow: 43 Folders, Dumb Little Man, Mashable, ReadWrite, TechCrunch. I was unable to find any specific feed for Mashable and ReadWrite, but from TechCrunch i followed gaming and startup news. From Dumb Little Man it was Tips for life and How To. And last from 43 Folders Decision-making. I've been following these sites from 16,10,2014 to 20,10,2014. In my time of using FeedDemon they have not sent me any spam or mail. The sites i'm following have been providing my feed reader with the news i've subscribed to and i haven't found any wrong type of articles or advertisement.

When i started this experiment i had very little knowledge of feed readers. I knew what Rss feed was, but i hadn't used them before. Trying to follow single section from TechCrunch took me to page that had me choose what feed reader i was using. Without any prior knowledge i just chose one at random and it was FeedDemon. After installing FeedDemon i'm greeted with new program that reminds me of browser and can be used as one. From hereon i'm using FeedDemon as my browser to go through the given sites and trying to find interesting ones. With plenty of sites it's easy to find ones that are interesting, but following them is not that simple always. I'm unable to find rss feeds foı different sections from TechCrunch, but since they we're shown in the example i know they must exist. Google search is my friend here as i couldnt find individual feeds from the site. After finally gaining two feeds from TechCrunch i start searching for more feeds and find that they are not as easy to find as i thought. Unlike TechCrunch some sites dont have they're own rss feed subscribe section, but instead they use browsers native solutions. FeedDemon shows Subscribe button at the end of the site url, but sometimes it doesn't come right away and sometimes the site doesnt have rss feed so it doesnt come at all. After finally getting enough feeds to satisfy our quota and not having the energy to try and find more i settle with the ones i have. Now that my setup is complete following the given feeds is easy and advertisement free, well atleast until i open the article. Overall using feed readers makes it much more manageable to follow many different sources, but it does take some time and effort to setup. I will most likely take advantage of FeedDemon now that i have an idea how it works.

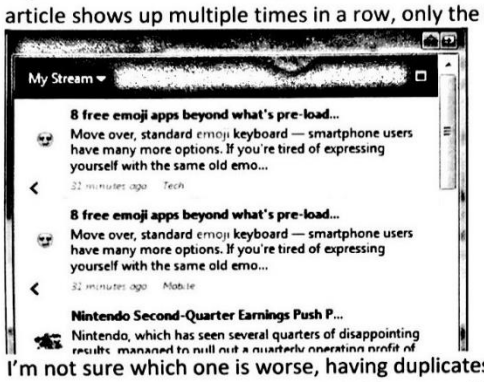
In this experiment I used NewzCrawler program to subscribe for few blogs. I tried to find out if any spam or unwanted content will appear on my email address or to the program itself. The blogs I followed were: Boing Boing, TechCrunch – gaming, SimplyRecipes, Dumb Little Man and John Battelle's Search Blog.

This was the first time ever I used any kind of RSS feeder program at all. First I had some difficulties to understand how the program works. After few minutes confusion, it turned out to be rather easy and quick way to read blogs and news.

I have been subscribing to these blogs for almost 5 days now, from Wednesday to Monday. I have not received any spam emails or any other messages or commercials that could be categorized as spam. I have to say that this was quite a narrow timespan, but on the other hand many programs and websites which you tell your email will start sending unwanted messages right away if they are willing to do so.

Overall this was an interesting experience, since I never followed any RSS feeds before. Only difficulties I faced were related to the program usage for the first time. Some of the blogs I followed were really great and I may continue follow those even though the experiment is over.

I changed my Mashable subscription from the whole site feed to cherry picked "channels". They weren't as well hidden as I thought at first but I still needed to search them for a while. Subscribing to many channels from the same site introduced some new problems. When the site tags a certain article to many feeds, FeedDemon doesn't filter them in any way and the exact same article shows up multiple times in a row, only the tag from which feed it was fetched changes.



I'm not sure which one is worse, having duplicates of interesting articles or having articles that I'm not interested at all in my stream. Adding more filters to the whole channel subscription over time would probably fix most problems with it and I'd probably choose that.

I've set up FeedDemon to run when I start my computer and I check the headlines when the window pops up. I don't actually read the whole articles that often. The way I've got it set up currently is nice when I'm not in a hurry and the window popping up doesn't interrupt me. But when I have to quickly check something like bus timetables the window popping up is pretty much the most annoying thing ever.

I'm still not convinced to keep using FeedDemon now that the test period is over. Old habits of checking the individual sites and the ease of just opening new browser tab instead of starting a new program keep me using normal browser and regular news sites as my primary of way of reading news. I don't know why but I also read the whole articles more often when browsing the site normally. Maybe it's the fact that I actually decided to go check the news instead of the automatically opened window forcing me to check them. Some easy to use and convenient browser based RSS readers like iGoogle widgets or Google Reader would be great, but Google decided otherwise.