Generating Activity Streams from
Events occurring in company-internal IT infrastructures

by

S. M. Khaled Reza

A thesis submitted in partial fulfillment for the degree of International Masters Program in Computational Engineering (Distributed Systems Engineering)
in the
Institute for Systems Architecture
Faculty of Computer Science
Dresden University of Technology

Supervisors:
Dr.-Ing. Marius Feldmann
Dipl.-Medien-Inf. Philipp Katz

Professor:
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

October 2011


Declaration of Authorship

I, S. M. Khaled Reza, declare that this thesis titled, ‘Generating Activity Streams from Events occurring in company-internal IT infrastructures’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ________________________________

Date: _________________________________
“Nothing is impossible in the world. To be special, you have to believe that I am special. There is no secret to win. To win you just have to believe that I can. You should have courage in truth.”

Master Shifu, Kung Fu Panda
In the current era of technology, the Internet and web technologies become the center source of information. Due to the huge amount of contents, one of the main challenges of modern information technology is aimed at how to reduce and manage information in a structured way with mobilizing users to the similar kind of relevant information. So, any intelligent system should be able to understand people’s interest about a particular type of information and automatically mobilize him to the similar kind of available information sources. The idea of high level Activity Streams along with its standardized format can play a vital role to solve this problem in the broader context. This master thesis introduces a novel system called CoASGen (Consolidation and Activity Streams Generator) system which is able to automatically generate high level Activity Streams after aggregating and consolidating from different independent systems (e.g. in a software company context: version management system, wikis, bug trackers etc.).

It retrieves life time information as heterogeneous web feed by sensing user activities from those independent systems and then it transforms several similar types of atomic activities into high level Activity Streams using semantic technologies along with its specific standardized format. Finally, it shows these high level Activity Streams to the user interface which is able to automatically motivate users to find relevant information easily without either missing any data or losing valuable time. This system solves the problem ‘information silos’ by reducing and managing information in a structured way.
Acknowledgements

I would like to thank all professors of Computational Engineering, TU Dresden for offering several excellent courses and their marvelous support during the study period. I specially thank Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill for accepting my master thesis in his chair and supporting my publication at IEEE ICCT 2011. I am grateful to work with him.

I specially thank with most sincere gratitude to my supervisors, Dr.-Ing. Marius Feldmann and Dipl.-Medien-Inf. Philipp Katz for helping in every part of my thesis to make this work efficient. I acutely appreciate their valuable advices during the discussion and writing comments. Thank you for your endless patience, support, supervision and consideration in my work.

I affectingly thank some of my friends: Mahmud Hossain, Sarfaraz Akhtar khan, S. M. Mohsin Reza, Ramzi Youssefi, Wahida Afroj Koli who had a great favor in my thesis. I also thank some of my friends: Ivan Gudymenko, Anna Utlik, Ehsan Khan, Kateryna Iermakova, Sudip Nath, Mahabub Al Rahman, Raihanul Islam, Farooq Anwar, Pooran who helped me during the work with discussion and my prototypical evaluation.

Finally, I would like to give a special thanks to my parents: Abdus Salam Miah, Begum Laila Anjuman Banu and my brother S. M. Mohsin Reza for encouraging me as well as mental support from my childhood.
# Contents

Declaration of Authorship .......................... ii

Abstract ................................................ iv

Acknowledgements ...................................... v

List of Figures ........................................ x

List of Tables .......................................... xii

Abbreviations .......................................... xiii

## 1 Introduction ........................................ 1
  1.1 Motivation .......................................... 2
  1.2 Main goals and Research Questions .................. 4
  1.3 Structure of the Thesis ............................ 5

## 2 Use cases and Requirements ....................... 7
  2.1 Use cases of Activity Streams ...................... 7
    2.1.1 Software Developing Company ................... 8
    2.1.2 Corporate Organization .......................... 8
    2.1.3 Network Management/Support Community .......... 10
  2.2 Requirement Analysis ................................ 11

## 3 Technical Background ............................ 13
  3.1 Web Feeds .......................................... 13
    3.1.1 RSS ........................................... 14
      3.1.1.1 RSS 2.0 ...................................... 15
      3.1.1.2 RSS 1.0 ...................................... 16
    3.1.2 Atom ........................................... 18
    3.1.3 Summary ........................................ 19
  3.2 Different Feed Aggregators ........................ 19
    3.2.1 Content-Based RSS and Broadcast News Streams Aggregation ........... 20
    3.2.2 Supporting Situation Awareness by Semantic Aggregation ............ 22
    3.2.3 Content based RSS Aggregation ................... 25
3.2.4 Affect Sensitive News Agent ........................................... 26
3.2.5 Coalescence of XML-Based RSS Aggregator ....................... 28
3.2.6 Summary .................................................................. 30
3.3 Activity Streams .............................................................. 30
3.4 Feed Aggregator vs. Activity Streams Generator .................... 32
3.5 State-of-the-art Generating Activity Streams System ................ 32
  3.5.1 Semantic Model from Heterogeneous Social Networks ............ 34
  3.5.2 Google Buzz ............................................................ 36
    3.5.2.1 Google Buzz Internal Concepts and Data Model ............ 37
    3.5.2.2 Discussion ......................................................... 38
  3.5.3 MySpace ................................................................. 39
  3.5.4 Gnip ................................................................. 41
  3.5.5 Gowalla ............................................................... 44
  3.5.6 Socialcast ............................................................. 46
  3.5.7 Final Discussion ......................................................... 48

4 Design ............................................................................ 50
  4.1 System Overview .......................................................... 51
  4.2 Mediator .................................................................. 53
    4.2.1 Heterogeneous Feeds ................................................ 53
    4.2.2 Format Syndicator ...................................................... 54
  4.3 Consolidator ............................................................... 54
    4.3.1 Aggregator .............................................................. 55
    4.3.2 Type Separator ........................................................ 56
      4.3.2.1 TitlePattern Comparator .................................... 56
      4.3.2.2 AuthorNameTime Comparator .............................. 57
      4.3.2.3 RegularExpression Comparator ............................ 57
      4.3.2.4 Frequency Comparator ....................................... 58
      4.3.2.5 Subject Decider .................................................. 58
    4.3.3 Activity Generation .................................................. 59
      4.3.3.1 Configuration Language ...................................... 59
  4.4 Activity Streams Generator ............................................... 61
    4.4.1 Activity Streams Manager ........................................ 62
      4.4.1.1 Activity Stream Validator .................................. 63
      4.4.1.2 Domain based Validator ..................................... 64
    4.4.2 Activity Streams Format ............................................ 64
      4.4.2.1 High level Activity Stream General Format ............. 64
      4.4.2.2 High level Activity Stream Prefix Format ............... 65
      4.4.2.3 High level Activity Stream Postfix Format ............. 65
      4.4.2.4 Subject Selection .............................................. 66
    4.4.3 Activity Streams View ............................................... 67
      4.4.3.1 MetaActivityStream .......................................... 67
  4.5 Real Time Stream Updates ............................................... 68
  4.6 Summary .................................................................. 68

5 Implementation ................................................................. 69
  5.1 Development Languages and Platform .................................. 69
5.2 Architecture of CoASGen system

5.2.1 Configuration Package

5.2.1.1 Percentage Level for the Isolator, Frequency Comparator

5.2.1.2 Percentage Level for the Isolator, Regular Expression Comparator

5.2.1.3 Activity Streams Generating Time Duration

5.2.1.4 Different Format of Activity Streams

5.2.1.5 Streams Life Duration in the Database

5.2.1.6 Locality order of Activity Stream in User Interface

5.2.2 Control Package

5.2.3 Graphical User Interface (GUI) Package

5.2.4 Mediator Package

5.2.4.1 Retrieve Stream sub Package

5.2.4.2 Format Recognition sub Package

5.2.4.3 Format Syndication sub Package

5.2.5 Consolidator Package

5.2.5.1 Control Package

5.2.5.2 Retrieve Stream Package

5.2.5.3 Title Pattern Comparator Package

5.2.5.4 Author Name Time Comparator Package

5.2.5.5 Regular Expression Comparator Package

5.2.5.6 Frequency Comparator Package

5.2.5.7 Subject Decider Package

5.2.5.8 Activity Generator Package

5.2.6 Activity Stream Generator Package

5.2.6.1 Control Package

5.2.6.2 Retrieve Stream Package

5.2.6.3 Activity Stream Format Package

5.2.6.4 Part Decider Package

5.2.6.5 Sub Activity Stream Generator Package

5.2.6.6 Activity Stream Validator Package

5.2.6.7 Domain based Validator Package

5.2.7 Persistence Package

5.3 Frameworks and APIs of CoASGen System

5.4 Databases

5.5 Summary

6 Evaluation

6.1 A Case System Study

6.1.1 Application Process Management Life cycle

6.1.1.1 Configuration

6.1.1.2 MetaActivityStream

6.1.1.3 High Level Activity Streams Generation

6.1.1.4 Streams Generation

6.1.1.5 Feed Item Description and View

6.2 Objective Evaluation

6.2.1 Evaluation of Consolidator
## Contents

6.2.2  Evaluation of Activity Stream Generation .......................... 101  
   6.2.2.1  High level Activity Stream Generation Process ............. 101  
   6.2.2.2  High level Activity Stream Format ............................ 102  
   6.2.2.3  Stream Generation Process .................................... 103  
   6.2.2.4  Generation of MetaActivityStream ............................ 105  
6.3  Real User Evaluation .................................................. 106  
6.4  Requirements Fulfillment .............................................. 109  
6.5  Summary ................................................................. 112  

7  Epilogue ............................................................................. 115  
   7.1  Summary and Conclusion ................................................ 115  
   7.1.1  Explored Research Questions ....................................... 116  
   7.1.2  Some features and Guide to adapt in other similar environment 117  
   7.2  Future work ............................................................... 118  

A  End User Questionnaires ..................................................... 119  

Bibliography ........................................................................... 122
# List of Figures

3.1 A real feed example of RSS 2.0 format ........................................ 16
3.2 A real feed example of RSS 1.0 format ........................................ 17
3.3 A real feed example of Atom format ........................................... 18
3.4 Schematic System Architecture of Content based RSS aggregator ....... 21
3.5 Framework of SA support tool for software forges ......................... 23
3.6 Supervision of PicoForge ......................................................... 24
3.7 Total activities in project in 60 days ........................................... 25
3.8 The Cobra content based RSS aggregation network ......................... 26
3.9 The architecture of ASNA ......................................................... 27
3.10 The framework architecture of PheRSS ....................................... 29
3.11 The execution flows of PheRSS ................................................ 29
3.12 Enterprise Activity Streams ..................................................... 31
3.13 Facebook’s Activity Stream ....................................................... 33
3.14 Source code of Facebook’s Activity Stream .................................. 34
3.15 The semantic web integration of heterogeneous online social websites 35
3.16 Mapping from Linkedin.com and DBLP to and extended FOAF ontology 36
3.17 Activity streams of Google Buzz ............................................... 37
3.18 MySpace Activity Stream ......................................................... 39
3.19 Activity subscription flow of MySpace ....................................... 40
3.20 Aggregating Activity Streams from several social networks ......... 42
3.21 GNIP data model ................................................................. 43
3.22 GNIP data flow ................................................................. 43
3.23 Activity Streams with Gowalla .................................................. 45
3.24 Activity Streams with Socialcast .............................................. 46
3.25 Socialcast activity stream engine .............................................. 47
4.1 CoASGen systems's overall system architecture ......................... 51
4.2 Mediator internal architecture .................................................. 53
4.3 Consolidator internal architecture ............................................. 55
4.4 Flow chart of Type Separator in Consolidator .............................. 57
4.5 Configuration language of Activity generation with different parts .... 59
4.6 Activity Streams Generator architecture .................................... 61
4.7 The flow diagram of Activity Stream generator ......................... 63
4.8 The semantic diagram of high level Activity Stream general model format 65
4.9 The semantic diagram of high level Activity Stream prefix model format .... 65
4.10 The semantic diagram of high level Activity Stream postfix model format 66
5.1 The core packages architecture of CoASGen system .................. 70
List of Figures

5.2 The packages of the Configuration of CoASGen system .......................... 71
5.3 Screenshot of a Configuration file .................................................. 72
5.4 Screenshot of Activity Streams Generator ......................................... 74
5.5 The packages of the Mediator of CoASGen system ............................ 75
5.6 Screenshot of RSS 2.0 feed .......................................................... 76
5.7 Screenshot of RSS 1.0 feed .......................................................... 76
5.8 Screenshot of Atom feed .............................................................. 77
5.9 Screenshot of generalized web feed .................................................. 77
5.10 The sub packages of the Consolidator ............................................ 78
5.11 Class diagram of different packages in Consolidator ........................... 79
5.12 The sub packages of the Activity Stream Generator .......................... 82
5.13 Class diagram of Activity Stream Generator .................................... 83

6.1 MVC pattern of CoASGen system application using SVN, Wiki and Bug tracker .......................................................... 89
6.2 Screen print of user configuration page ............................................ 90
6.3 Screen print of user configuration page after configuration applied ........ 91
6.4 Screen print of metaActivityStream for the case study application system 92
6.5 Screen print of one Activity Stream for the case study application system 94
6.6 Screen print of streams for the Activity Stream described in last figure ... 95
6.7 Screen print of one Feed Item’s description from the generated stream ... 96
6.8 Line graph of source feed items (manually selected) ......................... 98
6.9 Line graph of isolated feed items (Consolidator output) .................... 100
6.10 Comparison graph of source feed items and isolated feed items (by Consolidator) ......................................................... 101
6.11 High level Activity Stream of a type consisting 10 activities ............. 102
6.12 screen print of three different high level Activity Streams format ......... 103
6.13 Screen print of streams in one isolated feed type ............................ 104
6.14 Screen print of stream generating database .................................. 104
6.15 Screen print of technical term configuration .................................. 105
6.16 Screen print of MetaActivityStream in CoASGen system ................. 105
# List of Tables

2.1 Requirements analysis and goals in a table view ........................................... 12  
3.1 High level comparison of Generating Activity Streams ................................. 49  
6.1 Source table of input feed items ................................................................. 98  
6.2 Isolated feed items by Consolidator ............................................................. 99  
6.3 The comparison table of source and isolated feed items (by Consolidator) ....... 100  
6.4 The list of questionnaires conducted to end user and their result calculation .... 113  
6.5 Requirements analysis and achievement ...................................................... 114  
A.1 The result table of end user evolution .......................................................... 121
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoASGen</td>
<td>Consolidation and Activity Streams Generator</td>
</tr>
<tr>
<td>B2B</td>
<td>Business-to-business</td>
</tr>
<tr>
<td>RSS</td>
<td>Really Simple Syndication</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>BBC</td>
<td>British Broadcasting Corporation</td>
</tr>
<tr>
<td>CNN</td>
<td>Cable News Network</td>
</tr>
<tr>
<td>DTV</td>
<td>Digital Television</td>
</tr>
<tr>
<td>RSSF</td>
<td>Retrievable Surface Storage Facility</td>
</tr>
<tr>
<td>MMAS</td>
<td>Multimodal Aggregation Service</td>
</tr>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognizer</td>
</tr>
<tr>
<td>MAi</td>
<td>Multimodal Aggregation Index</td>
</tr>
<tr>
<td>DC</td>
<td>Dublin Core</td>
</tr>
<tr>
<td>DOAP</td>
<td>Description of a Project</td>
</tr>
<tr>
<td>FOAF</td>
<td>Friend of a Friend</td>
</tr>
<tr>
<td>EvoOnt</td>
<td>Evolution Ontology</td>
</tr>
<tr>
<td>VOM</td>
<td>Voice of the Martyrs</td>
</tr>
<tr>
<td>BOM</td>
<td>Bill of Material</td>
</tr>
<tr>
<td>SVN</td>
<td>Subversion</td>
</tr>
<tr>
<td>CVS</td>
<td>Concurrent Versions System</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>Cobra</td>
<td>Content based RSS Aggregation</td>
</tr>
<tr>
<td>ASNA</td>
<td>Affect Sensitive News Agent</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>CRM</td>
<td>Customer Relationship Management</td>
</tr>
<tr>
<td>HR</td>
<td>Human Resource</td>
</tr>
<tr>
<td>PSM</td>
<td>Probabilistic Semantic Model</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>UTC</td>
<td>Coordinated Universal Time</td>
</tr>
<tr>
<td>MDP</td>
<td>Multicast Dissemination Protocol</td>
</tr>
<tr>
<td>PSHB</td>
<td>PubSubHubbub</td>
</tr>
<tr>
<td>REST</td>
<td>Representation State Transfer</td>
</tr>
<tr>
<td>ASUpdate</td>
<td>Activity Streams Update</td>
</tr>
<tr>
<td>ASDisplay</td>
<td>Activity Streams Display</td>
</tr>
<tr>
<td>API</td>
<td>Application Package Interface</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>JSP</td>
<td>Java Server Pages</td>
</tr>
<tr>
<td>Std.</td>
<td>Standard Deviation</td>
</tr>
</tbody>
</table>
Dedicated to my family and specially to my mother,

Begum Laila Anjuman Banu.
Chapter 1

Introduction

From the beginning of the Web as a platform for exchanging important documents has transformed into a multi mass media which is capable of initiating interactive communications among a large group of people, literally from any part of the world [1]. The exciting development of computer applications and Internet technologies managed to become the center of attraction. In recent years, social networks took a remarkable place in the World Wide Web by motivating people’s interaction and expressing their professional and personal view as well as interest. After having a great number of such service, it has become a huge data source that is pretty hard to keep track. As a consequence, it is essential to reduce the contents effectively and reformat its structure to a certain standard where data format would be more users friendly and easily accessible. With this idea of interaction and standard format, corporate community can have internal cooperating system in where a lot of independent systems would aggregate their data as well as generate a special activity format called Activity Streams. This master thesis intend to specify a system with the idea of generating efficient high level Activity Streams along its standard format and structure for the corporate area. To motivate this system’s architecture and its technical infrastructure some research questions and requirement analysis are taking place which details and preferable solutions are described in the following chapters.

This chapter describes the motivation of high level Activity Streams in a short overview. It summarizes the trend and problems existed in the current aggregated systems which deals with a huge amount of feed items. And therefore, it runs through the importance and benefits of generating high level Activity Streams sophisticatedly.
1.1 Motivation

The use of modern technology made a significant change in our life style. The world becomes much smaller day after day by the blessing of advanced communication materials. Instead of spending long time for instructing employees, managers now have an opportunity to disseminate the same information through electronic communication. In Business-to-business (B2B) context, employees might use their team network to keep abreast of deadlines and stay on top of priorities by monitoring the actions of their colleagues [2]. As an example, when an employee or a manager leaves a commercial meeting he can post a micro-message about what work has been done so far and what went wrong. So, the team would have a broader idea about who’s in the pipe, can sharpen their arguments and share contacts[3]. In the software company perspective, the technical team can post their micro-message each time when there are problems on the technical platform or even if any new feature of software is released. They can share their experience and find standard solutions for a particular problem by collaborating with their idea. The great benefit of this technique is that other people of this group have a good chance to become aware of discussed problem.

With the development of advanced technology, people have a great chance to collect more information within very short time. Basically, every member of a company has distinct area of expertise, interest and intension. Each time someone found any relevant article or business information, they can share within their corporate network or system. Actually, there are lots of such independent systems available in the current trend of technology. And each of this system has a huge amount of data feeds to share within their group mates. It’s often very hard to keep track of every part of information that is published in different system. For example, this following use case explores the real problem in Software developing company: Let’s assume a company has couple of independent online systems such as Wiki to documenting particular stubs and article, SVN to maintain current and historical versions of software files or source codes, Bug tracker system to keep track of reported software bugs in their work.

- A worker of this company has to visit all particular systems separately to keep track of updating information. There is a great possibility for every worker of this IT Company to miss some particular stubs from there because all individual systems are continuously updating. They cannot follow these systems all day along. One smart solution would be aggregate these data into one particular system and view them gradually.

- But the problems of very frequent updating are still remaining. And the system would become a huge data bank within a very short time. So, it’s still hard to
keep track for workers. If someone needs to have a look past week’s posted feeds about any important things then it’s become a very complicated task because he has to find things from more than couple thousand of posts.

The described example shows the remaining problem and infeasibility of existing cooperating systems which would be strong cause of missing information. Basically, this data overload problem exists almost everywhere in the web where there is a collaboration of user. For example, social networks such as Facebook, MySpace, Twitter etc. has huge amount of updating data and it is really hard to keep in touch of particular’s interesting topics. This thesis aims to offer a feasible solution by reformat the Activity Streams structure to high level Activity Streams and therefore offer a user friendly automated system idea which gave users a guarantee of get in touch of every particular data he or she is interested in. As with the technology updating and more users are involving in World Wide Web day after day, there is no way we can keep the existing system up. So it has become an important task to chunk the relevant data feeds into much smaller pieces, then there is a way to organize one’s time around the ability to consume the information. So, it should be better look for an automatically controlled system. It would be very helpful if there is one system which is able to fetch data from different system and therefore view them gradually as individual’s interest or need. This system should be able to visit a number of independent systems and gather the information into a unique database. Then aggregate the data based on some intelligent technique and therefore organize them in a special format called high level Activity Streams. And the whole system could be called Activity Streams Generator. It aggregates the data from conversation, idea or technology based on similar type of contents. And therefore views the data to the user interface according to a predefined user friendly standard format and structure. This system uses some advanced intelligence which is basically based on syntactic and semantic logic technology.

Activity Streams can have an important for business or corporate organization to realize the true value of data aggregating from the social web [2]. It can save huge amount of data within very short time. The goal of Activity Streams is to provide sufficient metadata about an activity in a rich human friendly and machine processable format. This includes constructing readable sentences when there is any activity occurred, then visual representing of that Activity Streams by aggregating similar types of activities [4]. Activity Streams brings a better method of representing conversations because unlike with search, where the information is based on a keyword one came up with, Activity Stream gives user the opportunity to experience information favorably[2]. It gives machine the ability to monitor people’s behavior and take decisions based on different trends. So, the notions of discoverability and trend watching became possible.
by using standard format of Activity Stream. It should be clear at the beginning that there is a distinction between newsfeed and Activity Streams. A newsfeed (or web feed) is special data format used for providing frequently updated activities. As an example, popular social network Facebook uses news Feed that highlights information includes profile changes, upcoming events, and birthdays, among other updates. It also shows if any conversation is taking place between the walls of a user’s friends in Facebook. But Activity Stream consists of many small part of information which can be filtered, searched and automatically controlled by machine[5]. They are a combination of quantitative small stories which has definitely qualitative attributes. The part of data in Activity Streams can be used to predict what is more important for one people based on his or her past activities. In order to make Activity Streams more useful inside organizations, it is important to make it with some semantic format structure which would support employee easy to tune in and out of different activities, so that they can manage their attention on particular things in which they are interested.

Some few terms that will be helpful to know for the following chapters are defined here:

*Feed item*: is a entry in a detailed description of web feed or news feed. Entry may contain several parameters such as Author name, Feed id, Title, Text description, Published time etc.

*Atomic activity*: could be defined as a certain way to view feed item’s short overview. One activity is created from one feed item.

*Stream*: Several activities are represented in a row.

*Activity Streams*: In general, Activity Streams are the lists of activities performing by a user called Actor. Basically, these are atomic activities.

*High level Activity Stream*: In this thesis, Activity Streams are going to be reformatted to a standard and restructured with the high level activities which offer more users friendly environment as well as efficient data access.

### 1.2 Main goals and Research Questions

The central goal is to design and implement a system which should be able to automatically generate high level Activity Streams from different independent systems (e.g. Version management system, Wikis, Bug tracker etc.) for corporate use (e.g. Internal Software Company). This system will retrieve lifetime information as a web feed from different independent systems while any user activity happens. And by retrieving information it should have a standard methodology to aggregate this information
and generate high level Activity Streams with high level activities. High level Activity Streams should be represented in a standard format and structure where its syntactic as well as semantic structure should be finalized. Finally, this system should be able to show graphically these high level Activity Streams to the user interface.

The following research questions define the major goals which should be achieved in this thesis,

- How the system architecture and model would be designed considering requirements scenario?
- What should be the way to transform feed items to activities?
- How to realize the isolation of similar activities in an easily configurable manner?
- How to aggregate atomic activities to high level activities?
- What should be the standard format of high level Activity Stream which is applicable for corporate use?

These research questions and goals are addressed in the following chapters by designing, implementing and evaluating a system which can automatically generate high level Activity Streams from different source of feed items.

### 1.3 Structure of the Thesis

After introducing the main goals and the research questions in this section, the rest of this thesis is organized as follows.

Chapter 2 introduces some of the important use cases analysis of this system. In addition, this chapter describes a list of requirement analysis for this automated system in a clear view.

Chapter 3 describes some involving technologies and common approaches in existing standard systems. For example, Web feeds standard and its basic needs, existing feed aggregator’s approaches, definition of Activity Streams and its current standard. This chapter shows the clear difference between aggregators and Activity Streams generator. The most important part of this chapter explores State-of-the-art of generating Activity Streams. These sections show the existing standards and limitations of current Activity Streams generating systems.
Chapter 4 introduces the design of high level Activity Streams generating system called CoASGen (Consolidation and Activity Stream Generator) system with its internal system architectures. It describes what would be input of this CoASGen system and how these inputs (feed items) from heterogeneous sources are retrieved and aggregated together. Therefore, how these aggregated feed items are isolated to different type and finally how high level Activity Streams are generating with which standard semantic format structure. This chapter shows a clear graphical view as well as design description of every components in CoASGen system’s architecture.

Chapter 5 shows more detailed description about the prototype based implementation of CoASGen system. It emphasizes the architecture and components responsible for the CoASGen system which is explained in the design chapter. The important packages and decisions of implementations are described here with graphical diagram. In addition, responsible programming language, technique and frameworks are also discussed in the end of this chapter.

Chapter 6 evaluates implemented CoASGen system with some evaluating methods such as testing its functionality, performance and efficiency. At first, a case study is presented. Then there is a technical evaluation about the system performance and internal processes. An end user study is conducted here where there are number of questionnaires is conducted to evaluate the usability of this system’s application. Finally, the whole CoASGen system is evaluated with the goals and requirements analysis.

Chapter 7 concludes this thesis with describing main contributions and the research solution of explore research questions. The features of the developed prototype based implementation and future work proposals for further research are also described at the end of this chapter.
Chapter 2

Use cases and Requirements

2.1 Use cases of Activity Streams

Over the last couple of year’s communication patterns, information technologies had been motivated primarily from face-to-face to more online communication such as email, IM, text messaging and other tools. However, some people argue that email is now a slow and inefficient way of communication technology. As an example, time consuming ‘email chains’ can develop, whereby a small number of people are involved in a lengthy communications such as arranging an official or business meeting. The 'one-to-many' broadcasting technology offered by corporate social community is thought to increase productivity by circumventing this idea [6]. When we think about cooperative communication network, high level Activity Streams can offer an environment with more smart information share and time efficient communication. With the new era of information technology, high level Activity Streams has a vast area of application in the corporate area which would change the idea of business for the next generation. Based on different business objectives and function, different companies can design and build different types of social communities to serve and emphasize their information share. It can be also in terms of type and number of members, the type and frequency of editorial content and finally government, reputation and reward systems. It is not easy to define high level Activity Stream’s use cases concretely. It can be applied many field in many way. However, here in the following section there are some use cases included for high level Activity Streams which can have a significant impact in the corporate company area.
2.1.1 Software Developing Company

Usually Software Developing organization uses a number of independent online systems (e.g. SVN to keep track of frequently updating source codes, Wikis for documenting scientific implementations, Bug tracker system for keeping track of reported software bugs in the projects etc.) to cooperate and keep track their implementation with their coworkers. Typically, every employee of this organization has to visit these systems often to keep track which things are updating and which things are important and interesting for them. Actually, it would be a cause of losing lot of time. In such environment, it would be desirable to aggregate feed items from different independent systems and offer a special system which would gather and view these whole data feeds. But this aggregated system could overflow within a short time by huge amount of feeds due to the frequent update of these different source independent systems and then it would be still hard to keep track in this system. And other problem is finding information from the past (e.g. from the last week) would be lengthy and tedious because user has to find a particular feed item from very big amount of data.

The standard solution would be for such an environment, the development of a system where it generates high level Activity Streams from the similar type of isolated data and which would reduce the smart amount of contents and formulate a new standard to follow the relevant data. It would save a big amount of time and data lose, and therefore drive users to a particular direction for solving problem. For instance, a communication website where open source developers collaborate to share and improve their code, sets a good example of how corporate organizations might use the social web to improve their technical processes. So, by developing an independent system using high level Activity Streams idea, it is become easy to find relevant data and then follow similar activities based on particular’s interest that would save a smart amount of time and also become very helpful to prevent the missing of important data.

2.1.2 Corporate Organization

Within organization, there is a great application of high level Activity Stream to share the information and do flow more efficiently. But often there are some archetypal boundaries inside organizations which would restrict those data flows. Usually, in corporate company infrastructure, there is an intension to keep their data and interaction secret. But internally they can have a significant improvement of their business process by sharing their data internally with employee and special customers. It is very often in a company environment that they have lots of social collaborative networks for different aspects. For example, they could have different internal network service for marketing
experts, employee community service and a network service for technical support teams. If a particular member has access in all of these systems it is hard to keep track of all of these networks on time and it is also time consuming issue. So, there is a strong possibility for members to miss any important part of discussion that could be really important and relevant to them. In practice, it is impossible to keep in touch with the updating information on time because there would be more than thousand of topics of discussion within a hour in such a big organization. So, the solution would be to aggregate these systems to a general one system where feed items would be updated in real time. But due to aggregating lots of data this aggregated system will be overloaded for sure. And by very frequent update it would be very easy to lose in the data ocean within couple of hours. In this case, high level Activity Streams could make a super connection between these internal networks and would offer all the exciting and updating activities by aggregating similar type of information using a standard and user friendly format.

For instance, in the ‘Employee Community’ service there could be a discussion about last day meeting and the decision taken. They can discuss for any particular part of the decision with distinct idea from different direction. And the active members are connected in this service of that company who needed to know the discussed information, have to be always active in that service as well as look all discussion update by very frequent checking which is almost impossible because this discussion could long couple of days. And within this time others can have some discussion in this service about other issues. So, if any employee wants to find information from back date, he has to find from a big amount of data. There is a big possibility to miss some discussion information and this finding process would take long time. High level Activity Stream would aggregate this information and isolate them in a topics and organize these discussion (feed items) together such a way that is easily accessible. So, people of this company are not going to miss any important data without visiting continuously these internal social webs. Using Activity streams, it will become employee’s choice about which activities he is interested in and he will get a chance to have a look all relevant data in a blink by checking the metadata of updated high level Activity Streams. Using high level Activity Streams in internal communities, a particular company can communicate updates and information to employees across departments and even around the world in real time. From office, home or even across the continents, employees and team members can have encouragement faster, better team work and project collaboration among them. It has a great positive effect inside a company by reducing communication costs and increasing productivity.

Even if there is only one internal cooperative network or service with huge amount frequent activities, then high level Activity Streams can also play a vital role by generating a special format which would reduce a huge amount of data and then separate similar
type of interesting and important data for individuals. It would also save a good amount of time. With this kind of high level Activity Streams generated system, employee and co-worker can find their desired information at a glance. Getting the right information at the right moment would encourage employees to become more efficient. Organizations have a significant use of high level Activity Streams to improve their workflows. For example, if someone is working on a project with other employees, dependencies and milestones of project could trigger status updates to inform them if the project is ready for their contribution. People might have software agents which would look for certain types of activities or patterns, and then it will trigger responses or alerts which would make this workflow processes more efficient.

2.1.3 Network Management/Support Community

Network administrator could use high level Activity Streams to get connected with his or her client and therefore control the network in a sophisticated way. Today, we have a lots of network management system all over the country. And they have several distinct service systems such as internal network group (where administrator can let all users know if for certain amount of time network is not available for any reason, any certain software update could need certain system requirements or even taking care if any user has any particular network problems), wiki for sharing their technical experiences and information, online data store for different version of available software source etc. In this environment, it is desirable to aggregate activities from these different independent source systems and therefore gathers the frequently updated system’s activities into a general system where there will be all the current important and interesting information. And for saving users from any important data miss caused by data overflow, it is desirable to use a special reduced format and structure which could possibly offered by high level Activity Streams. Then it would become a time efficient system which is able to fetch lots of activities from different independent systems and view these informations in a standard sophisticated format. So, people don’t need to visit all the systems very often. High level Activity Streams will let them get in touch when there is any important event happens (caused by updates of activities). Therefore, network administrator would be able to solve problems and stay tuned about everything of this network without visiting all of these independent systems. Other peoples those are connected in this network also have a great opportunity to solve the basic problems by look at the similar type of discussion activities. So, high level Activity Streams makes the system easy to find any important information for every users and network administrator of this community or network.
2.2 Requirement Analysis

The research questions and requirement analysis are basically motivated by design, implement and evaluating of the CoASGen system which major requirements includes aggregating information activities from heterogeneous independent sources and therefore generating high level Activity Streams from these information activities. This thesis work should offer some standard high level Activity Streams format with its syntactic details description. More details of this system requirement analysis are addressed in the following.

1. The system should be able to deal with single and heterogeneous data sources.

2. System should have a standard and efficient aggregation and consolidation method for the real time feed items. Since the system works with big amount of data, using various method isolation of similar feed items would be a important task.

3. Generate high level Activity Streams from lot of atomic activities. System should have a idea about how to generate high level Activity Streams which would be a number of similar feed item’s (details of activities) representative. It should be easy to understand for the human and the feed items behind it should be easily accessible.

4. The proposed system should offer some standard of high level Activity Stream format structure. The system should give users the possibility to choose one of their recommended format to generate high level Activity Stream.

5. Should have a standard configuration language for generating streams from direct feed items. It should be able to recognize different parts (syntactic) of feed items automatically and generate activities by combining these parts. And it should have a standard way of representation of these generated activities as a stream in the user interface.

6. The system should be completely automatic which means without any help of other system or service, the system should work by itself.

7. The proposed system should be scalable with the size of information increase and with the number of different independent systems uses as source systems.

8. System should have good consistency in system level operation. Operation should be persistent using fault tolerance.

9. It should offer a way to configure the system by both administrator and end user.
10. The system should be designed by considering corporate company environment where a number of collaborating systems are used.

For more clear view of the requirement analysis and goals of the intended system, requirement parameters are described and scored here in the following table 2.1 with their importance level on a scale from one to five, where one represents the least level and five represents the top most required level to achieve. In this table, high level Activity Streams is represented as HLAS. Here, some parameters are described as goals and some of them are left with empty (-) which means they are needed to design the system but not a system goal.

Table 2.1: Requirements analysis and goals in a table view

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
<th>Importance</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous Sources</td>
<td>Able to deal with single and heterogeneous data sources.</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>Aggregation and Consolidation</td>
<td>Should have a standard and efficient aggregation and consolidation method for the real time feed items.</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>Generate HLAS</td>
<td>Able to generate high level Activity Streams.</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>HLAS semantic Format</td>
<td>System should offer standard semantic format structure of high level Activity Stream.</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>Stream Generation</td>
<td>Should have standard configuration language for generating streams from direct feed items.</td>
<td>4</td>
<td>Yes</td>
</tr>
<tr>
<td>Stream Representation</td>
<td>Should have a standard way of representation of streams in the user interface</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>Automatic System</td>
<td>The system should be totally automatic.</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>Scalability</td>
<td>Scalable with the size of information as well as with the number of different independent system.</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>Accuracy and Consistency</td>
<td>Should have good consistency in different system level operation.</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>Configurability</td>
<td>It should offer a way to configure the system by both administrator and end user.</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>System Development Intension</td>
<td>Intended to corporate company context</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Intended to enterprise or social network context</td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>
Chapter 3

Technical Background

Technical Background is organized with common approaches and technical tools which are necessary to understand for designing Activity Streams generating typical systems. First section deals with typical web feed standards with some real-time examples. Basically, each feed aggregator has been designed with particular intension of implementation. There is no standard aggregator found in this generation which can be used for Consolidation issue of CoASGen system. Furthermore, Different available feed aggregators and their technical details are described in this chapter. Third section deals with the existing standard of Activity Stream in the current web market with their format structure and description. The state-of-the-art of Activity Stream generating systems are described in section four. Basically, those systems offering Activity Streams are mostly in enterprise context of implementation and they do not provide that much technical detailed description. So, some existing system’s general idea, concepts, usage of Activity Streams and brief idea of API, format structure are described in this section. Finally, the Technical Background chapter ends with a high level comparison of existing Activity Streams offering systems with a number of comparison parameter of interest.

3.1 Web Feeds

Imagine a company updating some products every month and announcing periodically in their website. For visiting that information, customers have to remember and find in that particular website to get in touch with its update. If the company can provide these informations in a special format of data, then users or customers can set an aggregator in their own computer which is able to fetch data from that website with a link and they can be always informed as soon as there is any update in any product of that site. For frequently updating contents in the web, there is a special data format called web
feed. It is also called news feed or even sometimes syndicated feed. Typically, web feeds are syndicating by content distributor where users are allowed to subscribe it. Basically, web feeds are XML based content whose concrete part of the items are typically linked to the web page in its source content. The main sources of web feeds are news website and blogs, which are used to view and manage to the user interface in a user friendly manner. RSS and Atoms are the basic format of web feeds. Typically, web feeds are providing in the web interface using a special icon by that web pages original provider. Some websites still do not provide web feeds. In this case, it is possible to read this contents from the web page by a third party and generate web feed by scraping it.

Currently, most of the data visiting website through Internet are organized with list. For example, news headlines, search results, job vacancies etc. are following the idea of list with some links. And this is the idea to keep a large amount of data in an organized way. Otherwise, it could be hard to keep track of each handful sources. The basic structure of web feed contains a list of information items or entries which are typically followed by links. Each entry could have a list of metadata associated with it. The metadata contains some entries or items such as a title, a link for it and its description. For example, a title link is used for news feed to view in the user interface which is easily followable. A general web feed structure is given in the figures 3.1, 3.2, 3.3. The intention of designing web feeds was to make a machine readable format which is able to automatically transfer information from one website to another by feed format syndication. A web feed aggregator or feed reader collects and combines feeds from different places according to the users recommendation and thereby view to the users in a single web screen as a series of contents. A typical use case of web feed is for example, a content provider publish a web feed link in the web. An end user can subscribe it using feed aggregator or news reader which is running on his own remote machine. When it does so, the aggregator queries all the responsible server of its feed list to have a look if there is a new feed there. If the aggregator found any update or new feed arrives, it makes a note of this update or even downloads it to the aggregator or feed reader. Aggregator or feed reader can be configured for checking new contents from different site periodically.

3.1.1 RSS

RSS [7] is a XML based format structure which offers the syndication of information as a list following by links. It allows keeping format along with several useful information or metadata which helps each visitor to decide if they want to view it or not. This is a machine readable format that allows computer to fetch and understand the information. Which means, the machine can use some intelligence to view these information based
on user’s personal interest. This format is designed for directing several informations by computer on behalf of user, rather than user’s manual direction. For using this idea a website has to generate feed and make it available such as the contents are available in a server. Then computer can automatically search and gather the most recent information as a list from several websites very easily. For making available this RSS feed, it is important to indicate it to the viewers using link containing from any site. For example,

\[
\langle a \text{ type}="\text{application/rss+xml} \rangle \text{ href}="\text{feed.rss}"\rangle \text{RSS feed for this page} \langle /a \rangle
\]

Here, feed.rss is the URL link for a RSS feed. And the field 'type' indicates to the browser that this is a link for currently updated RSS feed. There is also one other technique called autodiscovery which is used to find feed from a page [8]. For example, XML::Atom::Feed\-new($stream) creates a new feed object which is used to specify data by $stream and XML::Atom::Feed\-find_feeds($uri) is used to link the atom feed with the page using link tags.

In the current age, there are couple of format structure offering by RSS feed. The most widely used format is RSS 2.0, which is simple and easy to use for semantic logic. Another version of RSS is RSS 1.0, which has more specified and extended format structure. Both of them are XML based format and easy to parse by any typical parser. The feed format Atom is comparatively well documented and standard syndicated format. It has a comparatively large described structure which is easily applicable for ontology and semantic application. This section describes a detailed overview of typical RSS format.

### 3.1.1.1 RSS 2.0

In RSS 2.0 [7], RSS means "Really Simple syndication” which is based on RSS 0.91. At first, RSS 2.0 was documented by Netscape which is later refined by Userland. The main intention of designing RSS 2.0 was to create its simple format in structure. The latest stable version of RSS 2.0 is 2.0.1, which has channel metadata that includes link, title, description etc. This metadata allows its user to look a thumbnail image to display this feed. It also shows some parameter for example webMaster, managingEditor, which describes about the responsible person of the feed, and the parameter lastBuildDate shows about last version updating time of the feed. And finally items (title, description etc.) might have a guid element to identify them uniquely. This functionality allows to provide some advanced functionalities to work with aggregators. In the following figure 3.1, there is an example of RSS 2.0 where the channel of RSS 2.0 describes with its title, description and followable links. In RSS 2.0, to avoid conflicts in the item names, XML
namespace can be used for extension and separate modules. This format structure is easy to reformatatable.

3.1.1.2 RSS 1.0

RSS 1.0 [7] is incorporate with RDF (Resource Description Framework) which is a web standard metadata. That is why for RSS 1.0, RSS abbreviates as RDF Site Summary. Because of using RDF processable format, any RDF processor is able to parse RSS 1.0 feed without knowing its any details. This idea makes RSS 1.0 easily applicable for semantic web. XML namespace also allows its extension and separate modules by RDF processor. There is an example of RSS 1.0 in the following figure 3.2. The RDF processor knows RSS 1.0, because the entire document of RSS 1.0 is wrapped in <rdf:RDF>... </rdf:RDF> elements. Each item of the RSS 1.0 has an rdf:about attribute matched with the link which work as an identifier. This format has an attribute called

![Figure 3.1: A real feed example of RSS 2.0 format [9]](image-url)
item which works as metadata of the feed and let RDF processor know the relationship between different items.

```xml
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns="http://purl.org/rss/1.0/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  >
  <channel rdf:about="http://www.xml.com/cs/xml/que/y/a/19">
    <title>XML.com</title>
    <link>http://www.xml.com</link>
    <description>XML.com features a rich mix of information.</description>
    <language>en-us</language>
    <items>
      <rdf:Seq>
      </rdf:Seq>
    </items>
  </channel>
</rdf:RDF>
```

**Figure 3.2:** A real feed example of RSS 1.0 format [9]

The Dublin Core Module\(^1\) is the most well known module of RSS 1.0. It uses set of metadata which is developed by librarians and some information scientists. This module standardizes a set of common metadata which is able to easily describe documents in compare to others. Both feed and individual information are attached into a metadata by using Dublin Core Module. For example, this module includes some useful parameter

---

\(^1\)Dublin Core Module is a standard that is easy to understand and widely used. Dublin Core (DC) defines for instance; the title, the subject, the date, the creator, the publisher of a resource.
such as dc:date which represents the date of the item, dc:rights is use to describe the property right of the contents. RSS 1.0 is more verbose than RSS 2.0 because its design intension was to support both general and RDF processor markup description.

3.1.2 Atom

Atom [7] is an XML based feed format which functionality is similar to RSS family. Unless RSS 1.0, 2.0 and Atom format specification is not informal to apply by a well known standards body and industry consortium. RSS 1.0 and 2.0 specifications can be changed based on people’s particular concern. But Atom has much straight forward specification which brings stability by limiting its structure changeability and having a standard established procedure to introduce in the environment. To standardize its format structure a group of people establish an Internet Engineering Task Force (IETF) group and they call it Atom. In the figure 3.3, there is an example of typical Atom feed which shows how Atom structure look like. Atom standardizes its format structure with

```xml
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Example Feed</title>
  <subtitle>A subtitle.</subtitle>
  <link href="http://example.org/feed" rel="self" />  
  <link href="http://example.org/" />  
  <id>urn:uuid:60a76e00-d399-11d9-b91c-0003939e0af6</id>  
  <updated>2003-12-13T18:30:02Z</updated>  
  <author>
    <name>John Doe</name>  
    <email>johndoe@example.com</email>
  </author>  
  <entry>  
    <title>Atom-Powered Robots Run Amok</title>
    <link href="http://example.org/2003/12/13/atoms01" />  
    <link rel="alternate" type="text/html" href="http://example.org/2003/12/13/atoms01.html" />  
    <link rel="edit" href="http://example.org/2003/12/13/atoms01/edit" />  
    <id>urn:uuid:1225c695-ef88-4e6b-8eaa-80bf34459a9e</id>  
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Some text.</summary>
  </entry>  
</feed>
```

Figure 3.3: A real feed example of Atom format

both feed level metadata as well as the entry. Entry elements could contain title, link, id and a brief text description where 'id' is similar as RSS 1.0's rdf:about and RSS 2.0's guid, 'brief text description' is similar as RSS's description.
3.1.3 Summary

The purpose of considering these structures is to show available the feed formats in the web. And each feeds are the input sources of the CoASGen system which works as every particular activity. There are some more available feed formats are using in the web for example RSS 0.9, 0.91, 0.92. However, RSS 2.0, 1.0 and Atom are the most common used format in the web. That is why those format structures are only considered. Nevertheless, it is easy to adapt with all other formats.

3.2 Different Feed Aggregators

In the current web technology, there are many mature feed aggregators available in the market. The feed aggregators are often called feed reader or news feed reader. Feed aggregators are applied in the system both stand alone applications as plug-ins and features implementation within larger application. Both feed readers or aggregators are retrieving feed from different websites and aggregates them into central database. Then it displays these feeds to a particular user according to users need. Most of the feed readers or aggregators provide opportunity to the users to choose any particular type of information retrieve and show them gradually. For example, there are several newspapers in the Internet such as British Broadcasting Corporation (BBC), Cable News Network (CNN), Fox news etc. and they are updating real time news. If someone is interested in political news or educational news, he may choose these options in the aggregator. So, this installed feed aggregator or feed reader on his computer will only retrieve these particular types of information from these newspapers and view him as soon as there is an update. The user neither has to visit every particular newspaper’s site nor that particular area of topics he was interested in, but the feed aggregator or feed reader will do it automatically for him.

There are several standard Feed aggregator available in the current age. But each of them developed with particular area of intension. For example, NewsStand is a news reader or feed reader which can retrieve information as a feed and display to the user based on both topical significance and geographical region [10]. It divides information by finding topic’s similarity and classifying regional aspects. Some of the current aggregators used in semantic web technologies which are providing unified information from heterogeneous source of information. In this section, there will be a brief description of some advanced feed aggregator’s internal techniques which has comparatively close development interest with CoASGen system.
3.2.1 Content-Based RSS and Broadcast News Streams Aggregation

In order to retrieve the information from variety of desired channels, users should have a low cost semantic aggregation of heterogeneous multimedia system. Content-Based RSS and Broadcast News Streams Aggregation system [11] offers an unsupervised framework for reading web based news article and broadcast news stream aggregation. The core materials of this system playing with a hybrid clustering algorithm that use information sources as RSS news feed and television newscasts programs. It builds a multimodal service which offers a combination of these two information sources. This system offers an integrated system which allows users to browse information coming from multiple data delivery and presentation platforms. The retrieving and managing data in the system is automatic.

The following figure 3.4 shows the schematic architecture of Content-based RSS and Broadcast News Streams Aggregation system. The system has two input data sources to retrieve the information. One input belongs to digitized broadcast news stream Digital Television (DTV) and another input source for online newspapers feed (RSSF). There is only one combined output system which is called Multimodal Aggregation Service (MMAS). MMAS is automatically determined from the semantic aggregation of the input streams and news feeds. Semantic aggregation works as a data input source A1 is semantically agreeable with another data input source A2. If A1 satisfies with the user expectation of A2 also satisfies. For example, different news sources often offer similar kinds of news (e.g. Japan affected by a tsunami) both in news channel and news feed. The MMAS is able to automatically detect the similarity and connections of these news sources, and then it shows these informations in the browser as a large heterogeneous multimedia archives. For semantic aggregation both of these input should be prepared for semantic structure. At first, this system makes semantic structure of the information sources and then applies to the semantic aggregation. News streams are automatically detected and segmented into their elementary structure. Using visual pattern matching technique the segmentation process is following through each of the DTV stream by exploiting aural and visual cues. Once segmented, an Automatic Speech Recognizer (ASR) tools is used in segmentation process to analyze news streams by transforming speech into textual description. The ASR tool is able to work with English and Italian language. Then each stream is categorized based on its main topics and textual description. Finally, these transcriptions are indexed to TVi documents catalogue. TVi documents catalogue is a kind of search engine server which allows full text searching and category filtering facilities.

RSSF streams comes up with RSS feeds which includes major online newspapers and press agencies. There is a brief description of title, description, publication date, time
and a corresponding link for each of the RSS feeds. Additional metadata which includes list of item’s category, some comments are also could be included in RSS specifications. This system stores every item as a tuple in the PostgreSQL database record which organizes as uuid, pubDate, link, feed, title and phrases. Then for identifying meaningful linguistic structure on each item, a linguistic analysis is performed to the title and description of each RSS item. By using a query constructor, a set of representative query expression is generating from the output of linguistic analysis. Finally, it submits and stores them to the index structure of TVi document catalogue.

Based on the semantic similarities, a hybrid clustering process is performed to aggregate the items (i.e. broadcast news stories and news articles) which is called ‘dossier generation’. Then, a unique identifier is given to each RSS feeds items and broadcast news stories for each multimedia dossier. Finally, this information is stored in the multimodal aggregation index (MAi) which is a permanent database and serves as a query answerer of front end user. It supports even more advanced query structure. The search results are providing in RSS feed format that’s why users can subscribe to the query and will be notified if any modification in the result page.

The main goal of this aggregator was to aggregate data from heterogeneous data and semantically aggregates them. Then show these informations as RSS feeds to the user.
according to their semantic similarity. The good part of this aggregator is that it uses semantic aggregation system which requires semantic knowledge about each of the stream. But this aggregator is made for a special aspect which is retrieving data from newspaper and television news history. So, it is not applicable for any other aspect. But the idea of this aggregator is quite nice and interesting.

### 3.2.2 Supporting Situation Awareness by Semantic Aggregation

Situation awareness by semantic aggregation [12] offers a Supervision tools to aggregated data and monitor data from number of different tools. This feed aggregator uses some approach to data collection from different interdependent system of a Software Forge\(^2\). The system is designed for some special independent system such as Wiki, bug tracker, mailing lists and source code management tools. It collects information and views them in one integrated PicoForge\(^3\) platform to the members. This feed aggregator aggregates information using semantic web standard: Resource Description Framework (RDF), Dublin Core (DC), Description of a Project (DOAP), Friend of a Friend (FOAF) and Evolution Ontology (EvoOnt). The aggregated data then displayed or re-published to the users of this system after processing inside. Basically, this system is developed to supervise a number of independent information systems which are used to collaborate and data management in a software company. This system offers a situation awareness semantic aggregation system for supporting tools. It provides three stages of situation awareness stage or steps such as perception, comprehension and projection. There is a brief description of these tools in the following figure 3.5. Perception deals with data collection for the tools of this feed aggregator. The responsibility this phase is to perceive the status, attributes and variants of relevant elements in the environment from every particular independent system. This level of feed aggregator deals with monitoring processes which helps to detect the awareness of multiple situation elements (i.e. object, events, people, systems and its environment factors) and their certain current status (e.g. modes, actions, locations and conditions). Finally, this phase identifies the situation of the entities. Comprehension works for data processing phase of this feed aggregator. The goal of this phase is to make a connection of situation entities. It integrates the information collected by perception phase and detects their impact upon the individual’s goals and objectives. This phase makes a comprehensive image of individual’s concern from the world view. The third level of tool, projection deals with the highest level of situation awareness feed aggregator which includes semantic analysis. This level predicts

\(^2\)Software Forge is a collaboration platform allowing collaborative software development over the Internet.

\(^3\)PicoForge is the name for a set of LibreSoftware applications which are integrated into a Web platform, which allow collaborative work, for software development (a Software "forge") or other collaborative work needs.
the future actions of the elements involve in the environment. It works with knowledge of the status, variation of the elements and comprehension of the situation which received from the earlier phase. With this knowledge, it determines and predicts how it will affect the future states of the operational environment.

Disparate and heterogeneous data are retrieving from multiple sources by the system and these data integrated together for further analysis. It aggregates feeds using RDF schema which uses RSS/RDF, FOAF, DOAP and EvoOnt schema to represent data. The first step of semantic analysis is to identify the syntactic elements of the feeds which are going convert to semantic information for further calculation. The resulting formats of semantics are following some web standard format such as DC, DOAP, FOAF and EvoOnt which are allowing their manipulation by different tools to understand these standard formats. The resulting formats are working in the following way. To help RSS/RDF items, this system integrate a FOAF schema which is responsible to describe about the author of the notified actions to the developer. When the item describes a commit or a bug, it also references to Voice of the Martyrs (VOM) and Bill of Material (BOM) documents. DOAP used to describe each project in the forge’s portal which provides a public feed to let everyone know about the update. The figure 3.6 shows the architecture of the Forge ’Supervision’ tool. To export a list of project in RSS or RDF format (as FOAF and DOAP), this tool offers a querying public projects to the platform. DOAP is describing about the project while FOAF is used to describe its members and the semantic notifications of the project activity is operating by public RSS/RDF feeds.

From the figure 3.6, forge integrates several web application software for example TWiki, Subversion (SVN), Concurrent Versions System (CVS), Lightweight Directory Access Protocol (LDAP), which includes RSS public feeds to track activity. This Supervision tool will retrieve, mix and process data from several independent systems for example,
Figure 3.6: Supervision of PicoForge

Wiki, Mailing list, SVN, Bug tracker etc. For each independent system, this Supervision tool retrieves all non-semantic RSS feeds. Then it uses semantic idea to publish in the Supervision environment. In addition, it is able to add other RSS feeds from outside which are related to these projects.

The ‘Supervision’ tool is able to offer visualization graph which describes different independent systems member’s visibility at a glance. The figure 3.7 describes the total activity graph of the members of different systems for example, wiki, SVN, Sympa within last 60 days. This is useful for the project manager to manage the environment by monitoring recent activities. The main aspects of this feed aggregator is to aggregate data from different sources and therefore adding semantic architectures. It gives a comprehensive vision about the collaboration in the community than the results analyzed on a single source. This offers users availability in the graphical interface and monitor different independent systems information. It works with semantic aggregation which is rare in other available feed aggregators. But, still it is not working with high level semantics. That means generating activities by following any standard format structure from feed items and then generate Activity Streams from there. Its resulting data can be processed and displayed to the user. However, it is still work as typical aggregator.
3.2.3 Content based RSS Aggregation

Cobra (Content based RSS Aggregation) is a RSS based feed aggregator [5]. It is designed for Weblogs, RSS feeds and other live Internet contents. The frequently increasing feeds in such environment leads the system to the data overload. From the overloaded system, finding particular’s interesting contents become difficult to manage. The vision of Cobra is to provide content based filtering and aggregation system with millions of real time updating web feeds which are matches with the user interest.

Cobra is a distributed scalable system that deals with huge amount of user with a personalized view of articles. It consists of three tier network, which includes a crawler (collect and extract data from web feeds), a filter (matching article with the user subscription) and a reflector (provides matching article as RSS feed and makes the feed easily accessible from typical RSS reader). These three tiers can be allocated over multiple hosts in the Internet. The system balances its load by placing some heavy loaded host’s tier to compare to the low load taken host. The design architecture of Cobra is described in figure 3.8. There are several raw web feeds or source feeds are periodically crawling as a series by the crawler. Source feed could consist a series of articles and their number of articles are depending on how the crawler is configured. Typically, for blog or news, it crawls as an average 10 feeds per batch. In addition, this crawler is able to reduce crawling load of source feed in a light weight manner. Then in the next step, crawler sends these crawled source feeds to the filter. Using a case-insensitive index based algorithm, filter matches the contents of the article with the users subscribed items. Then it sends those feeds to the appropriate reflectors which are matched. The reflector generates RSS feeds for the user (subscriber) and finally publishes it to the browser or standard feed reader. Notice that user are finally getting RSS feeds as they
Figure 3.8: The Cobra content based RSS aggregation network

are subscribed. The reflector typically caching last 10 matched articles in the system because if certainly any user querying about any subscribed item, it will be always ready to answer. When a service is unable to work with the any certain data rate of upstream service, then Cobra system uses a congestion control mechanism to balance this data rate. For each service, it keeps 1 MB data buffer to maintain its upstream service. If any upstream service sending data faster than the data processing in the queue, then sending data will be blocked for a while until it get balanced.

The main aspect of designing this feed aggregator was to work with huge amount of web feed crawling, filtering based on user’s subscription and showing them to the user. The important part of this feed aggregator is its scalability with large number of data and load balancing. However, it is still a typical aggregator which is developed for a fixed aspect. No semantic aggregation has been used to find out correlation between different source feeds. In the filter level of this aggregator architecture, it does not consider higher level of logic to make it more useful. But this feed aggregator gives a good idea of distributed scalable feed aggregator.

3.2.4 Affect Sensitive News Agent

Affect Sensitive News Agent (ASNA) [13] offers a feed aggregator which is dividing feeds according to user’s emotion model. The main goal of developing this system is
to categorize news stories based on people’s emotional affinity understanding and then using natural language processing idea, offer web feeds to the users which they are really looking for. In order to develop ASNA, they also had to design a linguistic tool called SenseNet which is able to read single line of text and assess them based on common sense and current-affairs knowledge (integrated with it). Based on this assessment, it generates some emotion model which is further used to offer particular types of feeds to the user. This system fetches information from several RSS news feeds and auto categorizes them based on affect sensitivity. This feed aggregator offers eight emotion-types including one neutral category and offer them different types of feeds based on these emotion-type.

The task analysis of emotion affinity deals with favorable or unfavorable choice or user which requires emotional intelligence, close understanding of textual context, common sense calculation, domain knowledge and linguistic knowledge. The operational architecture of ASNA is step by step and quite straightforward. The internal architecture of ASNA is describing in the following figure 3.9. At the first step, user chooses a number of feeds from the user interface according to his or her domain of interest. These user chosen feeds are then collected with RSS feeds as a source of the news in the next step. After that, News Fetcher collects this information from the Internet in details. It makes tuples of news topic and brief story by parsing the information with respect to these previously selected RSS feeds. In the next step, these plain text tuples are parsed by a language parser and the language parser makes the output tuple of subject, action and object, and their type, attributes respectively for each line of text. The output tuple of

![Figure 3.9: The architecture of ASNA](image)

Language parser is then examined by a linguistic tool called SenseNet which considers
each tuple as a sense and makes a numerical value as an output for each sentence (lexical unit). Then news texts are become classified according to eight emotion types (e.g. Happy, Sad, Hopeful, Fearful etc.) by Affect Sensing Engine. Finally, each user gets an emotion type and when he browses, he gets news feed based on his emotion group.

The basic aspect of designing ASNA was generating a feed aggregator, which is able to offer particular news for particular user based on their emotional status. The main achievement of this system is that it can isolate news to different class. However, the aggregator is still simple and no advance semantics used to calculate news group or type. The filter level (considered in this system) is limited to eight emotion type which is an interesting idea. This feed aggregator cannot offer information feeds in a dynamic way that means it cannot read user intension automatically and offer them similar types of data they are interested in. For example, if one user working with something related to research work and most of the time he is finding information related to that. Then this system is able to understand his thinking by looking at his searching history. So, it will always offer him some feeds when there is any updated news according his research area.

3.2.5 Coalescence of XML-Based RSS Aggregator

Coalescence of XML-Based RSS Aggregator offers a synthetic analyzer named PheRSS [14] which is able to work with RSS feed using different feed format and generate a wider set of metadata. By applying some rules and constraints, PheRSS can offer to the user more relevant information according to user interest. This system solves the ambiguity problem which was caused by retrieving feed from multiple websites with different feed format, indexing them with their important words and querying them with word matching. The idea of indexing word in PheRSS by matching user’s search with word stored in the Thesaurus which has comparatively similar semantic meaning. The PheRSS follows a standard component based framework by following hooks and slots idea. It has a collection of ‘plug point’ which enables it to be adopted in blogosphere environment. The framework architecture of PheRSS is described in figure 3.10. Each live time feeds are collected from different web site or blog with different feed format which are transformed to RSS 1.0 format. These feeds are indexed into the database with pointing its important keyword. As soon as any user query processed in the system, the expected results are returned by the filtering the keyword of word Thesaurus. Rules are basically processed by if else statement. The figure 3.11 shows a flowchart diagram of PheRSS which emphasizes more details description of its operational work flows. When a user searches for feeds, it starts keyword matching from the indexes which are pointing to the databases. Notice that every feed can have more than one indexed keywords.
Rule 1 is performing to identify content from the index stored in the database which has matching more than two keywords. For more advanced check this system is able to filter the contents, such as if content does not contain unwanted word then display it and otherwise not display. This is a simple system for syndication of different format and answer query result based on keyword matching. To design PheRSS, they put more attention on feed format, that is also important part to retrieve feeds from heterogeneous independent system with different feed format. The feed representing method to the user...
is quite straightforward. It uses comparatively poor level of semantics to emphasize the idea of PheRSS. It solves the word ambiguity problem.

### 3.2.6 Summary

From the discussion of this chapter, we have seen there are some standard Feed aggregators already available in the market. It would be a good option if we could use one of these aggregators for CoASGen system’s Consolidation process (described in chapter 4) which would save a lot of workload and time. Unfortunately, our developing intention and methodology does not match with any of these Feed aggregators. We have totally different ideas of aggregation system (e.g. isolate the activities (feeds) based on similarity and dissimilarity) which is not yet developed. There are some levels of filters to check similarity of activities in the Consolidation process. This part is important for the system because the whole system has a great impact of this Consolidation process. For example, the main intention is to develop CoASGen system for generating high level Activity Stream. And each of the Activity Streams will be made from one type of isolated feed. If isolation process in Consolidation is not perfect and efficient then high level Activity Stream will mislead its user because they are not representing similar activities. Notice that the idea of generating high level Activity Stream is to generate a representative from several similar types of activities.

### 3.3 Activity Streams

In the corporate area, there could be frequently updating thousand or even more amounts of data sources which are playing important roles to manage its internal infrastructure or managing collaborative works within its employee. These data sources are generated by people, for example, through emails or collaborative network messages. Sometimes these data can be generated by system such as inventory tracking. One of the most frequent problems in such environment is information silos which can be defined as overwhelming of vast amount of data generated by the people’s interaction and system. This problem asks a standard solution which explores the question as how to optimize the flow of information in a structured way to motivate the environment’s efficiency. Activity Stream could be the desired solution for the problem of information overflow. It can work like a central nervous system of a company which unifies every piece of corporate information. In the idea of Activity Stream, vast amount of information become instantly accessible using previously defined boundaries [15].
Previously defined Activity Streams are the list of activity performed by users which are typically in a single website. For example, Facebook news feed is called Activity Streams. There was no standardized format so that different websites can solely interact with each other. Generating Activity Streams in a company internal infrastructure is comparatively easier than aggregating all the companies to a public social network and play nicely with their interactions. One renowned company called IBM is using the standard idea of Activity Stream to create applications which is able to deal with aggregated sources of Activity Streams from some different independent sources. The idea follows through all kind of applications from ERP (Enterprise Resource Planning) to CRM (Customer Relationship Management) to HR (Human Resource) and so on; all collaborative departments will be able to post their important events and messages to the news feed. In the figure 3.12, there is an example system of generating Activity Streams in an internal company infrastructure which deals with several internal used independent systems for ERP, CRM, Wiki, Andere etc. Feeds are adapted by Activity Stream adapter to a central internal activity database. Using some idea and analyzing these adapter data, Activity Streams are generating and representing to different network. Activity Stream data could contain payload information of any particular information which could be used by the device or cloud based service to flow information. Any Activity Stream, for example, a new employee is joined to a company could contain his full contact information in the internal description of this Activity Stream. So, this information can be used to update a contact information database by that Activity Stream application which can work to identify logical interactions. However, the idea still deals with atomic
activities (which they called Activity Stream). It is not formulated yet to the level and format of high level Activity Streams.

3.4 Feed Aggregator vs. Activity Streams Generator

Feed Aggregator works with retrieving feeds (most of the cases real time feed) from single or heterogeneous independent systems and furnishes these feeds to the user interface with different order and intension (i.e. there are some advanced feed aggregators described in section 3.2). Often, it uses semantics to draw attention to the different types of feeds. Some feed aggregators even offering streams which are generated by user activities.

Activity Stream generator uses feed aggregator for retrieving and organizing feed. Then using its syntactic and semantic format, it offers high level Activity Stream which is a representative of lots of similar types of streams. Notice that there are three levels of calculation here. First, feed items which are generally RSS feed format are retrieved and transferred to activities (i.e. Alex committed a bug report on Java ArrayList at 3.30 pm). Then activities are being isolated based on their similarity. And finally, using semantic knowledge of these activities, Activity Stream generator generates Activity Stream from several isolated feed items.

3.5 State-of-the-art Generating Activity Streams System

The Activity Streams is a standardized method to representing human activity within an online status update which was originally formulated by Chris Messina. The activity streams have been promoted in order to allow one social network to include activity from other social networks in its news feed. The milestone of Activity Streams describes a flow of its implementation with its specification. As an enterprise use of Activity Stream, the popular social network Myspace first offered Activity Stream as a new feature for MySpaceID Live in March 2009. Then Facebook offered Activity Stream with Open Stream API at 2009-03-27. Activity Stream idea applied to Windows Live at 2009-09-14 where it adds the list of third-party partners to bring more updates which is used to contact within Windows Live network. At 2009-09-30, Cliqset offers real time community for sharing contents which was using Activity Stream’s idea. Google Buzz offers social web at 2010-02-09 which works as interaction of Activities and news aggregator. Then BBC used Activity Stream as a tweet at 2010-06-01. But all of these social network used Activity Stream’s idea which is not yet standardize formatted. In this thesis, these
used Activity Stream is considered as atomic activities and offers high level Activity Stream with its standard format which offers syntactic and semantic relationship of atomic activities [17]. This section describes the update of the field of Activity Streams that was offered before.

To standardize the idea of Activity Stream’s structure, a certain amount of research already been conducted. In every case, the data or activity has been fetched from single or different independent websites or system which is represented based on different perspective after aggregation. At most cases, they generate streams from direct user’s activity and there is no consideration of high level semantics. In this chapter, the standard Activity Streams which we found in the current available web technology have been discussed with some of its application examples and therefore analyzing their advantages and disadvantages. Basically, most of the Activity Streams generating systems are available in the market, are kind of enterprise (social network) product. The internal architecture of these systems has been intentionally kept secret. That is why, only the overview and functional details of these systems has been described in this section. In addition, an internal architecture of a system regarding semantic web integration of heterogeneous social networks has also been described at the beginning of this section.

Basically, Activity Streams had been implemented in two ways which includes Atom syndication format and JSON based format. Its basic format deals with verbs and objects. Verb deals with what people have done for example join, post, like, mark as favorite etc. Object deals with what have been acted by any of the actor for example article, bookmark, person, comments, photo etc. As an example of such Activity Stream, suppose a person X made a comments on another picture. Then the Activity Stream would be 'Person X comments on Y’s photo named Z' and is broadcast to the network which everyone else can see. In the popular social network Facebook, such a real time Activity Stream example is depicted by the figure 3.13, Here, Arif Mahamood is a person

![Facebook's Activity Stream](image)

**Figure 3.13:** Facebook’s Activity Stream [18]

or account name who tagged another person’s photo. So, the Activity Streams for such an activity has been shown as in the figure. The source code of this Activity Stream is describing in the figure 3.14,
3.5.1 Semantic Model from Heterogeneous Social Networks

The semantic web is playing a great role for web applications to integrate structured and semi-structured data. Some research has been conducted in this area, although still there are some new challenges regarding integrating social networks with the advantage of semantic web. Basically, conventional aggregation and reconciliation methods lose the ability to recall entity because it works with each entity at one level separately. But the semantic method can work with heterogeneous of social data in both logical schema and instance values.

This system uses semantic web as a mediation layer to integrate heterogeneous social networks [19]. It is able to learn a Probabilistic Semantic Model (PSM) to implement data application analysis. PSM is used to extend a standard attribute based Bayesian network (directed acyclic graphical model) to resolve atomic and associated properties of semantic models.

Ontology-based social network integration and Data extraction

There are two components included in this proposed approach which are an integration method of social network and learning semantic data using a PSM to identify their relationship. The figure 3.15 demonstrates the semantic web architecture which describes how the system is linking and unifying online social websites. The architecture consists of three layers of components which include some online social websites as heterogeneous data sources, a semantic layer to describe about elements identity and dependency, and data analysis layer describes the connection and relationship between different elements. The resulting platform of semantic web integration of heterogeneous online social website provides a unified dataset for advanced data analysis application which is capable to do community discovery, relationship identification, instance classification, semantic association analysis etc. In this semantic platform, each entity and their relationship depends on both their logical schemas that is usually resolved by reconciliation method and entity’s instance values that is resolved by numeric methods. Notice that reconciliation
Figure 3.15: The semantic web integration of heterogeneous online social websites describes a method of addressing source conflict between interpersonal levels or between entities. The whole method deals with a hybrid pipeline system where combining and calculating both logical reconciliation and numeric methods in parallel.

In this system, FOAF (Friend of a friend) technology has been used to share and reuse the information of people and their activities around the network. Notice that FOAF is a simple technology to share the information between websites which is able to extend, merge and reuse data automatically. FOAF is used in the mediation layer to support heterogeneous social network data where descriptive properties and relationships of each social network are mapped into it. Based on those mapping, the social network data are converted to semantic data. Some special atomic properties or relationships of social network that FOAF technology does not have need to be extended for combining these descriptive and relationship properties together. There is an example of mapping description and relationship properties of LinkedIn.com and DBLP site to an extended FOAF ontology with couple of additional relationship properties showing in the following figure 3.16. In this example, social data of LinkedIn.com and DBLP are collected and stored in the XML file which are then supplied and extended to FOAF ontology that are transformed to RDF data with universal ontology schema.
3.5.2 Google Buzz

Google has a series of offering system for example, Google Wave, Google Buzz and Google Plus. The updated one is Google Plus which is comparatively new system for social networking offered by Google and it does not have that much technical details in the web so far. The creation of Google Buzz was intended as an attempt to compete with popular social networking website such as Facebook and micro blogging website Twitter. Google Buzz is a social networking and messaging tools offered by the company Google which is integrated with Gmail. It offers Activity Streams by using some existing format. User can share their activities for example links, photos, videos, status, comments etc. with their friends. These activities are linked together and showing to user using the existing idea of Activity Stream.

In Google Buzz, users can choose to share their post publicly to the world or privately with their group of friends. Some similar products for example, Youtube, Flickr, Picasa, FriendFeed, Google Reader, Google Sidewiki, Google Latitude etc. are integrated with Buzz [20]. In the figure 3.17, there is a screenshot of Google Buzz which describes users message posts, photo sharing and some real time Activity Streams automatically offered by Google Buzz.
3.5.2.1 Google Buzz Internal Concepts and Data Model

Data model and internal architectural concepts of Google Buzz are described in the paper [21]. Google Buzz implemented its primary structure based on activity. This activity based structure follow through an action which is performed by a person. For example, Khaled posted a message. Khaled is a person here who does an action (posted a message). When any message is posted, uploaded or updated any photos or videos is considered as activity in Google Buzz. The basic structure or components of activities in Google Buzz are following with Actor, Verb and Object. An activity of Google Buzz contains some more information called metadata which contains timestamp, location
of actor and update of feeds. Users can post their activities and even can comment or participate in conversations of other activities. Google Buzz uses resource which is an individual data entity with unique identifier. Its API has three types of resources which includes activity resource (represents an activity), comment resource (represents a comment), and person resource (represents a Buzz user). The Google Buzz API uses data model that is based on groups of resources and therefore they named it as collections. In Google Buzz, each user has a number of activity collections which may include his all other connected users public and authorized private activities. It could also include all the activities he might be interested in. Then the algorithm of Google Buzz determines the posts and activities which has a match (similar) with these interest collection lists. Each user has its own comments and person collection list. Comments collection list works as for each individual activity resource collects all comments on this status and person collection which deals the list of people who are following that user.

Google Buzz has been implemented using different types of data format such as JSON (JavaScript Object Notation) which provides simple method of representing arbitrary data structures, Atom syndication format which is an XML based language used for web feeds and Activity Streams which is an extension of Atom feed format to express peoples interaction around the web. JSON is a text based format which is completely language independent and uses some C-family convention which includes the familiar languages for example, C, C++, C#, Java, JavaScript, Perl, Python etc.

3.5.2.2 Discussion

Google Buzz taken a beginning step in the field of generating Activity Stream. But it works with the first phase of generating Activity Stream which is still atomic stream. There is no advance semantic method used here to detect relationship among objects or some other elements. The method used by Google Buzz cannot solve the problem of data silos or data overload in a big system. Information such as any user activity or status are still atomic and there is no relationship detection among them. By representing this way, if any user is interested to find any past activity which has been conducted earlier is time consuming and even hard to find out. They did not describe sufficient description about their data aggregation system. Google Buzz used existing format structure of Activity Stream which is not yet standardized to high level Activity Streams. So, they cannot offer high level Activity Streams. There is no description included in Google Buzz documentation if it can work with heterogeneous data sources.
3.5.3 MySpace

MySpace is a social networking platform [22] developed at 2002 by Friendster which offers the facility to generating Activity Stream. Data can be retrieved by MySpace special real time stream API and it gradually views status and updates of user’s data in the MySpace user interface. Currently, MySpace has about 110 million users worldwide and it handles around 46 million of status updates every day. The functionality it offers include post messages, upload or update photos and videos, add a song to one’s profile and generating Activity Streams by analyzing and looking through users update [23].

In the figure 3.18, there is a screenshot of MySpace which shows some user status and generated Activity Streams from their activities. The API of MySpace is described in [24]. MySpace is developed with real time stream API which includes a rich stream of events. It uses ActivityStrea.ms atom extension with contextual information which is delivered via portable contacts. Notice that portable contacts shares information with the goal to provide a secure way to access their address books and friend lists [25]. MySpace API supports subscription query language which let users to specify multiple topics, for example, at the time of open MySpace, each application can save around 20 queries. It gives a way to separate private and public channels for the user with privacy.
MySpace API allows transport layer to operate retries on failure and batching. MySpace API can work with sample rate, target location and able to filter topics by objects type. It uses PUSH technology to publish data in the MySpace interface which includes PubSubHubbub’s hub-to-subscriber and industry-standard HTTP authentication.

Each Atom entries for each activity can be retrieved by the response of MySpace API call. The activity of MySpace is also following the format with Subject, Verb and Object. A variation of verb and object types is defined in the MySpace specification. Activities are coming to the system as activity feed. It tries to provide enough metadata with the feed for allowing MySpace activities to be consumed by any typical aggregator. The activity subscription flow of MySpace works as an application external to MySpace which invokes the subscription endpoint to create an Activity Stream subscription system. The subscription system has some parameters which includes filtering criteria and endpoint URL of stream recipient partner. Activities are triggered based on filter which depends on user subscription, rate limiting and batched. When the system find any trigger for activities or stream then it push to the partner endpoint for application consumption. The figure 3.19 describes the subscription flow of MySpace.

![Activity Subscription Flow of MySpace](image)

Figure 3.19: Activity subscription flow of MySpace [26]

Personal activities of MySpace are handling by getting updates for every single user. The activities of personal user are managed by considering unique ids for all activities, localized metadata, Coordinated Universal Time (UTC) timestamps and giving option to user for choosing 'composite' an atom entry which represents a single activity where partner can do their own aggregation. The activities are showing to MySpace interface by date and time in the descending order. Activity Stream application of MySpace can be Multicast Dissemination Protocol (MDP) applications or MySpaceID applications which has increasing of activities. Basically, these applications don’t have much semantics used and they can work such as, Here Issam Barhoum is a user who added an application with his profile. So, the system automatically realizes and generates this atomic Stream. For generating
this Activity Stream, this special activity should be requested by passing activity types in the string parameter. This system need a parser to decode if any contents of activities are html based. MySpace developed with the legal rights of JSON and XML.

Discussion

At the time of deploying Activity Stream in MySpace, it faced some troubles. Most of the cases of generating Activity Stream, the performance of filter was not good enough that it messed up with one stream to another which creates a lack of functionality perfection and was a cause of user dissatisfaction. The standard of MySpace do not have a certain standard which only describes about people’s gathering and makes a connection between their activities. It does not use the important benefit of semantic technology. They used existing format of Activity Stream and there is no relationship described between object and other elements in MySpace implementation. The Activity Stream it offers is still atomic stream and there is no high level Activity Stream standard. So, the problem of data silos or data overload are still existing in this platform. There is no description about heterogeneous data sources in MySpace.

3.5.4 Gnip

Gnip is a social media aggregation system which provides a platform as integration point of dozens of social media websites (Facebook, MySpace, Twitter etc.). It retrieves information via API and provides either original or Activity Stream format to the user. Data fetches by Gnip using polling based and gives back to the user as push based data. As a social media aggregation system Gnip was the first standard system which can work like grand central station for the social web. It collects information from different independent system, normalizes them with its own way and then delivers to the user based on their specification. By applying some Gnip offered methods, users can choose specific data that they care about and then Gnip box is responsible for fetching these data by polling endpoint such as PubSubHubbub (PSHB) sources, HTTP streaming sources etc.[27]. There is a screen shot taken in figure 3.20 for a general overview of Gnip interface and its operation. The data model of Gnip is followed by content social websites which are publishing their contents and Gnip retrieves these data, other social networking sites can then choose to either poll Gnip or receive updates when there is any match with their previously selected rules (i.e. notify us if you get comments in Facebook). More detailed description is depicted in the following figure 3.21. The benefit of current social networking is that they are using more or less same format to publish
their contents. So, it has become easy to collect data by API call and publish these data any where you need. If a data collected from any social network is not similar format then Gnip normalizes them into a common format. And after filtering, Gnip pushes some of these data to the user directly which matches with their selected rules and keeps other data into bucket called 'activity bucket' for future. Activity buckets are kind of container where activities can be saved based on fixed time duration. Web applications can poll these data in every minute if they need [28]. Gnip can deliver data to the system in both original formats (the source format from where Gnip fetches data) and Activity Streams format (which offers normalized form of activities). The data flow of Gnip is showing in the following figure 3.22. It shows the collected data from different social networking system of the web with different format. Then data ingested in Gnip system and after normalization it delivers to the user applications. A Gnip box is a virtual server which is running on Gnip platform and work as internet data collection device. Gnip uses a data collector to collect data from source system to Gnip box. Each data collection has its own API for endpoints to manage rules and received data. Webhooks, HTTP, PSHB and standard polling method used to deliver the data to the user applications. Gnip offers an event driven system which will notify users if there is any new content or update. The supported data format of Gnip includes XML, JSON,
SOAP and Plain text. And the delivery protocol Gnip can work with including HTTP GET (Polling), HTTP POST (Webhooks), Streaming HTTP (Comet), PubSubHubbub, XMPP, SUP etc [27].
Discussion

GNIP is designed for collaborating several social networks as a group where it will be useful for saving smart amount of time. It simplifies the process by getting API based data and normalizing this entire API’s data into one single stream. Notice that this normalization means generating Activity Streams from couple of streams. It aggregates the information from heterogeneous data sources and aggregation process is standard. But the Activity Stream it makes is still not high level Activity Streams. The weak part of Gnip is that there was no consideration in the design of Gnip about data overload or data silos problem. So, it is quite obvious that after running Gnip, within an hour this platform may have huge amount of data from where finding any particular type of data would be hard.

3.5.5 Gowalla

Gowalla, a location based social networking website which lets user share their experience of visiting different places or organizations within their friends and community. They can post their experiences by posting, comments, sharing photos, sharing videos etc. It generates Activity Streams based on its users activity which is basically focused on the visiting and experiences sharing of different places. When a person need to visit a new place he can go through the Gowalla and query about people experiences in that place. It even helps people to find the location of any renowned place. Gowalla can connect with Facebook, Twitter, by iPhone, Adroid, Blackberry, iPad etc. By using Gowalla API one can access Gowalla data which allows integration of different application and service using a platform [29]. In the following figure 3.23, there is a screen shot of Gowalla interface which shows automatically generated Activity Streams regarding visiting for a particular place Halkyon.

The real time update of Gowalla system is structured by topics. This topic could be either spot or a user. User can subscribe real time updates feed by following the link offered with every topics. Rather than polling activity feeds via API or Atom feeds, XMPP or PubSubHubbub is checking by applications for real time notification of activities. Real time feeds are structured by Atom entries which are marked up with Activity Stream, portable contacts and GeoRSS\(^4\) data.

Gowalla system API is developed with methods JSON responses, served as application/json. Atom and HTML format can be used for additional records in Gowalla. Gowalla uses Representation State Transfer (REST) protocol which is consisting of clients and

\(^4\)GeoRSS is an emerging standard for encoding location as part of a Web feed. In GeoRSS, location content consists of geographical points, lines, and polygons of interest and related feature descriptions.
server system. REST is a style of software architecture for distributed hypermedia sys-

tems. For getting activity update from the Gowalla service client initiate requests to

server by using REST protocol, server process that request and then return matching

responses [31].

Discussion

If someone wants to visit a place like a shop or restaurant, Gowalla can help him to
locate this place and can show him peoples idea about the place by Activity Streams
which is made by aggregating people’s comments. You can see how popular this place
and how peoples are interacting there. Basically, it is designed for mobile devices but
the idea and API can also work for desktop application and devices. But in technical
point of view it’s still kind of aggregating of newsfeeds. And it works with a basic part
of Activity Streams. User can view the interaction feeds about one place separately.
But it neither offers any high level Activity Streams nor work with semantic web. There
is no syntactic or semantic description included in Gowalla description [32]. It does not
work with different data sources. Using Gowalla data overload is not obvious because
feeds are always separated by place.
3.5.6 Socialcast

Socialcast is an Activity Stream generating engine for large enterprise company area which intention is to centralize the communication between people of a community and become a helping medium to emphasize company infrastructure education. It is deployed either on-premise or on-demand accelerate information awareness in between employee using Activity Streams [31]. Information awareness deals with sharing information between employees as soon as any new information arrives or updates. In the figure 3.24, there is a screenshot of Socialcast interface which describes status update concerning peoples works and activity. Some motivation of automatically generated Activity Streams from user activities are also depicted in this figure. User generated messages are coming to the Socialcast from different independent system via API call and connects together with existing business system in Socialcast platform. For example, in a company social feedback loop to SharePoint, CRM, intranets and may be with some other systems existed in the area. Employees of the company are collaborating around this loop of the systems and get real-time communication by Socialcast platform.
while they can analyze and measure about productivity. Socialcast generating Activity Streams by using existed standard of Activity Stream format. In the figure 3.25, there is a logical engine description of Socialcast Activity Stream where there are couple of systems (CRM, ERP, logistics, Warehouse etc.) are existed in the company and created a loop by operation and dependency. They need internal system collaboration which may call integration to a central platform. Socialcast Activity Stream engine took this role of integrated central platform and generating Activity Streams from the information or messages generated by the employees. The Socialcast API is a REST API for interacting with messages, comments and like abilities. It supports BASIC AUTH authentication and SSL connections. By logging into the Socialcast API, user gets an authentication to get access to the list of members of the communities. The request types of this API includes HTTP GET, POST, PUT and DELETE functionalities. It supports JSON and XML format for the stream messages. Supported development platform of Socialcast API includes Bash, Ruby, Perl, PHP, Java etc [32].

Discussion

Socialcast can work with large enterprise business network. They did not put enough information for Socialcast internal architecture. The good thing is that it is able to provide Activity Streams with existing available format structure. But it is more or less like aggregating the feeds from different social site and views them to the user graphically by putting a good ranking. Then the output system would be like thousand of streams within an hour because they are updating frequently. That is why, it could be hard to follow and some people may miss any important updated stream which they really wanted to know. Socialcast does not support high level semantics for example making semantic relationship between streams. It’s more or less like aggregating data and makes Activity Streams which we call atomic streams. So, it’s still typical system which cannot
solve the problem of data overflow. But this system can work with heterogeneous data sources.

3.5.7 Final Discussion

For enterprise area some company offers Activity Streams so far, but still all of them are activities which are not reducing contents and not followable. The Activity Streams they are generating are atomic streams. Notice that user activities should be normalized by atomic activities using existing standard and format. And from these atomic activities, Activity Streams should be generated automatically using some syntactic and semantic rules which is able to isolate and reduce huge amount of data. It would comfortably solve the problem of data silos. Unfortunately, there is no high level Activity Stream generating system in the current trend of technology. High level Activity Stream’s format is also not standardized yet by any of those described systems. Some of the systems uses very basic part of semantic web but most of the cases they did not take the advantage of semantic relationship among of Activity Stream format (e.g. Subject, Verb, Object etc).

Basically, in the corporate company context, there is even no system offered in the market which would offer Activity Streams. Here we included a compare table of some existing systems which offers typical Activity Streams with number of comparable parameter. In Table 3.1, a list of parameter is addressed to have a look more clearly that which system offers which functionality and their developing intention. These features are compared here as whether or not the system has certain functionality such as Generate Activity Streams, Any Standard of Activity Stream Format, Atomic or High Level Activity Streams?, Generate Atomic Streams to High Level Activity Streams, Syntactic Model Description of Activity Stream, Intended to Corporate Company Context, Intended to Enterprise or Social Network Context, Heterogeneous Data Sources, Data Aggregation Standard, Data Loading Capacity: Scalability and Configurability. In this table these parameters are evaluated as Normal (if it is not enough matured), Good (if it is standard) and Very Good (if it is advanced). Some of the parameters are also labeled with Yes or No depend on its aspect.
Table 3.1: High level comparison of Generating Activity Streams

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate Activity Streams</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Any Standard of Activity Stream Format?</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Atomic or High Level Activity Streams?</td>
<td>Atomic</td>
<td>Atomic</td>
<td>Normalized</td>
<td>Atomic</td>
<td>Atomic</td>
</tr>
<tr>
<td>Generate Atomic Streams to High Level Activity Streams</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Syntactic Model Description of Activity Stream</td>
<td>Normal</td>
<td>Normal</td>
<td>Unknown</td>
<td>Normal</td>
<td>Unknown</td>
</tr>
<tr>
<td>Intended to Corporate Company Context</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Intended to Enterprise or Social Network Context</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Heterogeneous Data Sources</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Data Aggregation Standard</td>
<td>Good</td>
<td>Unknown</td>
<td>Normal</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Data Loading Capacity: Scalability</td>
<td>Good</td>
<td>Normal</td>
<td>Good</td>
<td>Normal</td>
<td>Good</td>
</tr>
<tr>
<td>Configurability</td>
<td>Good</td>
<td>Good</td>
<td>Normal</td>
<td>Good</td>
<td>Good</td>
</tr>
</tbody>
</table>
Chapter 4

Design

This chapter describes the intended design and architecture of the high level Activity Streams generating system. First of all, there will be a system overview of the whole system which is consisting of a graphical architecture and general overview of all responsible components in the overall proposed system. And then it describes every particular component’s design intension and operation in a regular fashion. In addition, the configuration languages of activity generation are also described in this chapter. The final part of this chapter regards the most interesting part of this thesis which describes high level Activity Stream generating idea and their description. For example, how to formulate Activity Streams, how to visualize, how we can get high level activities from atomic activities etc.

For describing the system more easily the whole system is named as CoASGen (Consolidation and Activity Streams Generator) system. CoASGen system is completely automatic high level Activity Streams generating system which simplifies its system architecture.

Actually, the design work is motivated by the requirement analysis, research goals and research questions. CoASGen system describes its own idea in most of the part; therefore it reused some idea from previous literature to achieve the requirements and goals efficiently. Basically, from the review of different literature and systems it is quite clear that there was no particular system before which offers high level Activity Streams with semantic format structure. And the main contribution of this thesis work is generating high level Activity Streams in a human friendly standard format which would direct users to the right way to their intended information. Next few chapters will describe the prototypical implementation of CoASGen and its evaluation in a systematic way.
4.1 System Overview

The intension of developing CoASGen system is to make an efficient system which is able to offer high level Activity Streams from events or activities occurring in a number of different independent systems. For fulfilling this intension CoASGen system separated its responsibilities to different active components. The overall system architecture of CoASGen system is shown in the figure 4.1 where the components from the left hand sight describes a number of different Independent Systems. These Independent Systems are providing web feeds with different heterogeneous data format for example, RSS 2.0, RSS 1.0 and Atom. The next component of Independent systems is Mediator which is responsible for transform feed item’s data in a standard general format. Basically, it retrieves data from different Independent Systems with different format and makes a standard and general XML format for further processing. This Mediator makes the system flexible for existing different kind of web feed format. The next component of CoASGen system which is processing right after the Mediator is Consolidator. This component is responsible for processing and isolating feed items (e.g. XML feed) into different types. It aggregates the feed items (which are received from different independent source systems) and isolates them into different types such as all similar kinds of feed items are placed into each one type. Consolidator is the most important part of CoASGen system. Another important part of CoASGen system is Activity Stream Generator which is placed after the Consolidator in the diagram. Activity Stream Generator is responsible for generating high level Activity Streams from a list of feed items which are already isolated by type in Consolidator. It generates one high level Activity Stream from each one isolated type (may contain approx. 100 feed items). Activity Stream Generator also offers structured standard format of high level Activity Stream.

![Figure 4.1: The global view of the CoASGen's overall system](image)
with graphical user interface view. The final and the top component of the diagram is Graphical User Interface where the high level Activity Stream should be viewed to the users.

In a global view, CoASGen has major three components to perform its operation: Mediator, Consolidator and Activity Stream Generator. And these components main functionalities are,

**Mediator**

- Retrieve web feeds from different heterogeneous Independent Systems.
- Transfer different web feeds to one general XML feed.
- Store XML feed into the database.

**Consolidator**

- Aggregate feeds (generalized XML feeds) into one place.
- Isolate feed items into a number of different types. There are number of Isolator in the Consolidator where each Isolator uses some methods to isolate feed items into different smaller types. Each types should have a number of similar kinds of feed items as a result.
- Store isolated feed items into the database in an organized way.

**Activity Streams Generator**

- Generate high level Activity Streams from a number of isolated feed items.
- Update metaActivityStream of high level Activity Streams from frequently incoming new feed items.
- Validate high level Activity Streams by both structure based and domain based.
- Store high level Activity Streams into the database persistently.
- Offer an user friendly view of high level Activity Streams.
- Standardize some high level Activity Streams format with semantic structure.

The following sections of this chapter will describe more details about these components. The developing methodology and the idea behind every part also describes in the following sections.
4.2 Mediator

In the CoASGen system, Mediator is working for generalizing different heterogeneous web feed to XML feed. Mediator is working as feed admission handler in the system and feed provider of Consolidator. Its responsibility is to retrieve information from different heterogeneous independent source system, change its feed format to a standard XML format and then store these generalized feed items into the database which would be the source feed of Aggregator in Consolidator. The basic architecture and functionality of Mediator is shown in the figure 4.2, where Independent Systems used in the left sight of the architecture to describe about the possibility of working with heterogeneous independent source systems in the CoASGen system. It can work with a number of Independent source systems. The internal architecture of Mediator includes Format Syndicator and Feed Retriever which retrieves heterogeneous feeds found from different Independent Systems. Heterogeneous feeds (e.g. RSS 2.0, RSS 1.0, Atom etc.) are based on available web feeds in the web. Format Syndicator works for processing available format of web feeds and generating a general XML feed format. These internal components are described in the following sections more details.

4.2.1 Heterogeneous Feeds

Currently, web feeds are available in almost all systems. The importance of web feeds is increasing because of its organized structure. It can be used easily in any intelligence based processing, basically in the high level semantics calculation and decision making. And web feeds are comparatively easy to retrieve from another individual system using software API call. But in the broader web, all system is not offering same web feed format. There are some versions of RSS web feeds and Atom feeds in different systems. Basically RSS 2.0 is updated format of RSS family that is widely available in the web. But RSS 1.0 and Atom is also using in some systems. In chapter 3, different feed formats have been described in more details. CoASGen system can deal with all available web
feeds in the web. It provides standard flexibility to the user to use any existing format for further processing. Basically, it transforms these different formats to its own structured format in the phase of Mediator.

### 4.2.2 Format Syndicator

In CoASGen system, Format Syndicator is the main components of Mediator. It provides the most important role to generating general XML format. Format Syndicator is a kind of parser who is responsible for generating XML format from heterogeneous feed format. From a particular system it receives feed items, process it to expected format and therefore it is responsible for store it to the particular responsible table. Then these data items are become ready to the next phase for processing. Format Syndicator use to store data as in one particular table for one individual Independent System. It works for generating XML format and in parallel it behaves as logical information bus for each and every particular system.

Format Syndicator retrieves data using software API call from the different heterogeneous system as a real time update. After receiving, it parse the data feeds into XML format and instantly stores that data feed into the database. It has a significant good effect: consolidator does not have to wait for API call’s output which might be cause of system hangs out. So, instead of waiting, Consolidator of CoASGen system retrieves data feeds from a fixed source table of a database for any particular independent system. This idea makes the CoASGen system more flexible.

### 4.3 Consolidator

Currently, there are some Aggregators available in the market. Most of them are developed based on different system’s special aspects. More details of different Aggregator have been described chapter 3. CoASGen has its own Consolidator based on its special aspects. Consolidator is one of the most important components of CoASGen system which is responsible for aggregating the feed items from different source tables and isolate them into different type of the data feed based on their similarity and aspects. Consolidator is using in this system to organize the big amount of feed items in a sophisticated way. So that the next phase ‘Activity Streams Generator’ can find the expected isolated feed items easily. The basic architecture of Consolidator is shown in figure 4.3, where Consolidator retrieves generalized (XML form) feed items from different source system tables of database. It aggregates feed items them and isolates or separates feed items into a number of different types based on feed’s similarity. And finally, it stores
isolated feed items into the database with different type tables. The internal architecture of Consolidator has two major components named as Aggregator and Type separator. Aggregator is responsible for aggregating data which already retrieved from different independent systems. Type separator is isolating the aggregated feed items into different types using its different level of Isolator. The idea of aggregating in Aggregator, isolating in Type Separator, Activity Generation and activity updating in the Consolidator is described in the following section with more details.

4.3.1 Aggregator

In CoASGen system, Aggregator is a fundamental component of Consolidator. It retrieves lot of activities from different independent systems and aggregates them into one place which would be proper input for Type Separator. Actually Aggregator takes certain fixed amount of data from all source tables, aggregates them and therefore makes a batch which is the input for Type Separator. One batch is fixed input for one time execution of Type Separator. If any source table do not have enough feed items then it is possible to take fewer feed items from there. But it restricts upper the limit. Actually, it takes that fixed input at the first time of making high level Activity Streams. And then it works as real time update which means getting each particular feed items from source system, process and store it to the source table on real time, then after consolidation process store it to the type table also on real time, and finally in Activity Stream Generator CoASGen set some time frame to generate high level Activity Stream and showing to the user interface efficiently. Normally a running system would update real time feed items which have really less overload because of frequently processing. Because of having very frequent update, only social network system could have overload.

Figure 4.3: The internal architecture of Consolidator
which can be controlled by extending batch limit. For corporate system environment, it
does not have that much flow of input feed items. So, overflow or overload is not any big
headache here. Therefore, CoASGen system care about flow limit considering uncertain
situation.

4.3.2 Type Separator

Type Separator is playing an essentially important role in Consolidator which has five
different Isolator and each of them has separate method and techniques to find out feed
item’s similarity. Some of the Isolators are easily configurable by both system admin-
istrator and end user. Type Separator retrieves data from Aggregator and providing
a number of isolated types of feed item into the database which is the input source of
Activity Stream Generator. It is very important to isolate the feed items efficiently by
considering different parameter of feed items because the efficiency of final output of
CoASGen system (high level Activity Stream generating) is depends on the output of
Type Separators in Consolidator. The five Isolators of Type Separator follows some
method and techniques. There is a short description of these idea and methods are in
the following section. Type Separator receives feed items from Aggregator as batch by
batch (e.g. in each batch 50 feed items). Each batch is processing and next batch comes
after one batch processing completion. In general, each of the feed items in a batch
comes with number of parameters (e.g. Feed id, Author name, Title, Text description,
Published time etc.). For taking decision if they are similar or not, Type Separator of
CoASGen system takes care by comparing and considering some of these feed item’s pa-
rameters in each Isolator. The flow chart in the figure 4.4 describes the detailed steps of
Type Comparator in a graphical way. Since the CoASGen system is a automatic system
and working with real time incoming feeds, once Consolidator starts it frequently checks
about new incoming feed update and if there is any feed item update then it does these
steps in the following of Type Separator otherwise it keeps checking.

4.3.2.1 TitlePattern Comparator

First of all, its checks if any similar title named feed item exist in the database with the
incoming new feed item’s title. If yes then insert that feed item into that type (same
type with the matched feed item to the database) into the database. The idea behind
this Isolator is sometime peoples reply messages without changing its subject (which is
title here). But they are talking about the similar stubs or informations. This Isolator
also applies pattern matching for checking feed items titles similarity. After calculation
of first Isolator if proceed rest of the feed items to the next Isolator.
4.3.2.2 **AuthorNameTime Comparator**

This Isolator checks if there is two feed items which has same author and the published time duration is less than two hours. The idea behind this Isolator is that sometimes users sending messages through chatting or discussing similar information by very frequent mail. So, these frequent feed items with same author name contains similar information. After completion of its execution, this Isolator sends rest of the undecided feed items to the next Isolator of Consolidator.

4.3.2.3 **RegularExpression Comparator**

RegularExpression Comparator is an advanced level Isolator used in CoASGen system. This Isolator considers a number of self generated and already existed parameters (e.g. Author name, Time etc.) of feed items to compare feed item’s similarity with each other. Self generated parameters for example, Domain based technical keyword or pattern (e.g. bug#1045), Category, URL etc. (can be found by using regular expression from the
title and text field of feed items). By setting some preference value (e.g. category match = 5, time duration close = 2, author name match = 1), RegularExpression Comparator calculates (by comparing the data of these parameters) one feed items with others by a final integer value (by adding up all of their matching result). And then by considering this feed items, Isolator checks other feed items if they exceeds a fixed range called top value (configured by user e.g. 30). If any feed item exceeds top value then RegularExpression Comparator decides both of them are similar and keep them in one type. This way this Isolator can find lots of similar feed items easily. Then, it stores them into the database. If user or administrator set less value in top value configuration, then the system will generate coarse grained isolated type. This Isolator is a bit rich and efficient in practice. After completing its calculation with one batch of feed items, it sends rest of the feed item to the next Isolator.

4.3.2.4 Frequency Comparator

It concerns about frequency calculation of feed item. This level takes incoming feed item’s title and text description in a list array and keeps its noun and verbs in another list array for comparing with existing feed items in the database. Noun and verbs are taken into consideration because they are the meaningful words in any sentences. Notice that finding nouns and verbs of a sentence are done by special parts of speech tagger. For calculating frequency matching, nouns and verbs found in title should be treated as twice importance than feed item’s description. Comparing with the incoming feed items if Isolator found more than a certain percentage (this is user configurable e.g. 70%) of matching with any feed items into the database, then it keeps that incoming feed item as same type of that compared feed items into the database. Those feed items has matches less than that certain percentage, Frequency Comparator forwards these incoming feed items to the next Isolator.

4.3.2.5 Subject Decider

It decides feed items title’s subject by following a configuration language. And then compare with the table name if there is any table name in the database with this same subject name. Notice that some table’s are generated in this system based on feed items title’s subject name by using Hash map method. If it matches that means this table’s feed items has same subject and they are similar feed items. So, insert this feed item into that table because one table is holding one type of feed items and they are similar. If it does not find same subject named table in the database then create a new table with the name of this incoming feed item’s subject’s Hash map value.
Chapter 4. Design

Configuration language works as follows, at first take all the keyword of the title of that feed item. Then check these words if one of them can be found in predefined technical keywords. If yes then keep it as subject otherwise check with domain based keywords (predefined by administrator based on environment). If it can be found there then keep it as subject otherwise find parts of speech of these words (using parts of speech tagger) and take first noun as subject.

4.3.3 Activity Generation

Activities are the precise representation of feed items description. It gives short overview about the detailed data description of every feed items. Activities are generating automatically in CoASGen system using some configuration languages. For generating each activity, CoASGen system finds out three different parts which are aggregated together and forms an activity. In the following figure 4.5, there is a syntactic diagram of different parts of an activity is shown with their role. The part 'Author name' is an actor of activity who perform some work. This is also subject of activity. CoASGen systems uses a configuration language for finding out 'Author Name' (describes in the following section). 'Technical keywords' is a object of activity which describes with some technical keywords of a feed item’s description. It shows the basic information of feed items. For finding out 'Technical keywords’, there is a domain based configuration language (will be described in the following section). The last part of activity is 'Published Time' which can be found as a parameter in any feed item description. This is a weak entity. By aggregating these three parts each activities are forming and finally these atomic activities are representing in streams.

![Diagram](image.png)

**Figure 4.5:** Configuration language of Activity generation with its different parts

4.3.3.1 Configuration Language

Configuration languages are used to detect some special fields of activities from each feed item’s description. Each feed item has two important parameters in its detailed
description which are title and text. The configuration languages using these two parameters to detect their expected parts of activity. In the following sections, there is a description about how these configuration languages works.

Configuration Language for Author Detection

Author name is detecting from the title and text parameter of feed item description using a Configuration Language. For generating every activity, Configuration Language has to find out right author name from each feed descriptions. Basically, most of the feed item has a parameter called ‘authorname’ which has a mail address (from whom it comes from). But it does not describes the real author name. Some feed item offers their author name in the text and some others do not. Those feed items offering Author name in the text, Configuration Language keeping them (author name) using regular expression. And for those do not have direct name in the text description, in this case Configuration Language finds any renowned web address if it has in its text description. It become the author name of the generating activity. Configuration Language finds it (renowned web address) using regular expression general web address pattern (e.g. CNN.com, monster.com, dvb.de etc). Even if configuration language can not find renowned mail address, then it has to rely on the mail address given in feed items (came with the sender’s mail address of feed item). This sender mail address then become author name in the activity. So, its obvious that Configuration Language will find at least one author name for each feed items.

Configuration Language for Object Detection

Using domain based Configuration system Object part of activity is detecting from the title or text description of feed item. A set of domain based technical terms (technical keywords e.g. software, bug, commit etc.) are predefined (configured by administrator) in the system before deploying. The idea behind these technical term is, they depicts each feed information overview within very short description. Basically, system checks these technical terms with the fields of feed item (title and text description) as a keyword matching. Those keywords found as a match, this Configuration Language keeps them as list of words and finally merge them together as an Object. For selecting more and effective technical words in each feed item, there is a analysis system which checks about which part (title or text description) has found more technical terms. The part has more technical terms, is selected for Object creation. Finally, it creates Object from these selected technical parts by merging them.
4.4 Activity Streams Generator

Activity Streams Generator is one of the important and interesting part of the CoASGen system. It generates high level Activity Streams from a lot of isolated feed items which is found in different independent systems. It retrieves the data items from database which is already processed by Consolidator and then it generates high level Activity Streams with its own high level Activity Stream format and structure. And finally it saves these generated high level Activity Streams into the database which is going to view in the user interface. Actually, the main goal of designing all other components of CoASGen system was to make an efficient Activity Streams Generator. It offers a standard format of high Activity streams with syntactic and semantic format structures. The main task of this Activity Stream Generator is to generate high level Activity Streams from atomic activities which is basically precise version of data feeds. High level Activity Stream is a kind of representative of aggregated several similar activities or events that happening in different independent systems. The standard format and structure of high level Activity Streams is described in the following sections more details.

![Figure 4.6: The internal architecture of Activity Streams Generator](image)

If we look inside of CoASGen system, activities are coming from different heterogeneous independent systems in the Mediator, after generalizing XML feeds it comes to the Consolidator where it aggregates and isolates based on similar type. Actually, these operations are done in these components intended to provide a graceful and nice environment to Activity Streams Generator. The basic architecture of Activity Stream Generator is shown in figure 4.6, where isolated feed items are the input of Activity Streams Generator (from database). Inside Activity Stream Generator there is an Activity Streams Manager who is responsible for generating high level Activity Streams with standard format. Then these high level Activity Streams are validated and stored into the database. Database is used here with one replicated table for viewing data gracefully. Other table is intended to update high level Activity Streams very frequently. The main important component of Activity Stream Generator is Activity Stream Manager which is described in next section with more details. In next few sections, Activity Stream Format and its view also described with its syntactic and semantic format structures.
4.4.1 Activity Streams Manager

Activity Streams Manager is the central component of Activity Stream Generator. The main responsibility of Activity Streams Manager is to generate high level Activity Streams with the format structures described in the following sections. Activity Streams Manager retrieves isolated feed items from database, generate several high level Activity Streams from each of these isolated type of data feeds and therefore store it into the database. Isolated feed items means these data feed already have been processed by Mediator, Consolidator and finally stored into the database with different type (each type has similar isolated feed items). One type table is going to be the source of one high level Activity Stream because each source table one type (isolated) of similar feed items. At the first time of making high level Activity Stream, it makes high level Activity Stream from the whole feed items of a table. But with the frequently update of activities in source system, this table should be update over time. It works like when he got any update of data at any point of time, then it recalculate and regenerate the high level Activity Stream. There is an interesting part in here. Activity Streams Manager works like, it keep the previous calculation of high level Activity Streams generation, add one more activity’s calculation and then update just some partial part of previously generated high level Activity Stream. That’s how it saves processing power. It does not have to recalculate all the existing feed items from that type table and make completely new high Activity Stream. However, it recalculates the whole type table after a certain time interval, generates totally new and updated high level Activity Stream and replaces it to the existed high level Activity Stream from the database. It makes the CoASGen system more reliable. After generating high level Activity Stream, frequently it updates the ASUpdate table (Activity Streams Update table). There is also a redundant ASDisplay table (Activity Stream Display table) which is not frequently updating.

It updates with certain time distance. This time can set for different system needs. For corporate organization architecture it fits with 30 seconds. Because of not frequently updating ASDisplay table, it can serve with nice feedback for user’s query with smart speed. This Activity Stream Manager follows all the format and structures that describes in the following sections. Activity Streams Generator also offers a standard user interface view of high level Activity Streams which also describes in the next following sections. In the internal architecture of Activity Stream generator, it has two internal Validator to check if generated high level Activity Stream is human understandable (with enough sentence structure). One of them are language based Validator and another one is domain based. In the following sections, there is a brief description of these Validator.
Figure 4.7: The flow diagram of Activity Stream generator

The flow diagram 4.7 of Activity Stream Generator describes more details about the operation and execution hierarchy in a graphical way.

4.4.1.1 Activity Stream Validator

The responsibility of Activity Stream Validator is to check the generated high level Activity Stream with high level Activity Stream semantic format structure if it follows enough sentence structure (using grammar). If it finds that for generating high level Activity Stream, some parts (e.g. verb or object) are missing, then it recommends other format structure which could be used for high level Activity Stream. Notice that there are three semantic format structure in Activity Stream generator. If it finds that the subject (a semantic part) of high level Activity Stream is missing then it recommends the Activity Stream Manager to call Domain based Validator. If Activity Stream Validator finds that the generated high level Activity Stream is well structured and with valid format then it save this high level Activity Stream into the database.
4.4.1.2 Domain based Validator

If Activity Stream Manager fails to detect subject for a high level Activity Stream generation then it calls Domain based Validator to find out domain based subject. Domain based Validator has preselected domain based keywords (should be configured by system administrator) which are matched with keywords of isolated feed items. The keyword it finds maximum times appeared in the feed item description (from all feed items in one isolated type) is become the subject for this isolated type. Notice that one high level Activity Stream is generated from one isolated type (where there could several feed items with similar information).

4.4.2 Activity Streams Format

Activity Stream Format of CoASGen system standardizes some high level Activity Stream standard. Some other social network also offered Activity Streams where they have only atomic activities as Activity Streams but CoASGen system offers high level of Activity Streams. In high level of Activity Stream, several isolated atomic activities are aggregated and forms a new standard of high Activity Stream which would motivate users to the flow of information regarding these atomic activities. For making standard high level Activity Streams format, CoASGen system also reuses some idea of existing non standard Activity Streams. It partially reuses syntactic structure of available structure. CoASGen system offered syntactic and semantic structure of high level Activity Stream is following. These semantic format structure is not yet very rich. But using more semantics, this semantic format could be very efficient. Notice that semantic structures are conceptual structure of lexical and syntactic expression in English [35].

4.4.2.1 High level Activity Stream General Format

High level Activity Stream General Format has its own semantic format structure to generate high level Activity Stream. Syntactically, it has three main components which are Subject, Be Verb, Main Verb/Object. These syntax are automatically electing by the system using some logical calculation. Subject is specially very important among these syntax because it carries main information of high level Activity Streams. There is a Subject selection procedure to elect Subject from isolated feed items where it uses a Configuration Language for Subject Selection (described in the following section). Be Verb and Main Verb are electing by finding a common be verb and common verb among the isolated feed items. And each of these syntactic part has some semantic structures. For example, Be Verb can be are, has, have was, were etc. It has number of options to
adapt and generate a human understandable high level Activity Stream. In the figure 4.8, a semantic diagram describes the basic architecture of high level Activity Stream General Format where each of the syntactic part has number of semantic entities.

### 4.4.2.2 High level Activity Stream Prefix Format

High level Activity Stream Prefix Format consists of two syntactic format which are Fixed Prefix and Subject. The syntactic part Fixed Prefix containing a fixed string and Subject contains the central issue of the isolated feed items. Both of the syntactic part is automatically electing by system where Subject is deciding by a Configuration Language for Subject Selection (described in the following section) and Fixed Prefix is electing based on which one can be match with Subject to form a human understandable sentence.

![Figure 4.8: The semantic diagram of high level Activity Stream general model format](image1)

### 4.4.2.3 High level Activity Stream Postfix Format

The semantic architecture of high level Activity Stream Postfix Format has three syntactic parts. Subject, Verb/Be Verb and Fixed Postfix are electing by system automatically using some logical calculation. Subject is electing using a Configuration Language for Subject Selection where it uses number of parameter to detect right Subject. Second syntactic part Verb/Be Verb is electing by finding a common be verb or verb among

![Figure 4.9: The semantic diagram of high level Activity Stream prefix model format](image2)

Figure 4.9, describes the basic semantic architecture of high level Activity Stream Prefix Format where each of the syntactic part has number of semantic entities to formulate high level Activity Stream.
the isolated feed items. And Fixed Postfix has some fixed string which is used to add in a last part of high level Activity Stream. This high level Activity Stream format can generate high level Activity Stream in two way. If if can finds out three syntactic part, then generate with them otherwise it can generate high level Activity Stream with only two syntactic part (Subject and Fixed Postfix).

![Figure 4.10: The semantic diagram of high level Activity Stream postfix model format](image)

In the figure 4.10, there is semantic diagram of high level Activity Stream Postfix Format where each of the syntactic part has number of parameters to formulate human understandable high level Activity Stream (by maintaining sentence structure).

#### 4.4.2.4 Subject Selection

Subject is one of important part in semantic architectures of high level Activity Stream because it depicts the short information of high level Activity Stream with a simple string. In the Activity Stream Generator, CoASGen system uses a special configuration language to detect Subject among the isolated feed items. In the following, there is a description of Subject selecting procedure.

**Configuration Language for Subject Selection**

Configuration Language has a number of selection parameters to find out which part would be the suitable to find a Subject for a high level Activity Stream. From these parameters, it has a priority hierarchy from where it tries to find out Subject. If it fails to find out Subject from one parameter then it finds in another parameter. The hierarchy of the parameters are following is as follows: category, domain based keyword, technical keyword, renowned word and noun. In the text description of some feed item has a parameter called Category which has a brief description about feed item’s information. If system finds a common category name in all isolated feed items, then it decides this category name as a Subject. If its fails to find out category then it finds out a domain based keyword which is common in all isolated feed items. Notice that technical keyword should be configured by system administrator depends on particular domain environment. Further if it fails to detect Subject from there, then it finds if any
common technical keywords from entire isolated feed items. If it further fails to detect Subject from there, then it finds any renowned word (e.g Company name, Software name etc.) which is common in all isolated feed items. Even if fails to detect Subject there, then it finds first noun of a feed items if it is common in all isolated feed items will be the subject. So, its pretty obvious that system will be able to find a Subject. Notice that this Subject is used as MetaActivityStream for every particular high level Activity Stream.

4.4.3 Activity Streams View

It is important that how high level Activity Streams will be shown in graphical user interface. CoASGen system offers a graphical view of high level Activity Stream’s screen. The user interface of high level Activity Streams is like typical user interface but it offers some special and nice context. This view offers lots of isolated feed items from very small high level Activity Stream overview. And it is very easy to follow this information within short time. Nobody is going to miss any important data that he needed to follow. The Activity Streams View has a metaActivityStream which is easy to follow and gives users a short motivation about the generated high level Activity Stream. In the graphical view of high level Activity Stream, a number of generated Activity Streams is ordered with the time and importance of the high level Activity Stream. Importance is figured out by the user’s collaboration for these feed item (which are the isolated feed items of any generated high level Activity Stream). And with the high level Activity Stream, there is a information all responsible activities by following a link. For example, this high level Activity Stream made from that number of feeds. If any user wants to have a look any high level Activity Stream, then by clicking it enlarges to full screen page with details of that high level Activity Stream.

4.4.3.1 MetaActivityStream

MetaActivityStreams are the metadata of high level Activity Stream which gives a easy access to the users to all currently updated high level Activity Streams. This is the first view of Activity Streams. It is updated frequently with the update of high level Activity Streams. Users can have a look a short overview (short description) of high level Activity Stream and follow their expected high level Activity Stream by clicking in the metaActivityStream. It gives meta description of high level Activity Stream and their generated time. MetaActivityStreams are generated by the subject of high level Activity Stream which depicts the short information about that high level Activity Stream. It shows the basic information about the subjects of that high level Activity Streams. For
example, if the subject is "Football" then it offers the general description of Football linked by all other important feed items regarding Football.

4.5 Real Time Stream Updates

CoASGen system used completely real time update from retrieving data from the different heterogeneous independent systems. Events or activity is just happens in a particular system and its feed items are coming to the system for processing on real time. However, "Activity Stream Generator" has a small time frame to view the data to the user interface gracefully. There CoASGen used two tables in the database for processing. Second one is replicated from the first one within a fixed duration distance. One table is for frequently updating high level Activity Streams based on frequent update of new activities in the source system. And another table is for viewing the data to user interface. The idea behind the two databases is that user interface can be ask for some data to the database very often, and if it is continuous updating table then it won’t be able to answer the query very frequently which would be cause of user interface slow or hanged. And it could a cause of user dissatisfaction. These real time update and efficient user query feedback makes the system more efficient.

4.6 Summery

This chapter summarizes CoASGen system’s design overview by graphical architecture and description. The Consolidator and Activity Stream Generator sections are the central part of this chapter. Five different Isolators are described in Consolidator with graphical architecture which plays important role to make Consolidator efficient. In the Activity Stream Generator section, three semantic format of high level Activity Streams are described with semantic architectures. This chapter describes the main contribution of this thesis.
Chapter 5

Implementation

CoASGen system is completely automated system with advanced intelligence and it can perform its lots of operation dynamically. We developed a prototype for internal communication of Software Company where there are different kinds of independent systems (SVN, Bug Tracker, Wikis) updating their feed items very frequently. This chapter explains detailed description about the implementation of CoASGen system. First of all, it describes the development language of CoASGen system and cause of choosing these languages. Secondly, this chapter describes the core implemented package architectures of whole CoASGen system. And then for each of these core packages main classes and their functionalities are described with required diagrams and source code examples. Third part briefly describes about the most important frameworks and APIs (Application Package Interface) which are used to develop this prototypical implementation. And finally it summarizes this chapter.

5.1 Development Languages and Platform

All the core components of CoASGen systems have been developed in Java 1.6.0.24\(^1\) and with Eclipse platform. This programming language is chosen to develop this prototype because Java is an object oriented high level programming language and its source code files are platform independent and compiled into bytecode format which can be execute in Java interpreter. And this compiled Java code is able to run most of the computer because almost all operating systems have Java virtual machine. So, it has very good portability. Java has a nice standard of data structures, class and methods which can be applied for many sophisticated applications including home applications. It’s easy to handle error with Java’s exception handling and its memory management is very

---

\(^1\)Java Runtime Environment 1.6.0.24 update Security news and updates.
efficiently managed by itself. Java has nice functionality coverage with adding external API library which provides very big functionality with comparatively small stress and time. For example, in this prototype a linguistic tool LingPipe\textsuperscript{2} is used for deciding parts of speech of a sentence and therefore it helps with lots of intelligence processing of the prototype. And also for reading XML file, this prototype used standard XML parser which simplifies the intension and workload in a sophisticated way. So, for doing research or developing different prototype Java took a smart position to apply and getting desirable output in a user friendly nice format. Eclipse platform is chosen to develop this prototype because it has an amazing editor and user friendly debugger. This platform is extensible and lots of plug-in are available in the Internet to adding new of functionality. In addition, we can even write plug-in if needed.

5.2 Architecture of CoASGen system

CoASGen is a system which is combination of several core packages and its extended functionalities. Figure 5.1 shows the all the core packages involved in this system and in next few sections, there are more detailed description about these packages. Some less important packages of this system are not focused here as they are helper and test packages.

\textsuperscript{2}LingPipe is a toolkit for processing text using computational linguistics.
5.2.1 Configuration Package

The system and generating high level Activity Streams are easily configurable; the company could configure it in a flexible way with a certain rule as their requirements. For example, the system could be in a level of strict, less strict or even independent based on user requirements. Configuration package is one of the important packages of CoASGen system which deals with configurability. It includes three inner packages to manage its operations. Figure 5.2 shows all the configuration packages of this prototypical system. 'Control' package is one of them which consists a controller class which controls the entry point of this configuration package. It also instantiates all needed classes in this core package. 'ReadFile' is another sub package of Configuration packages which has necessary classes to read XML Configuration file and therefore store these necessary parameters described in the Configuration file. Finally, the classes inside 'ApplyConfiguration' configure all necessary components of the whole CoASGen system by applying its configuration parameters in other core and their sub packages. For example, In 'ActivityStreamGenerator' core package, there is a sub package called 'SubActivityStreamGenerator' which has a parameter called 'Activity Stream time duration'. This parameter is responsible for instantiating the Activity Stream Generator’s recalling time in after every certain duration. It would be 30 seconds for example. Then this Activity Stream Generator machine will generate high level Activity Streams after every 30 seconds. There is an example of Configuration file with its parameters in figure 5.3. Basically, CoASGen system has chosen its five configuration parameters which include high level Activity Stream format types, Compare frequency range, high level Activity Stream generating time durations, Time duration after feed items delete from table and Locality order of user interface.

5.2.1.1 Percentage Level for the Isolator, Frequency Comparator

In the Configuration file, it shown as Compare frequency range. In Consolidator part of CoASGen system, for deciding similar feed item types, the system will automatically check frequency matching of incoming feed item with existing feed items into the
database. With the user decided fixed frequency percentage the system will take decision if incoming feed item has similarity with any feed item existed into the database and based on that it will decide whether to add that incoming feed item into any certain type of feed item or not. So, user can choose, for example 40 which would be considered as, if 40 percentage matching found then yes signal otherwise no for adding incoming feed item into any certain type of stream. This parameter gives users the facility that how strict calculation he needs to manage this CoASGen system. For example, if he or she chooses more compare frequency range then the system will be coarse grained for choosing similar type of feed items into a table. And if user chooses less compare frequency range then the system will fine grained.

### 5.2.1.2 Percentage Level for the Isolator, Regular Expression Comparator

For selecting this parameter in the configuration file, it is important to know what Regular Expression comparator does. It is an Isolator which takes domain based keyword pattern from feed item’s text using regex. And then compare with one feed item’s founded keyword to another feed item’s founded keyword. By setting them some point in matching (different part of regex match has different point, for example, if feed item’s category matches, it add ten points in overall calculated value). More detailed description in chapter 4. RETopValue is configurable parameter which is used to decide at which condition this Isolator will take decision that these feed items are similar. In the
figure 5.3, this parameter is set as 30. So, by considering one feed item, Regular Expression comparator will decide those feed items are similar if they have overall calculated value is 30 or more 30.

5.2.1.3 Activity Streams Generating Time Duration

In the Configuration file, third parameter is shown as high level Activity Stream generating time duration. After consolidation process done one time, there are lot of similar type of feed items are stored in a single table which are responsible for generating one Activity Stream. And this consolidation process is a continuous process. So, into this table there will be always updated streams. We need a policy to generate Activity Stream from continuously updated table. So this parameter works as, after certain duration of time, let say 30 seconds, CoASGen system will automatically regenerate Activity Streams from updated stream table. This parameter makes the system more flexible in a way that how recent users need updated data.

5.2.1.4 Different Format of Activity Streams

In the CoASGen system, there are three special formats of Activity Stream which describes as Activity Stream types in the Configuration file. User can choose any of them to generate Activity Streams. This parameter makes the system flexible based on particular users interest because user has a nice chance to choose the way of Activity Stream he or she likes.

5.2.1.5 Streams Life Duration in the Database

In the configuration file this parameter is shown as Time duration after Stream delete from table. Streams will be automatically deleted from all tables after a fixed time period. For example, if user chooses 30 days, then all streams older than 30 days will be deleted automatically. CoASGen system always keep track about all streams life time by querying them after a certain time periods.

5.2.1.6 Locality order of Activity Stream in User Interface

This is shown as Locality order of user interface in the configuration file. This parameter determines how Activity Streams will be presented on the user interface. There are two choices here which are based on either updated time or the importance of an Activity
Chapter 5. Implementation

Stream. CoASGen system gives user this facility to choose one of them by filling up the XML file.

5.2.2 Control Package

Control package is one of the core packages in the CoASGen system which is basically starting and instantiating point of the system. There is a controller class which basically instantiates graphical user interfaces and gives user the way to access of all the functionality of the system. The operation of this package is like door opener of the system and directing to any particular’s desired destination.

5.2.3 Graphical User Interface (GUI) Package

In CoASGen system GUI is developed with basic features because this is a prototypical implementation system and it’s not very important to have very well graphical user interface. The user interface showing Activity Streams after calculation and showing some information about generated Activity Streams. The GUI has been developed with JSP and Servlet which is deploying with Apache Tomcat to view all the needed components of Activity Streams Generator. In the figure 5.4 shows a screen print of the
5.2.4 Mediator Package

Mediator package is one of the core parts of CoASGen system which has some important role playing in its sub packages. In CoASGen system, Mediator is used for generalizing feed format. For example, RSS 1.0, RSS 2.0 or Atom feed will transformed to a general XML feed for calculation and consideration. The figure 5.5 shows a schematic diagram of Mediator packages that has been used in CoASGen system. For developing Mediator, CoASGen system used three different sub packages such as, 'RetrieveStream', 'FormatRecognition' and 'FormatSyndication'.

![Figure 5.5: The packages of the Mediator of CoASGen system](image)

5.2.4.1 Retrieve Stream sub Package

'RetrieveStream' has DataRetriever classes to read streams from different independent systems (i.e. Wiki, SVN, Bug Tracker etc.) using Application Programming Interface (API) call. And therefore without any changing anything of incoming stream it sends this stream to next package 'FormatRecognition'.

5.2.4.2 Format Recognition sub Package

After fetching streams the FormatRecognizer classes of 'FormatRecognition' sub package are responsible to determine the new incoming stream has which feed format. Basically, so far in web we have three feed format which are RSS 2.0, RSS 1.0 and Atom. There are some formats examples of all available feed in the Internet are shown in the following few screenshot pictures and therefore describes briefly how they are generalized in standard format. From the figure 5.6, RSS 2.0 is last updated version so far. This is comparatively more flexible and widely available in the web. FormatRecognizer determine its format easily and retrieves it’s all important parameter data. Therefore, send it to the next level to generalize its format.
RSS 1.0 shown in figure 5.7, is more widely used web feed format in the web. RSS 1.0 is a bit verbose than RSS 2.0 because it uses some markup that Resource Description Framework (RDF) processor need to process. So, without going too much detail, FormatRecognizer reads important parameter of this type of feed format, for example, feed title, feed description, feed link, feed time, author etc. and immediately send it to the next level of operation. Another format called Atom is shown in figure 5.8, is bit rich format and it’s not that much widely used in the web but it has nice structure. It has
metadata inside the structure of the feed (i.e. entry in the figure 5.8) and from there FormatRecognizer is able to retrieve needed information parameters for further next processing.

![Atom.xml](image1)

**Figure 5.8:** Screenshot of Atom feed

### 5.2.4.3 Format Syndication sub Package

Finally, stream mediation process end with the sub package ‘FormatSyndication’ with its class FormatGeneralizer which is responsible for changing different dynamic feed format to a general standard of feed format (i.e. XML). FormatGeneralizer receives all important parameters of a feed and therefore generate another new file of feed with its own defined format. It uses hash mapped name for the new generated file. In the following figure 5.9, there is an example of generalized web feed format which is generated by FormatGeneralizer to process this stream or feed in the next phase consolidator.

![Generalized.xml](image2)

**Figure 5.9:** Screenshot of generalized web feed
5.2.5 Consolidator Package

Consolidator is one of the core packages of CoASGen system which has one of the main important roles of the system. In CoASGen system, the basic operation of Consolidator is isolating streams based on their similarity or aspects. This is the most important task because the ultimate goal, generating Activity Stream is depends on this isolating efficiency.

The Consolidator of CoASGen system has been developed with three levels of filters which imply the system’s accuracy and efficiency. Different level of filter in Consolidator has been developed in different sub packages in Consolidator package. Figure 5.10 gives an overview of the important sub packages needed to develop Consolidator.

And in the following sections, there will be brief descriptions of these sup packages. The class diagram presented in figure 5.11 describes graphically in more details about the internal interaction of Consolidator.

5.2.5.1 Control Package

‘Control’ package is the entry point of Consolidator package which has a ControllerConsolidator class to instantiate other sub packages. There is a connection between core control package and this ‘Control’ package. When the system needs to get access in Consolidator then core control package establish connection by instantiating this ‘Control’ package which is responsible to get access in other sub packages and their functionality. In addition, it retrieves the data from persistent hash map key value from database at
the beginning of launching Consolidator. And at the end when operation of Consolidator is done, it saves the current state of hash map key value into the persistent database.

5.2.5.2 Retrieve Stream Package

‘RetrieveStream’ package consists of RetrieveFeeds class which is responsible to retrieve stream from database as a batch and then supply for next operation in Consolidator. Notice that streams are already saved into the database by Mediator with their generalized format. This package also has DriverConnection class which has lot of pre-formatted methods for establishing connection with different database in MySQL via MySQL JDBC connector.

5.2.5.3 Title Pattern Comparator Package

‘StreamTitleChecker’ package is one of the important packages and one of the filters of Consolidator from its three levels of filters. It consists some classes for example SameStreamChecker, SaveSimilarStream to check firstly, whether the machine got same stream which they already got before or not. Secondly, it checks if stream has same or very closely similar title but not exactly same contents. That means their topic has similarity somehow. It checks title using string pattern matching. If titles are matching more than or equal 95% then it considers these streams are similar. Then using class SaveSimilarStream, it saves this stream into the similar stream’s table of the database.
And those streams for which SameStreamChecker does not found similarity are sending to the next level of filters.

5.2.5.4 Author Name Time Comparator Package

'AuthorNameTimeComparator' package works for finding out if there are feed items which has a same author and publishing time difference is in between three hours. Then it decide that they are similar feed items and keep them in one isolated group. CheckAuthorNameTime class checking these author name and publishing time condition if it satisfies, and then StorgeDatabaseAndClearList class stores the condition satisfied feed items to the database. This is comparatively small package.

5.2.5.5 Regular Expression Comparator Package

This is one the important and big packages of this prototypical implementation. The detailed description of this Isolator can be found in chapter 4. RetrieveRegularExpression class receives feed items and for each feed item it calls CheckUsingRegularExpression class to find out all the domain based pattern in this feed item using regex. Then these found keywords are saved into the persistent database by calling SaveRegularExpression-DataClearList class. This class saves the data into the database and clears all the data from list array. This way it stores all feed items regex data into the database. Notice that there are around 30 regex developed for this Isolator. For example, the regex for finding 'bug no' is ("bugs#\[0-9\]+"). RegularExpressionCompare class then start calculation of all the regex saved data. It does the operation like that considering one feed item’s regex data from the database, compare with others and using a compareValue (int) it saves a matching point (it remembers which feed item is considered one and which one is compared one by their id). Then RegularExpressionDecision class take decision by comparing with a topValue (user configured); it saves those feed items into the database which has compareValue more than topValue, and keep those considered one and compared one in same isolated group. It saves these data using SaveFeed-ItemAfterRECompareCheckTime class. In addition, this class checks those feed items if they have published time duration within the user configured range. These whole operations are persistent and consistent.

5.2.5.6 Frequency Comparator Package

'FrequencyComparator' package deals with some classes which are used to develop second level of filters in Consolidator. And this level of filter plays very important roles
in the whole system. This package consists of FrequencyComparator, the AdvancedCheck, the FrequencyPercentage, the IncomingStreamDataModel and the SaveFrequencyMatchStream classes. FrequencyComparator is responsible for comparing incoming streams frequency with existing streams in the database. It reads the incoming stream’s title and description, then keep its nouns and verbs list into a ListArray. It also reads stream from database in the same way of incoming stream and therefore compare them one by one to find how similar this database streams in compare to incoming stream. Class AdvancedCheck is used to keep track of matched frequency while they are in compare. Based on frequency matching it calculates frequency matching percentage for every database stream considering one incoming stream. FrequencyPercentage is used to calculate this frequency matching percentage between two streams. IncomingStreamDataModel helps by keeping track with stream information while there in comparison and then it stores data for example stream table name, id, number of useful words, matched frequency percentage etc. for both sides of streams. Then it finds out which stream of database has maximum match with all incoming stream one by one. And at the end of this package using SaveFrequencyMatchStream, it saves those streams into a table of the database which has a match above a certain percentage (let say 70% match: user defined). Those incoming streams have below this certain percentage of match with all the database streams are going to send to the next level of filter.

5.2.5.7 Subject Decider Package

'SubjectDecider' package consists of SubjectDecider, the PartsOfSpeechDecider and the SaveSubjectDecidedStream classes. This package implements third level of filter in Consolidator. SubjectDecider decides stream’s subject with the help of PartsOfSpeechDecider. LingPipe, a linguistic tool used in PartsOfSpeechDecider as a external library to decide which word has which parts of speech. Considering a stream’s title in PartsOfSpeechDecider, if first noun is following with a be verb or verb then it considering as stream’s subject. Otherwise, first noun of the title is subject. LingPipe is very sophisticated tools which give the opportunity to access word’s tag, chunk very easily. And it’s also light weight in compare to its operation. By calling PartsOfSpeechDecider when SubjectDecider get stream’s subject, it checks in the database if this subject named table exist or not. Notice that tables are basically created only in the 'SubjectDecider' package with the persistent hash mapped name of stream’s subject. Subject is considered as a key of hash map and values are the name of table which is sequentially generated by a function called hashValueGenerator(). So, if SubjectDecider found this subject named table into the database that means the streams inside this table are similar of that subject decided stream. Then that subject decided stream can be inserted into that
table using class SaveSubjectDecidedStream because as the basic idea of these filter we are isolating similar type of stream in one table. And if SubjectDecider don’t get any table same named with that subject, then it generates one table with the value of hash map using this subject as hash map key.

### 5.2.5.8 Activity Generator Package

When isolation process is complete, then those isolated feed items can be transform to activities. This package is responsible for doing that stub. For generating activity, this package have to detect different syntactic part of a activity. ConfigCentralAuthor class is responsible for detecting author name for this generating activity. For technical details please visit chapter 4. ObjectSelectionConfigLanguage selects object for generating activity using some technical methods (detailed description in chapter 4). Finally, FeedItemToActivityGeneration class generate activity using those syntactic parts found from last two class and that feed items published time. And by calling SaveActivityIntoDatase class it saves generated activity into persistent database. So, user interface can query any time into the database for publishing activities (updated) to the user interface.

### 5.2.6 Activity Stream Generator Package

Activity Stream Generator is one of the core packages in CoASGen system which has a major responsibility to generate Activity Streams from isolated tables.

![Figure 5.12: The sub packages of the Activity Stream Generator](image-url)
This package plays the main goals of this prototypical implementation. It offers three different types of Activity Stream’s format. To manage these works in Activity Stream Generator package, it uses some sub packages for every particular operation. In the following figure 5.12 gives an overview of the important sub packages needed to develop Activity Stream Generator in practice. And after that in the following sections, there will be a brief description of every sub packages with their particular developing intention and functionalities.

![Class diagram of Activity Stream Generator](image)

**Figure 5.13: Class diagram of Activity Stream Generator**

Class diagram of Activity Stream Generator in figure 5.13 depicts more details interaction of different packages used in Activity Stream Generator implementation.

### 5.2.6.1 Control Package

'Control' package of Activity Stream Generator package plays as an entry point to generate Activity Stream. It has ControllerActivityStreamGenerator class to instantiating other sub packages in Activity Stream Generator package based on users need and aspects. This package retrieves configuration data from core 'Control' package and then after applying this configuration, then it finds user chosen Activity Stream format and by instantiating it gives control to the responsible package to generate Activity Streams.

### 5.2.6.2 Retrieve Stream Package

'RetrieveStream' package has couple of classes for example, DataRetriever and the DriverConnectionASGen for retrieving isolated stream from type separated table in the database. DataRetriever retrieving all streams from one table and keeps it temporarily in an ArrayList for further calculations and therefore generating Activity Streams from there. Notice that the source of one Activity Stream is all streams available in one isolated table in the Activity Stream database. DriverConnectionASGen handles database connection related issues before every transaction with the database. It supports all classes in all other packages of Activity Stream Generator package too.
5.2.6.3 Activity Stream Format Package

'ActivityStreamFormat' is one of the very important packages in the implementation of Activity Stream Generator. This package offers three types of Activity Stream’s format with its syntactic and semantic model. This package has classes includes ActivityStreamGeneralStructure, the ActivityStreamFixedPrefix, the ActivityStreamPostFix, the ActivityStreamGeneralStructureModel, the ActivityStreamFixedPrefixModel, the ActivityStreamPostFixModel which are playing a vital role to generate Activity Streams.

From the 'control' package it comes with user selected Activity Stream type or format. Using this type selection, this package selects how it will create model for generating Activity Streams by choosing class to execute. For example, if user selects ”Activity Streams General Structure” format to generate Activity Streams, then this package chooses ActivityStreamGeneralStructure which uses ActivityStreamGeneralStructureModel to select all the needed parts to generating Activity Streams. Then it sends this model to 'PartDecider' package for comparing and selecting different important part (i.e. subject, be verb, verb etc.) of Activity Stream.

5.2.6.4 Part Decider Package

The basic intention of developing this package is to select different common and important parts of Activity Stream from a set of streams by using some advanced logical condition and comparison. 'PartDecider’ package consists of StreamPartChecker, the SaveDataModel and the CompareAndSelectParts classes which have different development intension and functionality. StreamPartChecker is responsible to detect important part of all streams from one table. Using the format and model found from 'ActivityStreamFormat' package, it chooses different parts of every stream in a table. For example, using Activity Stream prefix format, we only need a subject which is renowned noun. Then it compares if every stream in a table has this word common. Suppose from a stream title, the machine found a renowned word ”Wiki”. Then it will find this word ”Wiki” in all other streams in one table, if it found then the machine can decide this word as final renowned word which would be use to generate Activity Stream.

So, for generating this Activity Stream we only need to find out renowned word and therefore compare if it also exist in most of the stream in a table. After finding out renowned word of every stream SaveDataModel class used to save its data in an ArrayList model with their all other stream information (e.g. table name, stream id etc.). Finally, CompareAndSelectParts class compares saved words from the data model and decide
the common one. Then it sends this data to the 'SubActivityStreamGenerator’ package to creating Activity Streams.

5.2.6.5 Sub Activity Stream Generator Package

'SubActivityStreamGenerator’ is the goal package where Activity Streams are generated using already selected format and model. 'SubActivityStreamGenerator’ package consists of ActivityStreamGeneralStructureGenerator, the ActivityStreamFixedPrefixGenerator and the ActivityStreamPostFixGenerator classes. From the control package this package retrieves user selected format and model, and based on that it execute responsible class to generate Activity Streams. For example, if user chooses ”Activity Stream Fixed Prefix” as a format of Activity Stream, then it will choose ActivityStreamFixedPrefixGenerator class to generate Activity Stream. And for instance, if this class gets PhpMyAdmin as a renowned word from 'PartDecider’ package, so the generated Activity Stream would be, 'Some update about PhpMyAdmin’. 'Some update about’ is fixed prefix here and ’PhpMyAdmin’ is renowned word (noun). Notice that, one Activity Stream is a representative of all streams in one isolated table. In practice, if a table doesn’t have enough amounts of streams to generate Activity Stream, then the machine cannot make efficient Activity Streams from there. So, this class will take care in this situation as an exception and it will select one stream from there for showing as Activity Stream.

5.2.6.6 Activity Stream Validator Package

'ActivityStreamValidator’ package consists of ActivityStreamValidator, and the SaveInDatabase classes. The main intension to develop this package was to validate generated Activity Streams with its semantic structure. ActivityStreamValidator is responsible to validate Activity Stream where are two possibilities. If generated Activity Stream is valid with all of its structure then save it the persistent database using SaveInDatabase class. Otherwise, regenerate this Activity Stream again by resending to the ’ActivityStreamGenerator’ package.

5.2.6.7 Domain based Validator Package

'DomainBasedValidator’ package validates if any semantic parts (specially Subject) are missing in 'ActivityStreamGenerator’ package. SemanticPartChecker class checks if Subject is missing in 'ActivityStreamGenerator’ package. If yes then it calls DomainBasedSubjectFinder class to find a reasonable Subject for generating high level Activity
Stream. DomainBasedSubjectFinder class calls a DomainBasedKeywordFinder class to find a domain based keyword. Notice that this class is already configured for Software Company environment with the technical keywords. This class finds a common keyword from the isolated feed items which can be match with these configured domain based keywords. If it found then this class return this keyword to 'ActivityStreamGenerator' package as a Subject of one high level Activity Stream. 'ActivityStreamGenerator' package then generate high level Activity Stream with that Subject.

5.2.7 Persistence Package

Persistent package is basically used for managing data and writing data to the disk in a save manner where there is no possibility to loss it. Most of the classes of this package are used to keep a redundant copy of databases including hash map table databases. The responsibility of this package is to make the system persistent and secure. In the Consolidator core package, there are two issues that handle with persistent memory. First, saving isolated feed items into the persistent database and secondly, saving the hash mapped key after completing operation. Notice that hash map key values are saving in the database because the values of hash map are generating dynamically based on its key but it follows an order. And every time system starts, it initializes the updated order of hash map value, and then it becomes easy to follow next order. Hash map key values are used in the Consolidator to generate table names and their efficient searching. Activity Stream saving into the database is continuously updating process. For making the system more flexible to the user, PublishRedundant class of this package is used to make and update a redundant copy of database periodically. User query will be answered from that redundant database. This idea makes the system more persistent.

5.3 Frameworks and APIs of CoASGen System

For implementing CoASGen system’s prototype, some Application Package Interface (API) method and Framework has been used. XML parser used to read configuration file at the time of deploying system. XML parser converts an XML document to DOM object which can be easily access by system. LingPipe is a tool kit used in CoASGen system, can process text using linguistics and find out word’s parts of speech. For forming semantic structure of high level Activity Stream, it was used to determine parts of speech of different syntactic parts. The thin wrapper Java Timestamp was used to compare time of different feed items. For saving feed items dynamically into the database, Java Hash Map table was used to make the process transient. And the Framework and editor
was used to develop CoASGen system prototypical implementation is Eclipse Helios which makes this big project easy to handle.

5.4 Databases

CoASGen system's prototype was developed based on persistent database. To make the data persistent, CoASGen system uses redundant copy of each important data in most of the operation. MySQL 5.5.8\textsuperscript{3} database was used to store all of the data including feed items, activities, high level Activity Stream. Using Java Database Connectivity (JDBC) connector MySQL was connected with source framework. From the user interface (which was developed using JSP Servlets), it was quite friendly to access data (e.g. MetaActivityStream, different format of high level Activity Stream, activities etc.) from database.

5.5 Summary

The whole CoASGen system's implementation architecture is described in this chapter with most of the packages and classes. By using graphical picture of CoASGen system's configuration file, the complete configuration system is described here. The output of Mediator also described here with generated output. The two important packages, Consolidator and Activity Stream Generator is described here with graphical architecture. Furthermore, Framework and API used in CoASGen system development is also discussed briefly.

\textsuperscript{3}MySQL is the world's most popular open source database that is supported by an active community of open source developers and enthusiasts.
Chapter 6

Evaluation

The evaluation is intended to prove that the system fulfills its expected functionalities and requirements. In this chapter, the design methods, responsible algorithms, techniques and the complete results of CoASGen system are assessed. The evaluation of CoASGen system is divided into couple of parts. At first, a case system study is represented to evaluate its basic processes and their functionalities. There is also a technical evaluation with the system designs, methodologies and techniques in this section. The efficiency and effectiveness of CoASGen system is assessed by its performance analysis in the next section of this chapter. The succeeding section deals with an end users study where there are number of questionnaires are conducted to evaluate the usability of this system’s application. Then the whole CoASGen system is evaluated with the goals and requirements analysis (described in chapter 1.2 and 2.2) in this chapter. Finally, at the end it summarizes this chapter.

The technical evaluation of this chapter is structured in the following way. At first, the methods and techniques applied in different level of Isolator in Consolidator are evaluated in a regular fashion. Then there is an evaluation of high level Activity Stream generation and then briefly described about each component’s processing steps. The efficiency of CoASGen system is conducted this chapter by considering the performance analysis of every important role playing tools which is implemented in the prototypical implementation system of this thesis.

6.1 A Case System Study

As a case system study, CoASGen system is developed for verifying its basic processes and functionalities, which evaluates technically its design methodologies and applied
Chapter 6. Evaluation

For case study system application, different independent heterogeneous source systems has been chosen as as a feed items source provider. In this study, we used SVN (Subversion system), Wiki and Bug Tracker system for feed items source providing which provides lots of real time feed items. This application is aimed to retrieve approximately huge amount of lifetime feed items and isolating them based on some standard rules and conditions. Finally, it generates high level Activity Streams using different predefined high level Activity Streams format. In this application, users have an opportunity to configure the system using number of parameters which makes the system more flexible based of user’s personal desire or need.

First of all, the running environment of this case study system application is following: The server is running on Apache Tomcat 7.1 in embedded Eclipse Helios Java EE IDE. It retrieves data from MySQL 5.5 based on users query from user interface. User interface is designed with JSP (Java Server Pages) Servlets which is able to work with embedded Java code. And in the back end, real time feeds are retrieving from different independent source systems using Java API call. Then all the logical calculations in Consolidator and Activity Stream generator is done by standard Java object oriented programming. The case study is designed as application process management life cycle in two parts: One is about the calculation and methodologies includes in Consolidation process and another part deals with different isolated type descriptions as well as generation processes of high level Activity Streams to the user interface. The following figure 6.1 shows an overall overview of this application in a sophisticated way. The entry point of this system is its user interface with the configuration page where user get an opportunity to configure the system based on his or her requirements. Then it motivates user to the metaActivityStream page to get update of recent Activity Stream generations. Notice that when there is any new Activity Stream generation or update of existing one, metaActivityStream will be automatically updated. So, users are always connected.
with updated information by a brief overview of metaActivityStreams. Controller of this figure is responsible for controlling the system based on users need. It also passes information from different components of the system to the user interface.

A huge amount of real time feeds are retrieving in the Mediator, which are generalizing as a XML feeds and proceed to the Consolidator for further processing. Consolidator isolates these feed items using its five Isolators and periodically update these informations to the Activity Stream Generator to generate high level Activity Streams. User interface periodically send query to the Activity Stream Generator via Controller and get the updated information of new Activity Stream generation and display them as metaActivityStream. User interface also query to the consolidator often to recognize if there any update in configuration settings.

### 6.1.1 Application Process Management Life cycle

#### 6.1.1.1 Configuration

At the starting of the system when user interface is deploying, there is a configuration system for user where user can configure using a set of parameters based on their need. If users want to skip it, the system will work with default configuration. This configuration system saves information for particular user. User can choose high level Activity Stream type (from different three format structure) using combo button in the figure 6.2 from currently offered three types of Activity Stream format which are "using general

![Figure 6.2: Screen print of user configuration page](http://localhost:8080/ActivityStreamUI/View/UserConfiguration.jsp)
model format”, ”using prefix model format”, ”using postfix model format” (more details described in chapter 4.4.2). Currently for this system configuration ”Using general model format” is chosen in the figure 6.2. Using radio button users get an opportunity to choose reasonable frequency level for one Isolator of Consolidator. Notice that Consolidator has five different Isolators. This parameter taken care the isolation process’s level of strictness (see chapter 5.2.1). Currently, it has been chosen as 40% which means it will work with less strictness which means it will decide any feed Item is similar with another feed item if it found frequency matching (basically considers nouns and verbs because they are the meaningful word of a feed items) more than 40% in them. This user configuration option also gives user an opportunity to configure another Isolator of the Consolidator which is Regular Expression Comparator. As Frequency Comparator, it will decide similarity of the feed items based on comparing the certain pattern of text which found by using Regular Expression (more details in chapter four). User can set every atomic streams lifetime in the database by selecting next parameter’s radio button which is selected here as 30 days or two months. The system automatically checks after a fixed time interval if this time limit is over or not for every atomic streams and it will take action if condition satisfies. And the final parameter deals with how Activity

![Figure 6.3: Screen print of user configuration page after configuration applied](http://localhost:8080/ActivityStreamUIview/fullFormConfirm.jsp)

Streams will be shown in user interface which means based on what it will show Activity Streams. There are two options in this combo box which are ”Based on updated time” or ”Based on importance” (more details description in chapter 4.4.3). Currently, ”Based
on updated time” to show Activity Streams in the user interface based on its creation or updating time.

After filling up this configuration page, there is a button ”Set config” for apply this configuration in the system. When user confirms configuration then he or she get a confirmation page regarding all the selected information shown in figure 6.3. Once configuration is done user get a button for directly overview of the available updated Activity Streams metaActivityStream. MetaActivityStreams are frequently updated by Activity Stream creation or update (existing one). It shows as a sophisticated representative and works as metadata of high level Activity Stream’s current status.

6.1.1.2 MetaActivityStream

Metadata is a kind of structured information which represents an overview of a certain amount of information. It gives an easier way and link to retrieve the actual intended data. MetaActivityStreams are the metadata of every particular Activity Streams. In the figure 6.4, there is a screen print of metaActivityStream for the generated Activity Streams in this case study system which offers metaActivityStreams by following links and therefore its generation time.

![Welcome to Activity Streams generator](image)

**Figure 6.4:** Screen print of metaActivityStream for the case study application system

This metaActivityStreams are frequently updates by querying metaActivityStream table periodically. In this system, metaActivityStreams are designed in an interesting way. It
Chapter 6. Evaluation

creates a persistent metaActivityStream table. When there is any new Activity Stream generation or update, this system inserts a metaActivityStream to the metaActivityStream table. And then it automatically publish this information to the user interface, which describes one new Activity Stream is generated or has update. After a certain duration (Activity Streams life time), metaActivityStream is automatically being deleted from the metaActivityStream table. So metaActivityStream table is always updated. By querying that metaActivityStream table in a fixed time interval, it is easy to get in touch whether there is any update of Activity Stream or not. Users also have an opportunity to get updated metaActivityStreams by refreshing the page. Because refreshing executes the query on metaActivityStream table where it views the updated information of metaActivityStream table to the user interface. For the current application there are lots of metaActivityStreams for example ”Interface Improvements”, ”Displaying results”, ”Configuration”, ”Data Export” etc. which are the individual representative of lots of similar kind of atomic streams. These metaActivityStreams has been generated by the system automatically based on high level Activity Stream’s minimized information. For this application the generation times are same because this system generates these Activity Streams at one execution. By following the link given in metaActivityStream, it will derive user to the high level Activity Stream view page where it has more details description and option about that particular high level Activity Stream.

6.1.1.3 High Level Activity Streams Generation

From each metaActivityStream, user can access one high level Activity Stream page which offers a special page with that high level Activity Stream’s viewing time and responsible atomic activities (with a link). The system works as follows, by following the link of metaActivityStream, it directly queries to the high level Activity Stream database and view the responsible high level Activity Stream based on users preselected high level Activity Stream types which belongs to one format from generated three. Notice that there are three format of high level Activity Streams (one of them should be selected at the time configuration). The high level Activity Stream is shown in figure 6.5 by taken a screen print of this page. It is generated from 10 activities (short description of feed items). In this figure, the high level Activity Stream is ”Security /Restriction is recently updated” which means there are some feed item update (or some new feed items) about ”Security /Restriction” with the number of responsible updated activities. The number of activities (here in the figure 10 activities) is shown in the next parameter of the figure with a link which is able to mobilize users of its all activities (or streams). So, a user can easily understand by this high level Activity Stream that what kinds of information of these activities might have. Now they do not need to view every particular activity
from the streams to have a look any particular feed item’s information. Instead, they can find high level Activity Streams and if he found the content of high level Activity Stream (by looking its precise description) is important for him then he may go in and have a look its activities to find his desired feed item. So, creating high level Activity Stream, the system become fine grained to find any particular information. This way users can save lot of valuable time and efforts. If we think in another way, similar types of information are organized in one high level Activity Stream even with old atomic streams (for example, last month’s streams). So, by configuring the CoASGen system perfectly, users are not certain to miss any information which they are looking for even it is an old activity.

6.1.1.4 Streams Generation

By clicking the parameter of 10 activities in figure 6.5, the system gives a wide description of all activities in stream format of information in figure 6.6. It gives clear description by generating streams where stream format includes author name, some technical keywords from responsible activity and generation time. For example, in the stream generation page described in the figure 6.6, one of the atomic streams is "Will Palmar reported on server authentication, signon, multiple at 2010-08-04." where Will Palmar is the author, server authentication, signon, multiple are the technical keywords to describe what this activity about and 2010-08-04 is the publication time of this stream. There is a configuration language to find out the author information from the feed item or activity description. For finding technical keywords from the feed items, CoASGen system uses a
configuration language (described in chapter 4). These technical keywords emphasize the outlook of this particular feed item’s description. And it uses the time parameter as the published time of the feed item. A clear description of 10 similar activities is shown in the figure 6.6. It gives a way to visit a lot of similar kind of information stream in a regular fashion. This is an example of one high level Activity Stream’s responsible activities. All other high level Activity Streams generated in this system also has different amount of similar information streams. So, there is no doubt that these streams have similar information. If someone come to visit this High Level Activity Stream and for him it is easy to visit this organized data (atomic streams) within a short time. He does not have to visit particular data arbitrarily anymore.

6.1.1.5 Feed Item Description and View

By clicking the hyperlinked technical words of stream page in the figure 6.6, the CoAS-Gen system will directly let you go to the feed items details description which is depicted in the figure 6.7. This page describes real activity information in an organized way where title, description and generation time parameters taking place with full text. One can get back to stream page in the figure 6.6 from this page using ”Back” button. Then he may visit all other activities details by clicking on the link given there. By following the
back button one can go back to the metaActivityStream again and by looking if there is any update (new metaActivityStream generated or any existing metaActivityStream updated) therefore choosing and by visiting responsible streams for any particular high level Activity Stream, he or she can have a look in details of any activities or feed items in the user interface.

### 6.2 Objective Evaluation

The objective evaluation deals with the output of the system for number of different types of input sources. This is a way to evaluate the system if it can have expected output and performance using all of its designs, methodologies, techniques etc. For evaluating the system, 100 individual feed items are taken based on their content description (topics, description), category, published time, valid information etc. We collected these feed items manually with ten different types of activities (based on these parameters) and ten feed items in each type. Each type of feed items has similar kinds of information, for example, in first type we have 10 feed items where each of these feed item’s authors are talking about 'Interfaces of software’, the published time duration of these feed items are within two years, in the feed item’s text description they have similar category and their all parameters have valid information (sometimes some feed items could have corrupted parameters which are not processable by any intelligence system). We evaluated the
system part by part. Basically, there are two main parts of CoASGen system which are Consolidator and Activity Stream Generator. Mediator has comparatively short and precise responsibility. And Consolidator is one of the most important parts of them which have five Isolators to isolate feed items based on their similarity. For generating high level Activity Streams, good precision or accuracy of Consolidator’s output would be very important. Precision would be define as the degree of closeness of measurements of a quantity compare to the actual (true) value.

### 6.2.1 Evaluation of Consolidator

To evaluate Consolidator, we test the feed items which we get in the output of Consolidator with source feed items. Therefore, manually we took ten different types of feed items as an input source for the CoASGen system. These ten different types of 100 feed items are collected arbitrarily (no sequence or organization maintained) from the source database. To evaluate Consolidator we collected these input feed items from SourceForge\(^1\). They have been collected based on some parameters which are content description (topics, description), category, published time, valid information etc. And the configuration of CoASGen system has been set as, Top Value selection for taking decision in the Isolator (Regular Expression Comparator) is 30%, Reasonable percentage level for the Isolator (Frequency Comparator) is 50% and Time duration is selected as two years. These two important Isolator of Consolidator is configured with average value. So, the output should be not fine grained and not even coarse grained but in between them. After processing these feed items in the Consolidator of CoASGen system, the outputs are taken to compare with source feed items to examine the level of precision of Consolidator in this system. Each type of feed items in the source table 6.1 for example ”Interface Improvements”, ”Configuration”, ”Data Export” has ten individual feed items and they are considered with similar category of feed items. Notice that feed item’s description has a special part called category which describes about the intension of feed items.

For taking feed items in the source table (before processing by Consolidator), most of the feed items has been considered within certain time of publication, for example time between 2009 and 2010. Also some few feed items are taken with different published time because of checking the Consolidator if it is able to isolate feed item by taking care fixed time duration or not. And of course we confirm that these feed items in the source table have valid information. In the line graph figure 6.8, feed items are taken manually with different types which are depicted in the line graph horizontally and in the vertical

---

\(^1\)SourceForge is a web-based source code repository. It acts as a centralized location for software developers to control and manage open source software development.
Table 6.1: Source table of input feed items

<table>
<thead>
<tr>
<th>Number/Parameters</th>
<th>Feed items</th>
<th>Minimized Type Name</th>
<th>Number of feed items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interface Improvements</td>
<td>type1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Configuration</td>
<td>type2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Data Export</td>
<td>type3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Authentication Issues</td>
<td>type4</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Security / Restrictions</td>
<td>type5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Privileges</td>
<td>type6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Browser/W3C/JS compatibility</td>
<td>type7</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>Relations,Comments,History</td>
<td>type8</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>Data insertion/extraction..</td>
<td>type9</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>Displaying results</td>
<td>type10</td>
<td>10</td>
</tr>
</tbody>
</table>

Evaluation: 100

column of this graph describes number of feed items in each type. Here, in the figure 6.8, a line graph is going to straight which describes each type of feed item’s initial stage that means these feed items are not processed by Consolidator yet. The next table and graph depicts the result or isolated feed item by Consolidator. Then we will compare these two parts to find out approximate precision of Consolidator.

![Source Feed Items (manual)](image)

**Figure 6.8: Line graph of source feed items (manually selected)**

After processing (source feed items) by Consolidator, we have a bit change in the feed items number in some of the types. That means Consolidator of CoASGen system decides isolation of these 100 feed items in the following way in table 6.2. The reason and based on which parameters these number of feed items are changed are described in the next section. It almost same but "Browser/W3C/JS compatibility" type got two more feed items and "Displaying results" lose that two feed items. And rest of the types have same number of feed items which implies Consolidator of CoASGen system has good output by using it’s all of the five Isolators. In the end of this section there is a
comparison of these two tables which describes the precision level of Consolidator more clearly.

Table 6.2: Isolated feed items by Consolidator

<table>
<thead>
<tr>
<th>Number/Parameters</th>
<th>Types of Feed items</th>
<th>Minimized Type Name</th>
<th>Number of feed items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interface Improvements</td>
<td>type1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Configuration</td>
<td>type2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Data Export</td>
<td>type3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Authentication Issues</td>
<td>type4</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Security / Restrictions</td>
<td>type5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Privileges</td>
<td>type6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Browser/W3C/JS compatibility</td>
<td>type7</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>Relations,Comments,History</td>
<td>type8</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>Data insertion/extraction..</td>
<td>type9</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>Displaying results</td>
<td>type10</td>
<td>8</td>
</tr>
</tbody>
</table>

Evaluation: 100

The following line graph in figure 6.9 describes the table 6.2 output in graphical way. The vertical column of the graph describes the number of feed items exist in every types and horizontal column implies the name of different types found in the Isolated output of Consolidator. Here, we found first six points (which depicts as types in the type table) are absolutely straight which means these six types remain same as source table with same amount of feed items. These isolated types has same feed items with the parameters (e.g. feed item’s content description (title and text), category, time duration etc.) declared in the source. Then point seven shifted a bit to the upper and point ten shifted a bit to the downward that means there are some change with feed items in these types here with compare to source graph. In type seven two more feed items are added from any other type and type ten loses two feed items. In the next table, we have a clear comparison of these two tables.

Finally, table 6.3 describes the comparison between Consolidator’s input and output. First row (Interface Improvement), we find a number of feed items in input and Consolidator outputs are same and we checked feed items description in the database that they are same as source feed type had. So, for the first type (first row) of feed items, we can say that Consolidator works properly and its precision is 100%. The next five rows have same input and output sequence. Therefore, they have similar precision level. At the row no seven, the type ”Browser/W3C/JS compatibility” has some change in output. Two more feed items are added with this existing ten feed items. By checking these two feed items from database we found that they came from the type ”Displaying results” because they have similar frequency (noun and verb) with the existing feed items in ”Browser/W3C/JS compatibility” type. Because of two feed items have been affected in the seventh row of feed items types, in this type the precision level become 80%. And
these feed items shifted from other type ("Display Result" which was previously defined in the source table), so that the type10 also affected with two feed items in precision. So, tenth row’s precision level also becomes 80%. By calculating all types precision level, for these 100 feed items the Consolidator’s overall precision becomes 96%. Most of the feed items type has been decided by the Isolator, Regular Expression Decider which is responsible for domain specific identity and Frequency Comparator which is deciding feed items similarity by comparing their important words (noun and verbs are only considered here as important words) similarity. By visiting the log file, one can have a look about which feed item type has been decided by which Isolator in the Consolidator.

**Table 6.3**: The comparison table of source and isolated feed items (by Consolidator)

<table>
<thead>
<tr>
<th>Number/Parameters</th>
<th>Minimized type name</th>
<th>Number of feed items (Input)</th>
<th>Number of feed items (Isolated)</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>type1</td>
<td>10</td>
<td>10</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>type2</td>
<td>10</td>
<td>10</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td>type3</td>
<td>10</td>
<td>10</td>
<td>100%</td>
</tr>
<tr>
<td>4</td>
<td>type4</td>
<td>10</td>
<td>10</td>
<td>100%</td>
</tr>
<tr>
<td>5</td>
<td>type5</td>
<td>10</td>
<td>10</td>
<td>100%</td>
</tr>
<tr>
<td>6</td>
<td>type6</td>
<td>10</td>
<td>10</td>
<td>100%</td>
</tr>
<tr>
<td>7</td>
<td>type7</td>
<td>10</td>
<td>12</td>
<td>80%</td>
</tr>
<tr>
<td>8</td>
<td>type8</td>
<td>10</td>
<td>10</td>
<td>100%</td>
</tr>
<tr>
<td>9</td>
<td>type9</td>
<td>10</td>
<td>10</td>
<td>100%</td>
</tr>
<tr>
<td>10</td>
<td>type10</td>
<td>10</td>
<td>8</td>
<td>80%</td>
</tr>
</tbody>
</table>

**Evaluation**: 100 100 96%

Since Consolidator’s output is going to use for generating high level Activity Stream, Consolidator is the most important parts of CoASGen system and for keeping its precision level good, we used five separate Isolators in Consolidator. And their operational
efficiency proved by the table 6.3 and compare graph of figure 6.10. In this graph, different feed item's type is described in the horizontal row and the vertical row is responsible for number of feed items exists in each type. The blue line is for source feed items and red line is for output of Consolidator. By comparing their statistics, blue line and red line are going almost same direction which indicates that Consolidator is able to isolate feed items and separate them as in different type of feed items.

![Consolidator's output comparison](image)

**Figure 6.10:** Comparison graph of source feed items and isolated feed items (by Consolidator)

And the red line has a bit change in last two points (7th and 10th) which means it has a bit inaccuracy that is ignorable because of some feed items could have unorganized text and data and they do not have any precise rules and regulations in the information or text. And the machine decision has to depend on some of the parameter of the feed items (i.e. Title, Description, Time, Author, Category, Domain specific regular expression, URLs etc.). Moreover, considering everything Consolidator’s overall efficiency level is very good.

### 6.2.2 Evaluation of Activity Stream Generation

#### 6.2.2.1 High level Activity Stream Generation Process

High level Activity Streams are the representative of similar kind of lots of feed items in each type. So, it is important that if this high level Activity Stream is able to represent the overall feed item’s common and general view. In the figure 6.11, high level Activity is generated by using its predefined format structure. Notice that there are three formats of high level Activity Streams described in chapter 4. This high level Activity Stream is generated by using high level Activity Stream postfix format structure. Since there
Chapter 6. Evaluation

is no high level Activity Streams generating system offered available to compare with, it will be described with the contextual and information subset it belongs to. Each of the format structure has different semantic structure. For example, general format of Activity Stream has a combination of subject, be verb, object/verb. Subject is decided by using a configuration language where it keeps one of the keyword pattern from feed item’s category (if it has in the text description), domain based keyword, technical term, renowned word or first noun from the title to depicts the idea of each activities. And this high level Activity Stream is generated by semantic combination of subject, be verb/verb and fixed postfix.

![Figure 6.11: High level Activity Stream of a type consisting 10 activities](image)

And the high level Activity Stream is generated by using common subject of activities or feed items. So, it is pretty obvious that it describes the general overview of all activities in the particular isolated type's feed items has. This high level Activity Stream has its generation time embedded with the number of activities (following by a link) which it is responsible for. By using this link user can easily follow the streams activities in a second where they have also a possibility to view the full description of feed items by following a link. This high level Activity Stream shown in the figure 6.11, has subject, be verb and verb/object. Here subject is "Relation, Comments, History" decided by the system, be verb is "is" and it could not find any reasonable verb or objects. Therefore, it gives an alternation objects which fits with the subject and be verb. But the system output of Activity Stream is to show the general overview of these feed items of one particular type.

6.2.2.2 High level Activity Stream Format

In this master thesis three format structure of high level Activity Streams are finalized. Here, in figure 6.12, a database screenshot is taken where three formats of two high level Activity Streams are described. In this three format structure, "General Activity Stream Format", "Prefix Activity Stream" and "Postfix Activity Stream" has different semantic structure. First one "General Activity Stream Format" has subject, verb and main verb/object in structure. These parts of the sentences are configured by system automatically itself using a configuration language where few sentence structure format is described. By having a look at the database screenshot, "Interface Improvements" is
Chapter 6. Evaluation

the subject, "are" is be verb and "recently updated" is helper objects used here because for this Activity Stream, system could not find any acceptable main verb or objects which is fit with the sentence. But in the next Activity Stream (next row) "Display result" is the subject, "is" is be verb and shown is decided as main verb/object.

<table>
<thead>
<tr>
<th>generalactivitystream</th>
<th>prefixactivitystream</th>
<th>postfixactivitystream</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface improvements are</td>
<td>Some update about Interface</td>
<td>Interface Improvement has</td>
</tr>
<tr>
<td>recently updated</td>
<td>Improvements</td>
<td>ignored recently</td>
</tr>
<tr>
<td>Displaying results is shown</td>
<td>Some update about Displaying</td>
<td>Displaying results shown</td>
</tr>
<tr>
<td></td>
<td>results</td>
<td>recently</td>
</tr>
</tbody>
</table>

**Figure 6.12:** screen print of three different high level Activity Streams format

The second format of high level Activity Stream "Prefix Activity Stream" is pretty straightforward. It has a fixed prefix which is "Some update about" here in the first row and then following by its subjects. Sometime it could happen like be verb and main verb/object is not found by using configuration language in the feed item’s data then this format is very useful to view the high level Activity Stream. In the third high level format structure "Postfix Activity Stream" has subject, verb and fixed object. Here, in figure 6.12, "Interface Improvement" is selected as subject, "ignored" is verb and "recently" is fixed object. In this structure the system is able to generate high level Activity Streams with subject and fixed object only. For showing the idea works, we just developed a prototypical system with some few semantics in these high level Activity Stream format. By using more semantic structure in these format structure high level Activity Stream can be develop with more rich structure.

6.2.2.3 Stream Generation Process

Streams are the short and precise representation of feed items for every particular isolated type. It gives short overview about the detailed data of every feed item. So, the challenge is to keep some technical term for every particular feed item by developing some configuration language. Every feed item has mainly two parts of data description (title, text) where we can find some technical terms for viewing this feed item’s general overview. Basically, there are three parts of data in every stream generated in here in figure 6.13. They are author name, object and published time where object deals with finding out some technical word from the feed item which depicts the basic information of that feed item. For finding author name and object there are special configuration language to figure out from the feed items data. For finding author name the configuration language has to find out right author name from each feed description. Basically, most of the feed item has a parameter authorname which has a mail address (from whom it comes from). Some feed item offers their author name in the text and
some others do not have. Those feed items do not have author name in the text and even mail address in that parameter in that case we have to rely on any renowned web address it gives in the text description. So, we set the priority for finding author name in this way, first it looks for if it can find out author name in the text description and if it fails to find out then it goes for that parameter (authorname) where we find sender mail address. If it fails to detect further then it looks for any renowned web address in the text description. So, it is pretty obvious that it will find one author name for each feed items. For finding object from the feed item, it uses a domain based configuration

![Figure 6.13: Screen print of streams in one isolated feed type](image)

![Figure 6.14: Screen print of stream generating database](image)

system where a set of technical terms are already predefined (configured by administrator). A screen print of such a technical terms configuration is showing in the figure 6.15. Here, this terms are selected for Software Company environment. These technical terms depicts feed information overview in a second. Basically system check these terms with feed titles and text description fields. For selecting more and effective technical word for each feed item, there is a checking system which checks about which part (title or text description) has more technical terms. The part has more technical term (found by matching this configured technical terms with feed item’s terms) is selected for object generation. Finally, it generates object from these technical part. And time is basically the publication time of feed items. Using all of these parameter (author name, object, time), the system generate streams efficiently for each feed items in a feed type. Figure 6.14 gives a database screen print of the detailed creation process of each part in a stream generation of feed items. CoASGen system makes the streams easily accessible from user interface.
Chapter 6. Evaluation

6.2.2.4 Generation of MetaActivityStream

MetaActivityStream is the metadata of lots of high level Activity Stream generated by the CoASGen system. From this metadata user can have access of updated data by every 60 seconds (user configurable). Since the system is frequently updating, this page automatically refreshes itself with the updated data within every 60 seconds. In the figure 6.16, there is a screenshot of one MetaActivityStream generated by CoASGen system. It has basically two parameters which are MetaActivityStream and Generation time. MetaActivityStream is simply taken from the subject of each high level Activity Stream. Basically, subject depicts the main idea of high level Activity Stream and that is why it works as very short description of the high level Activity Stream. Here, in the figure 6.16, "Relations, Comments, History" is one metadata of one high level Activity Stream which was selected from the subject of that high level Activity Stream. MetaActivityStream is following by a link where user can click and visit the details of high level Activity Stream.

```java
// Configure your domain based technical terms here
}
```

Figure 6.15: Screen print of technical term configuration

Figure 6.16: Screen print of MetaActivityStream in CoASGen system
6.3 Real User Evaluation

To make end user evaluation, a number of questionnaires have been conducted to 10 users by let them visiting user interface of CoASGen system. All of the peoples are major in Computer Science who were involved in this end user evaluation. The questionnaires were made by considering different technical stubs of CoASGen system. There are 100 different feed items taken as an input source and they were chosen based on content description, published time duration, category and valid information.

The questions were formulated using few of the different categories (based on CoASGen systems different components and methods). And it was requested to the users that please think the system as a real world system and give us your feedback. There was no right or wrong answer but every question had a satisfaction level to answer from 1 to 10 where 1 was worst and 10 were the best. Here, in the following there is a precise description about the end users response by visiting the overall system. The questionnaires conducted to the end user is described in the table 6.4 by categorized them in the left sight of the table and after the description of the questions in the middle there is a calculation in the right sight where Mean (average) and Standard Deviation (Std.) is calculated by user answered numbers for each questions. The following paragraph will discuss the result of end user feedback.

1. There are certain parameters in the configuration system which will be applied to the system for deploying. How helpful was the configuration system of this CoASGen system?

   The first question was about configuration system of CoASGen system. The mean value of the result was 8.7 and standard deviation was 1.33749. The end user experiment result was good but not very interesting for the users because it asks some of the system level configuration parameter (i.e. Isolator of Consolidator configuration) which requires prior knowledge about the system internal design and algorithms. That is why it was comparatively not easy to understand for the user and as a result the standard deviation\textsuperscript{2} is a bit high.

2. MetaActivityStreams are the metadata of every particular high level Activity Stream. How helpful is it in CoASGen system?

   The answer of second question achieves the best result among all other questions in the end user questionnaires. That means the idea of MetaActivityStream is pretty helpful for the user. It gives them an immediate general overview of the

\textsuperscript{2}Standard Deviation is a widely used measure of variability or diversity used in statistics and probability theory. It shows how much variation or "dispersion" there is from the average (mean, or expected value).
whole high level Activity Stream generated by CoASGen system which are easy to follow. The mean value of this question’s result was 9.8 and standard deviation was 0.42164.

3. Is high level Activity Stream able to mobilize users to find their expected data or feed items by following these streams that it offers?

The first question of third category ”High level Activity Stream” describes if the generated high level Activity Stream good enough to motivate user to the feed items which they are looking for. The result mean value was 9.3 and standard deviation was in this case 0.8756. That means users think that the generated high level Activity Stream can motivate them to their expected information. But some users find some challenge to judge this question because it requires a bit research to find perfect feedback.

4. Do you think this high level Activity Stream will be helpful to reduce data loss and save user’s time?

The second question of third category was a general question that was asking, by using this high level Activity Stream if the system is able to give the expected benefit (reduce data loss and save user time) which was systems design intention. The mean value of the result was 9.5 and standard deviation was 0.97183. The result shows that user are agreed about the motivation of the system.

5. How do you like three different kinds of high level Activity Stream format?

The question in category four was to get feedback from the user if they like CoASGen system’s generated high level Activity Stream format. End user feedback about this question was for mean value 9 but the standard deviation was a bit high (1.05409). Actually, for the prototype development we used some few semantics to show that the idea of CoASGen system is works. One can add more semantics to the format to make the high level Activity Stream more attractive where it will have more dynamic and well language structured sentence formation. Some users gave feedback to update CoASGen with rich semantic structure.

6. How appealing is the representation of streams in the user interface? Does it lead to the feed items data description?

The first question of category five asking about the representation of streams in the user interface. The result shows as a mean value 9 and standard deviation was 1.24722. User thinks it is working well and some of them feels like it could have a bit more impressive representation by putting more detailed information (to the user interface) about the feed items. Basically, this is a prototypical implementation
and representing streams in the user interface was not a major task. So, it was not very important part to show streams in the user interface nicely. Any graphical designer can make this representation more attractive.

7. Please give your perfection level about the stream generation from direct feed items

The second question of fifth category was about generation of streams from feed items. The end user evolution for this question was such as mean value 8.8 and standard deviation 0.91894. Actually, understanding this question was a bit hard for the end user. We had to describe it. And some users have comparatively less impression about this question. The deviation level was not very impressive because some few users think that it could be a bit improved for the sake of more users friendly.

8. Following one high level Activity Stream, all streams are just consolidated (isolated) streams of similar types. What do you think about the accuracy of consolidation process?

At the sixth category, the question was about accuracy of Consolidator of CoASGen system. The mean value of the result was 9.5 and standard deviation was 0.70711. That means users think that the consolidator can isolate feed items with standard precision. And the standard deviation level for this question shows that most of users have similarity for answering this question. They appreciated about the five Isolators role in the Consolidator.

9. Is created UI easy to understand?

The seventh category question gets 8 as a mean value and 1.41421 as standard deviation. User thinks the user interface is not very standard yet and in some point it is not easy to understand. This question gets the worst mean and standard deviation among all the questionnaires. Some user experienced little bit complicacy to get back using button of the user interface which is not efficiently formulated for the back mode. It does not remember the last executed query. They recommended for creating more user friendly user interface for dummy user where there will be less technical stubs. Actually, designing very good user interface was not the central intention about this thesis work. User interface creation was just to show the design and idea is works.

10. What do you think about the response time of this system?

The first question of eighth category result has 9.1 as a mean value and 1.28668 in standard deviation. It was asking about the response time of the system. Users think that the response time of CoASGen system was ok. But some user finds
a bit complicacy due to the lack of user interface design that’s why the standard deviation of this question was a bit fluctuated.

11. Please give your satisfaction level for the system’s overall structure and performance.

The overall satisfaction level is depicted in the question two of last question of last category which got 9.3 as a mean value and 0.82327 as standard deviation. End users think the idea of the system is difficult to understand but it is a impressive idea and they believe that the idea will work in the real field. The standard deviation level of this question also shows that users have similar idea about the system design and goals.

6.4 Requirements Fulfillment

There are a number of requirements already discussed for this master thesis in chapter 2 which is going to be reconsidering in this section to make confirm about the final achievement of this work. The requirements fulfillment is basically motivated by the design of CoASGen system, implementation of a prototypical system and evaluating the prototypical system by both technical and end user evaluation. In the following table 6.5, there are some parameter used to describe the system requirements and final achievement. At the left sight, Category describes different requirement name precisely and the next right parameter describes their description. Next four parameters describes about the importance level, implementation workload, if these requirement were one of the goal of this thesis and finally Achievement shows the level of success in this master thesis for that particular requirement. The level indicates from 1 to 5 here where 5 is the best and 1 is worst. The following paragraph will describe every particular requirement and their achievement is a brief discussion.

1. Heterogeneous Sources: The design and implementation of CoASGen system can deal with different heterogeneous sources. Feed items with different feed format (e.g. RSS 2.0, RSS 1.0 and Atom) are transferred to a general feed format (XML format) structure for fast processing in the Consolidator. So, the CoASGen system is able to deal with heterogeneous source feed items. It was one of the goals of this thesis and the level of importance was very high. And the success of this requirement is fulfilled with good degree.

2. Aggregation and Consolidation: This is one of the most important design goals of CoASGen system because the output of the Consolidator should be isolated to different feed item types which are the input of final system (Activity Stream
generator). That’s why the five isolators of Consolidator are specially taken care in this system design and implementation. The two most important Isolators (Regular Expression Comparator, Frequency Comparator) in Consolidator were the difficult level of implementation task and finally they are performing well. The overall importance of this parameter was very high and the achievement is also satisfied with very good degree (proved by Consolidator’s objective evaluation).

3. **Generate AS**: High level Activity Stream generating is a central task and requirement of this master thesis which was also a goal of the system design and implementation. And the CoASGen system is able to generate standard high level Activity Streams from different types of feeds where lots of similar feed items are isolated to one type. It maintains some semantic structure to generate human understandable sentences which are the representative of lots similar feed items for each feed types. This requirement’s importance and difficulty level of implementation was very high and CoASGen system also has good level of achievement.

4. **AS Semantic Format**: Generating high level Activity Stream requires semantic format to formulate human understandable sentences. Standardizing some semantic structure of format was a goal of the CoASGen system design and implementation. It was taken care by implementing three different format of high level Activity Stream format which are General Activity Stream format, Prefix Activity Stream format, Postfix Activity Stream format. Each of these format has different semantic structure (described more details in chapter 4). The importance level of this requirement was high but level of difficulty was just next (below) level of very high. And the achievement is also good but it can have a bit more achievement level by using more sentence formation grammar and semantics.

5. **Stream Generation**: Stream generation from the direct feed items was a minor goal of the CoASGen system. By developing some configuration language CoASGen system generating stream from feed items. The importance of this parameter was not the highest and its difficult level was also in a friendly range. But the achievement level for this requirement in CoASGen system is impressive. The system can automatically configure different parts of stream from feed item’s description and therefore generate user friendly streams from these different parts.

6. **Stream Representation**: Since CoASGen system is developed as a prototypical system, stream representation was not a major goal in this system. So, CoASGen system has typical stream representation in the user interface. And its implementation difficulty was in average level. The achievement level of this parameter was not very impressive but more than average.
7. **Automatic System:** Making the system totally automatic was a design and implementation goal. The CoASGen system is able to perform itself without any manual instruction or any help of other system. After configuration, the Mediator and Consolidator are calling by itself with a certain fixed duration to check if any new incoming feed items are available. And Activity Stream generator also continuously checks the log file if there is any feed items update in any available or new generated feed types. So, the system is totally automatic. The implementation difficulty was more than average and the achievement level was very good for this requirement.

8. **Scalability:** Scalability is important issue for developing any big system. Since the CoASGen system was a prototypical implementation that is why it was not that much important in this case. But we checked this system with big amount of feed items processing and its performance was good. It also works well with different heterogeneous independent system’s feed item. Since we do not run the system continuously long time in the real field, our achievement level for this parameter is considered as more than average.

9. **Accuracy and Consistency:** Since CoASGen system was a prototypical implementation, maintaining accuracy and consistency level very good was not a central goal. CoASGen system is just developed to show that the idea works. Managing accuracy and consistency level in a perfect way is very time consuming task. CoASGen system is maintained with every system level operation data input output perfection and some fault tolerance check with some few test cases. So, the achievement level is considered as more than average.

10. **Configurability:** Configurability was one of the central goals of developing CoASGen system. The system is easily configurable by both administrator and end user. There is a configurable file which has number of parameter to configure; for example, Activity Stream format selection, two different Isolators configuration of Consolidator, Stream’s life time in database, Activity Stream generating duration, Activity Stream displaying preference selection etc. It also asking at the time of deploying the system by an optional configuration page to the end user. Since the system is an automatic system, if any user does not want to configure the system then it will work with by default configuration. In addition, CoASGen system can be configured by domain based configuration. System administrator should configure this part. The development difficulty for this parameter was more than average but achievement level is very good.

11. **System Development Intension:** One of the goals was to design and implement CoASGen system for corporate company context but not public network.
That’s why the CoASGen system’s prototypical system was developed for a software company internal uses with the different independent systems they are using (e.g. SVN, Wiki and Bug Tracker). Every design and implementation method was developed by considering the corporate environment. The difficult level was just considering the environment in the design and the requirement is completely fulfilled.

6.5 Summary

In this chapter, different types of evaluation were conducted. We took 100 different feed items as input sources based on some parameter. The system performance and end user evaluation was pleasant. The very important component of CoASGen system, Consolidator has approximately 96% of accuracy in the technical evaluation. Requirements of CoASGen system are discussed in this section with the thesis result. This prototypical system is developed as domain independent development and in this case it was Software Company context. To adapt this system in any other environment (e.g. Corporate Company, Networks Management/ Support Community), it has to be configured based for that certain domain.
Table 6.4: The list of questionnaires conducted to end user and their result calculation

<table>
<thead>
<tr>
<th>Category</th>
<th>Questions</th>
<th>Mean</th>
<th>Std.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>1. There are certain parameters in the configuration system which will be applied to the system for deploying. How helpful was the Configuration system of this CoASGen system?</td>
<td>8.7</td>
<td>1.33749</td>
</tr>
<tr>
<td>MetaActivityStream</td>
<td>2. MetaActivityStreams are the metadata of every particular high level Activity Stream. How helpful is it in CoASGen system?</td>
<td>9.8</td>
<td>0.42164</td>
</tr>
<tr>
<td>High level Activity Stream</td>
<td>3. Is high level Activity Stream able to mobilize users to find their expected data or feed items by following these streams that it offers?</td>
<td>9.3</td>
<td>0.8756</td>
</tr>
<tr>
<td></td>
<td>4. Do you think this high level Activity Stream will be helpful to reduce data loss and save user’s time?</td>
<td>9.5</td>
<td>0.97183</td>
</tr>
<tr>
<td>High level Activity Stream Format</td>
<td>5. How do you like three different kinds of high level Activity Stream format?</td>
<td>9</td>
<td>1.05409</td>
</tr>
<tr>
<td>Atomic Streams</td>
<td>6. How appealing is the representation of streams in the user interface? Does it lead to the feed items data description?</td>
<td>9</td>
<td>1.24722</td>
</tr>
<tr>
<td></td>
<td>7. Please give your perfection level about the stream generation from direct feed items.</td>
<td>8.8</td>
<td>0.91894</td>
</tr>
<tr>
<td>Consolidation</td>
<td>8. Following one high level Activity Stream, all streams are just consolidated (isolated) streams of similar types. What do you think about the accuracy of consolidation process?</td>
<td>9.5</td>
<td>0.70711</td>
</tr>
<tr>
<td>User Interface</td>
<td>9. Is created UI easy to understand?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>10. What do you think about the response time of this system?</td>
<td>9.1</td>
<td>1.28668</td>
</tr>
<tr>
<td></td>
<td>11. Please give your satisfaction level for the system’s overall structure and performance.</td>
<td>9.3</td>
<td>0.82327</td>
</tr>
</tbody>
</table>
Table 6.5: Requirements analysis and achievement

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
<th>Importance</th>
<th>Difficulty</th>
<th>Goals</th>
<th>Achievement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneous Sources</td>
<td>Able to deal with single and heterogeneous data sources.</td>
<td>5</td>
<td>2</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td>Aggregation and Consolidation</td>
<td>Should have a standard and efficient aggregation and consolidation method for the real time feed items.</td>
<td>5</td>
<td>5</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td>Generate AS</td>
<td>Able to generate high level Activity Stream.</td>
<td>5</td>
<td>5</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td>AS semantic Format</td>
<td>System should offer standard semantic format structure of high level Activity Stream.</td>
<td>5</td>
<td>4</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>Stream Generation</td>
<td>Should have standard configuration language for generating streams from direct feed items.</td>
<td>4</td>
<td>4</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td>Stream Representation</td>
<td>Should have a standard way of representation of streams in the user interface</td>
<td>3</td>
<td>3</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Automatic System</td>
<td>The system should be totally automatic.</td>
<td>5</td>
<td>4</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td>Scalability</td>
<td>Scalable with the size of information as well as with the number of different independent system.</td>
<td>3</td>
<td>4</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Accuracy and Consistency</td>
<td>Should have good consistency in different system level operation.</td>
<td>4</td>
<td>5</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Configurability</td>
<td>It should offer a way to configure the system by both administrator and end user.</td>
<td>5</td>
<td>3</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td>System Development Intension</td>
<td>Intended to corporate company context</td>
<td>5</td>
<td>3</td>
<td>Yes</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Intended to enterprise or social network context</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>
Chapter 7

Epilogue

This chapter summarizes the conclusion of this master thesis. In the second section, it answers the discovered each research questions of this research work. Third section proposes the features of CoASGen system and their adaption idea to different corporate environment. Finally section four proposes some future works.

7.1 Summary and Conclusion

In this master thesis, a system is presented called CoASGen system which is able to solve the problem of data silos by reducing arbitrarily unmanaged huge amount of feed items and generate efficiently managed high level Activity Streams where users could follow the exact information they are interested in. In the current age of technology, Internet and Web technology become the center of information source. The main problem is the huge amount of data in the Internet which is managed as arbitrarily and unstructured way. It is difficult to keep track the information and a user has to spend a lot of time and effort in order to find the particular types of information he is interested in. The prototype of CoASGen system is basically implemented in the direction of corporate company (e.g. Software Company). However, it can be applied for other purposes in the web technologies (such as in Network management, Corporate Company’s discussion group, Job searching website etc.) to reduce the information load and directing someone in a sophisticated way to his interested area of information without losing a smart amount of time and data. In the corporate company area, people are using various independent systems together and due to having huge amount of data, the aggregated system very often forms ”information silos”. This problem is solved by CoASGen system by managing and structuring information using the idea of high level Activity Streams.
7.1.1 Explored Research Questions

The result of the research questions which were explored in Chapter one are addressed in these sections which are motivated by design, implementation and evaluation of CoASGen system. The results are summarized briefly in the following paragraph.

**How the system architecture and model would be designed considering requirements scenario?**

CoASGen system uses simple mechanism to retrieve feed items, generating activities, consolidation and generating high level Activity Stream. It has been designed with the idea of persistent data and efficient user access. The design and architecture (described in chapter 4) is fully satisfied with the system’s goal and requirements. For example, it is able to deal with heterogeneous data sources from different independent systems, it has an efficient aggregation and consolidation policy using five different level of Isolator (see chapter 4.3.2) and using its own standardized semantic format structure CoASGen system is able to generate high level Activity Stream which is easy to understand as well as works as several similar activity’s representative to mobilize user for finding their expected information. By giving graphical architecture working model in every small component as well as by evaluating the system, CoASGen system proves that the system architecture and model really works.

**What should be the way to transform feed items to activities?**

In CoASGen system, feed items are transformed to activities by using some configuration languages (see chapter 4.3.3). Using an activity format structure, different parts of each activity are selected using one of the configuration languages. For example, author names are configured by a configuration language where system automatically finds if feed item description contains sender name otherwise it will find the responsible organization name. And sender mail address would be selected if it cannot find any of the sender name or organization name. Other part for example, ”short technical description of feed item” is configured by configuring domain specific technical word selection from the feed item description. Finally, it generates human understandable activities by combining every particular selected part to represent in the user interface.

**How to realize the isolation of similar activities in an easily configurable manner?**

The isolation of similar activities is done by Consolidator (see chapter 4.3) using five different level of Isolator. Since Consolidator is very important component for the whole system, it has been specially taken care by maintaining efficient method of different Isolators and keeping them in a hierarchy of execution. Two Isolators (Regular Expression
Comparator and Frequency Comparator) plays a very important role in Consolidator to isolate similar feed items which are easily configurable by both system administrators (by setting XML file) and end user (in the configuration page).

How to aggregate atomic activities to high level Activity Stream?

High level Activity Streams are generating from several similar activities (see chapter 4.4) which are isolated by Consolidator. Using some predefined high level Activity Stream format, CoASGen system generates high level Activity Stream where it combines different semantic parts. The CoASGen system has intelligence to calculate and select different semantic parts of high level Activity Stream format (e.g. Subject, be verb, verb, object etc.) from feed items description of activities. Before generating final high level Activity Stream, CoASGen system has a checking system with sentence structure which confirms that the sentence it makes is human understandable. If the system realizes some mandatory part of a high level Activity Stream formats are not found by calculation, then it recommends to user another high level Activity Stream format to generate high level Activity Stream. There is also a domain based checking system which helps to identify some semantic parts (if system fails to identify these parts) to generate high level Activity Stream.

What should be the standard format of high level Activity Stream which is applicable for corporate use?

Three formats of high level Activity Stream (see chapter 4.4.2) have been standardized in CoASGen system which are General Activity Stream Format, Prefix Activity Stream Format and Postfix Activity Stream Format. Each of the formats has different semantic structure. By configuring the system user of administrator can choose any of the format to view high level Activity Stream. When CoASGen system generates high level Activity Stream from isolated activities, it tries to make a version of high level Activity Stream with all formats. It has a fault tolerant system which stops the system to generate any particular format of high level Activity Stream if it realizes that some mandatory semantic parts are missing (system could not find that parts because of lack of information in the isolated feed items) then it recommends the system to use another format. But it happens only when there are very few feed items in one isolated type.

7.1.2 Some features and Guide to adapt in other similar environment

CoASGen system is developed with the idea of domain specific language (DSL) which is easily adaptable with lot of similar system. The prototype of CoASGen system is developed for Software Company where number of heterogeneous independent system
could be running in parallel. It is applicable for other similar environment for example; in Corporate Organization, very often we see there are numbers of independent systems (e.g. internal communication service for marketing experts, employee community service and service community for technical support teams) are running to emphasize their internal interaction speed. If it is a big company (in this case there will huge feed update within short time) then they should configure the system in a way that it generated fine grained isolated high level Activity Stream. In this case system administrator should configure the system with low percentage in the Isolator; Regular Expression Comparator and Frequency Comparator. In addition, he should configure the activity generator with frequently available technical terms (words) which are meaningful. If the company is small and they do not have very frequent feed update, then system administrator can leave the system with high percentage in those Isolators configuration. This idea is similar in high level Activity Stream generating time duration, if the system is very frequent then system will get lots of new feeds within short time. So, they can keep it low range in configuration. Locality order based on importance is always preferable and stream deleting time duration from the database is depending on company’s internal thinking; if old data is useful for them then they should select it for long time.

7.2 Future work

CoASGen system’s design and architecture fulfills all of its goals and requirements. Furthermore, during the time of research some new questions arrives to enhance the system and the idea a bit further.

Use more semantics in AS Generator

By applying more semantics in CoASGen system’s Activity Stream Generator, it is possible to get more dynamic high level Activity Streams. It requires more linguistic research for formulating complex sentences by following grammar structure. Since CoASGen system is developed with extendable architecture, it can be easily extended by adding additional advance semantics.

Apply the idea of CoASGen system to Social Network

The idea of CoASGen system can be applicable to Social Network (e.g. Facebook, Google Plus, Twitter etc.) environment where several independent systems could be aggregated and frequently updating news feeds could be isolated as well as formulate to a new format of high level Activity Streams. In this case, it would be better to process data with parallel computing and the database should be transactional database with multi tier architecture.
Appendix A

End User Questionnaires

Questions

Configuration

1. There are certain parameters in the configuration system which will be applied to the system for deploying. How helpful was the Configuration system of this CoASGen system?

MetaActivityStream

2. MetaActivityStreams are the metadata of every particular high level Activity Stream. How helpful is it in CoASGen system?

High level Activity Stream

3. Is high level Activity Stream able to mobilize users to find their expected data or feed items by following these streams that it offers?
4. Do you think this high level Activity Stream will be helpful to reduce data loss and save user’s time?

High level Activity Stream Format

5. How do you like three different kinds of high level Activity Stream format?

Atomic Streams

6. How appealing is the representation of streams in the user interface? Does it lead to the feed items data description?
7. Please give your perfection level about the stream generation from direct feed items?
Appendix A. Questionnaire

Consolidation

8. Following one high level Activity Stream, all streams are just consolidated (isolated) streams of similar types. What do you think about the accuracy of consolidation process?

User Interface

9. Is created UI easy to understand?

Performance

10. What do you think about the response time of this system?
11. Please give your satisfaction level for the system’s overall structure and performance.

Results

The following table A.1 describes end user evaluation results.

Suggestions

1. Use transactional database with multi tier architecture to get better scalability.

2. Sentence formation should be updated with more grammar rule.

3. You should provide more information of activities in the User Interface (UI).

4. Consider UI from the point of view of dummy user.

5. May be you can make higher level system or block diagram to present business logic (backend logic) for end user understanding.
### Table A.1: The result table of end user evolution

<table>
<thead>
<tr>
<th>Category</th>
<th>Questions</th>
<th>User1</th>
<th>User2</th>
<th>User3</th>
<th>User4</th>
<th>User5</th>
<th>User6</th>
<th>User7</th>
<th>User8</th>
<th>User9</th>
<th>User10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>1</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>MetaActivityStream</td>
<td>2</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>High level Activity Stream</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>High level Activity Stream Format</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Atomic Streams</td>
<td>6</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>Consolidation</td>
<td>8</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>User Interface</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>9</td>
<td>5</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Performance</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>9</td>
</tr>
</tbody>
</table>
Bibliography

[1] Stacey Lan. Modern technology such as personal computers and the internet have made it possible for many people to do their work from home. what are some of the advantages and disadvantages of this situation? March 2011. URL http://www.oppapers.com/essays/Modern-Technology-Such-As-Personal-Computers/642311.


